Lucent Technologies
Bell Labs Innovations

VitalQIP®
Release 6.2
Administrator Reference Manual

190-409-042R6.2
Issue 1
June 30, 2005

Contents

About this document

Welcome to VitalQIP® - a powerful IP name and address management tool. VitalQIP simplifies the assignment and allocation of IP addresses and services, such as Dynamic Host Configuration Protocal (DHCP) and Domain Name System (DNS). This product is a comprehensive collection of management tools and user interfaces. Each management tool and user interface provides the ability to plan, manage, and locally administer IP addresses and services across Linux, UNIX and Windows 2000 platforms. VitalQIP works with directory services and RDBMS database configurations.

Purpose

This manual was created to assist you in preparing for your installation and rollout of the VitalQIP product. It contains an overview of the product and its relationship to DNS and DHCP, as well as network planning information. The remainder of the manual is intended as a reference to the more advanced aspects of using VitalQIP, such as working with policy files and message routes, setting up advanced DNS and DHCP configurations, and database management. The last part of the manual contains comprehensive troubleshooting information for VitalQIP, DNS, and DHCP.

It is important that you review this guide first, before you begin your installation of VitalQIP, especially if you are a first-time VitalQIP user.

Reason for reissue

Table 1 lists the changes to the VitalQIP GUI in version 6.2 that required the User's Guide to be reissued.

Table 1 Administrator Reference Manual changes

Feature name	Description	Feature impact
Redundant Schedule Service	VitalQIP 6.2 supports a redundancy mechanism to minimize VitalQIP downtime for Schedule Service.	• New Redundant Schedule Service section in Chapter 4. • New policies added to VitalQIP Schedule Service policies section in Chapter 4.

Feature name	Description	Feature impact
GSS-TSIG updates to Microsoft DNS	VitalQIP 6.2 supports GSS-TSIG update of Microsoft 2000 and 2003 DNS servers. VitalQIP also supports checking of the client's privileges for secure dynamic updates in support of MS Windows 2000/2003 secure update compliance.	• Changes to Windows 2000 secure zones support section in Chapter 7. • Additional policies for the VitalQIP MS DNS Update Service in Chapter 4.
Secure messaging	VitalQIP 6.2 supports the ability to encrypt and authenticate DHCP updates from a DHCP server to the QIP Update Service, as well as push requests and all other VitalQIP message traffic.	• New VitalQIP SSL Tunnel Service section in Chapters 3 and 4, and new global policy in Chapter 4. • Addition of Chapter 6, "Secure message routes".

How to use this information product

This manual is organized as follows:

Part I: Planning and configuration	Chapter 1: Background	This chapter provides background about the technology behind VitalQIP (DNS and DHCP) and the background on VitalQIP.
	Chapter 2: Plan and configure your network	This chapter provides information on planning and configuring your VitalQIP network.
Part II: Advanced management and configurations	Chapter 3: Manage VitalQIP services	This chapter describes the VitalQIP services and how to start and stop them.
	Chapter 4: VitalQIP policy files	This chapter describes the VitalQIP policy files.
	Chapter 5: Authenticate administrators	This chapter describes how to use the administrator authentication callout tool.
	Chapter 6: Secure message routes	This chapter provides detailed information on how to set up secure message routes.

	Chapter 7: Advanced DNS configurations	This chapter describes advanced DNS configurations, such as support of external objects, BIND, and so on.
	Chapter 8: Advanced DHCP configurations	This chapter describes advanced DHCP configaration, such as failover servers, AIX DHCP, and so on.
	Chapter 9: Database administration	This chapter covers database administration mainly for Sybase databases. Oracle databases must be maintained by an onsite Oracle database administrator.
Part III: Troubleshooting	Chapter 10: Troubleshooting VitalQIP	This chapter offers advice for resolving problems with VitalQIP.
	Chapter 11: Troubleshooting DNS	This chapter offers advice for resolving problems with DNS.
	Chapter 12: Troublestooting DHCP	This chapter offers advice for resolving problems with DHCP.
	Chapter 13: Other messages	This chapter contains common database error messages, service error messages, and information messages.

Table 2 lists the typographical conventions used throughout this manual.

Table 2 Typographical conventions

Convention	Meaning	Example
boldface	Names of items on screens. Names of commands and routines. Names of buttons you should click.	Select the **Client** check box. The **qip_getappllst** routine returns the entire list of existing applications. Click **OK**.

Convention	Meaning	Example
Helvetica bold	Names of keys on the keyboard to be pressed.	Press **Enter** to continue.
Letter Gothic	Output from commands, code listings, and log files	`# Name: Share` `shared-network _200_200_200_0 {`
Letter Gothic bold	Input that you should enter from your keyboard.	Run the following command: **`c:\setup.exe`**
<angle brackets>	Variables that you must substitute another value for.	*<debugfile>.bak.log*
italics	Manual and book titles.	See the *VitalQIP User's Guide* for more information.
Helvetica italic	Directories, paths, file names, and e-mail addresses.	A symbolic link must be created from */etc/named.conf* that points to *named.conf*.
bold italic	Uniform Resource Locators (URLs) and emphasis	The VitalQIP Web site is ***http://qip.lucent.com***.
click	Click the left button on your mouse once.	To delete the object, click **Delete**.
right-click	Click the right button on your mouse.	Right-click on a service.
double-click	Double-click the left button on your mouse.	Double-click the book icon.

Related information

The following documents are referenced in this manual:

- *VitalQIP Installation Guide* (part number: 190-409-043)

  This guide describes how to install the VitalQIP product.
- *VitalQIP Command Line Interface User's Guide* (part number: 190-409-044)

  This guide discusses and describes how to use the VitalQIP Command Line Interface.
- *VitalQIP User's Guide* (part number: 190-409-068)

  This guide describes how to set up and use the VitalQIP user interface on Linux, UNIX and Windows platforms.

Information product support

The Information Products & Training Group within Lucent offers cost-effective educational programs that cover the VitalQIP products and add-ons. Our offerings also include courses on the underlying technology for the VitalQIP products (for example, DNS and DHCP). Our classes blend presentation, discussion, and hands-on exercises to reinforce learning. Students acquire in-depth knowledge and gain expertise by practicing with our products in a controlled, instructor-facilitated setting. If you have any questions, please email us at *vitaleducation@lucent.com*.

Technical support

If you need assistance with VitalQIP, you can contact the Technical Assistance Center for your region. Contact information is provided in Table 3.

Table 3 Technical support information

Region	Address	Contact information
North, Central, and South America	Lucent Technologies 400 Lapp Road Malvern, PA 19355 USA	Phone: 1-866-LUCENT8 (582-3688) Option 5 Web: ***www.lucent.com/support***
Europe, Middle East, Africa, and China	Lucent Technologies Chiltern House Sterling Court Broad Lane Bracknell, RG12 9GU UK	Phone: 00 800 00 LUCENT or +353 1 692 4579 E-mail: *emeacallcenter@lucent.com* Web: ***www.lucent.com/support***
Asia Pacific	Lucent Technologies Australia 68 Waterloo Rd North Ryde NSW 2113 Australia	Phone: 1800-458-236 (toll free from within Australia) (IDD) 800-5823-6888 (toll free from Asia Pacific - Hong Kong, Indonesia, South Korea, Malaysia, New Zealand, Philippines, Singapore, Taiwan, and Thailand) (613) 9614-8530 (toll call from any country) E-mail: *apactss@lucent.com*

How to order

Customers can order additional VitalQIP manuals online at ***http://www.cic.lucent.com/documents.html***.

Suggested Reading and Internet Sites

The following books and Internet sites are suggested if you want additional information:

- *DNS and BIND, Fourth Edition*, Paul Albitz & Cricket Liu, O'Reilly & Associates, Inc.
- *UNIX in a Nutshell*, Daniel Gilly and the staff of O'Reilly & Associates, Inc.
- *Using Sybase System XI*, Special Edition, Peter Hazlehurst, Que
- *Oracle 9i Database Online Documentation*, Release 2 (9.2). http://otn.oracle.com/pls/db92/db92.homepage
- ***www.ietf.org*** – The Internet Engineering Task Force responsible for RFCs and Drafts.
- ***www.isc.org*** – The Internet Software Consortium for information on BIND.

How to comment

To comment on this information product, go to the Online Comment Form or email your comments to the Comments Hotline (comments@lucent.com).

□

Part I: Planning and configuration

Overview

Purpose

In Part I, you will find background information on the VitalQIP and the underlying technology, planning your network, and configuration options. Refer to Part I whenever you are setting up your system or need to review DHCP and DNS.

Contents

Part I contains the following chapters:

□

1 Background

Overview

Purpose

This chapter provides background information on VitalQIP and the technology behind VitalQIP. This chapter is intended to provided an introduction to VitalQIP and its infrastructure, a DNS overview in relation to VitalQIP, a DHCP overview in relation to VitalQIP, and an Internet Protocol overview in relation to VitalQIP.

This chapter is meant to aid you in your understanding of VitalQIP. See this chapter if you ever need to a simple description of the VitalQIP Management System or a review of DNS, DHCP, and Internet Protocol.

Contents

This information presents the following topics.

VitalQIP Management System Introduction

Network Administrators and operations groups must face the challenge of creating and managing their IP network address infrastructure. In a static environment, this may not be too difficult – but today's business environment is not static. Networks grow, organizations change, and technology is moving forward faster than ever before. Maintaining control of each element of the network is the challenge that the Lucent IP Management System, VitalQIP, has been designed to handle.

VitalQIP allows you to bring organizational control to the management of the IP infrastructure, in either a centralized or decentralized network administration structure. It allows you to create addressing and naming schemes within the desired corporate model, or to continue the existing scheme.

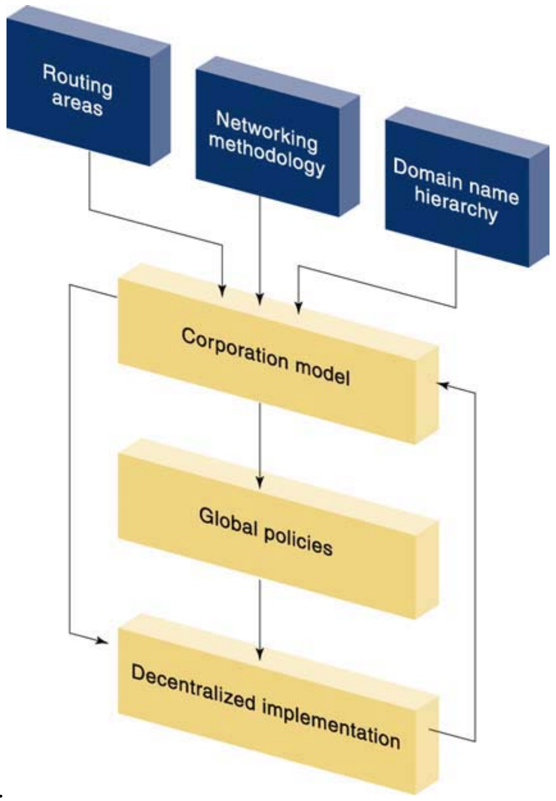
With VitalQIP, you can assign, reserve, move, allocate, reclaim, and view the addresses of networks, subnets, and network elements. In association with the IP addresses, VitalQIP maintains the corporate-wide Domain Name System, including such records as the hostname, alias, and mail exchangers.

VitalQIP maintains a complete view of the corporate address infrastructure, the domain name hierarchy, networks and subnets, DNS servers, Bootp servers, and DHCP servers. VitalQIP enables a comprehensive device profile to be maintained for every IP address and the state of the address. The state of an address can be unused, used, reserved, pending a move, or available for dynamic assignment. Reserved addresses and pending moves are time and date sensitive. This complete view is available to all authorized administrators, and partial views, such as a single subnet, can be made available to subnet administrators. VitalQIP supports a flexible administrative model with either update or read-only access on various components of your network.

The VitalQIP Infrastructure

The philosophy of the VitalQIP infrastructure is to mirror the organizational and networking environment of an organization. This infrastructure definition is then expanded to include every object that has a claim to reside on the network. To ensure a true reflection, the infrastructure environment developed must have the flexibility to accommodate multiple scenarios, such as accommodating a corporate acquisition or an autonomous division.

VitalQIP allows the network infrastructure to be defined from multiple viewpoints (as shown in Figure 1). Once defined, the internal policies can be applied, and then administrators are assigned to deployment and maintenance activities. The definition is inherently hierarchical, always providing a two-way interaction: from the top down through the enforcement of network policies, and from the bottom up through the authorized allocation of objects to their network locations.



.

Figure 1 Influences on the Network Infrastructure

VitalQIP utilizes the industry-standard Sybase database or Oracle database, which conforms to open systems architecture standards, as well as providing an excellent environment for scalability, as shown in Figure 2. Whether maintaining the infrastructure environment for a small company or single user or for a very large organization with many users, the database design model ensures a high level of performance.

Inherent in a proven relational database are the operational aspects expected of a high-availability system. Factors, such as backup, recovery, audit and history processing, SQL interface, and database decentralization are readily available.

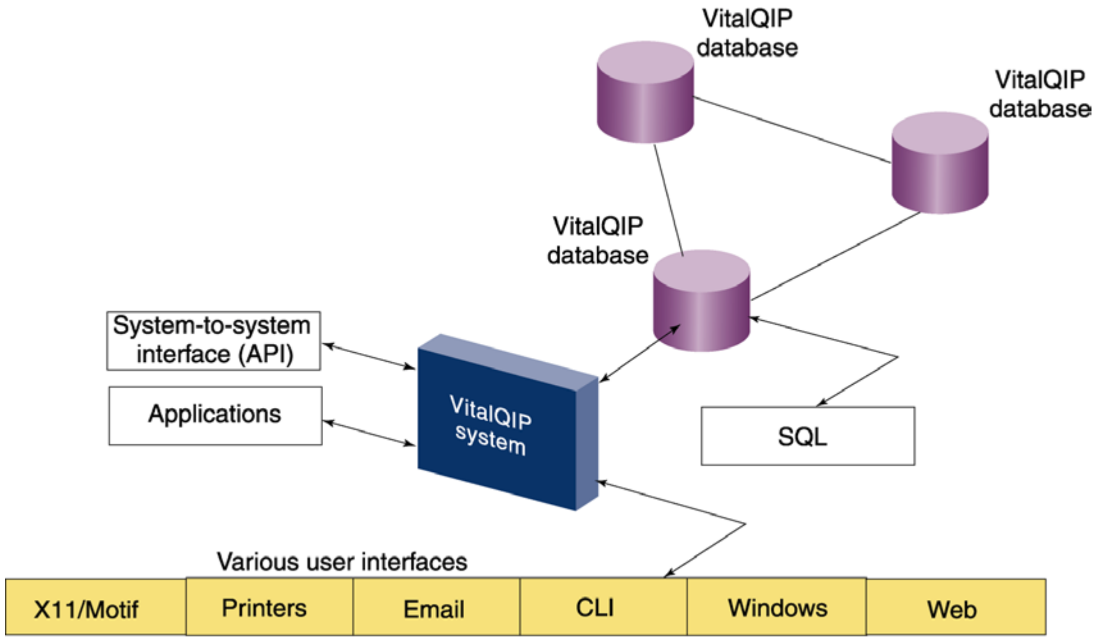


Figure 2 VitalQIP Open Architecture

VitalQIP has been designed so administrators have an easy-to-use interface that will guide them through all transaction processing. Remote users can also interact with VitalQIP through the command line interface (CLI) and the World Wide Web.

DNS Overview and VitalQIP

The technology behind VitalQIP is reliant on DNS (Domain Name Services). This section provides a brief background about DNS in relation to VitalQIP.

Domain Name Services

The theory behind domain names and the use of hierarchical name space is discussed in this section. You gain an understanding of domain names and how they are developed. Domain Name servers, those machines that utilize DNS and the records they store and process, are also covered, as well as the reasoning behind primary and secondary servers and how they benefit a network.

For further information on DNS, see *DNS and BIND, Fourth Edition* by Paul Albitz and Cricket Liu, published by O'Reilly and Associates, Inc. You can also read RFC 1034 and RFC 1035 for more information.

Dynamic DNS and DNS Overview

The DDNS/DNS is a system for mapping names to IP addresses and vise versa. On the Internet, DNS is used by virtually all inter-networking software, such as remote terminal programs like telnet, file transfer programs like FTP, and electronic mail like SMTP (Simple Mail Transfer Protocol). Its basic function is to provide name and address resolution on a network.

At the most basic level, DNS maps a name to an IP address. DNS has seen several enhancements since its inception. Security, flexibility, and ease of administration are addressed. DNS provides a single naming model throughout the distributed environment, based on the idea of a hierarchical name space, accommodating a large, rapidly expanding set of names.

DNS makes its information available throughout the network, regardless of whether the network is departmental, organization-wide, the Internet, or any mixture of the above.

For example, a user may request a host service by entering **telnet hostservice** at a command line interface, as shown in Figure 3.

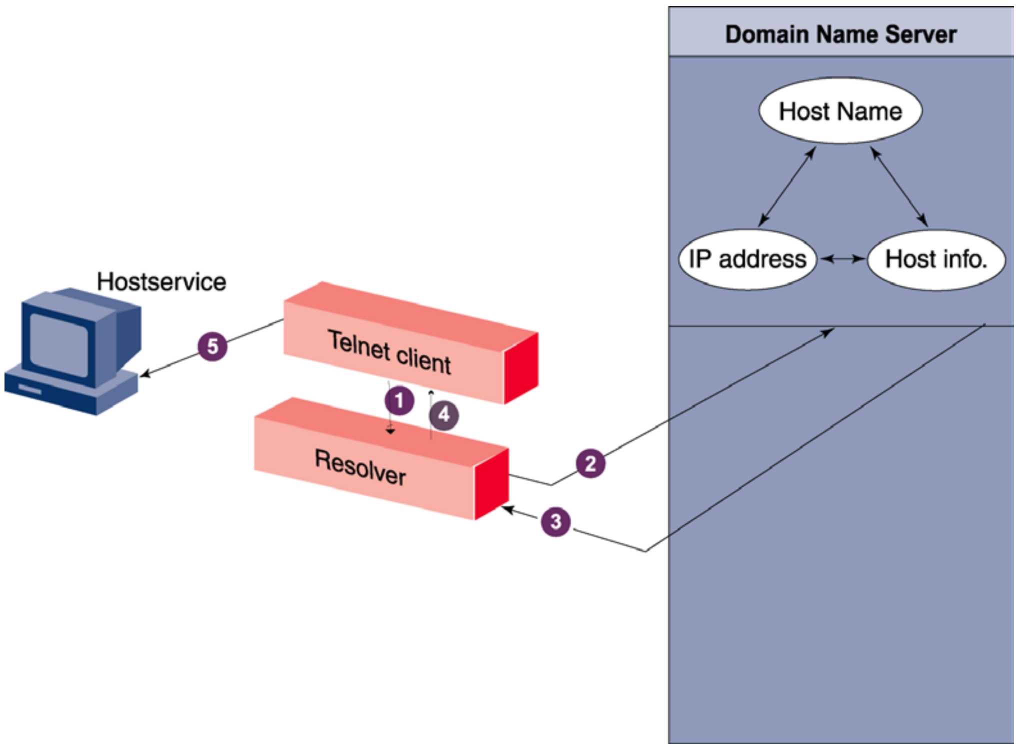


Figure 3 Routing of a Host Service Request

In Routing of a Host Service Request, the following occurs:

1. A client requests the name resolution of “hostservice”. The telnet client sends the request to the client's resolver.
2. The client resolver sends the request (query) to the DNS server, which determines corresponding IP address for “hostservice”.
3. The DNS server sends the IP address of “hostservice” to the client resolver.
4. The resolver sends the IP address to the telnet client.
5. Telnet uses the IP address received in step 3.

   Important! Host names are used in configuration, policy, and other types of files. Host names must be able to be resolved. DNS is one method to resolve host names.

The name space can be extended to roughly correspond to the organizational structure and needs. The model allows users to identify named network objects (for example, the servers, terminal servers, routers, gateways) and gain access to them without any knowledge of their physical location or address within the network. Furthermore, a user can continue referring to a network object by the same name, even when the characteristic of the object has changed.

Here is an example:

6. Host A's IP address is 111.111.111.101 and is used for a Payroll application. Payroll is a CNAME (alias) for HostA.
7. The Payroll application is moved to Host B at IP address 111.111.111.102. The DNS is updated to reflect the new IP address of the Payroll application. Payroll is no longer pointing to CNAME Host A but is pointing to CNAME Host B.
8. The user continues to access the application using the Payroll CNAME. The user is unaware of any application movement or address changes.

This provides flexibility in terms of change control, transparency, and consistency to the user community, program-to-program, or computer-to-computer environments.

DNS is a distributed database system for resolving host names into IP addresses. The host information is distributed among thousands of name servers organized into a hierarchy. This way, it does not require a central site to administer it. Each Domain Name server is designated responsibility for one or more domains and may refer requests for unknown hosts to another DNS server in the hierarchy, as shown in Figure 4.

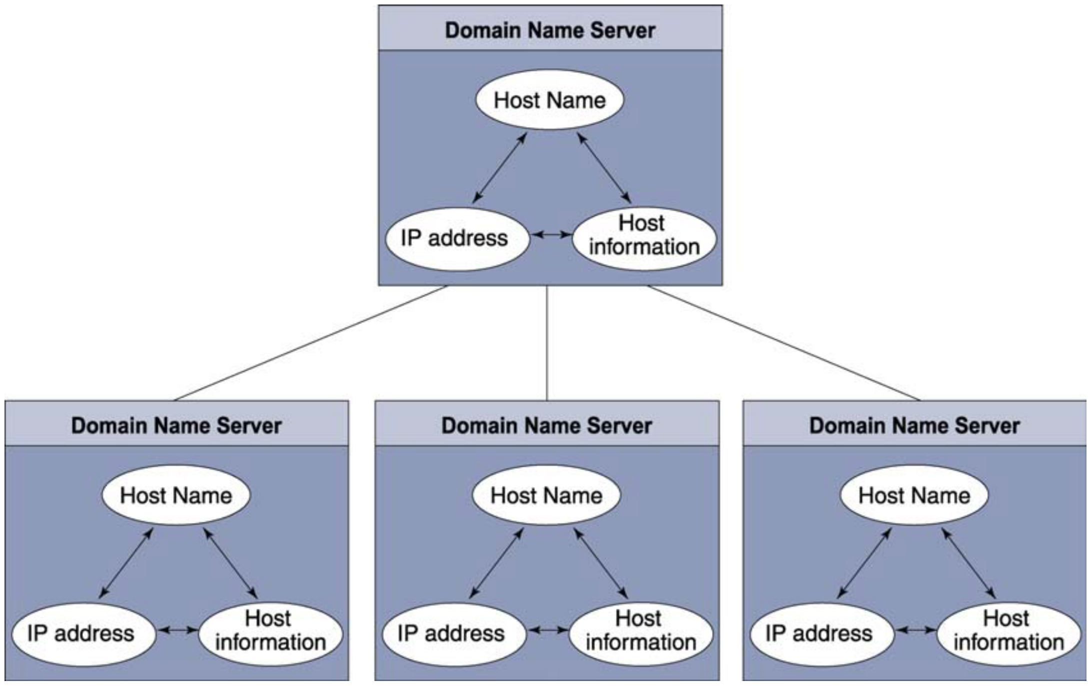


Figure 4 Host Name Resolution throughout the Domain Name Server Hierarchy

The three bottom Domain Name servers have been assigned responsibility for a domain. The top Domain Name server passes requests for unknown hosts to the bottom Domain Name servers. A bottom Domain Name server returns information about the unknown host if it has been assigned to a domain the Domain Name server administers.

The Berkeley Internet Name Domain (BIND) server implements DARPA Internet Domain Name specifications and protocol. BIND is based upon a client/server architecture where the client and the server are comprised of “resolver” and “named”.

The DNS primary function is to resolve a network object’s logical name to its network address(es) and vice versa. The client sends a query to the name server that the client has been configured to use. If the query can be resolved by that server, a response containing the resolution is sent back to the client. If the query cannot be resolved, a reason is sent back to the client.

Domain Names and their Relation to Hierarchical Name Space

DNS is inherently hierarchical and distributed in nature. DNS name space has a hierarchical organization, consisting of domains nested within each other. This hierarchical naming scheme is known as **domain names**.

Hierarchical name space provides autonomous control of domains and sub-domains, flexibility for change control, minimum requirements on both the server and network resources, which follow an administrative or an organizational structure.

The top of the domain hierarchy is the “root” (.), which is served by a group of name servers called “root” servers. Directly under the root domain are the top-level domains. Conceptually, there are two types of top-level domains: geographical and organizational. The geographic scheme divides top-level domains by country, and it is identified by a two-letter code (for instance, CA for Canada, JP for Japan). In the United States, the most commonly used top-level domains are organizational, including:

- *com* - commercial organizations
- *edu* - educational organizations
- *gov* - government organizations
- *mil* - military organizations
- *net* - major network centers
- *org* - organizations other than those above

No servers, including the root servers, have complete information about all domains, but most servers have pointers to other servers in the DNS hierarchy.

Domain names are written left to right from most significant (host name) to least significant (top level domain) with each domain level separated by a period (“.”). For example, the domain name, *cs.mit.edu,* contains three levels: *cs*, *mit*, and *edu.* In the above example, the lowest level domain is *cs.mit.edu*, the second level domain is *mit.edu*, and the top-level domain is *edu.*

The root of this hierarchy is maintained by the Network Information Center (NIC) and other organizations, as shown in Figure 5.

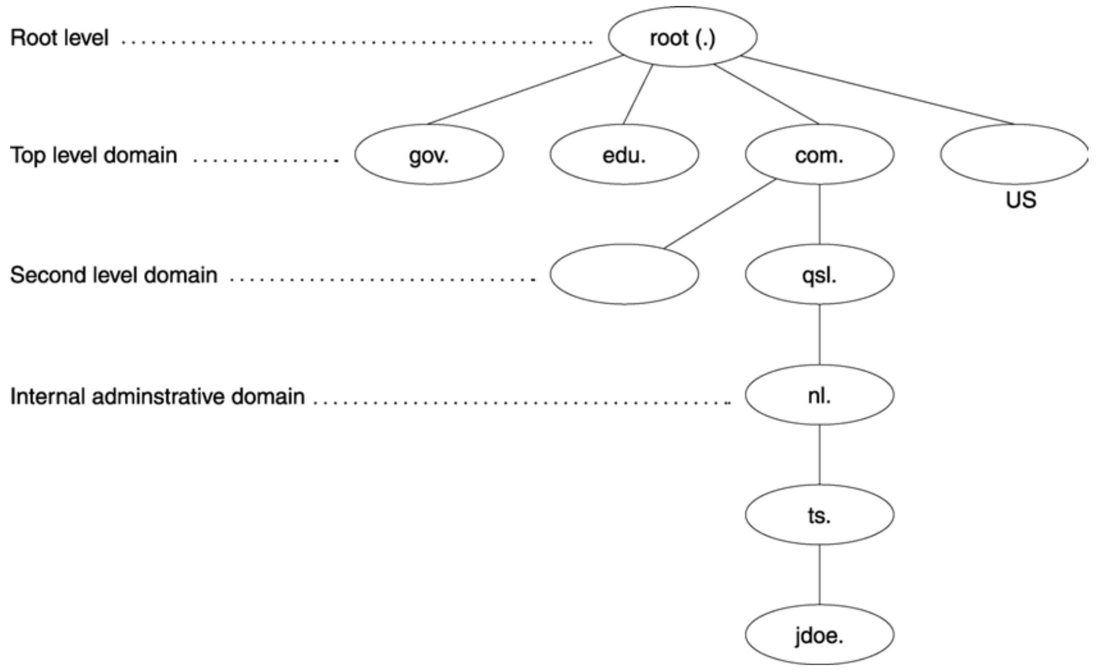


Figure 5 DNS Name Space Structure

In Figure 5, the different domain levels are shown:

1. Root level domain: The root of the tree is maintained on several servers across the world. The root domain name servers maintain information about name servers at the next lower level.
2. Top level domain: Every domain must be registered in one of the top level domains, such as COM, EDU, GOV, and so on. These top level domains, and their DNS, are maintained by the registrars.
3. Second level domain: These are the corporate domains. Each organization has the responsibility to maintain and administer its own domain.
4. Internal administrative domain: These are autonomous sub-domains within an organization. These sub-domains can be further nested into smaller sub-domains. The maintenance and administration of each sub-domain can be a centralized function or the responsibility of sub-organizations.

Name Servers, Zones, and Records

A name service resolves a computer name into a numeric address. Likewise, a name *server* stores (caches) information that makes up the domain database. The database is divided into sections called zones (domains), which are distributed among the name servers. The essential duty of a name server is answering queries using data in its zones.

Each DNS server contains resource records (RRs), which provide the data that make the DNS system work. VitalQIP generates RRs. Each zone can be available on a primary name server or secondary name servers. A zone is a set of RRs.

Start of authority (SOA) records specify version, refresh, retry, expire, and minimum values for a zone. SOA also specifies the primary name server and email address of the person/organization responsible for the zone. Name server (NS) records indicate the DNS servers that maintain a zone. Address (A) records indicate the IP address for a host.

Servers configured to reduce network overhead can cache these records. Each record includes a time-to-live (TTL) field that acts like the freshness date on food packaging.

Every master or slave zone has SOA and NS records indicating what machines have information about the zone. A host on your network that has a TCP/IP address may have a pair of RRs: an A (address) record to map the host name to an address, and a PTR record to map the address to a host name.

Primary and Secondary DNS Servers on a Network

A primary (authoritative) domain server must be assigned to each domain. Secondary (authoritative) servers can also be defined. In VitalQIP, one or more networks or subnets must also be assigned to each domain. A network can belong to multiple domains, but it is considered the primary (default) network for one of them. A primary server can have multiple primary and secondary zones.

VitalQIP allows name servers defined within VitalQIP to be configured as secondary servers against name servers and domains which are not managed by VitalQIP (non-managed servers). In establishing secondary servers for non-managed servers, you can set up a domain name that is not defined in VitalQIP, and the address of an authoritative server for this domain. VitalQIP automatically builds the *named.conf* file to include these non-managed zones.

For example, a company with the domain *stuff.com* has two sub-domains: *admin.stuff.com* and *research.stuff.com*. The Administration area has a DNS server that is primary for both *stuff.com* and *admin.stuff.com* and is not managed by VitalQIP. The Research area has a DNS server that is primary only for *research.stuff.com*, and is managed by VitalQIP. In this option, you can assign the Research DNS server as a secondary server for *stuff.com* by defining *stuff.com* in the non-managed servers section of VitalQIP.

When the inevitable changes are made, they need to be distributed to all name servers. Changes are coordinated at the primary server. The other non-master (secondary) servers for the zone periodically check for changes or are notified of changes to obtain new zone copies.

Each secondary server is required to check for changes and obtain new zone copies against the master. (The internal for checking is specified in the SOA as the zone refresh value.) A secondary may have multiple primary (master) servers defined. This list of master servers may include other secondary servers. This improves the transfer process when the primary is unavailable (due to host downtime or network problems).

Your servers, both primary and secondary, with their associated records are defined within VitalQIP. Each server's configuration is "pushed" to these servers with the **Network Services** function. See Chapter 5 in the *VitalQIP User's Guide* for more information.

VitalQIP Dynamic Update of Name Services – Lucent DNS Service

The Lucent DNS Service is based on code originally developed at U.C. Berkeley and currently maintained by the Internet Software Consortium. The Berkeley code is commonly referred to as **BIND**.

Lucent packages a BIND-compliant DNS server with VitalQIP. This is commonly referred to as Lucent DNS. Lucent DNS includes support of RFC 2136-compliant DDNS update. Lucent DNS is available on AIX, HP-UX, Sun Solaris, Linux Red Hat, and Windows 2000.

On a top level, a client requests information from a DNS server to resolve a name to an address. If the DNS server does not recognize the name (outside its scope of authority) or cache, it uses iterative queries to search through the name space hierarchy to find the authoritative server (either Internet or Intranet) that can answer the client's request. The originating DNS server (the one that the client originally asked), after searching the name space hierarchy, retrieves the answer from the authoritative server. It stores that answer and information collected during the search through the name space hierarchy in memory (cache).

The VitalQIP client can also dynamically update the Lucent DNS server. When an administrator adds an object in VitalQIP, the VitalQIP client sends an update to authoritative (primary and secondary) servers, causing that static object information to be updated immediately in DNS. Administrators do not need to wait until a VitalQIP push update of DNS files occurs to get static object information into DNS. If desired, you can turn dynamic update off for an object via the **Dynamic DNS Updates** check boxes in the Object Profile screen, or you can set a policy for all objects via policies in **Policies|Global Policies|Dynamic DNS**. Refer to "Global policies" in Chapter 2 of the *VitalQIP User's Guide* for more information.

Lucent DNS servers can update VitalQIP with external objects and resource records. Refer to "About external objects and resource records", on page 264 for more information.

DHCP Overview and VitalQIP

The technology behind VitalQIP is reliant on DHCP (Dynamic Host Configuration Protocol). This section provides a brief background about DHCP in relation to VitalQIP.

DHCP and Bootp Protocols

Bootp and DHCP are protocols that use a TCP/IP protocol stack (actually UDP) to automatically assign IP addresses and configure the operating environment of objects. These protocols provide a way for a server to configure the TCP/IP settings automatically for client workstations on the same network. This saves network administrators from having to set IP addresses and gateways, for instance, manually on every client on a network. DHCP is a very big time-saver for network administrators.

The DHCP server can be configured to assign Bootp addresses in addition to DHCP. The scope of the DHCP server can be defined at the corporate, domain, network, OSPF area, subnet group, or subnet level. This provides the flexibility to adopt a corporate, regional (for instance, network or OSPF area), or individual subnet Bootp and/or DHCP methodology.

Dynamic allocation is the function located in the VitalQIP client's Object Management window that is used to define multiple IP addresses at one time. Dynamic allocation assigns the addresses immediately and lets the user fill in information about the objects later. The following dynamic configuration methods can be used to assign addresses automatically:

- **None** allocates addresses without using Bootp or DHCP. No Bootp or DHCP servers or files are involved.
- **Manual Bootp** (MAC-based Bootp) allocates addresses using Bootp where the MAC address is defined.
- **Automatic Bootp** (MAC-less Bootp) allocates addresses using Bootp where the individual MAC addresses are not known.
- **Manual DHCP** (MAC-based DHCP) allocates addresses using DHCP where the MAC address is defined.
- **Automatic DHCP** (infinite lease DHCP) allocates addresses to a DHCP template with an infinite lease time.
- **Dynamic DHCP** (finite lease DHCP) allocates addresses to a DHCP template for a specific lease time.

About Bootp

A network object (the client) first broadcasts a Bootp request on the network. The broadcast contains the client's MAC address. A server on the network, designated as the Bootp server, receives the broadcast and checks the bootptab (Bootp table) for the client's MAC address. If the server finds the MAC address in its bootptab, it broadcasts a Bootp reply on the network. The Bootp reply contains essential network configuration information, such as its IP address, subnet mask, default router, DNS servers, and so on. The client receives the broadcast and configures its TCP/IP protocol stack.

The client has its protocol stack configured based on the Bootp reply. The client sends a configuration file transfer (using *tftp*) to the server that originally sent the Bootp reply or to another server as indicated in the Bootp reply. Once the client obtains the configuration details, the client sends file transfer requests (again using *tftp*) for the system's operating image.

The request for the operating image is optional; it is configured in the configuration file. For example, the operating image for PCs could be DOS utilities; for diskless workstations, it could be UNIX or file systems and so forth. Upon receiving the operating image, the client normally reboots and becomes operational on the network.

About DHCP

DHCP uses the UDP broadcast mechanism to automatically assign IP addresses and configure the operating environment of an object.

Although Bootp, when centrally administered, reduces the need to individually configure the desktop and the possibility of human errors, it introduces a second tier of problems. The MAC address has to be known, and the Bootp tables have to be constantly maintained for users' moves, or whenever network interface cards are replaced. In addition, Bootp table maintenance cannot accommodate multiple mobile users (using laptops and notebooks) without incurring significant personnel costs.

DHCP offers the promise of eliminating these problems, but it must be considered in relation to the entire enterprise. Today DHCP can operate at the LAN level and in isolation, but it needs to become part of the corporate address infrastructure strategy. For example, address strategy and the local DHCP implementation need to be the same, redundancy of DHCP servers must be assured, and allocation of duplicate addresses in a redundant server environment cannot be tolerated. Additionally, primary and secondary domain name servers must be dynamically updated, and network management systems must be fully aware of name to address bindings.

Today, companies implement DHCP at a local (subnet) level, with a single server, without any redundancy and without using the network management system to manage DHCP devices. There is no corporate address strategy adherence (other than an administrator manually typing in the correct address range). Typically, DHCP is a LAN implementation restricted to wide area Internet implementation.

VitalQIP enables DHCP to become strategically aligned with the corporate address strategy. VitalQIP maintains an address infrastructure as a single data source for all users, administrators, DNS servers, Bootp servers and DHCP servers to share. This enables infrastructure planning to become meaningful and makes it possible for an address strategy to be enforced. Redundant DHCP servers, dynamic update of DNS, and the network management system ensure network and desktop reliability and access.

VitalQIP is designed to be scalable and flexible in a multi-server environment, and to have a high availability mandate. Multiple DHCP servers can be configured for redundancy, ensuring that access is possible, even in the event that the main server is not accessible.

This DHCP server environment also allows the MAC addresses to be captured. With the host name, MAC address and IP address information in the database, help desk resolutions are simplified.

The overall design of VitalQIP enables a total address strategy to be implemented. The central data source of the network, domain infrastructure, device profile and IP address assignment enables all personnel within the organization to utilize the same data. The VitalQIP Command Line Interface enables this data source to interface with other applications and programs such that duplication of data and erroneous data is eliminated.

How DHCP Works

DHCP follows a process for the DHCP client (an Internet host using DHCP to obtain configuration parameters, such as a network address) to broadcast a discover request on its subnet. Through the broadcast request, the DHCP client asks any listening DHCP server (an Internet host that returns configuration parameters to DHCP clients) on the network to supply them with an IP address and associated parameters. A DHCP server on the network that is authorized to configure a particular client presents the client with an address through a reply to the client, in the form of an offer packet. Figure 6 shows the flow of the process.

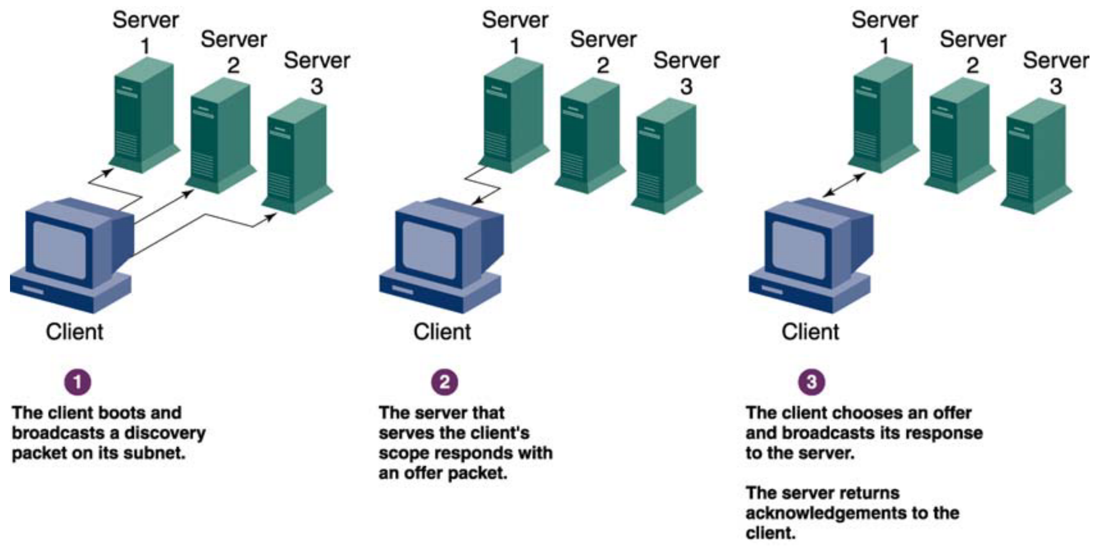

Server 1
Server 2
Server 3
Client
1
The client boots and broadcasts a discovery packet on its subnet.
Server 1
Server 2
Server 3
Client
2
The server that serves the client's scope responds with an offer packet.
Server 1
Server 2
Server 3
Client
3
The client chooses an offer and broadcasts its response to the server.
The server returns acknowledgements to the client.

Figure 6 DHCP Packet Flow

When the client receives the offer, it may decide to accept it or wait for other offers from other servers on the network. Finally, the client decides on a specific offer, and sends a request to accept the offer made by the DHCP server. At that time, the DHCP server replies with an acknowledgement of the client's choice and the subsequent assignment of the IP address and accompanying parameters. It is the client's decision as to what parameters to accept, and the client is required to ignore any unsupported parameters.

All the functions offered by VitalQIP software enable the network administrator to maximize the use of the DHCP protocol with relative ease.

Lucent DHCP

Lucent DHCP is an RFC 2131 and RFC 2132 compliant DHCP server with extensions that support responding to the older bootstrap (Bootp) protocol. It provides dynamic addresses to IP devices operating on the HP-UX, AIX, Sun Solaris, Linux Red Hat, and Windows platforms. Lucent DHCP is tightly integrated with the VitalQIP server, allowing dynamic updates from any number of Lucent DHCP servers to a central database of all dynamically assigned addresses. Lucent DHCP can reside on either the enterprise server or the remote server.

Lucent DHCP supports different client configuration options. Thus, the Lucent DHCP configuration file is created with five discrete sections, one for each of these modes. The VitalQIP server creates the configuration files for these dynamic modes. They are as follows:

- **Manual Bootp** - Manual Bootp is the base support for the older bootstrap protocol. It requires that a MAC address be known and assigned to a specific IP address in VitalQIP. VitalQIP creates entries in the Lucent DHCP configuration file, *dhcpd.conf*, similar to the old *bootptab* file. When a host issues a Bootp request, Lucent DHCP issues a Bootp response with the configured parameters for that device.
- **Automatic Bootp** - Automatic Bootp allows a host using the bootstrap protocol to obtain an address from a pool of IP addresses. The MAC address does not need to be known beforehand. When Lucent DHCP gives an IP address to a requesting device, the VitalQIP server is notified of the MAC address of the requestor. The VitalQIP database is updated to reflect this assignment. Once the IP address is assigned, that device is permanently given that IP address (in effect becoming a Manual Bootp-like entry).
- **Dynamic DHCP** - Dynamic DHCP is the standard DHCP object state. IP addresses are assigned to a pool and handed out to DHCP clients as they request them. This assignment is only valid for a finite period called the “Lease Time”, after which the client must ask to extend the use of that address with a “Lease Renewal”.
- **Manual DHCP** - Manual DHCP is analogous to Manual Bootp, except for a DHCP request. The MAC address of the DHCP client must be known beforehand, and a one-to-one relationship is established between the MAC address of a specific DHCP client and an IP address. The “Lease Time” associated with these objects is unlimited by default, but it may be configured to a shorter interval.
- **Automatic DHCP** - This is a hybrid between Dynamic and Manual DHCP. Like Dynamic DHCP, no MAC address is assigned to the object, so any DHCP client on the network can obtain an address. Since Automatic DHCP has an unlimited lease time, the IP address is permanently assigned to that MAC address when a client obtains an address. It becomes like a Manual DHCP address.

Lucent DHCP updates VitalQIP and the Lucent DNS server (Lucent DNS) with several components of lease information, including hostname, domain name, MAC addres, IP address, grant time, expiration time and relay agent information (option 82). These updates will only be forwarded from the DHCP server when the server has been properly configured, via the

UpdateQIP parameter. The server can be configured to support updates , or any combination if the following request types: grant, renew, release, decline, bootp, autorelease, delete, and/or expiration. A value of “all” can also be configured. The default is “none”.

Integration of DHCP and DNS with VitalQIP

VitalQIP provides a centralized approach to the management of the corporate-wide Domain Name System (DNS) hierarchy for all platforms. The VitalQIP Administrator defines one or more domains for your VitalQIP system. VitalQIP automatically associates the domains into a logical hierarchy. For example, if the domains *qtek.com* and *nj.qtek.com* are defined, VitalQIP creates the necessary hierarchy (that is, glue records).

The Lucent DNS server is dynamically linked to the VitalQIP database. It eliminates the need to generate DNS configuration files manually and restart DNS every time a change is made to it. You can remotely manage all DNS servers through VitalQIP.

A primary (authoritative) domain server must be assigned to each domain. Secondary (authoritative) servers can also be defined. One or more networks or subnets must also be assigned to each domain. A network can belong to multiple domains, but it is considered the primary (default) network for only one of them. A primary server host computer can represent multiple primary and secondary servers simultaneously.

Figure 7 shows in the case of Lucent DNS, VitalQIP can also update secondary servers, and the secondary servers "pull" their updates from the primary servers. VitalQIP also enables you to manage secondary servers for domains and networks that are not managed by VitalQIP, which is illustrated in the left portion of Figure 7.

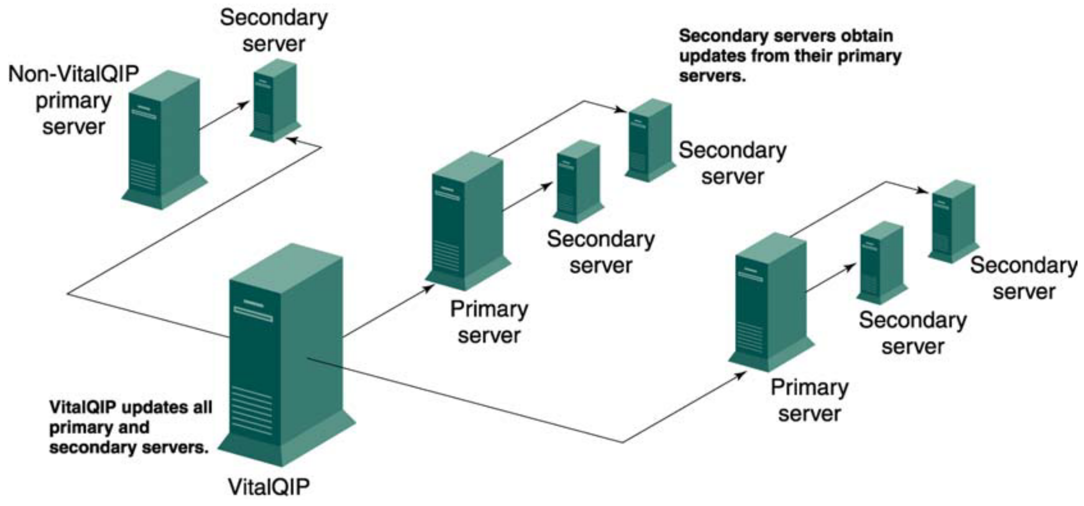

Non-VitalQIP primary server
Secondary server
Secondary servers obtain updates from their primary servers.
Secondary server
Secondary server
Primary server
Secondary server
Secondary server
Primary server
VitalQIP updates all primary and secondary servers.
VitalQIP

Figure 7 VitalQIP Updating DNS Servers

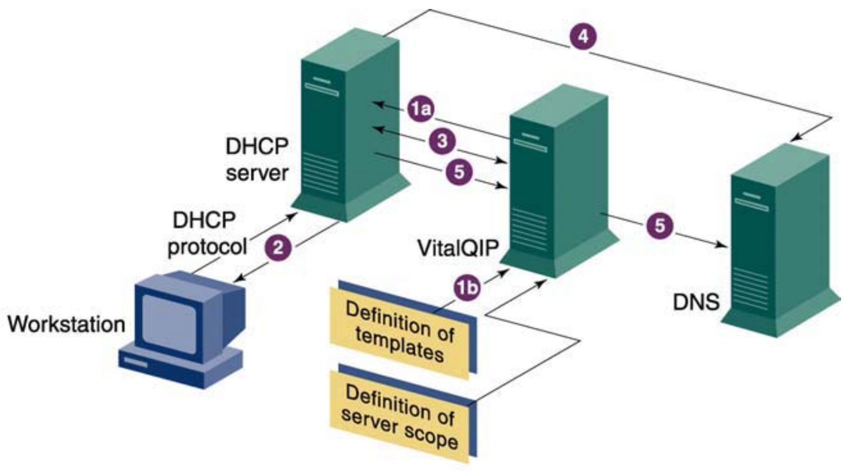
DDNS/DNS enables DHCP servers to dynamically update DNS servers in real time. As a DHCP server grants DHCP leases, VitalQIP automatically generates the appropriate records. The records are inserted into all authoritative DNS servers (primary and secondary), giving DHCP clients valid DNS entries immediately.

When the DHCP client accepts a lease containing the IP addresses and host name from the DHCP server, the DDNS/DNS primary and the Lucent DNS server (if it is a secondary server) are dynamically updated by VitalQIP with the IP address(es) and DHCP client's host name.

VitalQIP provides different methods of configuring your DHCP server and assigning them to network objects and subsequently to subnets:

- You can identify a list of MAC addresses (addresses assigned to specific hardware), which exist in a “pool”. These addresses can be assigned to registered clients only. When a client boots up, that client is checked by the specified server as to whether or not it should be assigned from this pool of MAC addresses. If it is a registered client, it is assigned an address from the pool. If the client is not a registered client, it is rejected.
- Exclusion MAC pool (XMAC) can be used to exclude (groups of) clients from getting an address through DHCP.
- A second method of DHCP/Bootp configuration is manual (static) addressing. This process assigns an address to a specific object manually. Each time that client boots, it receives the same address and no other client can have that address.
- A third method is dynamic configuration. This method allows any client who is assigned to a specific subnet to access a pool of addresses and have the first available address be assigned automatically.

In all cases, the server creates records for the database and updates the appropriate files on the DHCP server and, thus, the DNS server, how VitalQIP works with DHCP and DDNS/DNS (Figure 8).



1a A VitalQIP user, via the *Management* menu, allocates a range of addresses on a subnet to be defined to a particular DHCP server.

1b Network Services generate configuration files.

2 The network object (in this case, a workstation) uses the DHCP protocol to configure its network attributes.

3 The DHCP server immediately updates VitalQIP with the MAC address, IP address and workstation name.

4 The DHCP server updates DNS dynamically.

5 VitalQIP and the DHCP server and DNS server can be synchronized at prescheduled intervals if required (for recovery purposes only).

Figure 8 VitalQIP Interaction with DHCP and DNS

Internet Protocol and VitalQIP

VitalQIP is geared to the management of Internet Protocol (IP) as defined in RFC 791, the basic packet delivery service on which TCP/IP networks are built. The Internet Protocol is the foundation of the Internet and its functions include:

- Defining the datagram, which is the basic unit of transmission on the Internet
- Defining the Internet addressing scheme
- Routing datagrams
- Fragmenting and reassembling datagrams
- Moving data between the layers above and below the 7-layer OSI model

This section limits the discussion of the IP protocol to defining addressing schemes and explaining how an addressing scheme is implemented within VitalQIP.

IP Addresses

An IP address is a 32-bit binary address (consisting of four strings of eight bits each) that contains enough information to uniquely identify a network or a specific host. It is made up of two parts, a network part and a host part. The format of the address and the two parts is not uniform; it varies according to the class of address.

Network address class is determined by the following rules:

- A Class A address has the first bit set to zero:
  ```
  00000000     00000000     00000000     00000000
  ```
  The address range for class A is 00000001 to 01111110, which represents a network address from 1 to 126.
- A Class B address has the first two bits set to one and zero (10):
  ```
  10000000     00000000     00000000     00000000
  ```
  The address range for class B is 10000000.00000000 to 10111111.11111111, which represents a network address from 128.0 to 191.255.
- A Class C address has the first two bits set to one and the third bit set to zero (110):
  ```
  11000000     00000000     00000000     00000000
  ```
  The address range for class C is 11000000.00000000.00000000 to 11011111.11111111.11111111, which represents a network address from 192.0.0 to 223.255.25.5.
- A Class D address has the first three bits set ones and the fourth bit set zero (1110):
  ```
  11100000     00000000     00000000     00000000
  ```
  The address range for Class D is 11100000.00000000.00000000 to 11101111.11111111.11111111, which represents a network address from 224.0.0 to 239.255.255.5
- Class E addresses are reserved for future use and are not discussed in this document.

The network definition defines the number of hosts supported by a network, as shown in Figure 9 (shaded area equals network address; non-shaded area equals host address).

> **Important!** *The first address within a subnet is designated the subnet address. The last address is used to designate the broadcast address for the subnet. A simple way to calculate the number of hosts or subnets is to use 2 to the nth power -2. “N” is the number of bits used for hosts or subnets, and –2 is for excluding the network address and broadcast address.

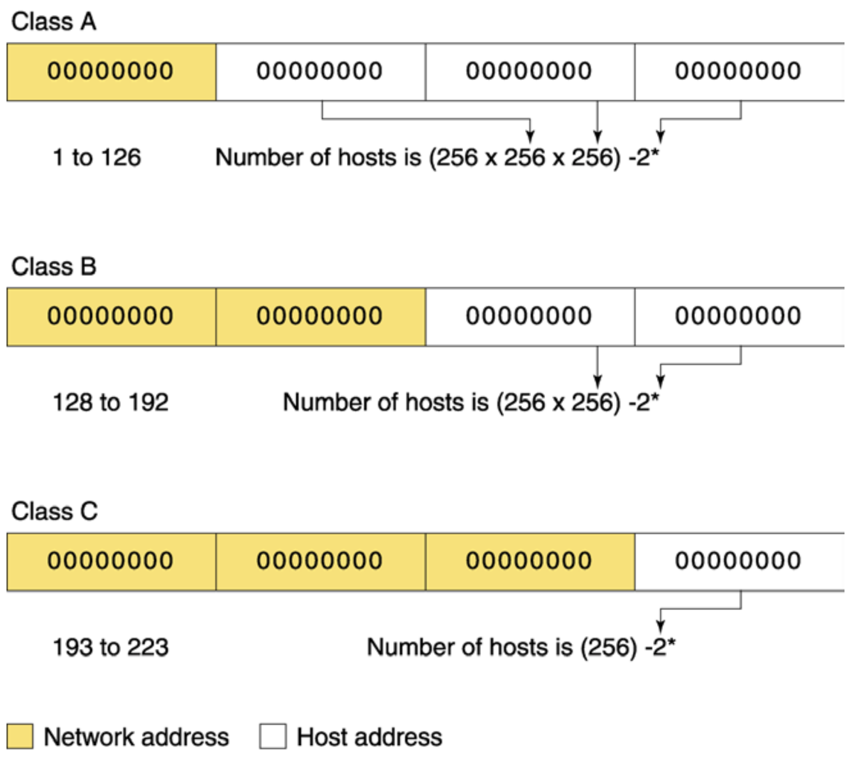


Figure 9 Number of Hosts supported by a network

The network definitions (shown in the shaded areas) are assigned by the Network Information Center (NIC). These definitions remain static for an organization. Organizations can use the host addresses in any way they choose.

Three networks are reserved for private use. These networks can be used internally by any organization.

The following table describes the private network address and their respective classes.

Table 4 Private Network Addresses

Network	**Network address class**
10.0.0.0/8	A
172.16.0.0/16	B
192.168.0.0/24	C

Subnetting

Subnetting is used to divide one large network into multiple smaller subnetworks, or subnets. The size of the subnets is typically determined by network administrators to best reflect the requirements of the organization. All networking equipment and routers understand and operate with subnet addresses.

A subnet mask is characterized in the same way as an IP address, with a 32-bit address broken into four octets. If there is no subnetting, then the following would apply:

Class A subnet mask 255.0.0.0	number of hosts = (256x256x256) - 2
Class B subnet mask 255.255.0.0	number of hosts = (256x256) - 2
Class C subnet mask 255.255.255.0	number of hosts = (256) - 2

The value of the subnet masks shown above would be the default masks for the respective network classes. However, these network definitions are completely impractical. If the value of the subnet mask is changed, the definition of the number of bits used to define the network and number of hosts changes. Table 5 shows what can happen by changing the subnet mask.

Table 5 Subnetting Example 1

Type	Address	1st octet	2nd octet	3rd octet	4th octet
Host Address	129.111.6.161	10000001	01101111	00000110	10100001
Subnet Mask	255.255.0.0	11111111	11111111	00000000	00000000
Subnet	129.111.0.0	10000001	01101111	00000000	00000000

In this example, the router directs traffic to the network defined as 129.111.0.0, so the whole Class B network is one large network with 256x256 hosts. If the subnet mask is changed, Table 6 shows what can be achieved.

Table 6 Subnetting Example 2

Type	Address	1st octet	2nd octet	3rd octet	4th octet
Host Address	129.111.6.161	10000001	01101111	00000110	10100001
Subnet Mask	255.255.255.0	11111111	11111111	11111111	00000000

Type	**Address**	**1st octet**	**2nd octet**	**3rd octet**	**4th octet**
Subnet	129.111.6.0	10000001	01101111	00000110	00000000

The router directs traffic to the network defined as 129.111.6.0, and the actual address is viewed as:

129.111.	6.	161
Network	Subnet	Host

Table 7 shows another example to create more networks and fewer host addresses.

Table 7 Subnetting Example 3

Type	**Address**	**1st octet**	**2nd octet**	**3rd octet**	**4th octet**
Address	129.111.6.161	10000001	01101111	00000110	10100001
Subnet Mask	255.255.255.128	11111111	11111111	11111111	**1**0000000
Subnet	129.111.6.128	10000001	01101111	00000110	10000000

Table 8 shows another example to create more networks and fewer host addresses.

Table 8 Subnetting Example 4

Type	**Address**	**1st octet**	**2nd octet**	**3rd octet**	**4th octet**
Address	129.111.6.161	10000001	01101111	00000110	10100001
Subnet Mask	255.255.255.248	11111111	11111111	11111111	**11111**000
Subnet	129.111.6.160	10000001	01101110	00000110	10100000

When subnet masks are employed, the first and last address of each subnet is reserved for the network and broadcast addresses, respectively. This means that for a Class B address 129.111.6.161 using the subnet mask 255.255.255.128:

- The subnet address is 129.111.6.128
- The broadcast address is 129.111.6.255

All subnet hosts are within the range 129.111.6.129 and 129.111.6.254.

Subnet Mask / Bit Numbers Reference

The following tables indicate how Class A, Class B, and Class C networks can be subdivided into subnets, and a quick reference for subnet mask /bit numbers.

Classes of Internet Addresses

Table 9 describes the classes of Internet Addresses.

Table 9 Internet Address Classes

Class	Range
A	1.0.0.0 to 126.255.255.255
B	128.0.0.0 to 191.255.255.255
C	192.0.0.0 to 223.255.255.255
D	224.0.0.0 to 239.255.255.255
E	240.0.0.0 to 254.255.255.255

Important! The network 0.0.0.0 is reserved for use as the default router, and the network 127.0.0.0 is reserved for the “loopback” function.

CIDR Address Block and Bits

Classless Inter-Domain Routing (CIDR) is an IP addressing scheme, which replaces older IP address scheme systems, based on A, B, and C. One IP address can be used to make many unique addresses. A CIDR address looks like a normal IP address, such as 188.122.122.1, but ends with a slash followed by a number. For example, 172.200.1.1/16. The slash and number are referred to as a bit.

A bit determines how many addresses are covered by the CIDR address. Lower numbers cover more addresses while higher numbers cover fewer addresses.

A group of addresses assigned to an organization is called a **block**.

Table 10 describes CIDR address block or bits. Two hosts have been subtracted for first and last addresses in the subnet.

Table 10 CIDR Address Block or Bits

Bits	Subnet Mask	Hosts
/8	255.0.0.0	16777214
/9	255.128.0.0	8388606

Bits	Subnet Mask	Hosts
/10	255.192.0.0	4194302
/11	255.224.0.0	2097150
/12	255.240.0.0	1048574
/13	255.248.0.0	524286
/14	255.252.0.0	262142
/15	255.254.0.0	131070
/16	255.255.0.0	65534
/17	255.255.128.0	32766
/18	255.255.192.0	16382
/19	255.255.224.0	8190
/20	255.255.240.0	4094
/21	255.255.248.0	2046
/22	255.255.252.0	1022
/23	255.255.254.0	510
/24	255.255.255.0	254
/25	255.255.255.128	126
/26	255.255.255.192	62
/27	255.255.255.224	30
/28	255.255.255.240	14
/29	255.255.255.248	6
/30	255.255.255.252	2
/31	255.255.255.254	0
/32	255.255.255.255	loopback

Class A

Table 11 describes Class A subnets. Two hosts have been subtracted for the first and last addresses in the subnet.

Table 11 Class A

Bits	Subnet Mask	Hosts	Subnets
/8	255.0.0.0	16777214	1
/9	255.128.0.0	8388606	2
/10	255.192.0.0	4194302	4
/11	255.224.0.0	2097150	8
/12	255.240.0.0	1048574	16
/13	255.248.0.0	524286	32
/14	255.252.0.0	262142	64
/15	255.254.0.0	131070	128
/16	255.255.0.0	65534	256
/17	255.255.128.0	32766	512
/18	255.255.192.0	16382	1024
/19	255.255.224.0	8190	2048
/20	255.255.240.0	4094	4094
/21	255.255.248.0	2046	8192
/22	255.255.252.0	1022	16384
/23	255.255.254.0	510	32768
/24	255.255.255.0	254	65536
/25	255.255.255.128	126	131072
/26	255.255.255.192	62	262144
/27	255.255.255.224	30	524288
/28	255.255.255.240	14	1048576
/29	255.255.255.248	6	2097152
/30	255.255.255.252	2	4194304
/31	255.255.255.254	0	8388608
/32	255.255.255.255	loopback	16777216

Class B

Table 12 describes Class B subnets. Two hosts have been subtracted for first and last addresses in the subnet.

Table 12 Class B

Bits	Subnet Mask	Hosts	Subnets
/16	255.255.0.0	65534	1
/17	255.255.128.0	32766	2
/18	255.255.192.0	16382	4
/19	255.255.224.0	8190	8
/20	255.255.240.0	4094	16
/21	255.255.248.0	2046	32
/22	255.255.252.0	1022	64
/23	255.255.254.0	510	128
/24	255.255.255.0	254	256
/25	255.255.255.128	126	512
/26	255.255.255.192	62	1024
/27	255.255.255.224	30	2048
/28	255.255.255.240	14	4096
/29	255.255.255.248	6	8192
/30	255.255.255.252	2	16384
/31	255.255.255.254	0	32768
/32	255.255.255.255	loopback	65536

Class C

Table 13 describes Class C subnets. Two hosts have been subtracted for first and last addresses in the subnet.

Table 13 Class C

Bits	Subnet Mask	Hosts	Subnets
/24	255.255.255.0	254	1

Bits	Subnet Mask	Hosts	Subnets
/25	255.255.255.128	126	2
/26	255.255.255.192	62	4
/27	255.255.255.224	30	8
/28	255.255.255.240	14	16
/29	255.255.255.248	6	32
/30	255.255.255.252	2	64
/31	255.255.255.254	0	128
/32	255.255.255.255	loopback	256

□

2 Plan and configure your network

Overview

Purpose

This chapter covers the planning and configuration of your network. In this chapter, you will find descriptions of VitalQIP components and services. Also, network planning, network configuration considerations, system requirements, and configuration scenarios are covered. Please read this chapter thoroughly before planning and configuring your network.

Contents

This information presents the following topics.

VitalQIP components

VitalQIP is comprised of many components and services. This section describes each component and service. System requirements for installing the components are covered in the *VitalQIP Installation Guide*.

VitalQIP server

The term **VitalQIP server** or **enterprise server** typically refers to the system that contains the relational database management system and all of the required enterprise server services. The VitalQIP enterprise server can be installed on Windows or UNIX platforms (including LINUX).

VitalQIP remote server

The term **VitalQIP remote server** or **remote server** typically refers to the systems that contain your network services, such as DNS, DHCP, and all the required remote services. VitalQIP remote servers can be installed on supported Windows or UNIX platforms (including Linux).

Distributed server

The term **distributed server** is used for servers that run across independent systems. Refer to “VitalQIP services”, on page 35 for information about distributed services.

VitalQIP Web Client Interface

The term **VitalQIP web client** interface or **web client** typically refers to a set of Common Gateway Interface (CGI) scripts installed on a web server system. These scripts serve as a VitalQIP administrative user interface to industry standard web browsers. The web client interface is not a fully functional administrative user interface, but it provides basic administrative functions. The web client can be installed on Windows or UNIX platforms.

VitalQIP client interface

The term **VitalQIP client** or **VitalQIP graphical user interface** (GUI) typically refers to the system(s) that runs the VitalQIP Windows client interface or VitalQIP Motif/UNIX client interface. The VitalQIP client provides all user and infrastructure functions of VitalQIP for client computers running Windows or UNIX.

VitalQIP Command Line Interface

The term **VitalQIP Command Line Interface** (CLI) refers to a command line interface. The CLI provides another alternative process to using the VitalQIP GUI. Commands permit the use of a prompt to carry out VitalQIP functions.

□

VitalQIP services

The majority of VitalQIP's communication and maintenance is based upon the use of services. Services are installed during a VitalQIP component's installation, such as the VitalQIP enterprise or remote server. Some services are installed with only one component. For example, the VitalQIP Active Lease Service is installed only on the VitalQIP remote server. Other services are included in several component installations, such as the VitalQIP Message Service. The Message Service is included in the installation of the enterprise, remote, and distributed servers.

Many services are versatile and can be optionally installed on various servers. The term **distributed services** refers to a select group of VitalQIP services that can be installed on any number of servers, as needed, for load balancing or off-load processing. The following services are part of the distributed services:

- VitalQIP DNS Update Service
- File Generation Service
- VitalQIP Message Service
- VitalQIP SSL Tunnel Service
- VitalQIP Login Service
- VitalQIP Schedule Service
- VitalQIP QIP Update Service

These services are included with component installations. However, they can be optionally installed on servers not running a VitalQIP component.

The behavior of VitalQIP services is controlled by a policy file. A policy file contains parameters referred to as **policies**, which affect a service's behavior. Refer to "Manage the VitalQIP policy file", on page 95 for more information on policies.

For UNIX platforms, a **daemon** is the equivalent of a Windows service. See "VitalQIP services on UNIX", on page 67 for more detailed information about daemons.

A brief description of the service/daemon follows.

VitalQIP Schedule Service (qipd)

The VitalQIP Schedule Service handles all scheduled events managed by the VitalQIP server. These events include:

- Scheduled moves
- Scheduled reclaims
- Automatic updates of DNS, Bootp, and Local files
- Generating Domain Controller configurations
- Validating the license key
- Purging tombstoned records

Redundant Schedule Service

To eliminate the possibility that the Schedule Service may become a single point of failure, two or more Schedule Services may be run simultaneously against the same database. For this reason, this service may reside on the VitalQIP enterprise server and on distributed servers.

VitalQIP QIP Update Service (qip-qipupdated)

The VitalQIP QIP Update Service sends updated data for a DHCP address and binding information as well as external DNS updates to the VitalQIP enterprise server. The service optionally updates DNS dynamically.

The QIP Update Service receives DHCP and DNS messages on the port specified by **qip-qdhcp** from the Message Service (**qip-msgd**). Each message received results in an action to add, renew, or delete DHCP leases or external objects/resource records from the VitalQIP database, as well as to update DNS information. Multiple Message Services may connect to the Update Service. Since the QIP Update Service processes requests in a synchronous manner, when the connection to the database goes down, all connections to the Message Services are closed. The Message Service queues DHCP and DNS requests until it reconnects to the Update Service.

This service can exist on the VitalQIP enterprise server and distributed servers.

VitalQIP Message Service (qip-msgd)

The VitalQIP Message Service (**qip-msgd**) does the following:

- Queues messages until they can be processed by another service
- Forwards SSL-enabled or cleartext messages to multiple destinations
- Provides message queue length restrictions
- Provides disk-based storage for messages
- Accepts messages from more than one source
- Provides reliable transport for remote sources
- Maintains compatibility with previous versions of the Message Service

The VitalQIP Message Service queues all messages from the DHCP server, DNS server, DHCP monitor services, the VitalQIP GUI, and the VitalQIP QIP Update Service and forwards the messages to other services. The final destination of messages sent to the VitalQIP Message Service depends upon the message type. For example, the DHCP server sends messages of DHCP; DHCP messages are typically sent to the QIP Update Service. These messages are not sent directly to the final destination (and are routed through the VitalQIP Message Service) because the update service typically cannot process messages as fast as the DHCP server can generate them. The Message Service has multiple queues, one for each message type.

- The services that are the destination for most messages are:
- VitalQIP QIP Update Service
- VitalQIP Audit Update Service (For information on the VitalQIP Audit Update Service, see the *Audit Manager User's Guide*)

- VitalQIP DNS Update Service
- VitalQIP Login Service

The Message Service also acts as conduit for all communication with the Login, Remote, Active Lease, QIP Update, DNS Update and SSL Tunnel services. Starting in VitalQIP 6.2, only the Message Service port will be well known and active on the network interface by default. All of the other VitalQIP services (Login, Remote, Active Lease, QIP Update, DNS Update and SSL Tunnel) grab ephemeral ports and register those connections with the Message Service. This action reduces the number of ports open for firewalls.

The Message Service can be configured through its policy file, allowing the user to specify details about message queuing and message delivery (message routing).

Important! The Message Service has three policies, MessageQueue, MessageRoute, and SecureMessageRoute, that control message queues and message routing. A Message Queue defines attributes of a ***queue*** defined by a message type. A Message Route defines attributes of a ***destination*** defined by a message type. A SecureMessageRoute defines a message type that connects through the SSL Tunnel Service to make use of message traffic authentication and encryption.

VitalQIP SSL Tunnel Service (qip-ssltd)

The VitalQIP SSL Tunnel Service provides you with the option to secure all communication between services tunneled through the Message Service with Secure Socket Layer encryption.

VitalQIP Active Lease Service (qip-netd)

The VitalQIP Active Lease Service enables VitalQIP to display the current active leases and enables leases to be deleted. This service resides on a remote server running DHCP services.

Since the Active Lease Service binds to an ephemeral port on the loopback address and then registers with Message Service, all communication with the Active Lease Service passes through the Message Service. This action reduces the number of ports open for firewalls.

VitalQIP Microsoft DHCP Monitor Service (MSDHCPMonitorService)

The VitalQIP Microsoft DHCP Monitor Service is a service used to support Microsoft DHCP 2000 servers. Since this service only runs on Windows, there is no daemon associated with it.

VitalQIP Login Service (qip-logind)

The VitalQIP Login Service passes user names and passwords from the Message Service to all services and processes, including the VitalQIP clients that connect to the database.

The implementation of the Login Service allows VitalQIP to use its own username password mechanism. A database administrator can keep passwords for the VitalQIP databases secret and store passwords in a central location. Users of VitalQIP applications need only know the VitalQIP-level usernames and passwords (not the actual database usernames and passwords.)

When the VitalQIP client is started, it queries the Login Service for the database username and password, and logs into the database with these values. VitalQIP ensures that the user-supplied username and password matches what is stored in the VitalQIP database. The VitalQIP administrator does not need to know the database username and password.

The Login Service receives login request messages on the port specified by **qip-login**. It responds with all servers, usernames, and passwords configured for VitalQIP or Audit Manager databases. For information about the Audit Manager database, see the *Audit Manager User's Guide*.

This service exists on the VitalQIP enterprise server or a distributed server.

VitalQIP DNS Update Service (qip-dnsupdated)

The VitalQIP DNS Update Service propagates external dynamic DNS messages from other servers. It also processes DNS updates from the VitalQIP client and QIP Update Service. The service can be used to limit zone Access Control Lists for dynamic DNS updates. This service is installed on the VitalQIP enterprise server or a distributed server.

VitalQIP MS DNS Update Service

The VitalQIP MS DNS Update Service handles DNSDeltaNotification messages. These messages are received when the Remote Service sends data to DNS servers. The message triggers the MS DNS Update Service to read the add and delete files that are generated by the push. The service then sends the files to the DNS server.

The MS DNS Update Service also transfers zones when the remote server is configured with proxies or when the global proxies are set for a server’s organization. This functionality is used when there is a NAT or firewall between the remote server and the File Generation Service that causes AXFRs to be denied or cannot be routed between the two machines.

In this case, the AXFR is done locally by the MS DNS Message Service. The retrieved AXFR information is routed through the Message Service proxies to the File Generation Service.

This service uses an ephemeral port to register with the Message Service on the remote server. This service is only available on Windows 2000 and 2003 remote servers.

VitalQIP Remote Service (qip-rmtd)

The VitalQIP Remote Service transfers DHCP, Bootp, or DNS files from the VitalQIP enterprise server to remote servers. This service is only installed on a remote server.

Since the Remote Service binds to an ephemeral port on the loopback address and then registers with Message Service, all communication with the Remote Service passes through the Message Service. This action helps reduce the number of ports open for firewalls.

VitalQIP File Generation Service

The VitalQIP File Generation Service transfers Bootp, DHCP, DNS, NEX and/or Local Host Data and configurations files from the enterprise server to the remote server. The File Generation Service can run on your enterprise server or on a distributed server, and controls the generation of entire files for remote servers. Since this service acts as a go-between for the remote and enterprise servers, it prevents database communication between the two servers and thereby reduces overall traffic and improves file generation times.

This service is comprised of two "sub-services" called the VitalQIP Remote Method Invocation (RMI) Scheduler Service (**qip-rmisched**) and the VitalQIP RMI QAPI Service (**qapi**). These "sub-services" are referred to as services, but keep in mind that they work together to create the File Generation Service. The RMI QAPI Service is a subset of the RMI Scheduler Service.

The VitalQIP RMI Scheduler Service (**qip-rmisched**) is used to schedule access to the RMI QAPI Service. (An RMI QAPI Service is a separate executable started by the RMI Scheduler Service that gives clients access to the QAPI libraries.) This service is a Java-based RMI Service and does the actual file generation for the remote server configuration requests. The RMI server keeps track of which services are free to perform file generation.

Important! By default, the RMI Scheduler Service creates a RMI registry on port 1099 and registers itself as a RMI QAPI Scheduler. The port number and the name under which the service registers is configurable in the service's policy file.

RMI Scheduler Service

All file generation is performed on the server running the File Generation Service. Since the QAPI and **qipdb** libraries are not "thread safe", it is necessary to start multiple services (one service for each transfer) to generate files for multiple remotes simultaneously. This service controls what client has access to a QAPI service at a given time (effectively doing the work the kernel does if the libraries are thread safe and only one multithreaded RMI QAPI service is needed). Thus, the RMI Scheduler Service starts a specified number (5 by default) of RMI QAPI Services and controls access to them by forcing all clients to ask permission to communicate with a RMI QAPI Service. The term used to see a resource managed by the RMI Scheduler Service is called an **executor**. By default, each RMI QAPI Service is an executor.

RMI QAPI Service

The RMI QAPI Service is responsible for generating files for the remote servers. Although it is a Java application, it loads most VitalQIP libraries (**common**, **net**, **qipdb**, **qapi**, **qapijni**, **qsijni**) as well as the Rogue-Wave tools library. This service is a copy of the Java binary, which produces meaningful output in a process list.

The RMI QAPI Service is started automatically by the RMI Scheduler Service. Additional RMI QAPI Services on other servers can be started to improve overall performance. See "qapi - VitalQIP RMI QAPI Service daemon", on page 88 for information on starting other RMI QAPI Services. One RMI QAPI Service is started for each remote push. The RMI QAPI Service is a Java-based application and uses the Java debug policies.

These services can be installed on the VitalQIP enterprise server or a distributed server.

Lucent DHCP Service (dhcpd)

The Lucent DHCP Service provides Dynamic Host Configuration Protocol services to the network.   The service runs on the VitalQIP enterprise and remote server.

Lucent DNS Service (named)

The Lucent Domain Name Services (**named**) provides DNS support to the network. The service runs on the VitalQIP enterprise and remote server.

□

VitalQIP Service Configurations

VitalQIP services can be configured in a variety of ways. Typically, a group of VitalQIP services resides on the enterprise server and another group resides on the remote server. Figure 10 shows how services are typically configured for VitalQIP.

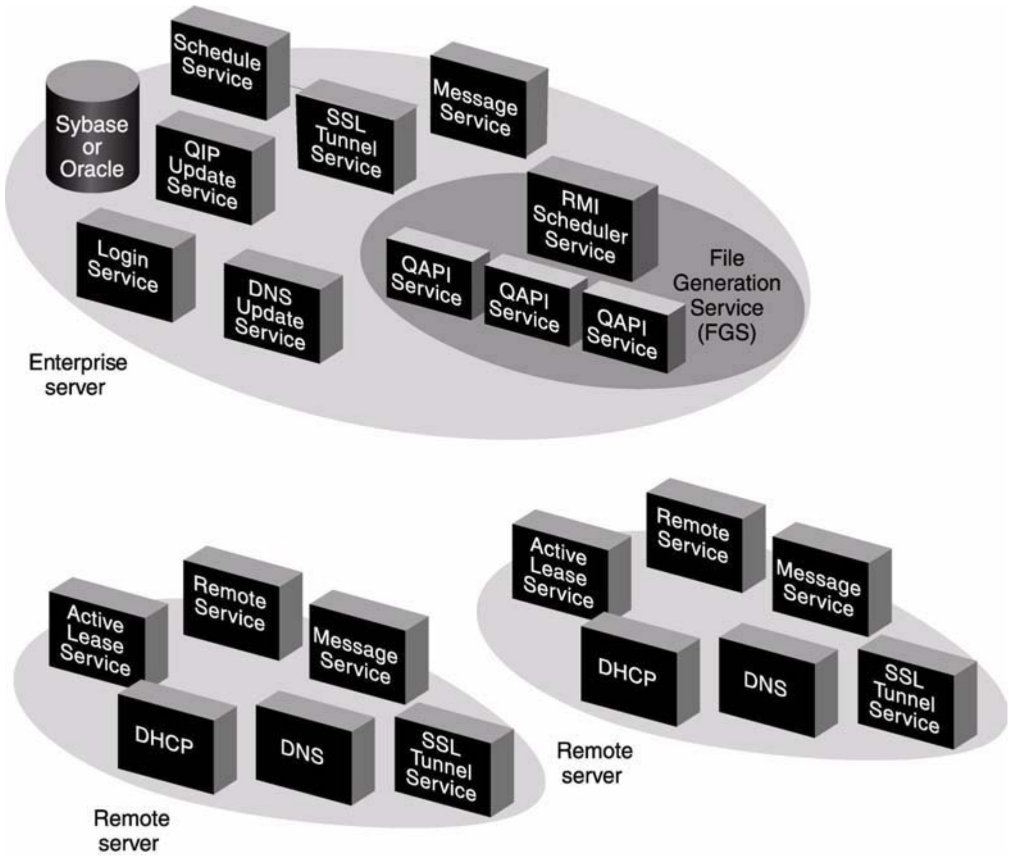


Figure 10 Typical VitalQIP service configuration

In Figure 10, the Schedule Service, Login Service, DNS Update Service, QIP Update Service, File Generation Service, Message Service, SSL Tunnel Service, and database reside on the VitalQIP enterprise server. The remote server has the Active Lease Service, Remote Service, DHCP, DNS, Message Service, and SSL Tunnel Service installed on it.

Alternatively, VitalQIP services can be installed in another configuration, which involves a distributed server. A service can reside on the enterprise server, a group of VitalQIP services can be installed on a distributed server, and another group can be installed on the remote server. This alternative configuration distributes the load to more machines. Figure 11 shows how services can be alternatively configured for VitalQIP using a distributed server.

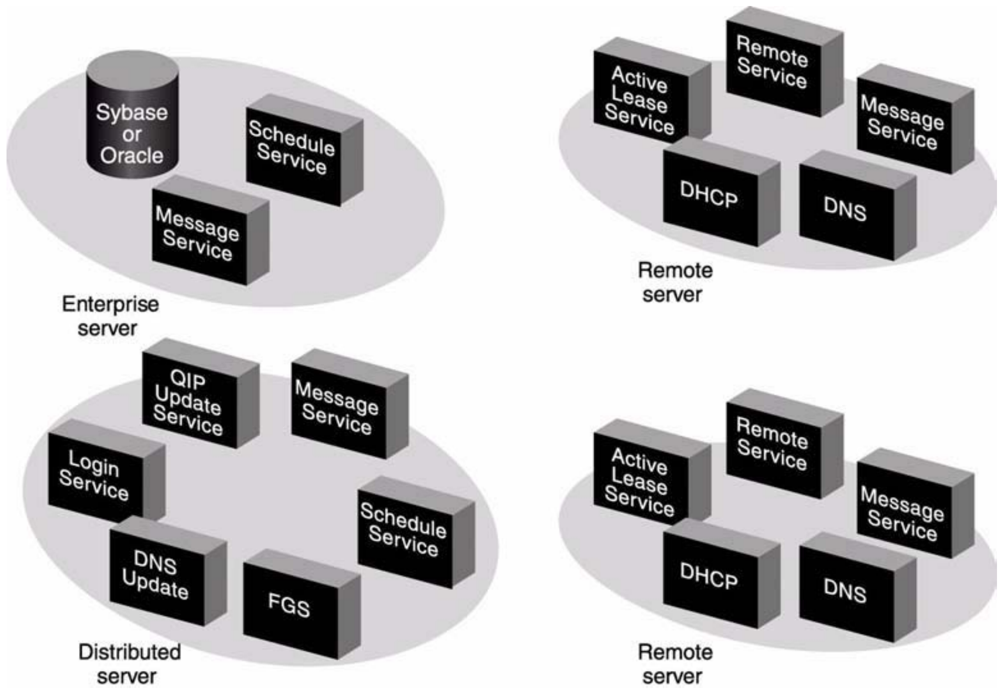


Figure 11 Alternative VitalQIP service configuration

In Figure 11, the Schedule Service, Message Service, and database reside on the enterprise server. The Login Service, DNS Update Service, QIP Update Service, File Generation Service, Schedule Service, and Message Service reside on a distributed server. The remote server has the Active Lease Service, Remote Service, DHCP, DNS, and Message Service installed on it.

Important! Figure 11 shows a configuration in which SSL is not being used. If it were being used, it would reside on the enterprise, distributed, and remote servers.

Other service configurations are possible. See "Installation Configurations" on page 44 for more information on VitalQIP configurations.

☐

Plan Your Network

The VitalQIP components are available on different platforms and can be organized into a variety of ways to best suit your system needs. The VitalQIP server can be either Windows or UNIX. The VitalQIP remote server can also be either Windows or UNIX servers (including Linux), as can the VitalQIP clients and the web servers.

VitalQIP can be installed in a variety of configurations - a single server or in a multi-server environment. You must decide prior to the installation of VitalQIP what your configuration will be, so that you select the right method of installation. Sample scenario configurations are shown in "Installation Configurations" on page 44.

Network Configuration Considerations

Planning your network topology is a necessary preliminary step. Before beginning the installation, you should have answers to the following questions.

1. Which server will be the VitalQIP enterprise server (also referred to as "VitalQIP server")? (The enterprise server contains the database system, such as Sybase or Oracle.)
2. How many VitalQIP remote servers will you have?
3. How many primary (master)/secondary (slave) servers will there be?
4. Will a failover server be used?
5. Will the VitalQIP remote server be DNS servers, DHCP servers, or both?
6. Which server will act as the VitalQIP web server?
7. How will my services be deployed?
8. Will a distributed server be used? If so, how many?
9. Would you like to generate SSL public/private keys to encrypt the message service transport?

Answering questions like these before you begin your installation makes the installation more effective and minimizes the time you spend installing the VitalQIP services.

The system requirements for each of the configurations may vary. Before you proceed, you must ensure your environment meets the configuration requirements. Look over the following configurations for one that may suit your environment.

Important! To determine system requirements for specific platforms and components, see the *VitalQIP 6.2 Release Notes and Upgrade Instructions*.

Installation Configurations

VitalQIP can be configured in a variety of ways. This section discusses different configuration scenarios in which VitalQIP can be set up. For more information about system and installation requirements, see the *VitalQIP Installation Guide*.

Single Server Scenario

Figure 12 shows a scenario where all VitalQIP components are installed on one server. This includes VitalQIP enterprise server, distributed services, VitalQIP remote servers, VitalQIP web client interface, and VitalQIP client.

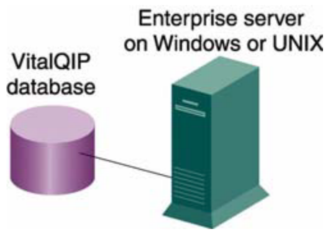


Figure 12 All VitalQIP components on a single server

The product components required are as follows:

- Sybase or Oracle database system
- VitalQIP server components involving the client, all required services, and the Command Line Interface
- Distributed services
- VitalQIP remote servers
- VitalQIP web client interface
- Web server

□

Single Enterprise and Remote Server Scenario

Figure 13 shows a scenario where VitalQIP enterprise server, distributed services, and client are installed on one server. The VitalQIP remote server is installed on a separate server.

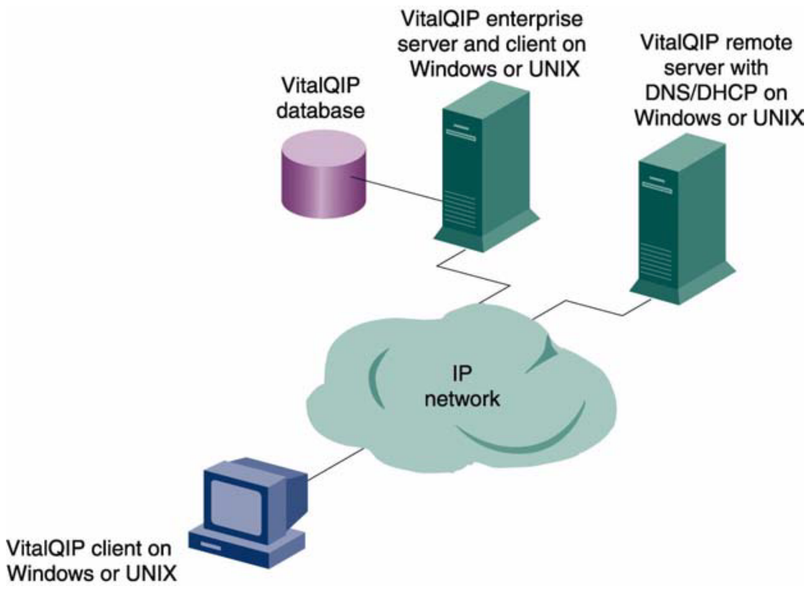

VitalQIP enterprise server and client on Windows or UNIX
VitalQIP remote server with DNS/DHCP on Windows or UNIX
VitalQIP database
IP network
VitalQIP client on Windows or UNIX

Figure 13 Enterprise server, client, and remote services on different servers

The product components required are as follows:

- Sybase or Oracle database system
- VitalQIP server components involving the client, all required services, and the Command Line Interface
- Distributed services
- VitalQIP remote servers
- VitalQIP web client interface
- Web server

□

Single Enterprise, Remote, and Distributed Server Scenario

Figure 14 shows a scenario where a VitalQIP enterprise server and client are installed on one server. The VitalQIP remote server is installed on another server, and distributed services are on another server.

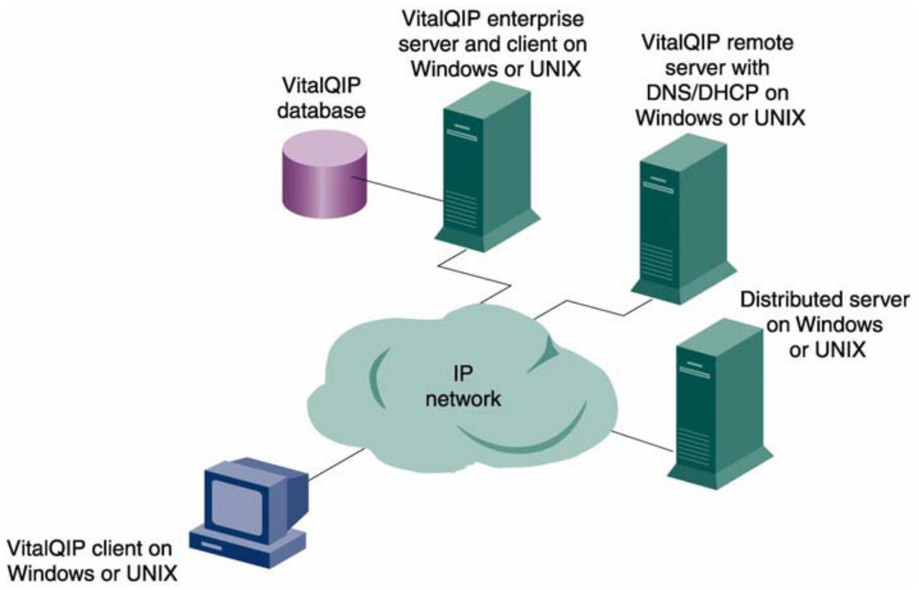

VitalQIP enterprise server and client on Windows or UNIX
VitalQIP remote server with DNS/DHCP on Windows or UNIX
VitalQIP database
IP network
Distributed server on Windows or UNIX
VitalQIP client on Windows or UNIX

Figure 14 Enterprise server, client, distributed services, and remote services on different servers

The product components required are as follows:

- Sybase or Oracle database system
- VitalQIP server components involving the client, all required services, and the Command Line Interface
- Distributed services
- VitalQIP remote servers
- VitalQIP web client interface
- Web server

□

Single Enterprise Server, Multiple Remote Servers, and Multiple Clients Scenario

Figure 15 shows an installation where VitalQIP enterprise server and distributed services are installed on one server. The VitalQIP remote services and clients are installed on different servers.

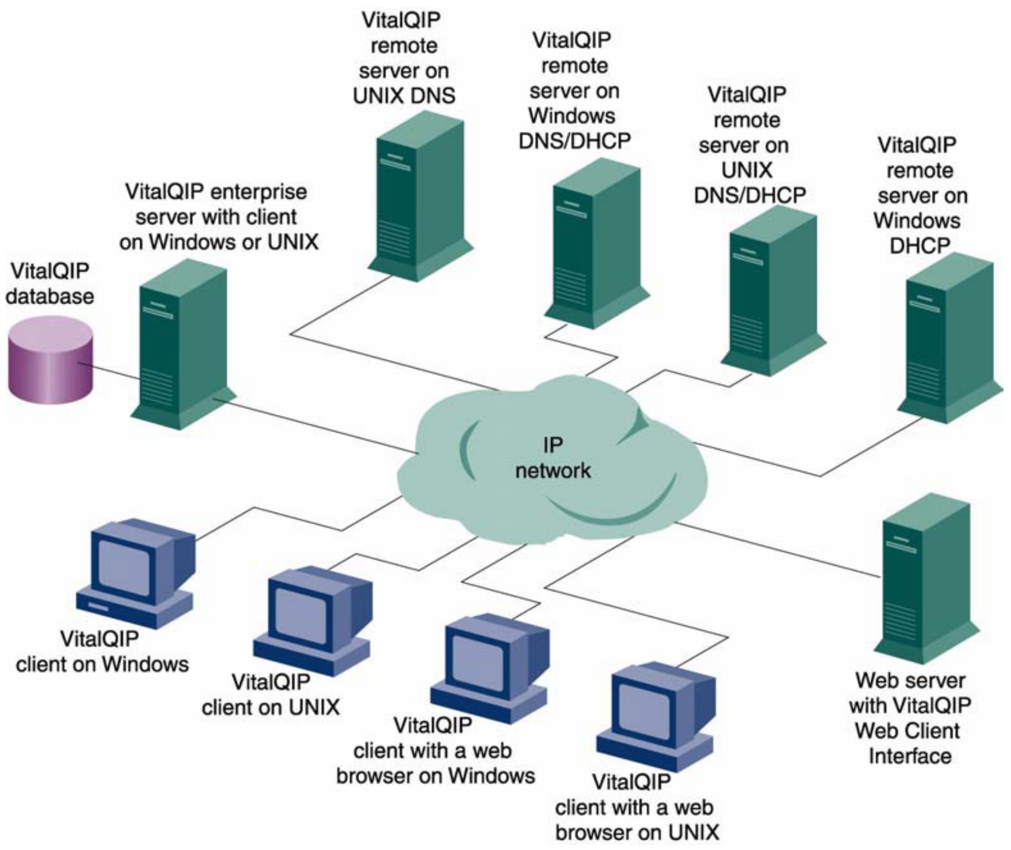

VitalQIP remote server on UNIX DNS
VitalQIP remote server on Windows DNS/DHCP
VitalQIP remote server on UNIX DNS/DHCP
VitalQIP remote server on Windows DHCP
VitalQIP enterprise server with client on Windows or UNIX
VitalQIP database
IP network
VitalQIP client on Windows
VitalQIP client on UNIX
VitalQIP client with a web browser on Windows
VitalQIP client with a web browser on UNIX
Web server with VitalQIP Web Client Interface

Figure 15 Single VitalQIP enterprise server with multiple remote servers and clients

The product components required are as follows:

- Sybase or Oracle database system
- VitalQIP server components involving the client, all required services, and the Command Line Interface
- Distributed services
- VitalQIP remote servers
- VitalQIP web client interface

- Web server
- VitalQIP client for Windows or UNIX

□

Single Enterprise Server, Multiple Remote Servers, Multiple Distributed Servers, and Clients Scenario

Figure 16 shows an installation where VitalQIP enterprise server is installed on one server. The VitalQIP remote services, clients, and distributed services are installed on different servers.

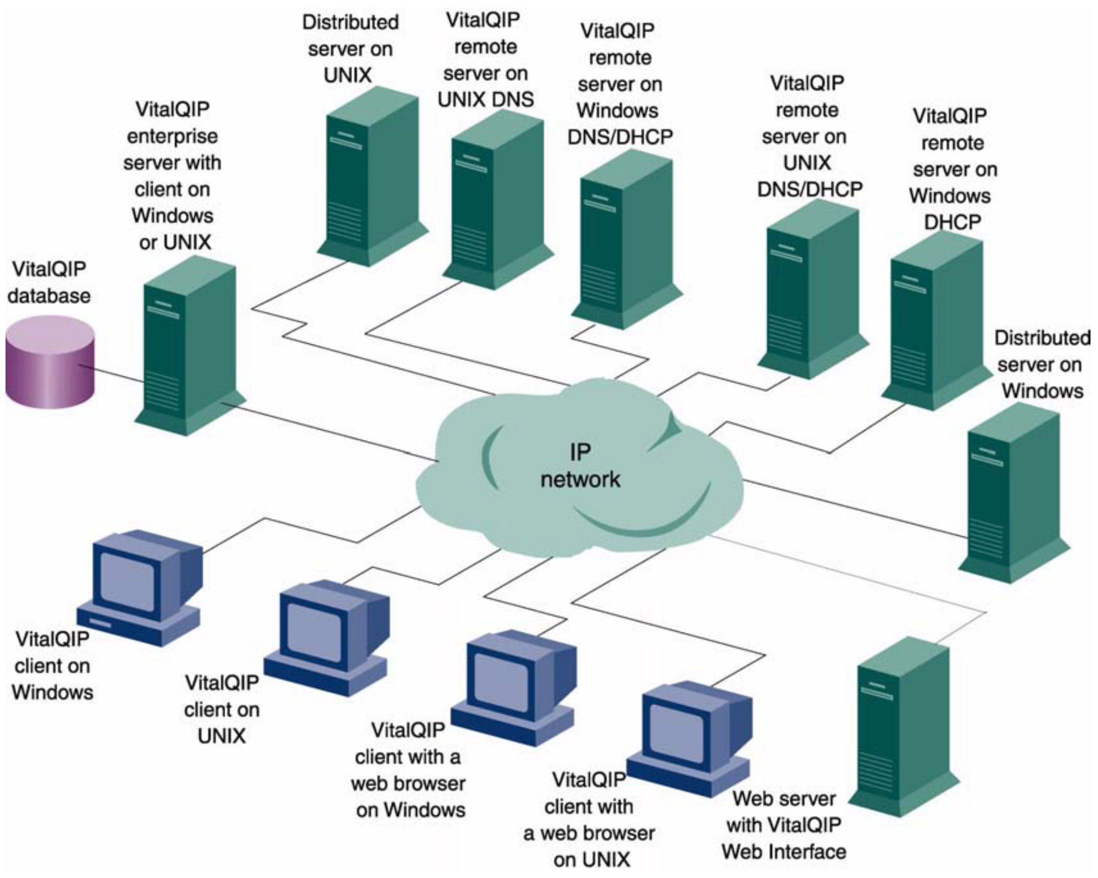

Distributed server on UNIX
VitalQIP remote server on UNIX DNS
VitalQIP remote server on Windows DNS/DHCP
VitalQIP remote server on UNIX DNS/DHCP
VitalQIP remote server on Windows DHCP
VitalQIP enterprise server with client on Windows or UNIX
VitalQIP database
Distributed server on Windows
IP network
VitalQIP client on Windows
VitalQIP client on UNIX
VitalQIP client with a web browser on Windows
VitalQIP client with a web browser on UNIX
Web server with VitalQIP Web Interface

Figure 16 Single VitalQIP enterprise server with multiple remote servers, clients, and distributed servers

The product components required are as follows:

- Sybase or Oracle system database system
- VitalQIP server components involving the client, all required services, and the Command Line Interface
- Distributed services
- VitalQIP remote servers

- VitalQIP web client interface
- Web server
- VitalQIP client for Windows or UNIX

□

Part II: Advanced management and configurations

Overview

Purpose

In Part II, you will find information on advanced DNS and DHCP configurations, managing VitalQIP services, setting up database failover, and VitalQIP database administration for Sybase. You can use Part II to set up your DNS and DHPC servers, manage VitalQIP services (such as starting and stopping services and managing policies), and manage your VitalQIP database (back up, restore, and so on).

Contents

Part II contains the following chapters:

☐

3 Manage VitalQIP services

Overview

Purpose

This chapter contains information about how to run VitalQIP services/daemons, and manage VitalQIP service policies. Information about using the VitalQIP Service Controller, startup scripts, and user-defined naming policy are also included in this chapter.

Contents

This information presents the following topics.

qipd - VitalQIP Schedule Service daemon	75
qip-qipupdated - VitalQIP QIP Update Service daemon	77
qip-msgd - VitalQIP Message Service daemon	79
qip-logind - VitalQIP Login Service daemon	81
qip-dnsupdated - VitalQIP DNS Update Service daemon	83
qip-netd - VitalQIP Active Lease Service daemon	85
qip-rmtd - VitalQIP Remote Service daemon	86
qip-rmisched - VitalQIP RMI Scheduler Service daemon	87
qapi - VitalQIP RMI QAPI Service daemon	88
qip-ssltd - VitalQIP SSL Tunnel Service daemon	89
named - Lucent DNS Service daemon	90
dhcpd - Lucent DHCP Service daemon	91

□

VitalQIP services on Windows

VitalQIP services can be started on all supported Windows platforms in two ways: through the VitalQIP Service Controller or through the Windows Control Panel (accessed via **Start|Settings|Control Panel|Services**). Before you begin running services, ensure you have completed the items in “Services checklist for Windows”, on page 55.

Important! VitalQIP services contain policies that affect the behavior of the services. For information on the policy files for VitalQIP services, see “Manage the VitalQIP policy file”, on page 95.

□

Services checklist for Windows

Ensure you have completed the following before you start VitalQIP services:

- The following VitalQIP services must be started on the VitalQIP server in the following order before users can access the VitalQIP software:
  1. SQL server
  2. VitalQIP Message Service
  3. VitalQIP SSL Tunnel Service
  4. VitalQIP Login Service
  5. VitalQIP Schedule Service
- Table 14 displays a checklist of all services to ensure that you are starting the correct VitalQIP services on your network. Table 14 also indicates where those services should be installed. If more than one type of server is listed, the service can be installed on one or many types of servers, depending on the service. The * indicates the service is not supplied by Lucent. The following abbreviations are used:
  – E/S - enterprise server
  – D/S - distributed server
  – R/S - remote server

  Important! If you are running DNS or DHCP services on the same system as your enterprise server, all services referred to as running on a remote server in the following table also run on the enterprise server.

Table 14 Checklist for services running on Windows

Type of service	Services	Where?
LUCENT DNS	Lucent DNS Service VitalQIP DNS Update Service VitalQIP Login Service VitalQIP Message Service VitalQIP SSL Tunnel Service VitalQIP Schedule Service VitalQIP Remote Service VitalQIP RMI Scheduler Service VitalQIP QIP Update Service	R/S E/S, D/S E/S, D/S E/S, R/S, D/S E/S, R/S, D/S R/S E/S, D/S E/S, D/S
LUCENT DHCP	Lucent DHCP Service VitalQIP Active Lease Service VitalQIP DNS Update Service VitalQIP Login Service VitalQIP Message Service VitalQIP SSL Tunnel Service VitalQIP Schedule Service VitalQIP Remote Service VitalQIP RMI Schedule Service VitalQIP QIP Update Service	R/S R/S E/S, D/S E/S, D/S E/S, R/S, D/S E/S, R/S, D/S R/S E/S, D/S E/S, D/S
Microsoft 2000 DHCP	MS 2000 DHCP Service* VitalQIP Active Lease Service VitalQIP DNS Update Service VitalQIP Login Service VitalQIP Message Service VitalQIP SSL Tunnel Service VitalQIP MS DHCP Monitor Service VitalQIP Schedule Service VitalQIP Remote Service VitalQIP RMI Scheduler Service VitalQIP QIP Update Service	R/S R/S E/S, D/S E/S, D/S E/S, R/S, D/S E/S, R/S, D/S E/S R/S E/S, D/S E/S, D/S
Microsoft 2000 DNS	MS 2000 DNS Service* MS DNS Update Service VitalQIP DNS Update Service VitalQIP Login Service VitalQIP Message Service VitalQIP SSL Tunnel Service VitalQIP Schedule Service VitalQIP Remote Service VitalQIP RMI Scheduler Service	R/S R/S E/S, D/S E/S, D/S E/S, R/S, D/S E/S, R/S, D/S E/S R/S E/S, D/S

Type of service	Services	Where?
Microsoft 2003 DHCP	MS 2003 DHCP Service*	R/S
	VitalQIP Active Lease Service	R/S
	VitalQIP DNS Update Service	E/S, D/S
	VitalQIP Login Service	E/S, D/S
	VitalQIP Message Service	E/S, R/S, D/S
	VitalQIP SSL Tunnel Service	E/S, R/S, D/S
	VitalQIP MS DHCP Monitor Service	R/S
	VitalQIP Schedule Service	E/S
	VitalQIP Remote Service	R/S
	VitalQIP RMI Scheduler Service	E/S, D/S
	VitalQIP QIP Update Service	E/S, D/S
Microsoft 2003 DNS	MS 2003 DNS Service*	R/S
	MS DNS Update Service	R/S
	VitalQIP DNS Update Service	E/S, D/S
	VitalQIP Login Service	E/S, D/S
	VitalQIP Message Service	E/S, R/S, D/S
	VitalQIP SSL Tunnel Service	E/S, R/S, D/S
	VitalQIP Schedule Service	E/S
	VitalQIP Remote Service	R/S
	VitalQIP RMI Scheduler Service	E/S, D/S

VitalQIP Service Controller

You can run VitalQIP services on Windows with the VitalQIP Service Controller (**Start|Programs|VitalQIP|Service Controller**). The Service Controller allows you to start and stop VitalQIP services.

The VitalQIP Service Controller displays messages about the events that occur related to VitalQIP services. It also allows you to save the VitalQIP service log as a text file.

The VitalQIP Service Controller can be configured to start, stop, and view events for *any* Windows services, in addition to VitalQIP services. For example, the Apache and SQL server can be added to the VitalQIP Service Controller to monitor their operability.

The list of services accessible through the Service Controller is referred to as the “Managed Services”. The list of Managed Services is stored in the Windows Registry so that the service can be retrieved the next time you run the VitalQIP Service Controller. If the Managed Service list is not found in the Registry, a default list of VitalQIP services is provided.

Additionally, when selecting the Lucent DNS Service, the **Options** button is enabled.

Run services with the VitalQIP Service Controller

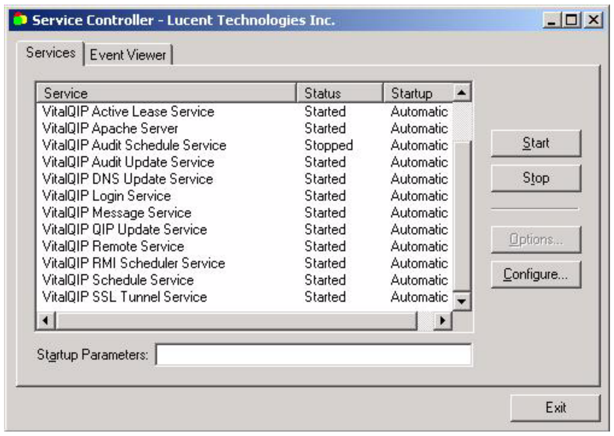
When to use

This section describes how to use the VitalQIP Service Controller to start and stop a service on Windows platforms.

Procedure

To start or stop a service, follow these steps:

1 Access **Start|Programs|VitalQIP|Service Controller**. The Service Controller window opens.



Service	Status	Startup
VitalQIP Active Lease Service	Started	Automatic
VitalQIP Apache Server	Started	Automatic
VitalQIP Audit Schedule Service	Stopped	Automatic
VitalQIP Audit Update Service	Started	Automatic
VitalQIP DNS Update Service	Started	Automatic
VitalQIP Login Service	Started	Automatic
VitalQIP Message Service	Started	Automatic
VitalQIP QIP Update Service	Started	Automatic
VitalQIP Remote Service	Started	Automatic
VitalQIP RMI Scheduler Service	Started	Automatic
VitalQIP Schedule Service	Started	Automatic
VitalQIP SSL Tunnel Service	Started	Automatic

2 To start any service, select the service in the Service list and click **Start**.

OR

To stop any service, select the service in the Service list and click **Stop**.

The Status column changes from “Stopped “ to “Started” or vice versa. Hold down **Ctrl** to highlight and select multiple services to start or stop.

Optionally, **Startup Parameter**s can be specified when starting a single service. (The VitalQIP services do not require any startup parameters.)

The status of the services is updated every 10 seconds. If the service does not start, click the **Event Viewer** tab to determine the cause.

Configure the VitalQIP Service Controller

When to use

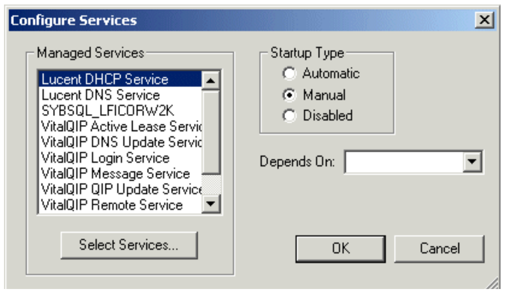
The VitalQIP Service Controller can be configured to start services automatically when a server starts or to be started manually by a user. Services can be disabled or made dependent upon other services.

Procedure

To configure the VitalQIP Service Controller, follow these steps:

1 Access **Start|Programs|VitalQIP|Service Controller**. The Service Controller window opens.

2 In the Service Controller window, click **Configure**. The Configure Services window opens.

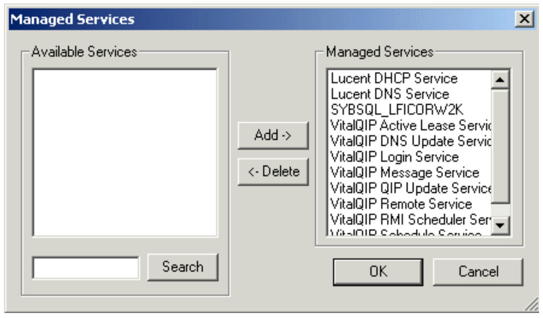


3 In the Configure Service Controller, select a service from the Managed Services list. Table 15 describes the options.

Table 15 Configure services options

Option	Description
Automatic	This option starts the selected service when Windows boots.
Manual	This option forces a user to start the service through the VitalQIP Services Controller.
Disabled	This option disables a service from functioning.
Depends on	The start of a service can be set to be contingent on the status of another service. The dependent service ***must*** be in the Managed Services list. For example, you can select the Lucent RMI Scheduler Service and make it dependent on the Sybase SQL Server. When you start the VitalQIP RMI Scheduler Service, it starts the Sybase SQL server before starting the VitalQIP RMI Scheduler Service. Only one service may be dependent upon another service. Otherwise, you create an infinite loop! For example, do ***not*** make Service A dependent on Service B and Service B dependent on Service A.

4 Click **Select Services** to add or remove a service from the VitalQIP Service Controller. The Managed Services window opens.



Lucent DHCP Service
Lucent DNS Service
SYBSQL_LFICORW2K
VitalQIP Active Lease Servic
VitalQIP DNS Update Servic
VitalQIP Login Service
VitalQIP Message Service
VitalQIP QIP Update Service
VitalQIP Remote Service
VitalQIP RMI Scheduler Ser

5 In the Managed Services window, the **Available Services** list displays ***all*** services defined for Windows. A search string and **Search** button are available to search for specific services. To search for a services, type a few characters in the service's name, and click **Search**. Any service containing the typed characters is returned. If no characters are typed in the field, a list of all Windows services is returned.

6 To add a service to the VitalQIP Service Controller, select a service from the **Available Services** list and click **Add**. The service is added to the **Managed Services** list.

7 To remove a service from the VitalQIP Service Controller, select a service from the **Managed Services** list and click **Delete**. The service is removed from the **Managed Service** list.

8 Click **OK** to return to the Configure Services window, where you can customize the newly added **Managed Services**.

END OF STEPS

Lucent DNS Service options

When to use

The Lucent DNS Service provides an interface for routine maintenance functions of DNS.

Procedure

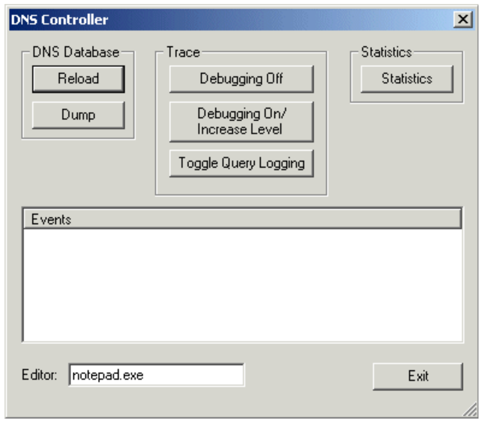
To use the Lucent DNS Service options, follow these steps:

1 Access **Start|Programs|VitalQIP|Service Controller**. The Service Controller window opens.

2 In the Service Controller window, click the **Services** tab.

3 In the **Services** tab, select the “Lucent DNS Service”.

4 Click **Options…** and the DNS Controller window opens.



Editor: notepad.exe

5 In the DNS Controller window, select the function you want to perform by clicking the appropriate button. Table 16 describes the buttons.

Table 16 DNS Controller buttons

Button	Description
DNS Database	To reload the DNS database, click **Reload** under DNS Database. The *named.conf* file and zone files (for example, reverse zone and forward zone files) are reloaded into the DNS database. To dump the DNS database, click **Dump** under DNS Database. This dumps the DNS server's status and information about what the server has and does not have authority over. The database image is saved to the *namedump.db* file in *%SYSTEMROOT%\system32*.
Trace	Trace is used to provide debug information from the DNS service. To turn trace off, click **Debugging Off**. To turn debugging on or to increase the debug level, click **Debugging On/Increase**. The debug level defaults to 0. There are 11 debugging levels. The first level gives you basic startup information. For more information on each level and what it gives you, see "The debug policy", on page 97, or see *DNS and BIND, Fourth Edition* by Paul Albitz & Cricket Liu. The debug information is stored in the *named.run* file, located wherever the DNS Zone files are located. You can also trace the queries that are made to your DNS server by clicking **Toggle Query Logging**. Clicking this button once turns query logging on, clicking it again turns query logging off. The queries are sent to the Windows Event Log.
Statistics	To obtain a DNS statistics file, click **Statistics**. For more information on the statistics file, reference *DNS and BIND, Fourth Edition*, by Paul Albitz & Cricket Liu. The statistics file is stored in the *named.sts* file and located in the *winnt\system32* directory.
Specifying an Editor	To specify the editor you wish to have files dumped to, type its name in the **Editor** field.

□

Diagnose problems with the Event Viewer tab

When to use

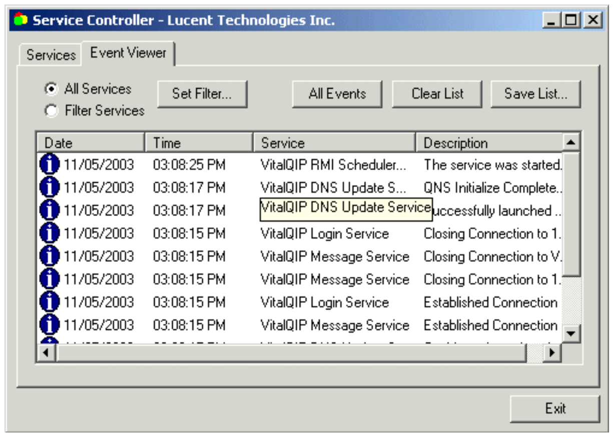
The **Event Viewer** tab is useful for diagnosing a potential problem. It is the first thing that should be reviewed to determine the problem.

Procedure

To use the **Event Viewer** tab, follow these steps:

1 Access **Start|Programs|VitalQIP|Service Controller**. The Service Controller window opens.

2 In the Service Controller window, click the **Event Viewer** tab. The Event Viewer appears. By default, the Event Viewer displays only events that have occurred since VitalQIP was started.



Date	Time	Service	Description
11/05/2003	03:08:25 PM	VitalQIP RMI Scheduler...	The service was started.
11/05/2003	03:08:17 PM	VitalQIP DNS Update S...	QNS Initialize Complete..
11/05/2003	03:08:17 PM	VitalQIP DNS Update Service	uccessfully launched ..
11/05/2003	03:08:15 PM	VitalQIP Login Service	Closing Connection to 1.
11/05/2003	03:08:15 PM	VitalQIP Message Service	Closing Connection to V.
11/05/2003	03:08:15 PM	VitalQIP Message Service	Closing Connection to 1.
11/05/2003	03:08:15 PM	VitalQIP Login Service	Established Connection
11/05/2003	03:08:15 PM	VitalQIP Message Service	Established Connection

3 Select the function you want to perform by clicking the appropriate button. Table 17 describes the options and buttons.

Table 17 Event Viewer tab options and buttons

Option/Button	Description
All Services	Lists events that have occurred for any of the Managed Services.
Filter Services	Lists events that belong to the Filtered Services list. See the next section, “Using Filtered Services”.
Set Filter…	Manages the list of services that appear in the Event Viewer.

Option/Button	**Description**
All Events	Scans the entire event log for all events in the system application event log. Clicking this button also performs an update.
Clear List	Clears the list of events. All the existing messages are cleared from the Event Viewer, but the events are kept in the event log.
Save List…	Saves the list to a text file.

□

Use Filtered Services

When to use

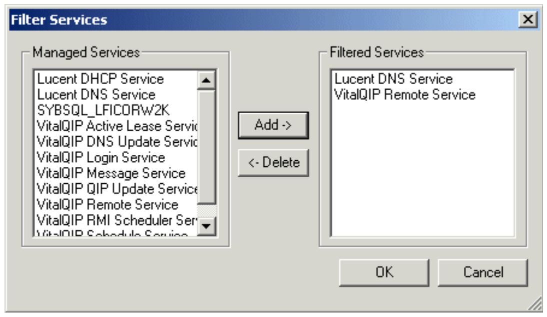
Filtered Services allows you to limit the messages displayed in the **Event Viewer** tab. You can display messages for one service or several services.

Procedure

To use Filtered Services, follow these steps:

1 In the **Event Viewer** tab, select **Filter Services**.

2 Click **Set Filter…**, and the Filter Services window opens.



Managed Services:
Lucent DHCP Service
Lucent DNS Service
SYBSQL_LFICORW2K
VitalQIP Active Lease Servic
VitalQIP DNS Update Servic
VitalQIP Login Service
VitalQIP Message Service
VitalQIP QIP Update Service
VitalQIP Remote Service
VitalQIP RMI Scheduler Ser

Filtered Services:
Lucent DNS Service
VitalQIP Remote Service

3 To filter services, select them from the list and click **Add->**. To remove service from the **Filter Services** list, select the **Managed Service** from the Filtered Service and click **Delete**. The list of Filtered Services is not saved if **Cancel** is clicked, and the **Event Viewer** tab defaults to display all Managed Services.

4 Click **OK** to return to the **Event Viewer** tab.

END OF STEPS

VitalQIP services on UNIX

The VitalQIP services are comprised of daemons/services that run on the VitalQIP enterprise, remote, and distributed servers, such as the Lucent DHCP Service and Lucent DNS Service. They are installed during the installation process, based on the VitalQIP components you selected.

Before you begin running services, ensure you have completed the items in the next section, “Services checklist for UNIX”.

Each daemon is described in more detail later in this section. The “Synopsis” shows the command you use to execute the daemon.

☐

Services checklist for UNIX

Ensure you have completed the following before starting VitalQIP services:

- The environment variables need to be set each time you open a window. You cannot run VitalQIP daemons/services or the application unless all the VitalQIP-related environment variables are set.

  When multiple VitalQIP components are installed on the same machine (for example, enterprise server, web client interface, remote server or client interface), a backup of the *shrc* (Bourne shell) or the *cshrc* (Cshell) file is created if the file already exists. The file is called *shrc* (or *cshrc).20050510.1130* where the numbers correspond to the date and time that the backup is made. The backup file enables you to preserve any customization you might have made to the file, so that you can modify the new environment variable file after installation.

  Important! The file is a single file and not broken out per component.

  A script is available to set the environment variables. Follow these steps to set your environment variables:

  a. Change to your *$QIPHOME/etc* directory.

  b. Run the following script (# at the beginning assumes you are root):

`# . ./shrc` OR `# source cshrc` (for C shell)

  c. If you need to add additional variables to this script, set the variables using the following commands depending on your shell script uses:

```
ksh: export MYVAR="My Value"
csh: setenv MYVAR "My Value”
sh:  MYVAR="My Value"; export MYVAR
```

  d. Ensure your environment variables are set. The values of the variables depend on whether you are using a Sybase or Oracle database. See the *VitalQIP Installation Guide* for more information.

- The following VitalQIP services must be started on the VitalQIP server before users can access VitalQIP:
  - SQL server
  - Message Service
  - VitalQIP SSL Tunnel Service
  - VitalQIP Login Service
  - VitalQIP Schedule Service
- Table 18 displays a checklist of all services to ensure that you are starting the correct VitalQIP services on your network. The table also indicates where those services should be installed. If more than one type of server is listed, the service can be installed on one or many types of servers, depending on the service. The * indicates the service is not supplied by Lucent.

  Important! If you are running DNS or DHCP services on the same system as your enterprise server, all services referred to as running on a remote server in Table 18 also run on your enterprise server.

  The following abbreviations are used:
  - E/S - enterprise server
  - D/S - distributed server
  - R/S - remote server

Table 18 Checklist for services/daemons on UNIX

Type of service	Services/daemons	Location
LUCENT DNS	Lucent DNS Service (**named**)	R/S
	VitalQIP DNS Update Service (**qip-dnsupdated**)	E/S, D/S
	VitalQIP Login Service (**qip-logind**)	E/S
	VitalQIP Message Service (**qip-msgd**)	E/S, R/S, D/S
	VitalQIP SSL Tunnel Service (**qip-ssltd**)	E/S, R/S, D/S
	VitalQIP Schedule Service (**qipd**)	E/S
	VitalQIP Remote Service (**qip-rmtd**)	R/S
	VitalQIP RMI Scheduler Service (**qip-rmisched**)	E/S, D/S
	VitalQIP QIP Update Service (**qip-qipupdated**)	E/S, D/S

Type of service	Services/daemons	Location
LUCENT DHCP	Lucent DHCP Service (**dhcpd**) VitalQIP Active Lease Service (**qip-netd**) VitalQIP DNS Update Service (**qip-dnsupdated**) VitalQIP Login Service (**qip-logind**) VitalQIP Message Service (**qip-msgd**) VitalQIP SSL Tunnel Service (**qip-ssltd**) VitalQIP Schedule Service (**qipd**) VitalQIP Remote Service (**qip-rmtd**) VitalQIP RMI Schedule Service (**qip-rmisched**) VitalQIP QIP Update Service (**qip-qipupdated**)	R/S R/S E/S, D/S E/S E/S, R/S, D/S E/S, R/S, D/S E/S R/S E/S, D/S E/S, D/S
IBM DHCP	IBM DHCP Service (**dhcpsd**)* VitalQIP Active Lease Service (**qip-netd**) VitalQIP IBM DHCP Monitor Service (**qip-ibmdhcpd**) VitalQIP Login Service (**qip-logind**) VitalQIP Message Service (**qip-msgd**) VitalQIP SSL Tunnel Service (**qip-ssltd**) VitalQIP Schedule Service (**qipd**) VitalQIP Remote Service (**qip-rmtd**) VitalQIP RMI Scheduler Service (**qip-rmisched**) VitalQIP QIP Update Service (**qip-qipupdated**)	R/S R/S R/S E/S E/S, R/S, D/S E/S, R/S, D/S E/S R/S E/S, D/S E/S, D/S

VitalQIP startup scripts

VitalQIP comes with four scripts **qip-es-startup**, **qip-rs-startup**, **qip-ds-startup**, and **qip-cs-startup** that start several daemons. These scripts start daemons automatically. They are located in the *$QIPHOME/etc* directory.

The **qip-es-startup** script is used to start services on the enterprise server: The script starts the following daemons:

- qip-logind
- qapi
- qip-rmisched
- qipd
- qip-dnsupdated
- qip-qipupdated
- qip-msgd
- qip-ssltd

The **qip-rs-startup** script is used to start services on the remote server. The script starts the following daemons:

- qip-rmtd
- qip-dnsupdated
- qip-msgd
- qip-ssltd
- qip-netd
- dhcpd
- named

The **qip-ds-startup** is used to start distributed services. The script starts the following daemons:

- qip-logind
- qapi
- qip-rmisched
- qipd
- qip-dnsupdated
- qip-qipupdated
- qip-msgd
- qip-ssltd

The **qip-cs-startup** is used to start client services on the client for secure communication. The script starts the following daemons:

- qip-msgd
- qip-ssltd

□

Start IBM services

For a DNS server, the necessary daemons start automatically at system boot time if a file called *\/etc/named.conf* (for BIND 8.x or 9.x) exists. Otherwise, you can use the **qip-rs-startup** script to start services on the remote server. The script starts the following daemons:

- qip-rmtd
- qip-dnsupdated
- qip-msgd
- dhcpsd
- named

☐

Start DNS services manually

To start DNS Services (BIND 8.x or 9.x), run:

- For AIX, `startsrc -s named`
- For HP-UX, `/usr/sbin/named`
- For Linux, `/usr/sbin/named`
- For Solaris, `/usr/sbin/in.named`

Important! A symbolic link must be created from */etc/named.conf* that points to *named.conf* in the directory where DNS files are generated. Alternatively, the **-c** option must be used to specify the location of the startup file. See "named - Lucent DNS Service daemon", on page 90 for more information on the **-c** option.

Important! A symbolic link must also be created from */etc/rndc.conf* that points to *rndc.conf* in the directory where the DNS files are generated.

□

Start and stop daemons

Daemons (services) can be started or stopped on an individual basis by executing the daemon and its parameters from a command line interface. Parameters entered from a command line override policies specified in the *qip.pcy* file. For information on policy files, see "Manage the VitalQIP policy file", on page 95".

Daemons can be stopped by finding the daemon's process ID and using the **kill** <*process_ID*> command.

□

qipd - VitalQIP Schedule Service daemon

The VitalQIP Schedule Service daemon, **qipd**, must be started before you can access VitalQIP.

Synopsis

```
$QIPHOME/usr/bin/qipd [-c] [-d debug_level] [-f]
 [-g Login_Service's_IP_address] [-h] [-l policy_filename] [-s server]
 [-u username] [-p password] [-v] [-z] [checktime [check_interval]]
```

Parameters

The following parameters can be used with **qipd**:

`-c`	Do not close file descriptors from 1 to 63 on startup. Normally, file descriptors from 1 to 63 are closed when the daemon starts up. This option leaves the file descriptors opened for standard in, standard out, and standard error.
`-d` *debug_level*	Specifies the level of debugging to be performed. See "The debug policy", on page 97 for information on VitalQIP Services debugging.
`-f`	If specified, a child process is not forked when the server stars. Normally, the daemon is forked off as another process when you run the process. This parameter causes the daemon to run in the foreground.
`-g` *Login_Service's_IP_address*	Specifies the Login Service's IP address.
`-h`	Display help.
`-l` *policy_ filename*	Specifies an alternate policy file name.
`-s` *server*	Specifies the database server to connect.
`-u` *username*	Specifies the name of the user to log into the database. If you do not pass a username, the username in the *qip.pcy* file is used.
`-p` *password*	Specifies the password of the user. If you do not pass a password, the password in the *qip.pcy* file is used.

`-v`	This field displays version information. If this option is used, the daemon is not started. Instead, version information is displayed on the screen.
`-z`	Opens descriptors 0, 1, 2, standard in, standard out, and standard error as */dev/null*. This parameter eliminates standard I/O noise generated by indeterminate libraries. Messages that are generated by libraries linked with the daemons, are written to the standard in, standard out, and standard error. Such messages appear on the screen of the process that started the daemons. This option suppresses those messages from being displayed.
`checktime`	Specifies the time at which **qip-check** and **qip-unlock** are run. The default value is 00:00. For additional information about **qip-check** and **qip-unlock**, see the *VitalQIP Command Line Interface User's Guide*.
`check_interval`	Specifies the interval (in seconds) at which scheduled events are performed. This value specifies the process interval, but not necessarily the sleep time between each event. The default value is 60 seconds.

Something to keep in mind

– If you use the **-u** and **-p** parameters, the username and password are visible in the process list, that is they are displayed if you use **ps -ef|grep qipd**.

☐

qip-qipupdated - VitalQIP QIP Update Service daemon

The VitalQIP QIP Update Service daemon, **qip-qipupdated**, updates and performs routine maintenance for the VitalQIP database. The daemon must be running if you have installed Lucent DHCP, IBM AIX DHCP, or Microsoft DHCP on the enterprise server or on a remote server.

Synopsis

```
$QIPHOME/usr/bin/qip-qipupdated [-c] [-d debug_level] [-f]
 [-l policy_filename] [-m] [-S] [-s server] [-u username]
 [-p password] [-v] [-h] [-z]
```

Parameters

The following parameters can be used with **qip-qipupdated**:

`-c`	Do not close file descriptors from 0 to 63 on startup. Normally, file descriptors from 0 to 63 are closed when the daemon starts up. This option leaves the file descriptors opened for standard in, standard out, and standard error.
`-d` *`debug_level`*	Specifies the level of debugging to be performed. See “The debug policy”, on page 97 for information on VitalQIP Services debugging.
`-f`	If specified, a child process is not forked when the server stars. Normally, the daemon is forked off as another process when you run the process. This parameter causes the daemon to run in the foreground.
`-l` *`policy_filename`*	Reads the policy information from policy file.
`-m`	Specifies multiple QIP Update Services (Master Mode). In Master Mode, the QIP Update Service creates a new process for each new connection.
`-S`	Runs the daemon as an **inetd** service.
`-s` *`server`*	Specifies the database server to connect.
`-u` *`username`*	Specifies the name of the user to log into the database. If you do not pass a username, the username in the *qip.pcy* file is used.
`-p` *`password`*	Specifies the password of the username. If you do not pass a password, the password in the *qip.pcy* file is used.

-v	Displays version information for **qip-qipupdated** only. If this option is used, the daemon is not started. Instead, version information is displayed on the screen.
-h	Displays help.
-z	Opens descriptors 0, 1, 2, standard in, standard out, and standard error as ***/dev/null***. This parameter eliminates standard I/O noise generated by indeterminate libraries. Messages that are generated by libraries linked with the daemons could be written to the standard in, standard out, and standard error. Such messages appear on the screen of the process that started the daemons. This parameter suppresses those messages from being displayed.

Something to keep in mind

– If you use the **-u** and **-p** parameters, the username and password is displayed if you use **ps -ef|grep qipd**.

□

qip-msgd - VitalQIP Message Service daemon

The VitalQIP Message Service daemon, **qip-msgd**, provides communication between other VitalQIP services. The daemon must be running on enterprise, remote, and distributed servers (if they are utilized in your network).

Synopsis

```
$QIPHOME/usr/bin/qip-msgd [-c] [-d debug_level] [-f]
 [-l policy_filename] [-S] [-v] [-z] [-h]
```

Parameters

The following parameters can be used with **qip-msgd**:

`-c`	Do not close file descriptors from 0 to 63 on startup. Normally, file descriptors from 0 to 63 are closed when the daemon starts up. This option leaves the file descriptors opened for standard in, standard out, and standard error.
`-d` *debug_level*	Specifies the level of debugging to be performed. See “The debug policy”, on page 97 for information on VitalQIP Services debugging.
`-f`	If specified, a child process is not forked when the server stars. Normally, the daemon is forked off as another process when you run the process. This parameter causes the daemon to run in the foreground.
`-l` *policy_filename*	Reads policy information into a temporary policy file. This specification could be used if you want to test different policy options before you apply them.
`-S`	Runs the daemon as an **inetd** service.
`-v`	Displays version information for **qip-msgd** only. If this option is used, the daemon is not started. Instead, version information is displayed on the screen.
`-z`	Opens descriptors 0, 1, 2, standard in, standard out, and standard error as */dev/null*. This parameter eliminates standard I/O noise generated by indeterminate libraries. Messages that are generated by libraries linked with the daemons could be written to the standard in, standard out, and standard error. Such messages appear on the screen of the process that started the daemons. This parameter suppresses those messages from being displayed.

`-h`	Displays help.

□

qip-logind - VitalQIP Login Service daemon

The VitalQIP Login Service daemon, **qip-logind**, provides a mechanism to store database user names and passwords in a centralized location. The daemon must be started in order to use the VitalQIP system.

Synopsis

```
$QIPHOME/usr/bin/qip-logind [-c] [-d debug_level] [-f] [-u username]
 [-p password] [-l policy_filename] [-v] [-S] [-z] [-h]
```

Parameters

The following parameters can be used with **qip-logind**:

`-c`	Do not close file descriptors from 0 to 63 on startup. Normally, file descriptors from 0 to 63 are closed when the daemon starts up. This option leaves the file descriptors opened for standard in, standard out, and standard error.
`-d` *`debug_level`*	Specifies the level of debugging to be performed. See "The debug policy", on page 97 for information on VitalQIP Services debugging.
`-f`	If specified, a child process is not forked when the server stars. Normally, the daemon is forked off as another process when you run the process. This parameter causes the daemon to run in the foreground.
`-l` *`policy_filename`*	Reads policy information into a temporary policy file. This specification could be used if you want to test different policy options before you apply them.
`-S`	Runs the daemon as an **inetd** service.
`-u` *`username`*	Specifies the name of the user to log into the database. If you do not pass a username, the username in the *qip.pcy* file is used.
`-p` *`password`*	Specifies the password of the user. If you do not pass a password, the password in the *qip.pcy* file is used.
`-v`	Displays version information for **qip-logind** only. If this option is used, the daemon is not started. Instead, version information is displayed on the screen.

`-z`	Opens descriptors 0, 1, 2, standard in, standard out, and standard error as */dev/null*. This parameter eliminates standard I/O noise generated by indeterminate libraries. Messages that are generated by libraries linked with the daemons could be written to the standard in, standard out, and standard error. Such messages appear on the screen of the process that started the daemons. This parameter suppresses those messages from being displayed.
`-h`	Displays help.

□

qip-dnsupdated - VitalQIP DNS Update Service daemon

The VitalQIP DNS Update Service, **qip-dnsupdated**, is responsible for keeping DNS in sync with the VitalQIP database.

Synopsis

```
$QIPHOME/usr/bin/qip-dnsupdated [-c] [-d debug_level] [-f]
 [-l policy_filename] [-m] [-v] [-S] [-z] [-h]
```

Parameters

The following parameters can be used with **qip-dnsupdated**:

`-c`	Do not close file descriptors from 0 to 63 on startup. Normally, file descriptors from 0 to 63 are closed when the daemon starts up. This option leaves the file descriptors opened for standard in, standard out, and standard error.
`-d` *`debug_level`*	Specifies the level of debugging to be performed. See “The debug policy”, on page 97 for information on VitalQIP Services debugging.
`-f`	If specified, a child process is not forked when the server stars. Normally, the daemon is forked off as another process when you run the process. This parameter causes the daemon to run in the foreground.
`-l` *`policy_filename`*	Reads policy information into a temporary policy file. This specification could be used if you want to test different policy options before you apply them.
`-m`	Specifies multiple VitalQIP DNS Update Services (Master Mode). In Master Mode, the VitalQIP DNS Update Service creates a new process for each new connection.
`-S`	Runs the daemon as an **inetd** service.
`-v`	Displays version information for **qip-dnsupdated** only. If this option is used, the daemon is not started. Instead, version information is displayed on the screen.

`-z`	Opens descriptors 0, 1, 2, standard in, standard out, and standard error as */dev/null*. This parameter eliminates standard I/O noise generated by indeterminate libraries. Messages that are generated by libraries linked with the daemons could be written to the standard in, standard out, and standard error. Such messages appear on the screen of the process that started the daemons. This parameter suppresses those messages from being displayed.
`-h`	Displays help.

qip-netd - VitalQIP Active Lease Service daemon

The VitalQIP Active Lease daemon, **qip-netd**, must be running in order to use **Network Services|View Active Leases** (from the client) as well as the **qip-active** and **qip-dhcpsync** CLIs. **qip-netd** *must* run on the same server as the Lucent or IBM DHCP server. The daemon is invoked in all cases, regardless of where the enterprise server and the VitalQIP Remote Service are running.

Synopsis

```
$QIPHOME/usr/bin/qip-netd [-c] [-d debug_level] [-v] [-f] [-h]
 [-l policy_filename] [-z]
```

Parameters

The following parameters can be used with **qip-netd**:

`-c`	Do not close file descriptors 0 to 63 on startup. Normally, file descriptors from 0 to 63 are closed when the daemon starts up. This option leaves the file descriptors opened for standard in, standard out, and standard error.
`-d` *debug_level*	Specifies the level of debugging to be performed. See "The debug policy", on page 97 for information on VitalQIP Services debugging.
`-v`	Displays version information. If this option is used, the daemon is not started. Instead, version information is displayed on the screen.
`-f`	If specified, a child process is not forked when the server stars. Normally, the daemon is forked off as another process when you run the process. This parameter causes the daemon to run in the foreground.
`-h`	Displays help.
`-l` *policy_filename*	Specifies the policy filename.
`-z`	Opens descriptors 0, 1, 2, standard in, standard out, and standard error as */dev/null*. This parameter eliminates standard I/O noise generated by indeterminate libraries. Messages that are generated by libraries linked with the daemons could be written to the standard in, standard out, and standard error. Such messages appear on the screen of the process that started the daemons. This parameter suppresses those messages from being displayed.

qip-rmtd - VitalQIP Remote Service daemon

The VitalQIP Remote Service, **qip-rmtd**, starts the VitalQIP remote daemon that performs data transfer from a server running the File Generation Service, including primary and secondary DNS files, the Bootp table, and the DHCP configuration files to other services. The daemon must be running to use all the Network Services functions, with the exception of the **Network Services|View Active Lease**. If your enterprise server is also acting as a remote server, **qip-rmtd** must be installed on it, as well.

Synopsis

```
$QIPHOME/usr/bin/qip-rmtd [-c] [-d debug_level] [-v] [-f] [-h]
 [-l policy_filename] [-z]
```

Parameters

The following parameters can be used with **qip-rmtd**:

`-c`	Do not close file descriptors from 0 to 63 on startup. Normally, file descriptors from 0 to 63 are closed when the daemon starts up. This option leaves the file descriptors opened for standard in, standard out, and standard error.
`-d` *`debug_level`*	Specifies the level of debugging to be performed. See “The debug policy”, on page 97 for information on VitalQIP Services debugging.
`-v`	This field displays version information. If this option is used, the daemon is not started. Instead, version information is displayed on the screen.
`-f`	If specified, a child process is not forked when the server stars. Normally, the daemon is forked off as another process when you run the process . This parameter causes the daemon to run in the foreground.
`-h`	Displays help.
`-l` *`policy_filename`*	Specifies the policy filename.
`-z`	Opens descriptors 0, 1, 2, standard in, standard out, and standard error as */dev/null*. This parameter eliminates standard I/O noise generated by indeterminate libraries. Messages that are generated by libraries linked with the daemons could be written to the standard in, standard out, and standard error. Such messages appear on the screen of the process that started the daemons. This parameter suppresses those messages from being displayed.

qip-rmisched - VitalQIP RMI Scheduler Service daemon

The VitalQIP RMI Scheduler Service daemon, qip-rmisched, controls access to the RMI QAPI Services that generate configuration files.

Synopsis

```
$QIPHOME/usr/bin/qip-rmisched [-c] [-d debug_level] [-f]
 [-l policy_filename] [-v] [-S] [-z] [-h]
```

Parameters

The following parameters can be used with **qip-rmisched**:

`-c`	Do not close file descriptors from 0 to 63 on startup. Normally, file descriptors from 0 to 63 are closed when the daemon starts up. This option leaves the file descriptors opened for standard in, standard out, and standard error.
`-d` *debug_level*	Specifies the level of debugging to be performed. See "The debug policy", on page 97 for information on VitalQIP Services debugging.
`-f`	If specified, a child process is not forked when the server stars. Normally, the daemon is forked off as another process when you run the process . This parameter causes the daemon to run in the foreground.
`-l` *policy_filename*	Reads policy information into a temporary policy file. This specification could be used if you want to test different policy options before you apply them.
`-S`	Runs the daemon as an **inetd** service.
`-v`	Displays version information for **qip-rmisched** only. If this option is used, the daemon is not started. Instead, version information is displayed on the screen.
`-z`	Opens descriptors 0, 1, 2, standard in, standard out, and standard error as */dev/null*. This parameter eliminates standard I/O noise generated by indeterminate libraries. Messages that are generated by libraries linked with the daemons could be written to the standard in, standard out, and standard error. Such messages appear on the screen of the process that started the daemons. This parameter suppresses those messages from being displayed.
`-h`	Displays help.

qapi - VitalQIP RMI QAPI Service daemon

The VitalQIP RMI QAPI Service daemon, **qapi**, is responsible for generating files for the Remote Servers. This daemon is started automatically by the RMI Scheduler Service. There is no need to start this service manually unless you have more than one RMI QAPI Service. The RMI Scheduler Service starts the **qapi** daemon automatically.

Synopsis

```
$QIPHOME/jre/bin/qapi [--classname_name] [--schedulername_name]
[--servicename_name] [--id_number][--instance number]
```

Parameters

The following parameters can be used with **qapi**:

`--schedulername <name>`	Specifies the name of the Scheduler. The default is QAPI_Scheduler.
`--servicename _<name>`	Specifies the name of this Service. The default is QAPI.
`--classname <name>`	Specifies the name of the class. The default is QAPI.
`--id _<number>`	Specifies a unique number to register with the Scheduler Service.
`--instance <number>`	Specifies the instance of this scheduled process. This is useful for debugging.

A few things to keep in mind

– **qapi** is a copy of the Java executable. All Java Virtual Machine (JVM) options are valid for **qapi**.

– **qapi** is started automatically by the RMI Scheduler Service.

qip-ssltd - VitalQIP SSL Tunnel Service daemon

The VitalQIP SSL Tunnel Service, **qip-ssltd**, allows all VitalQIP message traffic to be encrypted and decrypted.

Synopsis

```
$QIPHOME/usr/bin/qip-ssltd [-c] [-d debug_level] [-v] [-f] [-h]
 [-l policy_filename] [-z]
```

Parameters

The following parameters can be used with **qip-ssltd**:

`-c`	Do not close file descriptors from 0 to 63 on startup. Normally, file descriptors from 0 to 63 are closed when the daemon starts up. This option leaves the file descriptors opened for standard in, standard out, and standard error.
`-d` *`debug_level`*	Specifies the level of debugging to be performed. See “The debug policy”, on page 97 for information on VitalQIP Services debugging.
`-v`	This field displays version information. If this option is used, the daemon is not started. Instead, version information is displayed on the screen.
`-f`	If specified, a child process is not forked when the server stars. Normally, the daemon is forked off as another process when you run the process . This parameter causes the daemon to run in the foreground.
`-h`	Displays help.
`-l` *`policy_filename`*	Specifies the policy filename.
`-z`	Opens descriptors 0, 1, 2, standard in, standard out, and standard error as */dev/null*. This parameter eliminates standard I/O noise generated by indeterminate libraries. Messages that are generated by libraries linked with the daemons could be written to the standard in, standard out, and standard error. Such messages appear on the screen of the process that started the daemons. This parameter suppresses those messages from being displayed.

named - Lucent DNS Service daemon

The Lucent DNS Service daemon, **named** or **in.named**, is the Internet domain name service. See RFC 1034 and RFC 1035 for additional information on the Domain Name System.

Without specified arguments, **named** loads the default boot file *named.conf*, reads any initial data, and listens for queries.

> **Important!** The *in.named* daemon is used on Solaris platforms. This program resides by default in */usr/sbin* directory.

Synopsis

```
named [-c conffile] [-d debuglevel] [-f | -g] [-n number_of_cpus]
 [-p port] [-s] [-t chrootdir] [-u username] [-v]
```

Parameters

The following parameters can be used with **named**:

`-c` *`conffile`*	Starts named using the specified conffile.
`-d` *`debuglevel`*	Starts named using the specified debug level.
`-f`	Runs named in the foreground.
`-g`	Runs named in the foreground and sends all output to stderr.
`-n` *`number_of_cpus`*	Tells named how many processors exist on the system.
`-p` *`port`*	Directs named to listen on the specified port -t chrootdir (Unix only) Causes named to chroot to the specified directory.
`-u` *`username`*	(Unix only) Runs named as the specified user.
`-v`	Prints named version information.
`-s`	Prints memory statistics at exit (only available if running in foreground).

□

dhcpd - Lucent DHCP Service daemon

The Lucent DHCP Service daemon, **dhcpd**, must be started to support DHCP Services. Whenever **dhcpd** grants releases or renews a lease, it writes the requests to the process **qip-msgd**. The **qip-msgd** reads these requests and forwards them to VitalQIP, DNS, or both VitalQIP and DNS.

Synopsis

```
$QIPHOME/usr/bin/dhcpd [-f configuration_file] [-d debug_level] [-v]
```

Parameters

The following parameters can be used with **dhcpd**:

`-f` *`configuration_file`*	Specifies the directory path where **dhcpd** reads configuration and active lease information.
`-d` *`debug_level`*	Initiates debugging in the DNS server and specifies the debugging level (an integer between 1 and 12). The output file (called *named.run*) is placed in the directory where **named** (or **in.named**) was started. Debugging can also be initiated using the **kill -USR1** UNIX command. An iteration of the **kill** command increases the debugging level by one.
`-v`	Displays version information. If this option is used, the daemon is not started. Instead, version information is displayed on the screen.

☐

4 VitalQIP policy files

Overview

Purpose

This chapter contains information about how to manage VitalQIP service policies.

Contents

This chapter presents the following topics.

Manage the VitalQIP policy file

During the installation, a policy file called *qip.pcy* is generated. The file contains policies for all installed services. The following services have sections in the *qip.pcy* file in the *QIPHOME* directory:

- VitalQIP QIP Update Service (**qip-qipupdated**)
- VitalQIP DNS Update Service (**qip-dnsupdated**)
- VitalQIP MS DNS Update Service
- VitalQIP Message Service (**qip-msgd**)
- VitalQIP Login Service (**qip-logind**)
- VitalQIP Schedule Service (**qipd**)
- VitalQIP Active Lease Service (**qip-netd**)
- VitalQIP Microsoft DHCP Monitor Service (**MSDHCPMonitorService**)
- VitalQIP RMI QAPI Service (**qip-qapi**)
- VitalQIP Remote Service (**qip-rmtd**)
- VitalQIP RMI Scheduler Service (**qip-rmisched**)
- VitalQIP SSL Tunnel Service (**qip-ssltd**)

Each service is separated with section headers, (for example, [VitalQIP Schedule Service]). Any policy specified before the first section header is passed to every service. See the sample *qip.pcy* file in “Sample qip.pcy file”, on page 155 for examples.

It is important to note that the service policies set in the client (discussed in Chapter 2 of the *VitalQIP User’s Guide*) override any policies you establish in the policy file.

If you opt for individual policy (*.pcy*) files rather than the larger *qip.pcy* file, your services read the individual files. In an upgrade, a *qip.pcy* file is installed but the smaller, individual service files override the *qip.pcy* files.

VitalQIP services policies

Policies for VitalQIP services can be stored in one large file, or they can be stored as separate files - one for each service. If you choose to store them in one large file, the file is called *qip.pcy*. In *qip.pcy*, each set of policy information must be preceded by a section name. For example, if you are using *qip.pcy* and you want to establish policies for the Schedule Service (**qipd**), the policies should be preceded with [VitalQIP Schedule Service].

By default, services look for their *.pcy* files in *QIPHOME*. If the service's *.pcy* file is not found, the *qip.pcy* file is processed. You can move them to a different file, but you must specify an optional environment variable, QIP_POLICYPATH. The value of this environment variable is set up like that of your PATH environment variable.

A policy file can be created or edited by using a text editor. The format of the policy in the policy file is `Policy=Value`. For example, if you were establishing a policy in the Schedule Service to identify the license interval to 1 hour (in seconds), the policy would look like:

```
LicenseInterval=3600
```

The services report configured and unrecognized policies. This information is reported in the debug log file. If a policy is unrecognized (typically the result of misspelling the policy), the specified policy will be displayed as such:

```
The following 2 policies were not recognized:
  Server
  qip-msgd
```

The services also report known policies that have been specified:

```
The following 3 policies were set:
  debug=All FeatureBackup FeatureTimeStamp FeatureRotateLogs
  FeatureFullStackTrace
  MessageServicePort=2222
  ListenPort=2366
```

□

The debug policy

Each VitalQIP service (with the exception of RMI QAPI, which uses Java debug policies only) has a **debug** policy that can be specified in either the Global section, if you wish to apply the same debug policy across all policies, or in a specific policy section of *qip.pcy*. Most clients and CLIs read the policy file in the same way that services do, and adhere to the value specified for the debug policy.

> **Important!** When you wish to diagnose a potential problem, first access the Event Viewer in Windows or *syslog* in UNIX to determine the problem.

The values for the debug policy can be a string composed of:

- Debug level
  - **All** - The maximum level of debugging; all levels.
  - **LevelCritical** - A critical error is one that shuts down the program. Only critical messages are logged.
  - **LevelError -** An error has occurred, but the program should continue. Critical messages are included.
  - **LevelWarning** - The program has encountered an unexpected issue but continues. Errors and critical messages are included with these warnings.
  - **LevelInfo** - These are informational messages about the program events and flow. These messages include critical messages, errors, and warnings.
  - **LevelDebug** - Indicates that all levels should be logged.
  - **None** - No debugging. This is the default.
- List of application/library layers from which to display debug messages
  - **Application** - The program itself (service, client, CLI).
  - **QSICommon** - The low level common library.
  - **QSINet** - The network related library.
  - **QIPDB** - The core VitalQIP database routine library.
  - **QAPI** - The VitalQIP business layer API.
- Additional debug features
  - **FeatureTimestamp** - Each message in the log is prefixed with a date/time stamp in *`DDD, MMM dd hh:mm:ss.ms`* format, for example, `Tue, May 31 13:10:30.456`.
  - **FeatureDeltaTime** - Time is displayed as the difference from the previous message.
  - **FeatureRelativeTime** - Time is displayed as the difference from the first message.
  - **FeatureISO8601** - Each line is prefixed with date and time in a format that conforms to ISO 8601. The date/time stamp format is *`YYYY-MM-DDThh:mm:ss.ms`*, where the capital letter T is used to separate the date and time components, for example, `2005-05-31T13:10:30.456`
  - **FeatureSeverityStamp** - Each line is prefixed with its severity.

- **FeatureBackup** - This creates a copy of a previous debug file to *<debugfile>.bak_1.log* before logging to the current debug file (*.log* is kept to maintain application association). To specify the number of backup debug files you wish to maintain, set the **DebugRotateMaxDepth** policy to a value greater than 1.
- **FeatureModule** - This displays the library/location of the original message.
- **FeatureThreadStamp** - Each line is prefixed with its thread ID.
- **FeatureStackTrace** - This displays function entrance and must be enabled when you submit bugs on VitalQIP.
- **FeatureFullStackTrace** - Displays function exit as well as entrance.
- **FeatureProfile** - Function execution times are displayed (also enables **FeatureStackTrace**).
- **FeatureValues** - Values read from or written to the VitalQIP database are printed to the debug log file.
- **FeatureRotateLogs** - Enables log file rotation (starts new log file).

For example, to enable debugging for the application and QSICommon, for warning messages and above with a timestamp, specify:

```
debug = LevelWarning Application QSICommon FeatureTimestamp
```

To enable maximum debugging, specify:

```
debug = All
```

By default, debug files are located in *$QIPHOME/log*.

Log file rotation

If you enable **FeatureRotateLogs**, you can ensure that log files rotate automatically using two additional debug policies: **DebugRotateFileSize** and **DebugRotateInterval**. Both policies have default values of zero, meaning that no rotation occurs based on size or time. These policies can be used together.

- Use **DebugRotateFileSize** to specify the maximum debug file size. For example:
  `DebugRotateFileSize=500K` - New debug file starts at 500K
  `DebugRotateFileSize=50M` - New debug file starts at 50 MB
- Use **DebugRotateInterval** to specify how often to rotate the debug file. For example:
  `DebugRotateInterval=90m` - New debug file starts every 90 minutes
  `DebugRotateInterval=12h` - New debug file starts every 12 hours
  `DebugRotateInterval=2d` - New debug file starts every 2 days

Debug filenames

Debug filenames may contain format characters that expand to date strings. For example, `DebugFile=qip-msgd.%A.%B.%d.log` would translate into a filename such as:

```
$QIPHOME/log/qip-msgd.Wednesday.April.2.log
```

Your platform may support many additional format strings. Use an Internet search engine to search for `strftime <platform>`.

> **Important!** On any platform except Windows, you can prefix a filename with the '|' character to send debug to a filter program (for example, `DebugFile="|my_debug_filter"`). Filter programs are restarted if they crash, although some debug output may be lost.

Sample debug filename format characters are described in Table 19.

Table 19 Debug filename format characters

Character	Description
%a	Abbreviated weekday name
%A	Full weekday name
%b	Abbreviated month name
%B	Full month name
%d	Day of month as decimal number (01 - 31)
%H	Hour in 24-hour format (00 - 23)
%j	Day of year as decimal number (001 - 366)
%m	Month as decimal number (01 - 12)
%M	Minute as decimal number (00 - 59)
%S	Second as decimal number (00 - 61)
%w	Weekday as decimal number (0 - 6; Sunday is 0)
%W	Week of year as decimal number, with Monday as first day of week (00 - 53)
%y	Year without century, as decimal number (00 - 99)
%Y	Year with century, as decimal number
%Z	Time-zone name or abbreviation; no characters if time zone is unknown
%%	Percent sign

☐

Java debug policies

VitalQIP integrates Java and C++ in several applications. The Java components have a slightly different format for specifying debug levels and features. This may seem to introduce inconsistencies but in reality provides greater flexibility with application debugging. **DebugLevel** can be applied to the following policies that contain Java code:

- VitalQIP Remote Service (page 142)
- RMI QAPI Service (page 146)
- RMI Scheduler Service (page 148)
- SSL Tunnel Service (page 152)

DebugLevel only accepts a debug level, as described in Table 20.

Table 20 DebugLevel policy

DebugLevel	Default: None Allowed: Debug | Info | Warning | Error | Critical | None	This policy sets the Java Debug level. The list describes the debug levels: **Debug** - Indicates that all levels should be logged. **Info** - These are informational messages about the program events and flow, and include critical messages, errors and warnings. **Warning** - The program encountered an unexpected issue but continued. Errors and critical messages are included with these. **Error** - An error has occurred, but the program should continue. Critical messages are included with these. **Critical** - A critical error is one that shuts down the program. Only critical messages are logged. **None** - No debugging. This is the default.

Java debug features are specified separately on different lines. The debug features listed in Table 21 can also be specified in the global section of the policy file.

Table 21 Java debug features

Debug feature	Description
FeatureTimestamp	When set to true, prefixes each line with time (with ms resolution).
FeatureProfile	When set to true, adds the function execution time to the beginning of each line in the debug file.
FeatureISO8601	Each line is prefixed with date and time in a format that conforms to ISO 8601. The date/time stamp format is *`YYYY-MM-DDThh:mm:ss.ms`*, where the capital letter T is used to separate the date and time components, for example, `2005-05-31T13:10:30.456`
FeatureSeverityStamp	When set to true, prefixes each line with its severity.
FeatureBackup	When set to true, backs up the previous debug file.
FeatureThreadStamp	When set to true, prefixes each line with its thread ID.
VerboseJNI	When set to true, displays details about what Java Native Interface (JNI) calls the Java Virtual Machine (JVM) is making.
VerboseClass	When set to true, displays details about class loader operations.
VerboseGC	When set to true, displays details about garbage collection operations.
UserJVMOpt	Used to specify options to the JVM directly.
LibraryPath	Used to specify what directories should be searched for native libraries. For multiple directories, use the separator value that is valid for your platform. On Windows, the separator value is ; On UNIX, the separator value is :

□

Global section

The global section of the policy file is where you set up policies that affect the entire *qip.pcy* file. For example, debug polices established in this section are treated as global and are passed to every service. Do not modify the Global Section header because the installation program uses it to insert specific values, such as the server name, user name and encrypted password.

Table 23 describes the available policies for the global section of the policy file.

Table 22 Global section policies

Policies	Values	Description
Debug	Default: None Allowed: See page 97	This policy sets the global debug level and is passed to every service. See "The debug policy", on page 97 for more information about the Debug policy.
CLIdebug	Default: None Allowed: See page 97	This policy sets the global debug level for CLIs and is passed to every CLI.
GUIdebug	Default: None Allowed: See page 97	This policy sets the global debug level for GUIs and is passed to every GUI.
LoginServer	Default: 127.0.0.1 Allowed: IP address	The IP address of the servers running the Login Service. More than one IP address can be specified to minimize VitalQIP's downtime in the event connectivity is lost with the Login Service. Each IP address must be separated by comma.

qip-msgd
qip-dns
qip-qdhcp
	Default: qip-msgd=2468
	qip-dns=3119
	qip-qdhcp=2490
	Allowed: Any port number
		The VitalQIP install no longer places entries into */etc/services*.
		Although */etc/services* is still scanned, for processes to find port numbers other than defaults, this policy file is used. These values are examined before anything in */etc/services*.
		All portname/number policies must reside in the global section.
		Important! The policies qip-msgd, qip-dns, qip-qdhcp always appear as not recognized in the debug log file because they are used by the qsinet library and not directly by the service.

Policies	**Values**	**Description**
SecureIncoming MessageService Connections	Default: False Allowed: True | False	If set to False, indicates that the Message Service should accept plaintext connections. The Message Service binds to the well known port qip-msgd on all interfaces including the loopback. If set to True, indicates that the Message Service should accept only SSL-enabled connections via the SSL Tunnel service. The SSL Tunnel Service binds to the well known port qip-msgd on all interfaces except the loopback. The Message Service binds to qip-msgd on the loopback. For more information on secure message transport, refer to Chapter 6, “Secure message routes” on page 195.

□

VitalQIP QIP Update Service policies

Behavior of the QIP Update Service (**qip-qipupdated**) is controlled by policies in the *qip-qipupdated.pcy* file or the *qip.pcy* file, located in the *QIPHOME* directory.   The *qip-qipupdated.pcy* file is used only by the QIP Update Service. This file is not created automatically. Since the *qip-qipupdated.pcy* file is not created automatically, you must create this file manually if you want to use a separate policy file for the service.   The *qip.pcy* file is installed automatically during the VitalQIP installation and contains policies for all services. Thus, the *qip.pcy* file can also be modified. By default, the *qip-qipupdated.pcy* file is processed. If it is not found, the *qip.pcy* file is used. Only one policy file is processed.

When modifying the *qip.pcy* file, the **[VitalQIP QIP Update Service]** section name must precede the service's policies.

Important! The QIP Update Service has a registered port number of 2490.

Table 23 describes the available policies for the policy file.

Table 23 VitalQIP QIP Update Service (qip-qipupdated) policies

Policies	Values	Description
Debug	Default: None Allowed: See page 97	This policy sets the debug level. See "The debug policy", on page 97 for more information about the Debug policy.
DebugFile	Default: *qip-qipupdated.log* Allowable value: Relative or absolute filename	This policy specifies the filename where the debug output is sent.
Master	Default: False Allowed: True | False | Any number	***UNIX only***. This policy determines if the service runs an instance for each message service connected. This may reduce some bottlenecks but Master mode can generate a substantial load on the database. If the load is too much, you can specify a number instead that indicates the number of processes to run. The following values can be specified: **True** - the service runs in Master mode. **False** - service does not run in Master mode. **Any number** - a number can be specified. This number determines how many processes are started (and may not increase). This policy is case insensitive.

Policies	Values	Description
AddIDToLogFilename OnFork	Default: True Allowed: True | False	***UNIX only***. If set to True, a separate log file with a unique ID in its name is created for each child process that is spawned (in Master mode). Otherwise, only one log file is created for all processes. Use this policy when **Master**=true and the **FeatureRotateLogs** debug policy is set.
AcceptSleepTime	Default: 0 Allowed: Any number	With Master mode process limit enabled, a process can essentially end up handling all of the connections. This is due to the socket assignment policy in UNIX. With UNIX, any process which is blocked on a select is eligible to receive another connection. Therefore, if Message Services attempt to connect to the QIP Update Service when the QIP Update Service is not busy processing requests, typically one process gets all the connections. To attempt to work around this, you can set AcceptSleepTime. This causes the QIP Update Service to sleep “n” seconds after each connect, allowing other QIP Update Services to see new connections. If the QIP Update Service is busy updating the database, however, a new process is given the connection.
UpdateDNS	Default: True Allowed: True | False	This policy determines if DNS is updated with the host name. The QIP Update Service sends messages to the DNS UpdateService to update DNS servers with information about new leases. The Message Service needs a route for messages of type DNSUpdateObject. The following values can be specified: **True** - DNS is updated if the hostname does not conflict with a static object **False** - DNS is not updated. This policy is case insensitive. See “Unique hostname resolution options”, on page 301 for more information about name resolution.

Policies	**Values**	**Description**
SendRenewsToDNS	Default: OnChange Allowed: Yes | OnChange | No	The policy controls whether or not the QIP Update Service sends dynamic updates to the DNS Update Service when it receives DHCP renewal messages from a DHCP server via a Message Service. Refer to "About external object updates to DNS", on page 272 for more information on External Dynamic Update Propagation (EDUP) updates to DNS. In many networks, most IP addresses are dynamic DHCP (D-DHCP) addresses used for desktop computers. These addresses are renewed frequently. However, the A and PTR records in DNS are not changed and remain until the records are deleted. They do not need to be renewed. If the QIP Update Service sends only DHCP grants, the amount of processing is reduced. The following values are valid: **Yes** - the QIP Update Service can handle cases where Windows 2000 DNS has "scavenging" turned on or can provide redundancy for DHCP clients that are registered even if DNS does not receive a grant message. **OnChange** - the QIP Update Service sends a renew message to the DNS Update Service only if the fully qualified host domain name of the DHCP client has changed since the lease has been granted. Clients may change their fully qualified host names on a DHCP renew when the DHCP server is configured to accept clients names, and the DHCP client has a modified configuration. **No** - the QIP Update Service never sends a renew message to the DNS Update Service.
MaxConnections	Solaris - 256 AIX - unlimited Windows - 1024 Linux - 1024 HP-UX - variable, depending upon physical RAM: 1GB - 512 files; >1GB - 2048 files Allowed: A number between 1 and 1014	This policy provides a mechanism for configuring the soft limit on the number of files (or sockets/connections) available to a process. This can also be configured external to each service by using the ulimit command. On most systems, `ulimit -a` shows the number of available files. **Important!** The service adds 10 to the configured value so that processes total 10 to 1024 available files and connections.
ConnectQueueDepth	Default: 10 Allowed: Any number between 5 and 1000	This policy specifies how large to make the pending connection queue. If the queue is too small and the Update Service is busy in a database request, connecting services start to see "Connection Refused" errors.

Policies	**Values**	**Description**
ConvertSpaces	Default: False Allowed: True | False	This policy determines if spaces in a hostname are converted to dashes before the hostname is stored in the database. The following values are valid: **True** - spaces in a hostname are converted to dashes. **False** - spaces in a hostname are not converted to dashes.
DropRequests	Default: False Allowed: True | False	If this policy is set to True, updates are dropped and no processing is done with the message. The ACK is sent back to the Message Service immediately.
DumpStatsOnExit	Default: False Allowed: True | False	This policy determines if statistics are dumped when the service exits. The following values can be specified: **True** - statistics are dumped to the event log when the service exits. **False** - statistics are not dumped to the event log when the service exits.
AuditUpdates	Default: False Allowed: True | False	This policy determines if DHCP update information is forwarded to the local Message Service for delivery to the Audit Update Service. The following values can be used: **True** - the DHCP update information is forwarded to the local Message Service. **False** - DHCP update information is not forwarded. See the *Audit Manager User's Guide* for more information on the Audit Update Service.
DenyConnectionList	Default: None Allowed: A comma delimited list of IP addresses and CIDR - style IP address ranges | All	This policy does not allow connections from listed IP addresses and networks. An example of listed IP addresses would be: DenyConnectionList=127.0.0.1,10.0.0.0/8. In this example, connections from the loopback address and the Class A 10 network are not allowed. Refer to "CIDR Address Block and Bits", on page 28 for information about CIDR. If this policy is set to "All", connections from all IP addresses and networks are not allowed. If **AllowConnectionList** and **DenyConnectionList** are set at the same time, **AllowConnectionList** takes precedence over the **DenyConnectionList**.

Policies	Values	Description
AllowConnectionList	Default: 127.0.0.1 Allowed: A comma delimited list of IP addresses and CIDR - style IP address ranges | All	This policy allows connections from all listed IP addresses and networks. An example of list IP addresses would be: AllowConnectionList=127.0.0.1,10.0.0.0/8. In this example, connections from the loopback address and the Class A 10 network are allowed. If this policy is set to “All”, connections from all IP addresses and networks are allowed. If **AllowConnectionList** and **DenyConnectionList** are set at the same time, **AllowConnectionList** takes precedence over the **DenyConnectionList**.
UpdatePassword	Default: Unpublished for security reasons Allowed: Alphanumeric	This policy specifies the VitalQIP administrator’s password that the QIP Update Service uses to authenticate itself to VitalQIP.
Server	Default: Valid database server name Allowed: Alphanumeric (up to 30 characters)	This policy specifies the name of the database server to use. Setting this value overrides the environment variable QIPDATASERVER.
LoginServer	Default: 127.0.0.1 Allowed: IP address	The IP address of the host running the Login Service. More than one IP address can be specified to minimize VitalQIP's downtime in the event connectivity is lost with the Login Service. Each IP address must be separated by comma.
ListenPort	Default: Ephemeral Allowed: Ephemeral | Any valid port number | Any service name in */etc/services*	This policy specifies which port the service listens for messages. Ephemeral indicates that the service will use a port that is dynamically allocated by the operating system. It will register this port with the local message service. To accept messages from previous releases of VitalQIP, set this policy to the service name `qip-qdhcp`, or the port number `2490`. Ports are usually less than 32,000.
ExternalStatusToSystemLog	Default: False Allowed: True | False	This policy allows you to specify whether the service should log events about the status of external updates to the system log.
MessageServicePort	Default : 2468	This policy allows you to specify an alternate port to communicate with the Message Service.

Policies	Values	Description
DNSUpdatePrincipal	Default: Derived Allowed: Derived | Weak | Strong	DHCP updates that will be forwarded to a secure zone on a Microsoft DNS server can be updated with one of two principals: weak or strong. This can be controlled using the DNSUpdatePrincipal policy. If this value is set to “derived”, the QIP Update service will look up the value of the “Allow DHCP Clients to Modify Dynamic Object Resource Records” policy on the object, subnet, and global levels and derive the correct value. If this value is set to “weak”, the QIP Update service will always use the weak principal. If this value is set to “strong”, the QIP Update service will always use the strong principal. Circumventing the database lookup by setting this value to “weak” or “strong” may give a performance improvement on DHCP updates at the expense of on-the-fly principal derivation.

Avoid dynamic name collisions

When to use

The QIP Update Service can be used to avoid dynamic name collisions occurring with static objects. The QIP Update Service (**qip-qipupdated**) has the capability to send messages to the DNS Update Service, which performs dynamic DNS updates.

Procedure

To allow the QIP Update Service to validate names assigned by the DHCP Server before updating DNS, follow these steps:

1 Configure one of two message routes in the **[VitalQIP Message Service]** section of the policy file to send DHCP messages to the QIP Update Service, rather than the DNS Update Service and send DNSUpdateObject to the DNS Update Service. For example:

```
MessageRoute=DHCP:A:0:QIP Update Service (DHCP):VitalQIP QIP Update Service:127.0.0.1
MessageRoute=DNSUpdateRR:A:0:QIP Update Service (Update RR):VitalQIP QIP Update Service:127.0.0.1
MessageRoute=DNSUpdateObject:A:0:DNS Update Service (Object):VitalQIP DNS Update Service:127.0.0.1
MessageRoute=DNSUpdateRR:A:0:DNS Update Service (Update RR):VitalQIP DNS Update Service:127.0.0.1
```

2 Set the **UpdateDNS** policy option to “True” (the default) in the [VitalQIP QIP Update Service] section of the policy file.

If you route DNS Updates through the QIP Update Service, it allows name checking to be performed before the DDNS packet is issued to the DNS server. Standard name checking prevails (for example, FIRST-IN or LAST-IN). Refer to “Unique hostname resolution options”, on page 301 for more information about FIRST-IN and LAST-IN.

For example, the DHCP server gives a lease to TEST xxx.xxx.xxx.3, which is sent to the QIP Update Service. The QIP Update Service inserts the name and IP address into VitalQIP and checks for name collisions. If a collision does not occur, a packet is sent to the DNS Update Service. If an object named TEST already exists, a name collision occurs. VitalQIP assigns a new name to the object. This new name is stored in the VitalQIP database and is forwarded to the DNS Update Service.

□

VitalQIP DNS Update Service policies

Behavior of the DNS Update Service (**qip-dnsupdated**) is controlled by policies in the *qip-dnsupdated.pcy* file or the *qip.pcy* file, located in the *QIPHOME* directory. The *qip-dnsupdated.pcy* file is used only by the DNS Update Service. This file is not created automatically. You must create this file manually if you want to use a separate policy file for the service. The *qip.pcy* file is installed during the VitalQIP installation. The file and contains policies for all services. Thus, the *qip.pcy* file can also be modified. By default, the *qip-dnsupdated.pcy* file is processed. If it is not found, the *qip.pcy* file is used. Only one policy file is processed.

When modifying the *qip.pcy* file, the **[VitalQIP DNS Update Service]** section name must precede the service's policies.

Important! The DNS Update Service uses unregistered port number 3119.

Table 24 describes the available policies for the policy file.

Table 24 VitalQIP DNS Update Service (qip-dnsupdated) policies

Policy	Values	Description
Debug	Default: None Allowed: See page 97	This policy sets the debug level. See "The debug policy" on page 97 for more information about the Debug policy.
DebugFile	Default: *qip-dnsupdated.log* Allowed: Relative or absolute filename	This policy specifies the filename where the debug output is sent.
DenyConnectionList	Default: None Allowed: A comma delimited list of IP addresses and CIDR-style IP address ranges | All	This policy does not allow connections from listed IP addresses and networks. An example of listed IP addresses would be: DenyConnectionList=127.0.0.1,10.0.0.0/8. In this example, connections from the loopback address and the Class A 10 network are not allowed. Refer to "CIDR Address Block and Bits", on page 28 for information about CIDR. If this policy is set to "All", connections from all IP addresses and networks are not allowed. If **AllowConnectionList** and **DenyConnectionList** are set at the same time, **AllowConnectionList** takes precedence over the **DenyConnectionList**.

Policy	Values	Description
AllowConnectionList	Default: 127.0.0.1 Allowed: A comma delimited list of IP addresses and CIDR-style IP address ranges | All	This policy allows connections from all listed IP addresses and networks. An example of listed IP addresses would be: AllowConnectionList=127.0.0.1,10.0.0.0/8. In this example, connections from the loopback address and the Class A 10 network are allowed. If this policy is set to "All", connections from all IP addresses and networks are allowed. If **AllowConnectionList** and **DenyConnectionList** are set at the same time, **AllowConnectionList** takes precedence over the **DenyConnectionList**.
ListenPort	Default: Ephemeral Allowed: Ephemeral | Any valid port number | Any service name in */etc/services*	This policy specifies which port the service listens for messages. Ephemeral indicates that the service will use a port that is dynamically allocated by the operating system. It will register this port with the local message service. To accept messages from previous releases of VitalQIP, set this policy to the service name `qip-dns`, or the port number `3119`. Ports are usually less than 32,000.
Master	Default: False Allowed: True | False | any number	***UNIX only***: for each connection to a Message Service, a new DNS Update Service is forked. This may reduce some bottlenecks. This policy is case insensitive. This policy determines if the service runs in Master mode or how many processes are started. The following values can be specified: **True** - the service runs in Master mode. **False** - service does not run in Master mode. **Any number** - a number can be specified. This number determines how many processes are started (and may not increase). Each process shares (see **AcceptSleepTime**) new connections.
AddIDToLogFilename OnFork	Default: True Allowed: True | False	***UNIX only***. If set to True, a separate log file with a unique ID in its name is created for each child process that is spawned (in Master mode). Otherwise, only one log file is created for all processes. Use this policy when **Master**=true and the **FeatureRotateLogs** debug policy is set.

Policy	Values	Description
AcceptSleepTime	Default: 0 Allowed: Any number	With Master mode limit enabled, a process can handle all connections. The socket assignment policy in UNIX allows this action. With UNIX, any process that is blocked is eligible to receive another connection. If a Message Service attempts to connect to the Update Service when it is not busy processing requests that Update Service receives all connections. You can set **AcceptSleeptime** to work around this action. The Update Service "sleeps" the specified number of seconds after each connection. In turn, other QIP Update Services see a new connection.
DDNS_Attempts	Default: 1 Allowed: Numeric	This policy specifies the number of attempts to make to update a BIND 8.x or 9 DNS server. A value of 0 turns off DNS updates.
DDNS_Timeout	Default: 5 Allowed: Numeric	This policy specifies the number of seconds to wait (per attempt) for a response from a BIND 8.x or 9 DNS server during an update.
DDNS_TTL	Default: 0 (no value is set) Allowed: Numeric	This policy is only useful to caching-only servers. When the DHCP server grants or renews a lease, the VitalQIP Update Service or Message Service updates DNS with the lease information and a calculated TTL value for the lease duration time. If shorter or longer values are needed for DNS, you can specify a non-zero value.
MaxConnections	Solaris - 256 AIX - unlimited Windows - 1024 Linux - 1024 HP-UX - variable, depending upon physical RAM: 1 GB - 512 files; >1 GB - 2048 files Allowed: A number between 1 and 1014	This policy provides a mechanism for configuring the soft limit on the number of files (or sockets/connections) available to a process. This can also be configured external to each service by using the ulimit command. On most systems, `ulimit -a` shows the number of available files. **Important!** The service adds 10 to the configured value so that processes total 10 to 1024 available files and connections.

Policy	Values	Description
SendRenewsToDNS	Default: No Allowed: Yes | No	Typically, DHCP renew messages need not be sent to DNS because the information is already there. However, if messages are not sent to the QIP Update Service first and the DNS update service gets messages directly from a DHCP server, this policy should be used. The default is not to send DHCP renew messages. The following values are valid: **Yes** - the DNS Update Service can handle cases where Windows 2000 DNS has "scavenging" turned on, or can provide redundancy for DHCP clients that are registered even if DNS does not receive a grant message. **No** - the DNS Update Service never sends an update to DNS on a renew. **Important!** If dynamic DNS updates are coming from a Message Service to the QIP Update Service before being routed to the DNS Update Service, this policy has no effect and should instead be set in the QIP Update Service section. Refer to "SendRenewsToDNS", on page 106.
ConnectQueueDepth	Default: 10 Allowed: Any number between 5 and 1000	This policy specifies how large to make the pending connection queue. If the queue is too small and the Update Service is busy with a database request, connecting services receive "Connection Refused" errors.
ConvertSpaces	Default: False Allowed: True | False	This policy determines if spaces in a hostname are converted to dashes before the hostname is stored in the database. The following values are valid: **True** - spaces in a hostname are converted to dashes. **False** - spaces in a hostname are not converted to dashes.
DumpStatsOnExit	Default: False Allowed: True | False	This policy determines if statistics are dumped when the service exits. The following values can be specified: **True** - statistics are dumped to the event log when the service exits. **False** - statistics are not dumped to the event log when the service exits.

Policy	Values	Description
DoSecureUpdates	Default: True Allowed: True | False	This policy determine if secure updates are sent to server and zone combinations that are configured for secure updates. The following values are not specified: **True** - secure updates are sent to server and zone combinations that are configured for secure updates. **False** - secure updates are not sent.
KerberosPrincipal	Default: (None) Allowed: Free form string	This policy applies to UNIX only. The name of the Kerberos principal to which the service is registered. This policy may be overridden with the QIP_KERBEROS_PRINCIPAL environment variable. "ddns.conf with proxies for Windows 2000 secure servers", on page 251 for information about Kerberos.
GSSAPIImmediateRetries	Default: 1 Allowed: Numeric	The number of times that the DNS Update Service tries a GSSAPI /SSPI operation before reporting a failure.
GSSAPIRetryDelay	Default: 900 Allowed: Numeric	The number of seconds after a failure the DNS Update Service waits before attempting to send another secure update to a DNS server.
DoKinit	Default: False Allowed: True | False	This policy determines if the DNS server performs its own Kerberos initialization. The following values can be specified: **True** - the DNS server performs its own Kerberos initialization. **False** - the DNS server does not perform its own Kerberos initialization. This policy is overridden by the **QIP_DO_KINIT** environment variable. The policy is used for UNIX only.
KeytabPath	Default: None Allowable value: */etc/krb5.keytab*	The *keytab* file to use when performing Kerberos initialization. This policy can be overridden with the **QIP_KEYTAB_PATH** environment variable. This policy applies to UNIX only.

Policy	**Values**	**Description**
DDNSGenerateSleep	Default: 0 (only generated at startup) Allowed: Numeric with an option modifier	If DDNSGenerateSleep is set to a non-zero value, the *ddns.conf* files are generated at the specified interval. Valid values are any number with an optional modifier. The following modifiers can be used: **s** - second (default modifier) **m** - minute **h** - hour **d** - day For example, DDNSGenerateSleep=30m regenerates the *ddns.conf* file every 30 minutes.
PauseBetweenChildren	Default: 0 Allowed: Numeric	When the Master policy is set to True or a number greater than zero, this policy determines how long to pause between sending signals to child processes. Set this policy to spread out the load if the CPU spikes when reading newly available *ddns.conf* files. This value is the number of seconds to pause between sending signals. It does not apply to SIGTERM.

□

VitalQIP MS DNS Update Service policies

Behavior of the MS DNS Update Service (**qip-msdnsupdated**) is controlled by policies in the *qip-msdnsupdated.pcy* file or the *qip.pcy* file, located in the *QIPHOME* directory. The *qip-msdnsupdated.pcy* file is used only by the MS DNS Update Service. This file is not created automatically. Since the *qip-msdnsupdated.pcy* file is not created automatically, you must create this file manually if you want to use a separate policy file for the service. The *qip.pcy* file is installed automatically during the VitalQIP installation and contains policies for all services. Thus, the *qip.pcy* file can also be modified. By default, the *qip-msdnsupdated.pcy* file is processed. If it is not found, the *qip.pcy* file is used. Only one policy file is processed.

When modifying the *qip.pcy* file, the **[VitalQIP MS DNS Update Service]** section name must precede the service's policies.

Table 25 describes the available policies for the policy file.

Table 25 VitalQIP MS DNS Update Service policies

Policy	Values	Description
Debug	Default: None Allowed: See page 97	This policy sets the debug level. Refer to "The debug policy", on page 97 for more information about the Debug policy.
DebugFile	Default: *msdnsupdated.log* Allowable value: Relative or absolute filename	This policy specifies the filename where the debug output is sent.
ListenPort	Default: Ephemeral Allowed: Ephemeral | Any valid port number | Any service name in */etc/services*	This policy specifies which port the service listens for messages. Ephemeral indicates that the service will use a port that is dynamically allocated by the operating system. It will register this port with the local message service. Ports are usually less than 32,000.
DenyConnectionList	Default: None Allowed: A comma delimited list of IP addresses and CIDR - style IP address ranges | All	This policy does not allow connections from listed IP addresses and networks. An example of listed IP addresses would be: DenyConnectionList=127.0.0.1,10.0.0.0/8. In this example, connections from the loopback address and the Class A 10 network are not allowed. For information about CIDR, see "CIDR Address Block and Bits", on page 28. If this policy is set to "All", connections from all IP addresses and networks are not allowed. If **AllowConnectionList** and **DenyConnectionList** are set at the same time, **AllowConnectionList** takes precedence over the **DenyConnectionList**.

Policy	Values	Description
AllowConnectionList	Default: 127.0.0.1 Allowed: A comma delimited list of IP addresses and CIDR - style IP address ranges | All	This parameter allows connections from all listed IP addresses and networks. An example of listed IP addresses would be: AllowConnectionList=127.0.0.1,10.0.0.0/8. In this example, connections from the loopback address and the Class A 10 network are allowed. If this policy is set to "All", connections from all IP addresses and networks are allowed. If **AllowConnectionList** and **DenyConnectionList** are set at the same time, **AllowConnectionList** takes precedence over the **DenyConnectionList**.
FailureThreshold	Default: 5 Allowed: A positive number	The number of resource record updates that can fail per zone before a push failure is reported to the caller.
UpdateRetry	Default: 5 Allowed: A positive number	The number of times the service tries to add a resource record before considering the resource record update a failure.
UpdateDelay	Default: 50 Allowed: A positive number of milliseconds	The number of milliseconds that the service is delayed after failing to add a resource record before attempting to reapply the update. The update delay allows the DNS server to process previously applied updates.
RecsPerPacket	Default: 100 Allowed: A positive number	The number of records that the service caches in memory before sending a packet to the caller. This parameter is used when the service proxies a zone transfer request to the DNS server for the caller. This parameter can affect the performance of the zone transfer.
UpdateQueueLength	Default: 10,000 Allowed: A positive number	The number of dynamic update requests to queue. Any requests attempted after this many are already pending are dropped. If many dropped requests are occurring, consider increasing the number of threads, increasing the queue length, or upgrading the machine.
ProcessFilesQueueLength	Default: 5 Allowed: A positive number	The number of push requests to queue. In theory, there should only be one push happening at a time, so this number should never have to change.

Policy	Values	Description
AXFRQueueLength	Default: 5 Allowed: A positive number	The number of AXFR requests to queue. AXFR happens on a push to screen, local or server through a proxy. Only one AXFR will be allowed to happen at a time, although if multiple users are pushing to the screen or local, there may be more than one queued.
ThreadPoolCount	Default: 50 Allowed: A positive number	The number of threads to listen for dynamic update requests. This should only need to be adjusted if you expect to have more than 50 VitalQIP GUIs updating secure zones on this server simultaneously.

☐

VitalQIP Message Service policies

The VitalQIP Message Service performs the following functions:

- Queues messages until they can be processed by another service
- Forwards SSL-enabled or cleartext messages to multiple destinations
- Provides message queue length restrictions
- Provides disk-based storage for messages
- Accepts messages from more than one source
- Provides reliable transport for remote sources
- Maintains compatibility with previous version of the Message Service

The VitalQIP Message Service queues messages from the DHCP server, DNS server, DHCP monitor services, the VitalQIP GUI, and the VitalQIP QIP Update Service and forwards the messages to other services. The final destination of messages sent to the VitalQIP Message Service depends entirely upon the message type. For example, the DHCP server sends messages of type 1 that are typically sent to the VitalQIP QIP Update Service.

The Message Service is configurable and allows the user to specify details about message queuing and message delivery (message routing). Depending on the setting of the **SecureIncomingMessageServiceConnections** policy, it allows messages to be SSL-enabled or plain text. The Message Service can accept secure connections or insecure connections, but not both.

> **Important!** The **SecureIncomingMessageServiceConnections** policy (located in the global section of *qip.pcy*) controls the behavior of both the SSL Tunnel Service and the Message Service. If this policy is set to True, the Message Service only binds to the well known port on the loopback address and the SSL Tunnel binds to the well known port on the network interface as well as the ephemeral port with which it will communicate with the Message Service. This allows the SSL Tunnel to accept only secure SSL-enabled external connections.
>
> If the **SecureIncomingMessageServiceConnections** policy is set to False (the default value), the Message Service binds to its well known port on the network and loopback interfaces, and the SSL Tunnel does not bind to any well known port. Instead it only binds to an ephemeral port and registers that port with the Message Service so that the Message Service can communicate with it. This allows the Message Service to accept only clear (or plain) text connections.

Message Queues, Message Routes (or Secure Message Routes) are configured in the *qip.pcy* policy file. A Message Queue defines attributes of a queue defined by a message type. Message Routes and Secure Message Routes define attributes of a destination defined by a message type. The VitalQIP Message Service has multiple queues, one for each message type.

Behavior of the Message Service (**qip-msgd**) is controlled by policies in the *qip-msgd.pcy* file or the *qip.pcy* file, located in the *QIPHOME* directory. The *qip-msgd.pcy* file is used only by the Message Service. This file is not created automatically. Since the *qip-msgd.pcy* file is not created automatically, you must create this file manually if you want to use a separate policy file for the service. The *qip.pcy* file is installed automatically during the VitalQIP installation and contains policies for all services. Thus, the *qip.pcy* file can also be modified. By default, the *qip-msgd.pcy* is processed. If it is not found, the *qip.pcy* file is processed. Only one policy file is processed.

When modifying the *qip.pcy* file, the **[VitalQIP Message Service]** section name must precede the service's policies.

> **Important!** The Message Service has a registered port number of 2468. The policy file can be edited to have ports other than the default for the Message Service.

Table 26 describes the policies available for the policy file.

Table 26 VitalQIP Message Service (qip-msgd) policies

Policy	Values	Description
Debug	Default: None Allowed: See page 97	This policy sets the debug level. See "The debug policy", on page 97 for more information about the Debug policy.
DebugFile	Default: *qip-msgd.log* Allowable value: Relative or absolute filename	This policy specifies the filename where the debug output is sent.
DenyConnectionList	Default: None Allowed: A comma delimited list of IP addresses and CIDR - style IP address ranges | All	This policy does not allow connections from listed IP addresses and networks. An example of listed IP addresses would be: DenyConnectionList=127.0.0.1,10.0.0.0/8. In this example, connections from the loopback address and the Class A 10 network are not allowed. Refer to "CIDR Address Block and Bits", on page 28 for information about CIDR. If this policy is set to "All", connections from all IP addresses and networks are not allowed. If **AllowConnectionList** and **DenyConnectionList** are set at the same time, **AllowConnectionList** takes precedence over the **DenyConnectionList**.

Policy	Values	Description
AllowConnectionList	Default: All Allowed: A comma delimited list of IP addresses and CIDR - style IP address ranges | All	This policy allows connections from all listed IP addresses and networks. An example of listed IP addresses would be: AllowConnectionList=127.0.0.1,10.0.0.0/8. In this example, connections from the loopback address and the Class A 10 network are allowed. If this policy is set to “All”, connections from all IP addresses and networks are allowed. If **AllowConnectionList** and **DenyConnectionList** are set at the same time, **AllowConnectionList** takes precedence over the **DenyConnectionList**.
AdditionalIPList	Default: Allowed: A comma delimited list of IP addresses	This policy specifies a comma separated list of IP addresses that should be treated as local addresses. This is used primarily for NAT separated machines trying to connect to a remote Message service. You can specify the NAT-assigned address to be treated as an address local to this machine.
MaxConnections	Default: AIX - unlimited HP-UX -variable, depending upon physical RAM: 1 GB - 512 files >1 GB -2048 files Solaris - 256 | Linux - 1024 Allowed: A number between 1 and 1014	This policy specifies the maximum number of Message Services that can connect (UNIX only). If you plan to “chain” Message Services together, the default values need to change. **Important!** The service adds 10 to the configured value so that processes total 10 to 1024 available files and connections. **Important!** This can also be configured external to each service by using the ulimit command. On most systems, `ulimit -a` shows the number of available files.
ReConnectTimeout	Default: 180 Allowed: Any positive number of seconds	When a connection is lost, the Message Service waits before attempting to re-establish a connection. The wait period is configured according to this policy and a random time that is up to 1/6 as large as the re-connect timeout, for example, 180 + (0...30).
DumpStatsOnExit	Default: False Allowed: True | False	This policy determines if statistics are dumped when the service exits. The following values can be specified: **True** - statistics are dumped to the event log when the service exits. **False** - statistics are not dumped to the event log when the service exits.

Policy	**Values**	**Description**
MessageServicePort	Default: 2468 Allowed: Any valid port number	This policy specifies on what port the service listens for UDP and TCP messages. Ports are usually less than 32,000.
ReceiveBufferSize	Default: 16K Allowed: A number between 4096 and 65536	This policy truncates messages larger than the specified value. This policy is useful for handling larger messages. This is used for UDP messages only.
AllowUDPControl Messages	Default: False Allowed: True | False	This policy determines if the Message Service can be dynamically configured throughout a network. The following values can be specified: **True** - the Message Service can be dynamically reconfigured throughout a network. The Message Service may create dynamic routes, accept service, and port publications, and act as a proxy. **False** - the Message Service ignores any requests from the network to reconfigure.
StatsLogPeriod	Default: 0 Allowed: A number between 0 and 2^{32} (that is, 4,294,967,295 seconds).	When set, this policy turns on periodic statistics logging to the system log and indicates the minimum number of seconds the Message Service waits between logging statistics. This policy is useful for determining whether message queues are growing, and can help determine whether Distributed Services should be installed. If the period is set to 0, no periodic logging occurs.
SecureOutgoingProxy Connections	Default: False Allowed: True | False	When this policy is set to True, the Message Service will communicate with Proxy Message Services via the SSL Tunnel Service to make a secure outgoing connection. The SSL Tunnel Service provides transport authentication and encryption. A setting of False causes the Message Service to attempt only a clear outgoing connection.
MessageQueue	Default: See Table 27 on page 125 for definitions of the values Allowed: <id>: <memq>: <diskq>: <ttl>: <warn>: <message directory>	The Message Service is generic. It forwards/routes multiple message types to any number of destinations. This policy and the MessageRoute policy control that. This policy defines the characteristics of the queue to be used with messages of type <*id*>. If this policy is not configured, the Message Service defaults to its previous queuing behavior with no limitations. The <*id*> must be the same as the <*id*> in the MessageRoute policy.

Policy	Values	Description
MessageRoute	Default: See Table 28 on page 127 for definitions of the values Allowed: <id>, <flags>, <ACK timeout>, <desc>, <port/servicename>, <Server IP>, <backup>	The Message Service is generic. It forwards and routes multiple message types to any number of destinations controlled by this policy. The MessageQueue policy is controlled by the MessageRoute policy. This policy defines a destination for message of type <*id*>. The <*id*> must be the same as the <*id*> in the MessageQueue policy.
SecureMessageRoute	Default: See Table 28 on page 127 for definitions of the values Allowed: <id>, <flags>, <ACK timeout>, <desc>, <port/servicename>, <Server IP>, <backup>	Identical to MessageRoute except that instead of making connections directly, SecureMessageRoutes make connections through the SSL Tunnel Service to make use of message traffic authentication and encryption. For more information on secure message routes, refer to Chapter 6, “Secure message routes” on page 195.

MessageQueue policy

This is an optional policy and is specified as:

```
MessageQueue=<id>:<memq>:<diskq>:<ttl>:<warn limit>:<message_directory>
```

Important! You can use either commas (,) or colons (:) to separate the values.

The MessageQueue policy examples can be found at the end of the **[VitalQIP Message Service]** section of the *$QIPHOME/qip.pcy* file.

If this policy is not specified, no disk queuing is done, and there is ***no limit*** to the memory queue. The fields of the MessageQueue policy are defined in Table 27.

Table 27 MessageQueue values

MessageQueue value	Description
Message ID <*id*>	Numeric. The Message ID uniquely identifies the type of message received by the Message Service. It must agree with the <*id*> of the MessageRoute policy. Currently, the following message types are handled: **1** or **DHCP** - Messages that come from a DHCP server. **3** or **Audit** - Audit messages that are sent from the VitalQIP client and QIP Update Service. **9** or **DNSUpdateObject** - Messages that describe an object to be updated in DNS. Such messages are sent from the VitalQIP client and from QIP Update Service to the DNS Update Service. **Important!** VitalQIP clients only send these messages if the **Use DNS Update Service** policy is set to True. **10** or **DNSUpdateRR** - Messages that describe a resource record added to or removed from a DNS server. Such messages are sent from DNS servers and from VitalQIP clients to the DNS Update Service. **Important!** VitalQIP clients only send these messages if the **Use DNS Update Service** policy is set to True.
Memory Queue Max <*memq*>	Each memory queue can have a pre-determined maximum memory size. When messages are received and the memory queue has reached its maximum permissible size, either the message is stored to disk or the oldest message is removed from the queue. A value of 0 configures the queue for infinite messages.
Disk Queue Max <*diskq*>	The maximum number of messages to store on disk (must be at least as large as memq). When *diskq* is enabled, all messages received of type id are stored to disk. When messages are received and *diskq* has reached its limit, the oldest message in the queue is marked for deletion.
Time to Live <*ttl*>	The number of seconds a message can live on queue until it is deleted. If set to zero, messages may stay on the queue indefinitely (until they are successfully routed).

MessageQueue value	Description
warn limit	Displays a warning message in the event log when the queue reaches (and doubles) this limit. Each subsequent doubling of the queue limit causes a warning to be displayed.
message directory	Indicates where to store the message queue directories. The default is *$QIPHOME/log/messages*.

Sample message queue

Examine a sample message queue:

```
MessageQueue = 1:5000:50000:3600:1000:
```

This message queue has the following characteristics:

- It defines a message queue for message type DHCP (1)
- The memory queue will store a maximum of 5,000 messages
- The disk queue will store a maximum of 50,000 messages
- Messages that are waiting on the queue for over an hour (3,600 seconds) are removed
- Once the queue hits 1,000, a message is sent to the syslog/event viewer (and again, at 2,000, 3,000 and so on)
- Disk messages are stored in the default location, *$QIPHOME/log/messages.001,* since no other is specified.

MessageRoute policy

This optional policy is specified as:

```
MessageRoute = <id>:<flags>:<ACK_timeout>:<desc>:<port>:<Server_IP>:…<*#>
```

Important! You can use either commas (,) or colons (:) to separate the values.

Several MessageRoute policy examples can be found at the end of the **[VitalQIP Message Service]** section of the *$QIPHOME/qip.pcy* file.

This policy defines where messages are sent/routed. There can be any number of MessageRoutes for a given message type. Table 28 describes the MessageRoute values.

Table 28 MessageRoute values

MessageRoute value	Description
Message ID <*id*>	Numeric. The Message ID uniquely identifies the type of messages received by the Message Service. It must agree with the <*id*> of the MessageRoute policy. Currently, the following message types are handled: **1** or **DHCP** - Messages which come from a DHCP server. **3** or **Audit** - Audit messages which are sent from the VitalQIP client and QIP Update Service. **9** or **DNSUpdateObject** - Messages that describe an object to be updated in DNS. Such messages are sent from the VitalQIP client and from QIP Update Service to the DNS Update Service. **Important!** VitalQIP clients only send these messages if the **Use DNS Update Service** policy is set to True. **10** or **DNSUpdateRR** - Messages that describe a resource record added to or removed from a DNS server. Such messages are sent from DNS servers and from VitalQIP clients to the DNS Update Service. **Important!** VitalQIP clients only send these messages if the **Use DNS Update Service** policy is set to True.

MessageRoute value	Description
Flags <*flags*>	The Flags field specifies a list of values for message processing. Each space in the flags field corresponds to one configuration value. Currently, three flag field spaces are defined. Valid values for field 1 are: **A** - **Asynchronous**. Messages are sent asynchronously with respect to other routes. See “Asynchronous example”, on page 129. **S** - ***Deprecated in VitalQIP 6.2***. **Synchronous**. Messages are sent to destination only after previous route has received ACK. Valid values for field 2 are: **L** - Lockstep. Used in combination with the first field as follows: **AL** - ***Deprecated in VitalQIP 6.2***. **Asynchronous Lockstep**. Next message is not sent until all routes have sent current message. **SL** - ***Deprecated in VitalQIP 6.2***. **Synchronous Lockstep**. Current message is not sent until previous route gets ACK, and next message is not sent until current message has been sent to all destinations. **.** - Indicates that the route uses the default behavior for a flag. It is used as a placeholder since the flags are position dependent. It does not need to be specified. **Important!** Users of deprecated flags should change them to A when upgrading to VitalQIP 6.2. Valid values for field 3 are: **B** - **Load Balance**. Balance message delivery between each destination. A MessageRoute "load balances" when it has a message to send and it has not received an ACK for a message that it has sent previously. See “Load Balance example”, on page 130. **R** - **Primary Reconnect**. Treats the first destination as primary and attempts to reconnect. It is independent of the preceding two flags (for example ALR, SLR, A.R, S.R). See “Primary Reconnect example”, on page 129.
ACK Timeout <*ack timeout*>	The number of seconds to wait for an ACK from the server (default is 0, no timeout). Use of this value is not recommended since another message is sent to the service immediately upon timeout. If multiple messages are sent due to multiple timeouts, the TCP queue to the service may fill up.
Destination Description <*desc*>	The description of the message route (for example, you could use the description “QIP Update Service” for messages destined for **qip-qdhcpd**).

MessageRoute value	Description
Destination Port Name <*port/servicename*>	The port or service name of where the message is being sent. Port names must be defined in the appropriate services file. The service and port names are as follows: **Service Name** — **Port Name** VitalQIP DNS Update Service — qip-dns VitalQIP QIP Update Service — qip-qdhcp VitalQIP Message Service — qip-msgd Use the port name if the target service is listening on its well-known port. Use the service name if the service is listening on an ephemeral port tunneled through the Message Service. The Message Service uses the MessageServicePort value to contact the IP address specified in the IP Address of the Primary Server <*server_ip*> and requests a conduit connection to the service specified in the port name of the message route.
IP Address of the Primary Server <*server_ip*>	The IP address of the enterprise server running on the specified port in the Destination Port Name <*port/servicename*>.
Backup Servers <…>	A list of alternate servers where there is a service running on the specified port specified in the Destination Port Name <*port*>.
<*#>	Destination multiplier. Allows you to specify the number of connections you wish to open for load balancing on the backup servers, rather than listing duplicate destinations multiple times.

The following are MessageRoute examples:

Asynchronous example

```
MessageRoute = 1:A:0:QIPUpdateService: VitalQIP QIP Update Service:192.0.2.22:192.0.2.25
MessageRoute = 1:A:0:DNSUpdateService: VitalQIP DNS Update Service:127.0.0.1:192.0.2.1:192.0.2.3
```

These message routes have the following characteristics:

- They define two routes for messages of type 1 (QIP Update messages).
- Due to the ‘A’ flag, messages are sent to both destinations as fast as possible. In other words, the Message Service does not wait for messages to be sent to the QIP Update Service before sending messages to the DNS Update Service.
- The first route is to the port defined by VitalQIP QIP Update Service, and messages are sent to 192.0.2.22 unless it is down, in which case messages are sent to 192.0.2.25.
- The second route tries to update the DNS Update Service on the local machine, but if that service is not up, it attempts to update the Lucent DNS Service on 192.0.2.1. If that service is down, it sends messages to 192.0.2.3.

Primary Reconnect example

You can configure the Message Service to reconnect to a primary destination for a given route by specifying “R” in the third space of the Flags field. Previously, if multiple destinations were configured for a route, each destination was treated equally: the Message Service would not attempt to “reconnect” to the first destination (primary).

```
MessageRoute = 1:A.R:0:QIPUpdateService:qip-qdhcp:127.0.0.1:3119:127.0.0.1
```

This message route has the following characteristics:

- Each type 1 message attempts to connect to **qip-qdhcp** on the local host.
- If it is down, it sends messages to the DNS Update Service on port 3119.
- The Message Service actively attempts to restore a connection to **qip-qdhcp** indefinitely.

Load Balance example

```
MessageRoute 1:A.B:0:QIPUpdateService:VitalQIP QIP Update Service:192.0.2.7*3
```

The on-demand load balancing example above has the following characteristics:

Each type 1 message is sent to the QIP Update Service:

- Upon startup, the Message Service will make three connections to 192.0.2.7.
- If the Message Service receives message *n*, it will send it to 192.0.2.7 and wait for an ACK (ACK1).
- If ACK1 is received from 192.0.2.7 before the next message (*n+1*) is received, message *n+1* will be sent to 192.0.2.7 on the first connection since that connection is free to process another message.
- If message *n+1* is received before ACK1, message *n+1* will be sent to 192.0.2.7 using a different connection.
- Similarly, if message *n+2* is received before ACK1 and ACK2, message *n+2* will be sent to 192.0.2.7 using the third connection.
- Suppose all destinations are processing a message and the Message Service is waiting for an ACK from each. If message *n+3* is received, it will be queued and the Message Service will wait for the ACKs. Whichever process/thread on 192.0.2.7 is the first to respond with an ACK will get message *n+3*.
- In all cases, the next message in the queue is sent to the first destination to respond with an ACK. If the first destination always responds with an ACK before the next message is queued, that destination will process all the messages and there is no reason to bother the other destinations.

Important! In the above example, where three different processes are running on the same host address, throughput is increased because each process can perform the necessary CPU processing while the others are blocked, waiting for disk access to update the database. For database operations where it may be beneficial to have the database server running on the same machine as the QIP Update Service and the Audit Update Service, this would provide nearly equivalent, and sometimes greater throughput than specifying different addresses.

☐

VitalQIP Login Service policies

Behavior of the Login Service (**qip-logind**) is controlled by policies in the *qip-logind.pcy* file or the *qip.pcy* file, located in the *QIPHOME* directory. The *qip-logind.pcy* file is used only by the Login Service. This file is not created automatically. Since the *qip-logind.pcy* file is not created automatically, you must create this file manually if you want to use a separate policy file for the service. The *qip.pcy* file is installed automatically during the VitalQIP installation and contains parameters for all services. Thus, the *qip.pcy* file can also be modified. By default, the *qip-logind.pcy* file is processed. If this file is not found, the *qip.pcy* file is used. Only one policy file is processed.

Important! If you move the VitalQIP Login Service to another host, the new address must be entered into the *qip.pcy* file. For example, if you placed the Login Service on host 192.0.2.3, you would type LoginService=192.0.2.3.

When modifying the *qip.pcy* file, the **[VitalQIP Login Service]** section name must precede the service's policies.

Important! The Login Service binds to an ephemeral port and registers itself with the Message Service. All communication with it is done through the Message Service.

Table 29 describes the policies available for the policy file.

Table 29 VitalQIP Login Service (qip-logind) policies

Policy	Values	Description
Debug	Default: None Allowed: See page 97	This policy sets the debug level. See "The debug policy", on page 97 for more information about the Debug policy.
DebugFile	Default: *qip-logind.log* Allowed: Relative or absolute filename	This policy specifies the filename where the debug output is sent.
ListenPort	Default: Ephemeral Allowed: Ephemeral | Any valid port number | Any service name in */etc/services*	This policy specifies which port the service listens for messages. Ephemeral indicates that the service will use a port that is dynamically allocated by the operating system. It will register this port with the local message service. To accept messages from previous releases of VitalQIP, set this policy to the service name `qip-login`, or the port number `2366`. Ports are usually less than 32,000.

Policy	Values	Description
ReConnectTimeout	Default: 5 Allowed: Any positive number of seconds	When the Login Service starts, it publishes with the Message Service. If the Message Service is restarted, the Login Service will need to re-connect to re-publish. This parameter specifies the maximum time in seconds that the Login Service will wait between connect attempts. It can be short since the connection is local.
DropRequests	Default: False Allowed: True | False	This policy determines if all requests are ignored. The following values can be specified: **True** - all requests are ignored. **False** - not all requests are ignored. This can be useful for debugging.
DumpStatsOnExit	Default: False Allowed: True | False	This policy determines if statistics are dumped when the service exits. The following values can be specified: **True** - statistics are dumped to the event log when the service exits. **False** - statistics are not dumped to the event log when the service exits.
ConnectQueueDepth	Default: 5 Allowed: A number between 5 and 1000.	This policy specifies how large to make the pending connection queue. If the queue is too small and the Login Service is busy connecting services the service receives “Connection Refused” errors.
MaxConnections	Solaris - 256 AIX - unlimited Windows - 1024 Linux - 1024 HP-UX - variable, depending upon physical RAM: 1 GB - 512 files; >1 GB - 2048 files Allowed: A number between 1 and 1014	This policy provides a mechanism for configuring the soft limit on the number of files (or sockets/connections) available to a process. This can also be configured external to each service by using the ulimit command. On most systems, `ulimit -a` shows the number of available files. **Important!** The service adds 10 to the configured value so that processes total 10 to 1024 available files and connections.

Policy	Values	Description
DenyConnectionList	Default: None Allowable value: A comma delimited list of IP addresses and CIDR-style IP address ranges | All	This policy does not allow connections from listed IP addresses and networks. An example of listed IP addresses would be: DenyConnectionList=127.0.0.1,10.0.0.0/8. In this example, connections from the loopback address and the Class A 10 network are not allowed. Refer to “CIDR Address Block and Bits”, on page 28 for information about CIDR. If this policy is set to “All”, connections from all IP addresses and networks are not allowed. If **AllowConnectionList** and **DenyConnectionList** are set at the same time, **AllowConnectionList** takes precedence over the **DenyConnectionList**.
AllowConnectionList	Default: 127.0.0.1 Allowed: A comma delimited list of IP addresses and CIDR-style IP address ranges | All	This policy allows connections from all listed IP addresses and networks. An example of listed IP addresses would be: AllowConnectionList=127.0.0.1,10.0.0.0/8. In this example, connections from the loopback address and the Class A 10 network are allowed. If this policy is set to “All”, connections from all IP addresses and networks are allowed. If **AllowConnectionList** and **DenyConnectionList** are set at the same time, **AllowConnectionList** takes precedence over the **DenyConnectionList**.
MessageServicePort	Default : 2468 Allowed: Any valid port number	This policy specifies the TCP and UDP ports to which the Message Service will bind.

Policy	Values	Description
qip.<*database_name*>.<*database_user*>.password	Default: None Allowed: Alphanumeric - up to 30 characters	The encrypted password that allows a database user to connect to the VitalQIP database running on the database server. The <*database_name*> must be configured on the Sybase or Oracle client, and the <*database_user*> must be the user who owns the VitalQIP database schema. The default user is qipadmin. The password must be encrypted using the **qip-crypt** routine. See “Encrypting your password”, on page 319 for more information about encrypting passwords.
audit.<*database_name*>.<*database_user*>.password	Default: None Allowed: Alphanumeric - up to 30 characters	The encrypted password that allows a database user to connect to the Audit Manager database running on the database server. The <*database_name*> must be configured on the Sybase or Oracle client, and the <*database_user*> must the user who owns the VitalQIP database schema. The default user is lamadmin. The password must be encrypted using the **qip-crypt** routine. See “Encrypting your password”, on page 319 for more information about encrypting passwords.
qip.<*backup_database_name*>.IsBackupOf = <*database_name*>	Default: None Allowed:	To support database failover, this policy defines a backup database server for a previously defined VitalQIP DatabaseServer. The server named in <*backup_database_name*> will be used in the event that <*database_name*> in unreachable.
AuthLibrary	Default: None Allowed: Any valid path name	The path name of a callout library which provides custom administrator authentication for VitalQIP or Audit Manager Administrators. See “Authenticate administrators”, on page 181.

☐

VitalQIP Schedule Service policies

Behavior of the VitalQIP Schedule Service (**qipd**) is controlled by policies in the *qipd.pcy* file or the *qip.pcy* file. The *qipd.pcy* file is used only by the Schedule Service. This file is not created automatically. Since the *qipd.pcy* file is not created automatically, you must create this file manually if you want to use a separate policy file for the service. The *qip.pcy* file is installed automatically during the VitalQIP installation and contains policies for all services. Thus, the *qip.pcy* file can also be modified. By default, *qipd.pcy* is processed. If it is not found, the *qip.pcy* file is processed. Only one policy file is processed.

When modifying the *qip.pcy* file, the **[VitalQIP Schedule Service]** section name must precede the service's policies.

Table 30 describes the policies available in the policy file.

Table 30 VitalQIP Schedule Service (qipd) policies

Policy	Values	Description
Debug	Default: None Allowed: See page 97	This policy sets the debug level. See ""The debug policy", on page 97 for more information about the Debug policy.
DebugFile	Default: *qipd.log* Allowable value: Relative or absolute filename	This policy specifies the filename where the debug output is sent.
ProcessInterval	Default: 60 seconds Allowable value: Numeric	The interval (in seconds) at which the scheduled events are performed. This value specifies the process interval rather than the sleep time between each event.
LicenseInterval	Default: 1800 seconds (30 minutes) Allowed: Numeric	The interval (in seconds) at which the license key is updated. This value must be greater than the **ProcessInterval**. Any value greater than 1 day is reset to 1 day.
LicenseWarnLevel	Default: 95 Allowed: Numeric	This policy specifies the level (as a percent) at which warnings about license key limits should be produced.
ObjectCountInterval	Default: 1800 seconds (30 minutes) Allowed: Numeric	Specifies how often to check if the maximum object count has been exceeded in any organization. Minimum value is 300.
CheckTime	Default: 00:00 Allowed: HH:MM	The time at which **qip-check** and **qip-unlock** are run.

Policy	Values	Description
DumpStatsOnExit	Default: False Allowed: True|False	This policy determines if statistics are dumped when the service exits. The following values can be specified: **True** - statistics are dumped to the event log when the service exits. **False** - statistics are not dumped to the event log when the service exits.
SchedulePassword	Default: Unpublished for security reasons Allowable value: Alphanumeric	The encrypted password of the Schedule Service administrator.
Server	Default: Appropriate database server name Allowable value: Alphanumeric (up to 30 characters)	The name of the database server to use. This is identical to setting the QIPDATASERVER environment variable. However, changing the policy's value overrides the QIPDATASERVER environment variable.
LoginServer	Default: None Allowable value: IP address	The IP address of the host running the Login Service. More than one IP address can be specified to minimize VitalQIP's downtime in the event connectivity is lost with the Login Service. Each IP address must be separated by comma.
SSFailoverDelay	Default: 1h Allowable value: time values with suffixes of `s` (seconds), `m` (minutes), `h` (hours), `d` (days)	This policy specifies the amount of time that this schedule service will allow another to work on a scheduled item before assuming that the other service is no longer functioning. The delay should be no longer than the time required for the longest configured scheduled push. The minimum permissible value is `1m` (one minute).
RemoveStaleSched ServiceInterval	Default: 7 Allowable value: 1 to 2 billion	This policy specifies the number of days all other Schedule Services can remain inactive before they are considered stale and are removed from the Schedule Service Map table. Removal of the Schedule Service from the map table is only done as a custodial action to ensure the table is not cluttered with inactive schedule services.

□

Redundant Schedule Service

VitalQIP users need to eliminate single points of failure within their infrastructure. One such point of failure is the VitalQIP Schedule Service. If this service stops operating for any reason, critical VitalQIP functionality begins to fail because the Schedule Service provides license key updates within the database. The VitalQIP GUI stops functioning if the license key is not updated periodically, and scheduled events such as pushes and moves, as well as network infrastructure changes, are not executed.

To eliminate this single point of failure, two or more VitalQIP Schedule Services are now run simultaneously against the same database.

ScheduleService.id file

The Schedule Service creates *$QIPHOME/etc/ScheduleService.id*. This file contains the database-determined ID. This file is used by Schedule Service to determine its identity with respect to the database. You should not modify this file, since it is created and used internally by the Schedule Service.

If the Schedule Service starts and *$QIPHOME/etc/ScheduleService.id* does not exist, it will have the database generate a new ID,and store that ID in the file.

SSFailoverDelay schedule service policy

The **SSFailoverDelay** policy (on page 136) in the VitalQIP Schedule Service section of *qip.pcy* specifies the amount of time that one schedule service will allow another to work on a scheduled item before assuming that the other service is no longer functioning, and taking over the tasks it was working on. The delay should be longer than the time required for the longest configured scheduled push. Values of less than one minute are set to one minute.

☐

VitalQIP Active Lease Service policies

Behavior of the Active Lease Service (**qip-netd**) is controlled by policies in the *qip-netd.pcy* file or the *qip.pcy* file, located in the *QIPHOME* directory. The *qip-netd.pcy* file is used only by the Active Lease Service. This file is not created automatically. Since the *qip-netd.pcy* file is not created automatically, you must create this file manually if you want to use a separate policy file for the service. The *qip.pcy* file is installed automatically during the VitalQIP installation and contains policies for all services. Thus, the *qip.pcy* file can also be modified. By default, the *qip-netd.pcy* file is processed. If it is not found, the *qip.pcy* file is used. Only one policy file is processed.

When modifying the *qip.pcy* file, the **[VitalQIP Active Lease Service]** section name must precede the service's policies.

Table 31 describes the available policies for the policy file.

Table 31 VitalQIP Active Lease Service (qip-netd) policies

Policy	Values	Description
Debug	Default: None Allowed: See page 97	This policy sets the debug level. See "The debug policy" on page 97 for more information about the Debug policy.
DebugFile	Default: *qip-netd.log* Allowed: Relative or absolute filename	The policy specifies the filename where the debug output is sent.
DenyConnectionList	Default: (None) Allowed: A comma delimited list of IP addresses and CIDR-style IP address ranges|All	This policy does not allow connections from listed IP addresses and networks. An example of listed IP addresses would be: DenyConnectionList=127.0.0.1,10.0.0.0/8. In this example, connections from the loopback address and the Class A 10 network are not allowed. Refer to "CIDR Address Block and Bits", on page 28 for information about CIDR. If this policy is set to "All", connections from all IP addresses and networks are not allowed. If **AllowConnectionList** and **DenyConnectionList** are set at the same time, **AllowConnectionList** takes precedence over the **DenyConnectionList**.

Policy	Values	Description
AllowConnectionList	Default: All Allowed: A comma delimited list of IP addresses and CIDR-style IP address ranges|All	This policy allows connections from all listed IP addresses and networks. An example of listed IP addresses would be: AllowConnectionList=127.0.0.1,10.0.0.0/8. In this example, connections from the loopback address and the Class A 10 network are allowed. If this policy is set to “All”, connections from all IP addresses and networks are allowed. If **AllowConnectionList** and **DenyConnectionList** are set at the same time, **AllowConnectionList** takes precedence over the **DenyConnectionList**.
ListenPort	Default: Ephemeral Allowed: Ephemeral | Any valid port number | Any service name in */etc/services*	This policy specifies which port the service listens for messages. Ephemeral indicates that the service will use a port that is dynamically allocated by the operating system. It will register this port with the local message service. To accept messages from previous releases of VitalQIP, set this policy to the service name `qip_ctl`, or the port number `1096`. Ports are usually less than 32,000.
MessageServicePort	Default : 2468 Allowed: Any valid port number	This policy specifies the TCP and UCP ports to which the Message Service will bind.
ConvertDomainNameCase	Default: None Allowed: None, ToLower, ToUpper	Converts the case of the domain names in the active lease display to upper/lower/no conversion (default)
ConvertHostNameCase	Default: None Allowed: None, ToLower, ToUpper	Converts the case of thehostnames in the active lease display to upper/lower/no conversion (default)
StripTrailingDots	Default: False Allowed: True/False	If true, trailing dots in the domain name are stripped before being sent to the GUI.
ACKPeriod	Default: 0 Allowed:	This policy specifies how often an ACK will be expected when leases are transmitted to the GUI. By default, only the last packet is ACKed.

☐

VitalQIP MS DHCP Monitor Service policies

The VitalQIP MS DHCP Monitor Service's behavior is controlled by policies in the *MSDHCPMonitorServiced.pcy* file or the *qip.pcy* file, located in the *QIPHOME* directory. The *MSDHCPMonitorServiced.pcy* file is used only by the MS DHCP Monitor Service. This file is not created automatically. Since the *MSDHCPMonitorServiced.pcy* file is not created automatically, you must create this file manually if you want to use a separate policy file for the service. The *qip.pcy* file is installed during the VitalQIP installation and contains policies for all services. Thus, the *qip.pcy* file can also be modified. By default, the *MSDHCPMonitorServiced.pcy* file is processed. If it is not found, the *qip.pcy* file is used. Only one policy file is processed.

In the *qip.pcy* file, the **[VitalQIP MS DHCP Monitor Service]** section name must precede the service's policies. Policies are entered in Policy=Value format.

Table 32 describes the policies for the policy file.

Table 32 MS DHCP Monitor Service policies

Policy	Values	Description
Debug	Default: None Allowed: See page 97	This policy sets the debug level. Refer to "The debug policy", on page 97 for more information about the Debug policy.
DebugFile	Default: *MSDHCPMonitorService.log* Allowable value: Relative or absolute filename	This policy specifies the filename where the debug output is sent.
OrgID	Default: None Allowed: Numeric	This policy specifies the organization of the DHCP service.
SleepTime	Default: 60 Allowed: Numeric	The number of seconds waiting between reading the *DhcpSrvLog* file, and thus the update rate to the VitalQIP enterprise server.

Policy	Values	Description
DenyConnectionList	Default: None Allowed: A comma delimited list of IP addresses and CIDR - style IP address ranges | All	This policy does not allow connections from listed IP addresses and networks. An example of listed IP addresses would be: DenyConnectionList=127.0.0.1,10.0.0.0/8. In this example, connections from the loopback address and the Class A 10 network are not allowed. For information about CIDR, see “CIDR Address Block and Bits”, on page 28. If this policy is set to “All”, connections from all IP addresses and networks are not allowed. If **AllowConnectionList** and **DenyConnectionList** are set at the same time, **AllowConnectionList** takes precedence over the **DenyConnectionList**.
AllowConnectionList	Default: 127.0.0.1 Allowed: A comma delimited list of IP addresses and CIDR - style IP address ranges | All	This parameter allows connections from all listed IP addresses and networks. An example of listed IP addresses would be: AllowConnectionList=127.0.0.1,10.0.0.0/8. In this example, connections from the loopback address and the Class A 10 network are allowed. If this policy is set to “All”, connections from all IP addresses and networks are allowed. If **AllowConnectionList** and **DenyConnectionList** are set at the same time, **AllowConnectionList** takes precedence over the **DenyConnectionList**.

☐

VitalQIP Remote Service policies

The behavior of the VitalQIP Remote Service (**qip-rmtd**) is controlled by policies in the *qip.pcy* or *qip-rmtd.pcy* file, located in the *QIPHOME* directory. The *qip-rmtd.pcy* file is used only by the Remote Service. This file is not created automatically. Since the *qip-rmtd.pcy* file is not created automatically, you must create this file manually if you want to use a separate policy file for the service. The *qip.pcy* file is installed automatically during the VitalQIP installation and contains policies for all services. Thus, the *qip.pcy* file can also be modified. By default, the *qip-rmtd.pcy* file is processed. If it is not found, the *qip.pcy* file is used. Only one policy file is processed.

When modifying the *qip.pcy* file, the **[VitalQIP Remote Service]** section name must precede the service's policies.

Important! The Remote Service binds to an ephemeral port and registers itself with the Message Service. All communication with it is done through the Message Service.

Table 33 describes the policies available in the policy file. In addition to **Debug**, the **DebugLevel** policy can be used to debug portions of this policy that contain Java code. Refer to “Java debug policies”, on page 100.

Table 33 VitalQIP Remote Service (qip-rmtd) policies

Policy	Values	Description
Debug	Default: None Allowed: See page 97	This policy sets the debug level. See “The debug policy”, on page 97 for more information about the Debug policy.
DebugFile	Default: *qip-rmtd.log* Allowed: Relative or absolute filename	This policy specifies the filename where the Java debug output is sent.
FileGenerationServer	Default: None Allowable value: RMI Scheduler Service's IP address	This policy specifies where this Remote Service has its files generated.
BackupFileGenerationServer	Default: None Allowed: Numeric	The IP address of the backup File Generation Service that will perform the file generation if the primary File Generation Service cannot be contacted.
SchedulerName	Default: QAPI_Scheduler Allowable value: Any name published in RMI Scheduler Service	This policy determines to which scheduler the Remote Service connects. The Remote Service contacts the RMI Scheduler Service via Java RMI. Multiple schedulers may be started.

Policy	Values	Description
RegistryPort	Default: 1099 Allowed: Any valid port number	The port on which the RMI registry is listening.
MaxFileCopyThreads	Default: 5 Allowed: Any number up to 100	After files are generated on the RMI Scheduler Server, they are copied to the remote server. Multiple copies occur simultaneously (one per thread). This policy specifies the maximum number of threads to start to copy files from the File Generation Server.
FileBufferSize	Default: 0x10000 Allowed: Number in decimal/hexadecimal/ octal format	This policy specifies the number of bytes to copy at one time during file copy. Larger values may be more efficient for extremely large files.
PublishOnMessage Service	Default: True Allowed: True | False	This policy determines if the Remote Service publishes its identity. By default, the Remote Service binds to a temporary port. The following values can be specified: **True** - the Remote Service publishes its identity "PublishID" and assigned port, so the clients can find the Remote Service. **False** - the Remote Service does not publish its identity "PublishID" and assigned port. All Remote Service communication passes through the Message Service.
PublishID	Default: VitalQIP Remote Service Allowed: Any string	Specifies what ID under which to publish the port. ***Do not change this value.***
PortNumber	Default: Ephemeral Allowed: Ephemeral | Any valid port number | Any service name in */etc/services*	This policy specifies on which port the service listens for messages. Ephemeral indicates that the service will use a port that is dynamically allocated by the operating system. It will register this port with the local message service. You can also set this to a port name or number.

Policy	**Values**	**Description**
GenDDNSConfOn Push	Default: One Allowed: None | One | All	This policy defines how many *ddns.conf* files are created when a DNS or DHCP push occurs. The following are valid values: **None** - no *ddns.conf* files are generated when a DNS push occurs **One** - one *ddns.conf* file is generated for the organization of the server being pushed **All** - generates *ddns.conf* files for all organizations
SpecifiedDirOnly	Default: False Allowed: True | False	This policy determines if the Remote Service can be run as non-root user. The following values can be specified: **True** - the Remote Service can be run as a non-root user. When a DNS or DHCP push occurs, ***all*** configuration files are placed in the directory defined for pushed files. Ensure that the user can write to the specified directory. Symbolic links need to be created manually in the appropriate configuration files. **False** - the Remote Service can only be run by the root user. The Remote Service places most configuration files in the directory assigned for pushed files. Some configuration files, such as the *named.conf* file, are placed in directories that are can only be written to by the root user.
RootReadOnlyNamedConf	Default: True Allowed: True | False	This policy determines if users can read the key information in a *named.conf* file during a BIND 9 transfer on a UNIX server. The following values can be specified: **True** - the named.conf file is created in mode 0600 on a UNIX server doing a BIND 9 data transfer. This prevents users from reading any key information. **False** - the *named.conf* file is created in mode 0644 on a UNIX server doing a BIND 9 data transfers. Users can read key information.

Policy	**Values**	**Description**
FailOnFailedUserExit	Default: False Allowed: True | False	This policy determines if a network generation fails when a user exit returns a non-zero value. The following values can be specified: **True** - a network generation fails by having the user exit return a non-zero value. **False** - the Remote Service continues network generation even if the user exit fails.
ErrorRedirectionToken	Default: 2>&1 Allowed: Token	Determines the token to tie STDERR to STDOUT. When the Remote Service runs a user exit, it sends all output from the user exit to a log. It ties STDERR to STDOUT with the 2>&1 operator. If a shell is used that does not understand the 2>&1 operator, specify an appropriate token.
RequireSSLConnection	Default: False Allowed: True | False	This policy determines if the Remote Service accepts data when a scheduler connecting is using a secure socket layer (SSL). The following values can be specified: **True** - the Remote Service only accepts data when the connecting scheduler has “SSL” or “sol” in its name. The RMI Scheduler Service ensures any scheduler with “SSL” in its name is using SSL. **False** - data is accepted when a scheduler without “SSL” in its name connects to the Remote Service.
MessageServicePort	Default: 2468 Allowed: Numeric	Specifies an alternate port to communicate with the Message Service.

☐

VitalQIP RMI QAPI Service policies

Behavior of the RMI QAPI Service (**qapi**) is controlled by policies in the *qip-qapid.pcy* or *qip.pcy* file, located in the *QIPHOME* directory. The *qip-qapid.pcy* file is used only by the RMI QAPI Service. This file is not created automatically. Since the *qapid.pcy* file is not created automatically, you must create this file manually if you want to use a separate policy file for the service. The *qip.pcy* file is installed automatically during the VitalQIP installation and contains policies for all services. Thus, the *qip.pcy* file can also be modified. By default, the *qip-qapid.pcy* file is processed. If it is not found, the *qip.pcy* file is used. This file is found in the *QIPHOME* directory. Only one policy file is processed.

When modifying the *qip.pcy*, the **[VitalQIP RMI QAPI Service]** section name must precede the service's policies.

> **Important!** The RMI QAPI Service uses RMI to communicate with other Services. The default port for RMI is 1099.

Table 34 describes the options available in the *qip-qapid.pcy* file. The **DebugLevel** policy can be used to debug this policy. Refer to “Java debug policies”, on page 100.

Table 34 RMI QAPI Service (qapi) policies

Policy	Values	Description
MaxZipFileSize	Default: 1M Allowed: Numeric	This policy specifies the maximum size (in megabytes) a zip file grows to before a new zip file is created. The absolute upper limit is 256MB. When the resultant file set is larger than **MaxZipFileSize**, the push results are contained in multiple files. Additional threads are used to copy these other files from the FGS. **Important!** It is recommended that **MaxZipFileSize** be set appropriately large to avoid file copier threads because connections back to the FGS will require holes in the firewall/NAT.

Policy	Values	Description
QAPISSL_Scheduler. Secure	Default: True Allowed: True | False	This policy determines if the files are transmitted by secure socket layer (SSL). The following values can be specified: **True** - the files are transmitted using secure socket layer (SSL). The content of the files are encrypted according to values in **QAPISSL_Scheduler.KeysFile**. You must set **QAPISSL_Scheduler.PassPhrase** and **QAPISSL_Scheduler.KeysFile** if this policy is set to True. **False** - the files are not transmitted using SSL. Refer to “Configuring SSL”, on page 203 for more information on how to set up secure file transmission with SSL.
QAPISSL_Scheduler. PassPhrase	Default: None Allowable value: Any string	A secret phrase used for key management. The **PassPhrase** policy is only needed if **QAPISSL_Scheduler.Secure** is set to True. The Qip-crypted form should be entered here.
QAPISSL_Scheduler. KeysFile	Default: *qipkeystore* Allowable value: Any filename	The file where the secure socket layer keys is stored. The **KeysFile** policy is only needed if **QAPISSL_Scheduler.Secure** is set to True.
DebugFilename	Default: *QAPI.%d.log* Allowed: Relative or absolute filename	The filename where Java debug output is sent. Since multiple QAPI services are started, the debug filenames are uniquely qualified by placing the instance ID in the filename. The QAPI Service replaces the “%d” with the instance of the QAPI service.
SchedulerServerIP	Default: 127.0.0.1 Allowed: IP address	This policy specifies where the RMI Scheduler Service can be found.
ServerAddressOverride	Default: IP address Allowed:Any valid IP address	If you have a multi-NIC machine and want the RMI clients (remote services) to use an IP address other than the default, you can specify it with this policy. **Important!** If you use this policy, you should move the **ServerAddressOverride** entry to the global section of *qip.pcy*. This will save you from having to enter the same IP address in the **ServerAddressOverride** policy in the RMI Scheduler section of the file.

□

VitalQIP RMI Scheduler Service policies

Behavior of the RMI Scheduler Service (**qip-rmisched**) is controlled by policies in the *qip-rmisched.pcy* or *qip.pcy* file, located in the *QIPHOME* directory. The *qip-rmisched.pcy* file is used only by the RMI Scheduler Service. This file is not created automatically. Since the *qip-rmisched.pcy* file is not created automatically, you must create this file manually if you want to use a separate policy file for the service. The *qip.pcy* file is installed automatically during the VitalQIP installation and contains policies for all services. Thus, the *qip.pcy* file can also be modified. By default, the *qip-rmisched.pcy* file is processed. If it is not found, the *qip.pcy* file is used. This file is found in the *QIPHOME* directory. Only one policy file is processed.

When modifying the *qip.pcy*, the **[VitalQIP RMI Scheduler Service]** section name must precede the service's policies.

Important! The VitalQIP RMI Scheduler Service uses RMI to communicate with other Services. The default port for RMI is 1099.

Table 35 describes the policies available in the policy file. In addition to **Debug**, the **DebugLevel** policy can be used to debug portions of this policy that contain Java code. Refer to "Java debug policies", on page 100.

Table 35 VitalQIP RMI Scheduler Service (qip-rmisched) policies

Policy	Values	Description
Debug	Default: None Allowed: See page 97	This policy sets the debug level. See "The debug policy", on page 97 for more information about the Debug policy.
DebugFile	Default: *qip-rmisched.log* Allowed: Relative or absolute filename	This policy specifies the filename where the Java debug output is sent.
RegistryPort	Default: 1099 Allowable value: Any valid port number	The port on which the RMI registry is listening. The registered port is 1099.

Policy	**Values**	**Description**
BasePort	Default: 0 Allowed: Any valid port number	If a firewall is in place between a remote server and the File Generation Service, a “base” port can be established for RMI QAPI Services. Each RMI QAPI Service starts and uses an incremental port with the first RMI QAPI Service using the port defined in this policy. The next RMI QAPI Service uses the next higher port, and so on. Suppose the defined base port is 5000. When the first RMI QAPI Service starts, it uses port 5000. When a second RMI QAPI Service service starts, it uses port 5001. If needed, more than one base port can be configured for each QAPI instance in this format: <*SchedulerName*>.BasePort. This is useful when multiple schedulers are configured.
ServerAddressOverride	Default: IP address Allowed:Any valid IP address	If you have a multi-NIC machine and want the RMI clients (remote services) to use an IP address other than the default, you can specify it with this policy.
SchedulerNames	Default: QAPI_Scheduler Allowed: List of strings	This policy specifies which schedulers to start (the initial number of Executors that this Scheduler manages by default). You typically only need to start one, but you can start any number with a comma-separated list of schedulers. The policy **SchedulerName** in the Remote Service (page 142) must match one of the names specified here.
<*SchedulerName*>.BasePort	Default: 0 Allowed: Any valid port number	This policy specifies the base port number that executors will be bound to.
<*SchedulerName*>.Port	Default: 0 Allowed: Any valid port number	This policy specifies to which port the Scheduler should bind. By default, the Scheduler binds to a temporary port but this may cause problems with firewalls. This policy allows you to specify the bound port. When multiple QAPI instances are configured, more than one port can be configured for each QAPI instances.
<*SchedulerName*>.ExecutorName	Default: QAPI Allowed: Any string	This policy specifies the name used when scheduling access. It is typically the same name as the class of the Executor.

Policy	**Values**	**Description**
<*SchedulerName*>.ExecutorClass	Default: QAPI Allowed: Any String	Any Java class is loaded as the executor. This class name is fully qualified or a class that exists in the *com.lucent.qtek.qip.qapi* package. Currently, the only valid Executor is QAPI. ***This policy is for future versions of VitalQIP.***
<*SchedulerName*>.ExecutorCount	Default: 1 Allowed: 0 - 64	This policy specifies the number of Executors to run (that is, the number of Executors that this scheduler will manage by default). This is the number of QAPI services to start and determines the number of pushes that may be performed simultaneously.
<*SchedulerName*>.Max ExecutorCount	Default: 5 Allowed: 1 - 128	This policy specifies the maximum number of Executors to run. Although only <*SchedulerName*>**.ExecutorCount** executors are started initially, the RMI Scheduler Service can start <*SchedulerName*>**.MaxExecutorCount** if the demand requires it.
<*SchedulerName*>.Executor RegistrationTimeout	Default: 5000 Allowed: Number of milliseconds until timeout	After the Scheduler grants the client access to the QAPI Service, that client must register within the specified time, or else the scheduler is free to reschedule that service.
<*SchedulerName*>.Secure	Default: False Allowed: True | False	This policy determines if the files are transmitted by Secure Socket Layer (SSL). The following values can be specified: **True** - the files are transmitted using SSL. The content of the files are encrypted according to values in <*SchedulerName*>**.KeysFile**. You must set <*SchedulerName*>**.PassPhrase** and <*SchedulerName*>**.KeysFile** if this policy is set to True. **False** - the files are not transmitted using SSL. Refer to “Configuring SSL”, on page 203 for more information on how to set up secure file transmission with SSL.
<*SchedulerName*>.PassPhrase	Default: None Allowable value: Any string	A secret phrase used for key management. The **PassPhrase** policy is only needed if <*SchedulerName*>**.Secure** is set to True.
<*SchedulerName*>.KeysFile	Default: None Allowable value: Any filename	The file where the secure socket layer keys is stored. The **KeysFile** policy is only needed if <*SchedulerName*>**.Secure** is set to True.

Policy	Values	Description
<*SchedulerName*>.Virtual MachineParams	Default: None Allowable value: Any parameters which the Java Virtual machine (JVM) accepts	The Scheduler starts a JVM to run the specified class (see (**<***SchedulerName***>.ExecutorClass**). The behavior of this JVM may be modified by supplying additional policies to the JVM.

VitalQIP SSL Tunnel Service policies

To secure VitalQIP message traffic, all communication between services is tunneled through the Message Service, which is located at both ends of any communication path. You can optionally encrypt and decrypt traffic to and from the Message Service with the VitalQIP SSL Tunnel Service (**qip-ssltd**).

SecureIncomingServiceConnections policy

The **SecureIncomingMessageServiceConnections** policy (located in the global section of *qip.pcy*) controls the behavior of both the SSL Tunnel Service and the Message Service. If this policy is set to True, the Message Service only binds to the well known port on the loopback address, and the SSL Tunnel binds to the well known port on the network interface as well as the ephemeral port with which it will communicate with the Message Service. This allows the SSL Tunnel to accept only secure SSL-enabled external connections.

If the **SecureIncomingMessageServiceConnections** policy is set to False (the default value), the Message Service binds to its well known port on the network and loopback interfaces, and the SSL Tunnel does not bind to any well known port. Instead it only binds to the ephemeral port and registers that port with the Message Service so that the Message Service can communicate with it. This allows the Message Service to accept only clear (or plain) text connections.

Table 36 describes the policies available in the *qip.pcy* policy file. In addition to **Debug**, the **DebugLevel** policy can be used to debug portions of this policy that contain Java code. Refer to “Java debug policies”, on page 100.

Table 36 VitalQIP SSL Tunnel Service policies

Policy	Values	Description
Debug	Default: None Allowed:See page 97	This policy sets the debug level. Refer to “The debug policy”, on page 97 for more information about the Debug policy.
DebugFile	Default: *qip-ssltd.log* Allowable value: Relative or absolute filename	This policy specifies the filename where the debug output is sent.
MessageBufferSize	Default: 0x2000 Allowed: Numeric	This policy specifies the maximum number of bytes to read/write between peers at once. All numerical arguments can be in decimal/hexadecimal/octal format. You cannot specify decimal values with a suffix, such as 64K or 1M.

Policy	**Values**	**Description**
PublishID	Default: VitalQIP SSL Tunnel Service Allowed:***Do not change***	This policy specifies under what ID to publish the service port. ***Do not change this policy.***
MessageServicePort	Default: 2468 Allowed: Any valid port number | Any service name in */etc/services*	This policy allows you to specify an alternate port to communicate with the Message Service. If the **SecureIncomingMessageServiceConnections** policy in the global section is set to True, the SSL Tunnel service will listen on the network interfaces on this port.
DenyConnectionList	Default: None Allowed: A comma delimited list of IP addresses and CIDR - style IP address ranges | All	This policy does not allow connections from listed IP addresses and networks. An example of listed IP addresses would be: DenyConnectionList=127.0.0.1,10.0.0.0/8. In this example, connections from the loopback address and the Class A 10 network are not allowed. For information about CIDR, see "CIDR Address Block and Bits", on page 28. If this policy is set to "All", connections from all IP addresses and networks are not allowed. If **AllowConnectionList** and **DenyConnectionList** are set at the same time, **AllowConnectionList** takes precedence over the **DenyConnectionList**.
AllowConnectionList	Default: All Allowed: A comma delimited list of IP addresses and CIDR - style IP address ranges | All	This parameter allows connections from all listed IP addresses and networks. An example of listed IP addresses would be: AllowConnectionList=127.0.0.1,10.0.0.0/8. In this example, connections from the loopback address and the Class A 10 network are allowed. If this policy is set to "All", connections from all IP addresses and networks are allowed. If **AllowConnectionList** and **DenyConnectionList** are set at the same time, **AllowConnectionList** takes precedence over the **DenyConnectionList**.
PassPhrase	Default: <*qip-crypted keystore password*> Allowed:	This policy specifies the password used to protect the keystore. This should be stored as an encrypted and encoded string.

Policy	Values	Description
KeysFile	Default: qipkeystore Allowed:	See “VitalQIP RMI Scheduler Service policies”, on page 148 for instructions on setting up the Keysfile. You will need to create the keys, create the certificate, and then import that certificate into the cacerts keystore. You can use the same generated keys and certificate on all VitalQIP installations.

☐

Sample qip.pcy file

The following *qip.pcy* file shows all the services mentioned in this chapter. Your *qip.pcy* file may differ from the example shown, due to your system configuration.

```
;-----------------------------------------------------------------------------
;                       The VitalQIP Policy File
;
; Any options specified before the first section header are treated as global
; and are passed to every application.
;
; Default or example values are shown for most options.  Setting an option to
; its default value is the same as not setting it at all.
;
; Vim users should set syntax highlighting to 'dosini.vim'
;-----------------------------------------------------------------------------

;-----------------------------------------------------------------------------
;                        Debug Levels Overview
;
; The value for debug can be a string composed of:
;   - A level specified by any one of:
;       LevelCritical LevelError LevelWarning LevelInfo LevelDebug
;
;     Each level is inclusive;
;       Critical -- produces the least debug messages
;       Debug    -- produces the most debug messages
;
;   - A list of layers you would like to debug:
;       Application QSICommon QSINet QIPDB QAPI
;
;   - Additional debug features:
;       FeatureTimestamp  FeatureDeltaTime  FeatureRelativeTime
;       FeatureModule     FeatureBackup     FeatureSeverityStamp
;       FeatureValues     FeatureRotateLogs
;       FeatureProfile    FeatureStackTrace FeatureFullStackTrace
;
; For example:
;   - To enable debugging for the application, and qsicommon for warning
;     messages and above specify debug = LevelWarning Application LibQSICommon
;   - To enable debugging specify: debug = All
;   - To add a timestamp to each debug output line, add FeatureTimestamp
;   - To backup debug files, add FeatureBackup (creates a .bak_##.log of the
;     debug file)
;   - To trace function exits, set FeatureFullStackTrace
;   - To display function execution time, set FeatureProfile
;   - To display values returned from the database in the debug log,
;     set FeatureValues
```

```
;
; ==============================   Debug Features   ===========================
; FeatureTimestamp      -- Prefix each line with time (with ms resolution)
; FeatureDeltaTime      -- Time is displayed as difference from previous message
; FeatureRelativeTime   -- Time is displayed as difference from first message
; FeatureISO8601        -- Prefix each line with time in ISO8601 format
; FeatureSeverityStamp  -- Prefix each line with its severity
; FeatureBackup         -- Backup previous debug file
; FeatureModule         -- Display source filename and line #
; FeatureThreadStamp    -- Prefix each line with its thread or process id
; FeatureStackTrace     -- Display function entrance (MUST be enabled when
;                          submitting bugs)
; FeatureFullStackTrace -- Display function exit as well as entrance
; FeatureProfile        -- Display function execution times (enables
;                          FeatureStackTrace)
; FeatureValues         -- Display values sent to/returned from the database
; FeatureRotateLogs     -- Enables log file rotation (start new log file)
;
; By default, debug files are placed in $QIPHOME/log
;
; DebugRotateMaxDepth specifies the number of backup debug files to maintain.
; You must also enable FeatureBackup for this to take effect.
;DebugRotateMaxDepth = 1
;
; Log file rotation -- If you enable FeatureRotateLogs and either/both of
; DebugRotateFileSize and/or DebugRotateInterval, log files will rotate
; automatically.
;
; You can specify the maximum debug file size with:
;DebugRotateFileSize=0    -- Do not rotate debug file based on size (default)
;DebugRotateFileSize=500K -- Start a new debug file at 500K
;DebugRotateFileSize=50M  -- Start a new debug file at 50 Meg
;
; You can specify how often to rotate the debug file with:
;DebugRotateInterval=0    -- Do not rotate debug file based on time (default)
;DebugRotateInterval=90m  -- Start a new debug file every 90 minutes
;DebugRotateInterval=12h  -- Start a new debug file every 12 hours
;DebugRotateInterval=2d   -- Start a new debug file every 2 days
;
; DebugRotateInterval can be used in conjunction with DebugRotateFileSize.
;
; Debug filenames can contain format characters that expand to date strings:
;      %a -- Abbreviated weekday name
;      %A -- Full weekday name
;      %b -- Abbreviated month name
;      %B -- Full month name
;      %d -- Day of month as decimal number (01 - 31)
;      %H -- Hour in 24-hour format (00 - 23)
;      %j -- Day of year as decimal number (001 - 366)
;      %m -- Month as decimal number (01 - 12)
;      %M -- Minute as decimal number (00 - 59)
;      %S -- Second as decimal number (00 - 61)
;      %w -- Weekday as decimal number (0 - 6; Sunday is 0)
;      %W -- Week of year as decimal number, with Monday as first day of
;            week (00 - 53)
;      %y -- Year without century, as decimal number (00 - 99)
;      %Y -- Year with century, as decimal number
;      %Z -- Time-zone name or abbreviation; no characters if time zone is
;            unknown
;      %% -- Percent sign
;         -- Your platform may support many more format strings.
;            Search google for 'strftime linux|solaris|microsoft|aix|hpux'.
; i.e.
; DebugFile=qip-msgd.%A.%B.%d.log = $QIPHOME/log/qip-msgd.Wednesday.April.2.log
;
; On any platform, except for Windows, you can prefix a filename with the '|'
; character to send debug to a filter program.  Filter programs will be
; restarted if they crash, but some debug output may be lost.
;
;  i.e.  DebugFile="|my_debug_filter"
;==============================================================================
;
;=================   J A V A   D E B U G   P O L I C I E S  ===================
; VitalQIP 6.x integrates Java and C++ in several applications.
; The Java components have a slightly different format for specifying debug
; levels and features.  This may seem to introduce inconsistencies but it
; provides greater flexibility with application debugging.
;
; Instead of Debug, Java components use DebugLevel.  This tag only accepts a
; debug level, not features.
; DebugLevel = { Debug | Info | Warning | Error | Critical | None }
;
; Features are specified separately.  One feature per line set to true or false.
; FeatureTimestamp     = true -- Prefix each line with time (with ms resolution)
; FeatureSeverityStamp = true -- Prefix each line with its severity
; FeatureBackup        = true -- Backup previous debug file
; FeatureThreadStamp   = true -- Prefix each line with its thread id
; VerboseJNI   = true -- display details about what JNI calls the JVM is making
; VerboseClass = true -- displays details about class loader operations
; VerboseGC    = true -- display details about garbage collection operations
; UserJVMOpt   = To specify options to the JVM directly
; LibraryPath  = Specify what directories should be searched for native libraries
;------------------------------------------------------------------------------

; Do not modify the Global Section header.  It is used by the install to
; determine where to insert specific values.
;------------------------------------------------------------------------------
;                                Global Section
;------------------------------------------------------------------------------
; Debug specified in the following are treated as global for services and are
; passed to every service.
;
;debug = None
; debug = All FeatureBackup FeatureTimestamp FeatureStackTrace
;------------------------------------------------------------------------------
; Debug specified in the following are treated as global for CLIs and are
; passed to every cli.
;
```

```
;CLIdebug = None
; CLIdebug = All FeatureBackup FeatureTimestamp
;-----------------------------------------------------------------------------
; Debug specified in the following are treated as global for GUIs and are
; passed to every GUI.
;
;GUIdebug = None
; GUIdebug = All FeatureBackup FeatureTimestamp
;-----------------------------------------------------------------------------
; A comma delimited list of names or addresses of the servers running the
; VitalQIP Login Service.
;
;LoginServer = 127.0.0.1
;-----------------------------------------------------------------------------
; The VitalQIP install no longer places entries into /etc/services.
; Although /etc/services is still scanned, for processes to find port numbers
; other than defaults, this policy file is used.  The hard-coded defaults are
; listed below.  These values are examined before anything in /etc/services.
; The default port mappings are: qip-msgd=2468, qip-dns=3119, qip-qdhcp=2490
; You can override by setting <service name>=<new port #>
; All portname/number policies must reside in the global section.
;
;qip-msgd=
;qip-dns=
;qip-qdhcp=
;-----------------------------------------------------------------------------
; Determines whether the Message Service should accept plaintext connections
; (value of 'false') or only SSL enabled connections via the SSL Tunnel service
; (value of 'true'). When this policy is set to 'true' the SSL Tunnel Service
; binds to the well known port qip-msgd on all interfaces except the loopback.
; Message Service will bind to qip-msgd on the loopback. When this policy is
; set to 'false', the Message Service binds to the well known port qip-msgd on
; all interfaces including the loopback.
;
;SecureIncomingMessageServiceConnections = false
;-----------------------------------------------------------------------------

[IPManage]

; The IPManage user interface looks at the following policies.
;LoginServer = None
;GUIdebug = None

;-----------------------------------------------------------------------------
; Most CLIs have debug tracing enabled using the name of the executable.
; A CLI first looks for a matching policy section name, and then looks at the
; cli debug policy 'CLIdebug'


[qip-syncexternal]

; An example of a generic CLI
;CLIdebug  = None
;DebugFile = qipsyncexternal.log
```

```
;----------------------------------------------------------------------------


[VitalQIP QIP Update Service]

; qip-qipupdated
; override globals
;debug = None
;DebugFile = qip-qipupdated.log

; If Master is true the QIP Update Service (qip-qipupdated) will run an
; instance for each message service connected (UNIX only).  Note that running in
; this mode can generate a substantial load on your database.  If the load is
; too much, you can instead specify a number, indicating the number of processes
; to run.
;Master = false

; If AddIDToLogFilenameOnFork is true, a separate log file with a unique ID
; in its name will be created for each child process that is spawned
; (Master mode; UNIX only).  Otherwise, only one log file will be created
; for all processes.  Use when Master=true, and FeatureRotateLogs is on.
;AddIDToLogFilenameOnFork = true

; With Master mode process limit enabled, 1 process can essentially end up
; handling all of the connections.  This is due to the socket assignment
; policy in UNIX.  UNIX will assign any process which is blocked on a select
; a new socket, and it will typically attempt to limit the number of processes
; which get new connections.  Therefore if Message services attempts to
; connect to the Update Service when it is not busy processing requests, 1 or
; 2 services will get all the connections.  To attempt to work around this you
; can set AcceptSleeptime.   This will cause the Update Service to sleep n
; seconds after each connect, allowing other Update Services to see new
; connections.
;AcceptSleepTime = 0

; If UpdateDNS is true, the  QIP Update Service send messages to the DNS Update
; Service to update DNS servers with information about new leases. The Message
; Service will need a route for messages of type DNSUpdateObject.
;UpdateDNS = true

; Typically DHCP renews need not be sent to DNS unless the client name has
; changed since the lease was granted. Set SendRenewsToDNS to Yes to always
; send DHCP renews to DNS, or to No to never send renews to DNS.  The OnChange
; value specifies that the renew will be sent if the name has changed.
;SendRenewsToDNS = OnChange

; MaxConnections specifies the maximum # of connected message services (UNIX
; only) Solaris and HPUX default to 60, AIX defaults to 2000.  If you intend
; on setting this value above 1000, contact the customer support center.
;MaxConnections = 60

; ConnectQueueDepth specifies how large to make the pending connection queue.
; If the queue is too small and the Update Service is busy in a database
; request, connecting services will start too see 'Connection Refused' errors.
```

```
;ConnectQueueDepth = 10

; If ConvertSpaces is true, spaces in hostnames are converted to dashes before
; they are sent to the QIP database.
;ConvertSpaces = false

; If DropRequests is true, updates are droppped (no processing is done with
; message).  The ACK is sent back to the Message service immediately.
;DropRequests = false

; If DumpStatsOnExit is true, statistics gathered during the lifetime of the
; process are dumped to the event log before the process terminates
;DumpStatsOnExit = false

; If AuditUpdates is true, DHCP update information will be forwarded to the
; local message service for delivery to the Audit Update Service.
;AuditUpdates = false

; Access Controls for Connections
; Access is granted to connect to a server based on the client's IP address.
; To allow all connections unless specifically denied, use the AllowConnection
; ACL.  To deny all access unless specifically allowed, use the DenyConnection
; ACL.  The AllowConnection ACL takes precedence over the Deny ACL if both are
; specified.  The ACLs take a comma delimited list of IP addresses and
; CIDR-style IP address ranges.  For example,
; DenyConnectionList=127.0.0.1,10.0.0.0/8 denies connections from the loopback
; address and the Class A 10 network.  The ACLs also interpret the word All to
; match all IP addresses.
;DenyConnectionList =
;AllowConnectionList = 127.0.0.1

; UpdatePassword specifies the encrypted password of the VitalQIP
; updateservice administrator.
; The default is not published for security reasons.
;UpdatePassword =

; The name of the database server that the service will use to connect to the
; database.  This must match one of the names returned from the login service.
; There is no default.
;Server =

; The name or address of the server running the VitalQIP Login Service.
;LoginServer = 127.0.0.1

; The port number for the service to bind to. Valid values are:
; Ephemeral | Any valid port number |  Any service name in /etc/services
; The default is Ephemeral
;ListenPort = Ephemeral

; Specify whether the service should log events about the status of external
; updates to the system log.
;ExternalStatusToSystemLog = false

; Allows you to specify an alternate port to communicate with Message service.
```

```
;MessageServicePort=2468

; DHCP updates that will be forwarded to a secure zone on a Microsoft DNS
; server can be updated with one of two pricipals: weak or strong. This can
; be controlled using the DNSUpdatePrincipal policy.
; If this value is set to "derived", the QIP Update service will look up the value
; of the "Allow DHCP Clients to Modify Dynamic Object Resource Records" policy on
; the object, subnet, and global levels and derive the correct value.
; If this value is set to "weak", the QIP Update service will always use the weak
; principal.
; If this value is set to "strong", the QIP Update service will always use the
; strong principal.
; Circumventing the database lookup by setting this value to "weak" or "strong"
; may give a perfomance improvement on DHCP updates at the expense of on-the-fly
; principal derivation.
;DNSUpdatePrincipal=derived
;
;-----------------------------------------------------------------------------


[VitalQIP DNS Update Service]

; qip-dnsupdated
; override globals
;debug = None
;DebugFile = qip-dnsudpated.log

;DenyConnectionList  -- see above
;AllowConnectionList -- see above

; The port number for the service to bind to. Valid values are:
; Ephemeral | Any valid port number |  Any service name in /etc/services
; The default is Ephemeral
;
;ListenPort = Ephemeral

; If Master is true the DNS Update Service (qip-dnsupdated) will run an instance
; for each message service connected (UNIX only)  Note that running in this mode
; can generate a substantial load on your system.  If the load is too much
; you can instead specify a number, indicating the number of processes to run.
;Master = false

; If AddIDToLogFilenameOnFork is true, a separate log file with a unique ID
; in its name will be created for each child process that is spawned
; (Master mode; UNIX only).  Otherwise, only one log file will be created
; for all processes
;AddIDToLogFilenameOnFork = true

; With Master mode process limit enabled, a process can essentially end up
; handling all of the connections.  This is due to the socket assignment
; policy in UNIX.  UNIX will assign any process which is blocked on a select
; a new socket, and it will typically attempt to limit the number of processes
; which get new connections.  Therefore if Message services attempt to
; connect to the Update Service when it is not busy processing requests, 1 or
; 2 services will get all the connections.  To attempt to work around this you
; can set AcceptSleeptime.   This will cause the DNS Update Service to sleep n
; seconds after each connect, allowing other Update Services to see new
; connections.
;AcceptSleepTime=0

; Since DNS updates are done using UDP you may need to configure
; time-out/attempts.  DDNS_Timeout unit is seconds.
; Setting DDNS_Attempts to 0 will turn off DNS updates.
; DDNS_Attempts was previously called DDNS_Retries.
;DDNS_Attempts = 1
;DDNS_Timeout = 5

; DNS TTL values are usually set to the duration of the DHCP lease, you can
; override that value by specifying DDNS_TTL (in seconds)
;DDNS_TTL = 0

; MaxConnections specifies the maximum # of connected message services (UNIX
; only) Solaris and HPUX default to 60, AIX defaults to 2000.  If you intend
; on setting this value above 1000, contact the customer support center.
;MaxConnections = 60

; Typically DHCP renews need not be sent to DNS (the information is already
; there).  The default is to NOT send it, set SendRenewsToDNS to Yes to enable.
; SendRenewsToDNS applies only to messages from the DHCP server, not the
; VitalQIP QIP Update Service.
;SendRenewsToDNS=No

; ConnectQueueDepth specifies how large to make the pending connection queue.
; If the queue is too small and the Update Service is busy in a database
; request, connecting services will start to see 'Connection Refused' errors.
;ConnectQueueDepth = 10

; If ConvertSpaces is true, spaces in hostnames are converted to dashes before
; they are sent to DNS.
;ConvertSpaces = false

; If DumpStatsOnExit is true, statistics gathered during the lifetime of the
; process are dumped to the event log before the process terminates.
;DumpStatsOnExit = false

; Send secure updates to server/zone combinations that are configured for
; secure updates
;DoSecureUpdates = true

; The name of the Kerberos principal that the Service will register as.
; (Unix only)
;KerberosPrincipal =

; The number of times that the DNS Update Service will try a GSSAPI
; operation before reporting a failure.
;GSSAPIImmediateRetries = 1

; The number of seconds after a failure the DNS Update Service will wait
; before attempting to send another secure update to a DNS server.
;GSSAPIRetryDelay = 900

; When true the DNS server will perform its own Kerberos initialization.
; (Unix only)
;DoKinit = false

; The keytab file to use when performing Kerberos initialization.
; (Unix only)
;KeytabPath = /etc/krb5.keytab

; When set to a non 0 value, the DNS Update Service will periodically
; run the qip-genddnsconfs cli.  This value determines the number
; of seconds between invocations.
;DDNSGenerateSleep = 0

; When the Master policy is set to True or a number, this policy
; determines how long to pause between sending signals to child
; processes.  Set this policy to spread out the load if the CPU
; spikes when reading newly available ddns.conf files.
; This value is the number of seconds to pause between sending
; signals.  It does not apply to SIGTERM.
;PauseBetweenChildren = 0
;-----------------------------------------------------------------------------


[VitalQIP MS DNS Update Service]

; override globals
;
; Do not turn on FeatureStackTrace for this service.
;
; If you have feature stacktrace turned on globally, you need to turn it off
; here. Feature Stacktrace will cause unexpected behavior in this service.
; The following "debug = None" statement will ensure that stacktrace does not
; accidentally get turned on for this service if it is turned on in the global
; section. If you wish to turn on debug for this service, you may use any feature
; EXCEPT FeatureStackTrace. For example, "debug = all FeatureTimeStamp" is a valid
; debug statement for this service.
debug = None
;debug = all FeatureTimeStamp
;DebugFile = qip-msdnsupdated.log

; The port number for the service to bind to. Valid values are:
; Ephemeral | Any valid port number |  Any service name in /etc/services
; The default is Ephemeral
;ListenPort = Ephemeral

;DenyConnectionList  -- see above
;AllowConnectionList -- see above

; The number of resource record updates that can fail per zone
; before a push failure is reported to the caller.
;FailureThreshold = 5
```

```
; The number of times the service will try to add a resource
; record before considering the resource record update a
; failure. The service will allow FailureThreshold resource
; record failures per zone before reporting a push failure to
; the caller.
;UpdateRetry = 5

; The number of milliseconds that the service will delay
; after failing to add a resource record before attempting
; to re-apply the update.  The update delay allows the DNS
; server to process previously applied updates.
;UpdateDelay = 50

; The number of records that the service will cache in
; memory before sending a packet to the caller.  This
; parameter is used when the service proxies a zone transfer
; request to the DNS server for the caller.  This parameter
; can affect the performance of the zone transfer.
;RecsPerPacket = 100

; Number of requests to queue for each activity type before
; rejecting them. You will probably have lots of updates happening
; at one time. This is ok, so give it a large Q size.
;UpdateQueueLength = 10000

; ProcessFiles happens on a push to server and AXFR happens on a
; push to screen, local or server through a proxy. So you should
; only need small values here.
;ProcessFilesQueueLength = 5
;AXFRQueueLength = 5

; Number of threads to accept update connection requests.
;ThreadPoolCount = 50
;---------------------------------------------------------------------------


[VitalQIP Message Service]

; qip-msgd
; override globals
;debug = None
;DebugFile = qip-msgd.log

;DenyConnectionList  -- see above
;AllowConnectionList -- see above

; AdditionalIPList specifies a comma separated list of IP addresses that should
; be treated as local addresses.  This is used primarily for NAT separated
; machines trying to connect to a remote Message service.  You can specify the
; NAT assigned address to be treated as an address local to this machine.
;AdditionalIPList=
; MaxConnections specifies the maximum # of connected message services (UNIX
; only) Solaris and HPUX default to 60, AIX defaults to 2000.  If you intend
; on setting this value above 1000, contact the customer support center.
;MaxConnections = 60

; When a connection is lost, the Message Service will wait before attempting
; to re-establish a connection.  This wait period is configured according to
; ReConnectTimeout and a random time that is up to 1/6 as large as the
; re-connect timeout i.e. 180 + (0...30).
;ReConnectTimeout = 180

; If DumpStatsOnExit is true, statistics gathered during the lifetime of the
; process are dumped to the event log before the process terminates.
;DumpStatsOnExit = false

; Specifies both the TCP and UDP ports that the Message Service will bind to.
; You should use this instead of ListenPort/AcceptPort.
;MessageServicePort=2468

; This policy truncates messages larger than specified number of bytes.
; This policy is useful for handling larger messages.  This is used for UDP
; messages only. Valid values are numbers between 4096 and 65536 bytes.
;ReceiveBufferSize = 16384

; 6.0 disables UDP based control messages by default.  Some software still
; requires this feature.
;AllowUDPControlMessages=false

; This policy, if set, turns on periodic statistics logging to the system log.
; This policy is useful for determining whether message queues are growing,
; which may indicate that Distributed Services should be installed.  The
; parameter specifies the minimum number of seconds the message service will
; wait between logging statistics.  If the period is set to 0, then no periodic
; logging will occur. The default is 0.
;StatsLogPeriod=0

; When this policy is set to true, the Message Service will communicate with
; Proxy Message Services via the SSL Tunnel Service. The SSL Tunnel Service
; provides transport authentication and encryption.
;SecureOutgoingProxyConnections = false

; MessageQueue/MessageRoute/SecureMessageRoute
; Each message type can have 1 MessageQueue and any number of MessageRoutes
; associated with it.  MessageQueue defines how to store the messages before
; they are routed.  MessageRoute defines how and where to route
; messages.  Use these options when configuring the Message Service:
```

```
;- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
;-                       M e s s a g e Q u e u e
;- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
; MessageQueue = {id}:{memq}:{diskq}:{ttl}:{warn limit}:{message dir}
; MessageQueue defines characteristics of how messages are stored in
; memory and on disk.  MessageQueue is not a required.  If no
; MessageQueue is defined for a message id, the queue will be memory
; only with no limit.
;
; ex. MessageQueue=1:32000:65000:3600:512:  -- will define a queue for
; messages of type 1 with a memory limit of 32000 nodes, a disk limit of
; 65000 nodes, nodes will be deleted after being on the queue for an hour
; if not already sent to every destination, a warning will be logged when
; 512 nodes have been queued, and it will store disk messages in the default
; location of $QIPHOME/log/messages/001

;- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
;-                       M e s s a g e R o u t e
;- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
; MessageRoute = {id}:{flags}:{ACK time-out}:{Description}:{service
 name}:{IP}[*#][:[{service name}]:{IP}[*#]...]
; MessageRoute can be specified any number of times for a message id.
; Each MessageRoute entry adds a route (i.e. destination) for a particular
; type of message (message id) received by the Message Service.  Currently
; there are 4 known types (1 - DHCP), (3 - Audit), (9 - DNSUpdateObject) and
; (10 - DNSUpdateRR) but the Message Service is capable of handling any type.
;
; 6.2 adds 2 changes to MessageRoute; a destination multiplier (10.1.2.3*15) 15
 connections to 10.1.2.3, this
; functionality is only usefull when load-balancing, and Service Names instead of port
 names/numbers.
; A Service Name can be one of the VitalQIP services that accepts routed messages.
; Currently only "VitalQIP QIP Update Service", "VitalQIP DNS Update Service",
; and "VitalQIP Message Service" can be routed to.
;
; Service Name may also be a port name or number if you have a non-QIP or
; legacy QIP service that you wish to route traffic to.
;
; ex. MessageRoute=1:A:0:My Update Service:VitalQIP QIP Update
 Service:10.2.3.4,10.4.3.2
; will configure a route for message id 1, sending messages independently
; of other routes (asynchronously), no time-out will occur while waiting for
; ACKs from the destination My Update Service on both servers 10.2.3.4 and 10.4.3.2.
;
; If another route of type 1 is added, messages on queue '1' will be sent there
; too. ex:
; MessageRoute=1:AL:0:My Other VitalQIP QIP Update Service:VitalQIP QIP Update
 Service:10.4.3.2,10.2.3.4
; this will send messages to My Other VitalQIP QIP Update Service on 10.4.3.2 or
; 10.2.3.4 (if 10.4.3.2 is down) The next message will not be sent until the
; first destination has processed the current message.
;
; SecureMessageRoute = as in Message Route above.
; Instead of making connections directly, SecureMessageRoutes make
; connections through the SSL Tunnel Service to make use of message
; traffic authentication and encryption.
;- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

;- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
;-      I  D o n ' t   C a r e   J u s t   S h o w   M e
;- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
; To achieve each of the following, use the example Message Routes to create
; your own implementation specific policies:

; To update the QIP Database only:
;MessageRoute=1:A:0:QIP Update Service:VitalQIP QIP Update
 Service:<ip_address_of_update_service>

; To update the DNS Running Locally:
;MessageRoute=DNSUpdateObject:A:0:DNS Update Service:VitalQIP DNS Update
 Service:127.0.0.1
;MessageRoute=DNSUpdateRR:A:0:DNS Update Service:VitalQIP DNS Update Service:127.0.0.1

; To update both the QIP Database and DNS Running Locally:
;MessageRoute=DHCP:A:0:QIP Update Service (DHCP):VitalQIP QIP Update Service:127.0.0.1
;MessageRoute=DNSUpdateRR:A:0:QIP Update Service (Update RR):VitalQIP QIP Update
 Service:127.0.0.1
;MessageRoute=DNSUpdateObject:A:0:DNS Update Service (Object):VitalQIP DNS Update
 Service:127.0.0.1
;MessageRoute=DNSUpdateRR:A:0:DNS Update Service (Update RR):VitalQIP DNS Update
 Service:127.0.0.1

; To update both the QIP Database and DNS Running on Enterprise Server:
;MessageRoute=DHCP:A:0:QIP Update Service (DHCP):VitalQIP QIP Update
 Service:<ip_address_of_qip_update_service>
;MessageRoute=DNSUpdateRR:A:0:QIP Update Service (Update RR):VitalQIP QIP Update
 Service:<ip_address_of_qip_update_service>
;MessageRoute=DNSUpdateObject:A:0:DNS Update Service (Object):VitalQIP DNS Update
 Service:<ip_address_of_dns_update_service>
;MessageRoute=DNSUpdateRR:A:0:DNS Update Service (Update RR):VitalQIP DNS Update
 Service:<ip_address_of_dns_update_service>

; Backup, updates DNS if connection to QIP Update Service is down:
; (attempt to reconnect to qip update service with A.R in flags)
;MessageRoute=1:A.R:0:QIP Update Service:VitalQIP QIP Update
 Service:<ip_address_of_qip_update_service>:VitalQIP DNS Update Service:127.0.0.1

; Make 20 load-balanced connections to a DNS update service at 10.1.2.3 and another 20
 to 10.1.2.4
;MessageRoute=9:A.L:0:DNS Update Service:VitalQIP DNS Update
 Service:10.1.2.3*20:VitalQIP DNS Update Service:10.1.2.4*20
;------------------------------------------------------------------------------
```

```
[VitalQIP Login Service]

; qip-logind
; override globals
;debug = None
;DebugFile = qip-logind.log

; The port number for the service to bind to. Valid values are:
; Ephemeral | Any valid port number |  Any service name in /etc/services
; The default is Ephemeral
;ListenPort = Ephemeral

; When the Login Service starts, it publishes with the Message Service.  If the
; Message Service is restarted, the Login Service will need to re-connect to
; re-publish.  This parameter specifies the max time in seconds that the Login
; Service will wait between connect attempts.  It can be short since the
; connection is local.
;ReConnectTimeout = 5

; When DropRequests is true, the service will not repsond to requests.
;DropRequests=false
;DumpStatsOnExit     -- see above
;ConnectQueueDepth   -- see above
;MaxConnections      -- see above
;DenyConnectionList  -- see above
;AllowConnectionList -- see above
;MessageServicePort = 2468

; The encrypted password that allows <DatabaseUser> to connect to the VitalQIP
; database running on <DatabaseServer>.  The database server name
; <DatabaseServer> should be configured on either Sybase or Oracle, and
; <DatabaseUser> should be the database user who owns the VitalQIP schema.  The
; encrypted password should be generated by the the qip-crypt CLI.  This policy
; can be specified multiple times for multiple <DatabaseServer>, <DatabaseUser>
; and <Password> combinations.  All listed QIP policies are passed to
; requesting VitalQIP clients.  The client may select the database to connect
; to by:
;      looking at the locally configured database environment,
;      looking at environment variables,
;      looking at local policy files, or
;      asking the user.
;
;QIP.<DatabaseServer>.<DatabaseUser>.Password =

; As QIP.<DatabaseServer>.<DatabaseUser>.Password, but for Audit Manager.
;Audit.<DatabaseServer>.<DatabaseUser>.Password =

; To define a backup database server for a previously defined DatabaseServer
; use the following config.  The server named in <BackupServer> will be used
; in the event that <DatabaseServer> in unreachable.
;QIP.<BackupServer>.IsBackupOf = <DatabaseServer>

; The pathname of a callout library which provides custom administrator
; authentication for VitalQIP or Audit Manager administrators.
; See the Administrators Reference Guide for more.
;AuthLibrary =
;-------------------------------------------------------------------------------


[QIPAuthCallout]

; The severity level at which successful authentications are written to
; the system log.  ( None | Information | Warning | Error )
;SuccessLogLevel = Information

; The severity level at which authentication attempts are written to
; the system log.   Authentication attempts are only written to the
; system log when the AuthenticationCli policy is not set.
; ( None | Information | Warning | Error )
;AttemptLogLevel = Information

; The severity level at which unsuccessful authentications are written to
; the system log.
; ( None | Information | Warning | Error )
;FailureLogLevel = Error

; The path name of a command to run to authenticate the administrator.
; Relative path names are resolved using the $PATH environment variable.
; The CLI should return 0 to allow the user to log into VitalQIP, 1 to
; defer authentication to VitalQIP, or 2 to deny the authentication.
; When this policy is not set, the administrator will authenticate
; against the password stored in the VitalQIP database.
; See the Administrator Reference Guide for a description of input arguments
; and expected return value.
;AuthenticationCli =

; The text of a message to be presented to the administrator in the case
; of failure.  The message will be displayed by the administrator's UI.
; Up to 253 characters.
;FailureMessage =

; The number of seconds the login service will tell the client to delay
; before notifying the administrator of an authentication failure.
; Any positive number.
;DelayOnFailure = 0
;-------------------------------------------------------------------------------


[VitalQIP Schedule Service]

; qipd
; override globals
;debug = None
;DebugFile = qipd.log

; ProcessInterval specifies how often to check for events.
;ProcessInterval = 60
```

```
; LicenseInterval specifies how often to update the license key.
;LicenseInterval = 1800

; LicenseWarnLevel specifies the level (as a percent) at which warnings about
; license key limits should be produced.
;LicenseWarnLevel = 95

; ObjectCountInterval specifies how often to check if the maximum object
; count has been exceeded in any organization.  Minimum value is 300.
;ObjectCountInterval = 1800

; Checktime specifies when to schedule cleanup operations.
;Checktime = 00:00

; If DumpStatsOnExit is true, statistics gathered during the lifetime of the
; process are dumped to the event log before the process terminates.
;DumpStatsOnExit = false

; SchedulePassword specifies the encrypted password of the VitalQIP
; scheduleservice administrator.
; The default is not published for security reasons.
;SchedulePassword =

; The name of the database server that the service will use to connect to the
; database.  This must match one of the names returned from the login service.
; There is no default.
;Server =

; The name or address of the server running the VitalQIP Login Service.
;LoginServer = 127.0.0.1

; The following policies allow time suffixes of m - minute, h - hour,
; d - day, w - week, (no suffix indicates seconds)

; This policy specifies the amount of time that this schedule service will
; allow another to work on a scheduled item before assuming that the other
; service is no longer functioning.  The minimum allowable time is 1 hour.
;SSFailoverDelay = 3h

; This policy specifies the amount of time the service will attempt to
; re-establish connection to the current database before attempting a
; connection to any configured backup.
; DBFailoverDelay = 5m

; This policy specifies the interval that schedule services can be inactive
; before being considered stale and removed from the Schedule Map table.
;RemoveStaleSchedServiceInterval = 7d
;-------------------------------------------------------------------------------
```

```
[VitalQIP Active Lease Service]

; qip-netd
; override globals
;debug = None
;DebugFile = qip-netd.log

;DenyConnectionList  -- see above
;AllowConnectionList -- see above

; The port number for the service to bind to. Valid values are:
; Ephemeral | Any valid port number |  Any service name in /etc/services
; The default is Ephemeral
;ListenPort = Ephemeral

; Allows you to specify an alternate port to communicate with Message Service.
;MessageServicePort=2468

; Specifies how to do case conversion of domain names before they are sent to
; the GUI.  Valid values are None (no conversion), ToLower (convert to lower
; case), ToUpper (convert hostnames to upper case)
;ConvertDomainNameCase=None

; Specifies how to do case conversion of hostnames before they are sent to the
; GUI.  Valid values are None (no conversion), ToLower (convert to lower case),
; ToUpper (convert hostnames to upper case)
;ConvertHostNameCase=None

; If true, trailing dots in the Domain name are stripped before being sent to
; the GUI.
;StripTrailingDots=false

; Specifies how often an ACK will be expected when transmitting leases to the
; GUI.  By default, only the last packet is ACK'd.
;ACKPeriod=0
;---------------------------------------------------------------------------


[VitalQIP Domain Controller Logon Audit Service]
; qip-dclas
; override globals
;debug = None
;DebugFile = qip-dclas.log

; Message server port - overrides qip-msgd.
;Message_Server_Address = 127.0.0.1
;SendPort =

; Organization ID
;OrgID = 1

; Send Logon Audit packets to the message service
;SendLogon = true
```

```
; Send Logout Audit packets to the message service
;SendLogout = true

; The address that this domain controller will identify itself as.
; By default, this is determined by gethostbyname(gethostname()).
;Domain_Controller_Address=

; Resolve the client's MAC address through NetBIOS.
;Resolve_Client_MAC = true

; Resolve the client's IP address through Winsock2.
;Resolve_Client_IP = true
;------------------------------------------------------------------------------


[VitalQIP Kerberos Domain Controller Logon Audit Service]
; qip-kdclas
; override globals
;debug = None
;DebugFile = qip-kdclas.log

; Message server port - overrides qip-msgd.
;Message_Server_Address = 127.0.0.1
;SendPort =

; Organization ID
;OrgID = 1

; Send Logon Audit packets to the message service
;SendLogon = true

; Send Logout Audit packets to the message service
;SendLogout = true

; The address that this domain controller will identify itself as.
; By default, this is determined by gethostbyname(gethostname()).
;Domain_Controller_Address=

; Resolve the client's MAC address through NetBIOS.
;Resolve_Client_MAC = true

; Resolve the client's IP address through Winsock2.
;Resolve_Client_IP = true
;------------------------------------------------------------------------------


[Lucent TFTP Service]

; luc-tfptd
; override globals
;debug = None
;
; DebugFile is the name of the file where debug/trace/log output will go.
; When not fully-qualified it is relative to $QIPHOME/log.
; Default is luc-tftpd.log.
;DebugFile = luc-tfptd.log

; Directory where files are read and written.  If not fully-qualified,
; the directory is relative to $QIPHOME.  Default is tftpboot.
;Directory = tftpboot

; DumpStatsOnExit will dump statistics in the syslog (UNIX)
; or the Event Log (Windows) upon exiting.  Default is false.
;DumpStatsOnExit = false

; Setting LingerConnections=true will keep connection alive
; after last packet is sent.  Default is false.
;LingerConnections = false

; ListenPort is the port to listen for TFTP requests.  Typically uses
; the 'tftp' value defined in the /etc/services file (UNIX) or the
; %systemroot%\system32\drivers\etc\services file (Windows).
; Default is tftp (typically this is port 69).
;ListenPort = tftp

; Timeout is the number of seconds the service will wait for a response
; before deciding that the client is not responding.  Default is 5.
;Timeout = 5

; Access Controls for Read and Write
; Access is granted to read or write based on the client's IP address.
; To allow all access unless specifically denied, use the Allow ACL.
; To deny all access unless specifically allowed, use the Deny ACL.
; The Allow ACL takes precedence over the Deny ACL if both are specified.
; The ACLs take a comma delimited list of IP addresses and CIDR-style
; IP address ranges.  For example,  WriteDenyList=127.0.0.1,8/10.0.0.0
; denies write access to the loopback address and the Class A 10 network.
; The ACLs also interpret the word All to match all IP addresses.
;ReadDenyList =
;WriteDenyList =
;ReadAllowList = All
;WriteAllowList = All
;-----------------------------------------------------------------------------


[VitalQIP MS DHCP Monitor Service]

; MSDHCPMonitorService
; override globals
;debug = None
;DebugFile = MSDHCPMonitorService.log

;OrgID = 1
;SleepTime = 60
;DenyConnectionList  -- see above
;AllowConnectionList -- see above
;-----------------------------------------------------------------------------
```

```
[VitalQIP IBM DHCP Monitor Service]
; qip-ibmdhcpd
;debug = None
;DebugFile = qip-ibmdhcpd.log
;orgid = 1

; The port number for the service to bind to. Valid values are:
; Ephemeral | Any valid port number |  Any service name in /etc/services
; The default is Ephemeral
;ListenPort = Ephemeral

;DenyConnectionList  -- see above
;AllowConnectionList -- see above
;-----------------------------------------------------------------------------


[VitalQIP Remote Service]

; qip-rmtd
; override globals
;debug = None

; Even though this service starts as a C++ application, most of the work is done
; in Java.  The Java debug level is set according to the C++ level but you may
; override it by setting DebugLevel.
;DebugLevel = NONE

;DebugFile = qip-rmtd.log
;FeatureTimestamp = false

; If the FGS can not be contacted, an attempt to the Backup FGS will be made.
;FileGenerationServer = IP Address of FGS;
;BackupFileGenerationServer =

; The name of the service that RMI Scheduler Service published.
;SchedulerName = QAPI_Scheduler

; Specifies what port the Remote Service should look for the RMI Registry on.
;RegistryPort = 1099

; The maximum number of threads to start to copy files from the FGS
;MaxFileCopyThreads = 5

; The max number of bytes to read from the FGS at once.   All numerical
; arguments can be in decimal/hex/octal format.
;FileBufferSize = 0x10000

; Specifies if we should publish our identity on the Message Service.
;PublishOnMessageService = true

; Specifies what id to publish our port under.  Do not change this.
;PublishID = VitalQIP Remote Service
```

```
; The port number for the service to bind to. Valid values are:
; Ephemeral | Any valid port number |  Any service name in /etc/services
; The default is Ephemeral
;PortNumber = Ephemeral

; Valid values for this option are:
;   none -- No ddns.conf files will be generated
;   one  -- Only generate the ddns.conf file for the org of the server being
;           pushed
;   all  -- Generate ddns.confs for all orgs
;GenDDNSConfOnPush = one

; By default the Remote Service will place most files in the 'Push Directory'
; but some files like named.conf are placed in directories writeable by root
; only.  If you'd like to run the Remote Service as a non-root user, you'll
; need to set SpecifiedDirOnly to true, causing files to be placed ONLY in the
; push directory.  Insure that the user can write to the specifed directory.
; You'll need to create symbolic links to the appropriate files manually.
;SpecifiedDirOnly = false

; If set to true, on UNIX, for a BIND 9 push, named.conf and rndc.conf will
; be mode 0600 to prevent users from reading any key info, otherwise the file
; will be 0644.
;RootReadOnlyNamedConf=true

; If set to true you can cause the push to fail by having the user-exit return
; a non-zero value.  Otherwise the Remote Service will continue with the push
; even if the user-exit fails.
;FailOnFailedUserExit=false

; When the Remote Service runs user-exits it sends all output from the user
; exit to the log.  It ties stderr to stdout with the 2>&1 operator.  If you
; are using a shell that does not understand this, specify an appropriate token.
;ErrorRedirectionToken=2>&1

; If set to true, the Remote Service will only accept a push when the
; scheduler it is connecting to has 'SSL' or 'ssl' within its name.  The RMI
; Scheduler Service insures any scheduler named with SSL in its name uses SSL.
;RequireSSLConnection=false

; Allows you to specify an alternate port to communicate with Message Service.
;MessageServicePort=2468
;-------------------------------------------------------------------------------


[VitalQIP RMI QAPI Service]

; The QAPI Service starts as a Java application.  It uses Java Debug policies.

; override globals
;DebugLevel = NONE
;FeatureTimestamp = false
;FeatureSeverityStamp = false
;FeatureThreadStamp = false
```

```
;FeatureProfile = false

; Specifies the maximum size a zip file can grow to.  When the limit is reached,
; a new zip file is created.  When disk space is extremely limited on the
; Remote Server it may be necessary to use smaller zip files.
;MaxZipFileSize = 1M

; If you are using SSL, you will need to configure the PassPhrase and KeysFile
; for qapi as well.  If you only have one RMI scheduler you can use a shorthand
; for the following options by omitting the scheduler name prefix and the dot.
;QAPISSL_Scheduler.Secure = true
;QAPISSL_Scheduler.PassPhrase = <qip-crypted keystore password>
;QAPISSL_Scheduler.KeysFile = qipkeystore

; Since multiple instances of QAPI will run, separate debug files are needed.
; The %d in the DebugFilename is replaced with the instance of QAPI.
;DebugFilename = QAPI_Scheduler.QAPI.%d.log

; This policy tells the QAPI server where to find the Scheduler
;SchedulerServerIP = 127.0.0.1

; If you have a multi-nic machine and want the RMI clients (Remote Services) to
; use an IP Address other than the default, you can specify it here.  You can
; move this policy to the global section.
;ServerAddressOverride={IP Address to use in RMI communication}
;-----------------------------------------------------------------------------


[VitalQIP RMI Scheduler Service]

; qip-rmisched
; override globals
;debug = None
; Even though this service starts as a C++ application, most of the work is done
; in Java.  The Java debug level is set according to the C++ level but you may
; override it by setting DebugLevel.
;DebugLevel = NONE
;DebugFile = qip-rmisched.log
;RegistryPort = 1099

; If you have a firewall between your Remote Servers and your FGS, you'll likely
; want to configure a BasePort for qapi instances.  The first instance of qapi
; will use the port you specify here.  Each additional instance will use a port
; number set by this number summed with it's instance number.
;BasePort = 0

; If you have a multi-nic machine and want the RMI clients (Remote Services) to
; use an IP Address other than the default, you can specify it here.
;ServerAddressOverride={IP Address to use in RMI communication}

; You can specify the name(s) of the scheduler(s) you would like to start.  You
; will typically only need to start one, but you can start any number by
; specifying a comma separated list of schedulers.  Options for each Scheduler
; must be prefixed with the name of the scheduler they pertain to.  'Executor'
; is the general term for what the RMI Scheduler Service schedules.  For most
; configs this will be the QAPI processes.
;SchedulerNames = QAPI_Scheduler

; Specifies the base port number that executors will be bind to.
;QAPI_Scheduler.BasePort = 0
;QAPI_Scheduler.Port = 0

; ExecutorName specifies the name to register with RMI for the loaded class.
;QAPI_Scheduler.ExecutorName = QAPI
; ExecutorClass specifies the name of the class to load.  It
; actually defaults to the same value that ExecutorName was set to.
;QAPI_Scheduler.ExecutorClass = QAPI

; Start 1 instance of QAPI initially (can be 0)
;QAPI_Scheduler.ExecutorCount = 1

; Specifies the maximum number of Executors to start (started as needed).
;QAPI_Scheduler.MaxExecutorCount = 5

; 5000ms time-out  Client must register with Scheduler within 5000ms (5 sec)
;QAPI_Scheduler.ExecutorRegistrationTimeout = 5000

; There is no default password.  This must be set to the qip-crypted
; keystore password entered in step 1 below.
;QAPI_Scheduler.Secure = false
;QAPI_Scheduler.PassPhrase =

; There is no default keysfile. This must be set to the keystore
; file entered in step 1 below, for example qipkeystore.
; Releative path names are relative to $QIPHOME.
;QAPI_Scheduler.KeysFile =
;
; See: http://java.sun.com/j2ee/1.4/docs/tutorial/doc/index.html for
; instructions on creating a keys file.  In summary you will need to
;
; 1) Create a set of keys for vitalqip running on this machine.
;    You will be prompted for a keystore password - this should be
;    the NON-qip-crypted passphrase. When you enter the password
;    into the PassPhrase policy above and in the SSL Tunnel section
;    of the qip.pcy file, it must be entered as the qip-crypted form.
;    Be sure to also enter the keystore file name in the KeysFile
;    parameter above and in the SSL Tunnel Service section of the pcy
;    file.
;    The certificate will be valid for ten years.
;
;    $QIPHOME/jre/bin/keytool -genkey -keyalg RSA
;                     -alias vitalqip
;              -validity 3652
;                     -keystore $QIPHOME/qipkeystore
;
; 2) Create a self signed certificate file for vitalqip.
;
;    $QIPHOME/jre/bin/keytool -export -alias vitalqip
;           -keystore $QIPHOME/qipkeystore
;           -file $QIPHOME/vitalqip.cer
;
; 3) Change the cacerts password to match the password you have
;    chosen above for the qipkeystore.
;
;    $QIPHOME/jre/bin/keytool -storepasswd -new <password>
;                     -keystore $QIPHOME/jre/lib/security/cacerts
;                     -storepass changeit
;
; 4) Import the self signed certificate into the enterpise server
;    cacerts keystore.
;
;    $QIPHOME/jre/bin/keytool -import
;                     -alias vitalqip
;                     -keystore $QIPHOME/jre/lib/security/cacerts
;                     -file $QIPHOME/vitalqip.cer
;                     -storepass <password>
;
; 5) Securely transfer the certificate file and the keystore to $QIPHOME
;    on the machine running the remote service.
;
; 6) Change the cacerts password on the remote machine to the password
;    you chose above.
;
;    $QIPHOME/jre/bin/keytool -storepasswd -new <password>
;                     -keystore $QIPHOME/jre/lib/security/cacerts
;                     -storepass changeit
;
; 7) Import the self signed certificate into the remote server cacerts
;    keystore.
;
;    $QIPHOME/jre/bin/keytool -import
;                     -alias vitalqip
;                     -keystore $QIPHOME/jre/lib/security/cacerts
;                     -file $QIPHOME/vitalqip.cer
;                     -storepass <password>
;
;
; VirtualMachineParams - anything specified here is passed to qapi (as an
; argument to the JVM)  Any 'java' arguments are valid.
;QAPI_Scheduler.VirtualMachineParams =

; This example configures a single scheduler to communicate with
; Remote Servers using the Secure Socket Layer (SSL).
;SchedulerNames = QAPISSL_Scheduler
;QAPISSL_Scheduler.ExecutorName = QAPI
;QAPISSL_Scheduler.Secure = true
;QAPISSL_Scheduler.PassPhrase = <qip-crypted keystore password>
;QAPISSL_Scheduler.KeysFile = qipkeystore

; This example starts 2 Schedulers, 1 which uses regular sockets, the other SSL
;SchedulerNames = QAPI_Scheduler QAPISSL_Scheduler
;QAPISSL_Scheduler.ExecutorName = QAPI
```

```
;QAPI_Scheduler.Secure = false
;QAPISSL_Scheduler.Secure = true
;QAPISSL_Scheduler.PassPhrase = <qip-crypted keystore password>
;QAPISSL_Scheduler.KeysFile = qipkeystore
;-------------------------------------------------------------------------------


[VitalQIP SSL Tunnel Service]

;Debug = None
;DebugLevel = NONE
; Even though this service starts as a C++ application, most of the work is done
; in Java.  The Java debug level is set according to the C++ level but you may
; override it by setting DebugLevel.
;DebugFile = qip-ssltd.log
;FeatureTimestamp = false
;
;MessageBufferSize = 0x2000
; The max number of bytes to read/write between peers at once.   All numerical
; arguments can be in decimal/hex/octal format.  No, you can't specify
; decimal values with suffixes i.e. 64k, 1M
;
;PublishID = VitalQIP SSL Tunnel Service
; Specifies what id to publish our port under.  Do not change this.
;
;MessageServicePort=2468
; Allows you to specify an alternate port to communicate with Message Service.
; Valid values are:
; Any valid port number |  Any service name in /etc/services
;
;DenyConnectionList -- see above
;AllowConnectionList -- see above
;
; See VitalQIP RMI Scheduler Service for instructions on setting up the Keysfile
; You will need to create the keys, create the certificate, and then import that
; certificate into the cacerts keystore. You can use the same generated keys and
; certificate on all VitalQIP installations.
;PassPhrase = <qip-crypted keystore password>
;KeysFile = qipkeystore
;-------------------------------------------------------------------------------

[VitalQIP Audit Update Service]

; qip-auditupdated
; override globals
;debug = None
;DebugFile = qip-auditupdated.log

;ListenPort          -- see above
;Master              -- see above
;AddIDToLogFilenameOnFork -- see above
;DropRequests        -- see above
;DumpStatsOnExit     -- see above
;ConnectQueueDepth   -- see above
```

```
;MaxConnections      -- see above
;DenyConnectionList  -- see above
;AllowConnectionList -- see above

; The name or address of the server running the VitalQIP Login Service.
;LoginServer = 127.0.0.1

; AuditUpdatePassword specifies the encrypted password of the Audit Manager
; updateservice administrator.
;AuditUpdatePassword =

; The name of the database server that the service will use to connect to the
; database.  This must match one of the names returned from the login service.
; There is no default.
;AuditServer =

; Additional arguments passed to qip-auditalertuserexit.
; Required parameters for the example NT user exit are
; AlertArgs = SMTPServer=<hostname> and
; AlertArgs = SMTPFrom=AuditManager
; There are no required parameters for the unix based qip-auditalertuserexit.
;-------------------------------------------------------------------------------


[VitalQIP Audit Schedule Service]

; qip-auditsched
; override globals
;debug = None
;DebugFile = qip-auditsched.log

;DumpStatsOnExit -- see above
;ProcessInterval -- see above
;LicenseInterval -- see above

; The name or address of the server running the VitalQIP Login Service.
;LoginServer = 127.0.0.1

; AuditSchedulePassword specifies the encrypted password of the Audit Manager
; scheduleservice administrator.
; The default is unpublished.
;AuditSchedulePassword =

; The name of the database server that the service will use to connect to the
; database.  This must match one of the names returned from the login service.
; There is no default.
;AuditServer =
;-------------------------------------------------------------------------------
```

5 Authenticate administrators

Overview

Purpose

This chapter describes how to use the authenticate administrators callout.

Contents

This chapter presents the following topics.

Administrator authentication tool

The VitalQIP Login Service supports a callout which can authenticate administrators against a third party authentication service, such as Radius or the Windows 2000 user database when an administrator logs into a VitalQIP, Network Allocator, or Audit Manager system. This authentication ability is done through an authentication callout or CLI command.

Important! If custom authentication routines take a long time to execute, administrators can experience delays logging into VitalQIP. In this case, distribute the authentication load by installing duplicate Login Services on other hosts. Configure a subset of the hosts running VitalQIP to use the duplicate Login Services.

Important! Custom authentication routines are invoked by the Login Service when attempting to confirm an administrator's identity. VitalQIP clients do not maintain a persistent connection with the Login Service or the Authentication Callout. Therefore, it is not possible to use the authentication callout to expire active user sessions.

☐

Sample administrator authentication callout

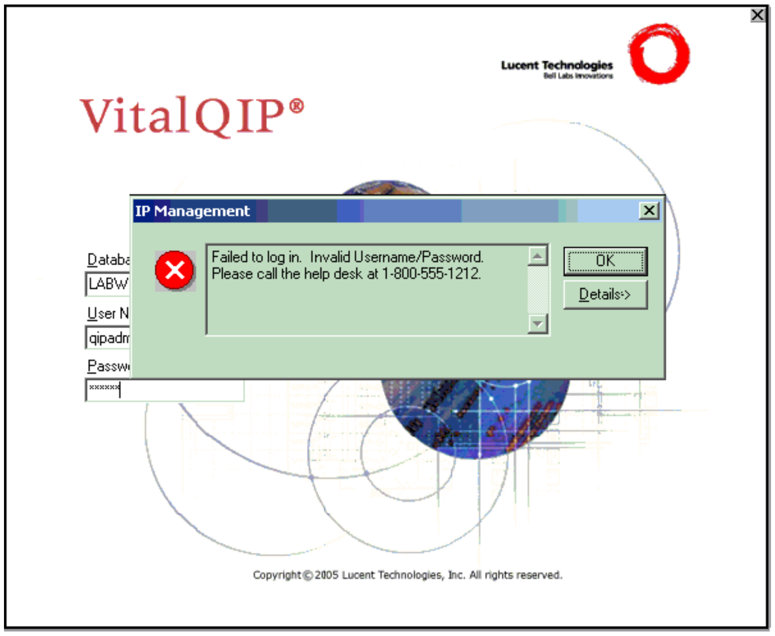
VitalQIP supplies a sample callout library called *QipAuthCallout.dll* (Windows) or *libqipauthcallout.so* (UNIX). This callout library can be configured to call an arbitrary command. The authentication callout library is loaded at run time by the Login Service based on policy options.

> **Important!** For the remainder of this section, *QipAuthCallout.dll* is used. You should assume *QipAuthCallout.dll* also includes the UNIX version *libqipauthcallout.so* (Solaris and AIX) and *libqipauthcallout.sl* (HPUX).

The *qipauthcallout.dll* library logs administrator authentications to the system log and can be configured to call an external CLI command. The external CLI command can then authenticate administrators.

To slow down password hackers, the *QipAuthCallout.dll* can lock an administrators' user interface for a specified number of seconds. This can be useful for slowing down "password hackers".

In case of an authentication failure, the *QipAuthCallout.dll* library can be configured to present the user with custom messages. The following is a custom message example:



Failed to log in. Invalid Username/Password.
Please call the help desk at 1-800-555-1212.

The sample authentication callout performs the following functions:

- Returns custom failure message messages (for example, a help desk number that varies according to the administrator's business unit)

- Returns custom failure delays
- Maps domain administrator names to VitalQIP administrator names
- Maps multiple domain administrator names to a single VitalQIP administrator

To perform these functions, you must replace the authentication callout with a custom library, using the authentication callout.

□

Authentication callout policies

The *QipAuthCallout.dll* library can be configured through the **[QIPAuthCallout]** section of the *QIPHOME/qip.pcy* policy file. Table 37 describes the authentication callout policies.

Table 37 QipAuthCallout.dll policies

Policy	Values	Description
SuccessLogLevel	Default: Information Allowed: None | Information | Warning | Error	The severity level at which successful authentications are written to the system log. The following values can be specified: None Information Warning Error
AttemptLogLevel	Default: Information Allowed: None | Information | Warning | Error	The severity level at which authentication attempts are written to the system log. Authentication attempts are only written to the system log when the AuthenticationCli policy is not set. The valid values are **None**, **Information**, **Warning**, and **Error**.
FailureLogLevel	Default: Error Allowed: None | Information | Warning | Error	The severity level at which unsuccessful authentications are written to the system log. The valid values are **None**, **Information, Warning**, and **Error**.
AuthenticationCli	Default: None Allowed: Any valid path name relative to *QIPHOME/userexits*	The path name of a command to run to authenticate the administrator. Path names are relative to *QIPHOME/userexits*. The CLI command should return one of the following: **0** - allow the user to log into VitalQIP **1**- defer authentication to VitalQIP **2** - deny the authentication When this policy is not set, the administrator authenticates against the password stored in the VitalQIP database. Refer to "Authentication CLI command", on page 189 for a description of input arguments and expected return values.
FailureMessage	Default: None Allowed: Text up to 253 characters	The text of a message to be displayed by the user interface when administrator authentication fails.

Policy	**Values**	**Description**
DelayOnFailure	Default: 0 Allowed: 0 or any positive number	The number of seconds the Login Service will tell the client to delay before notifying the administrator of an authentication failure.

☐

Use the authentication callout as a logging tool

When to use

The *QipAuthCallout.dll* library can be used as a logging tool to log administrator authentications to the system log.

Procedure

To do this, follow these steps:

1 In **[VitalQIP Login Service]** section of the *qip.pcy* file, set the **AuthLibrary** policy to **qipauthcallout**. Refer to "VitalQIP Login Service policies", on page 131 for more information.

2 Restart the Login Service.

Login attempt messages are written to the system log and administrator passwords are authenticated against the VitalQIP database.

END OF STEPS

☐

Use the authentication callout as authentication tool

When to use

The *QipAuthCallout.dll* library can be used to run a script or executable to authenticate administrators.

Procedure

To do this, do one of the following:

- In the **[VitalQIP Login Service]** section of the *qip.pcy* file, set the **AuthLibrary** policy to **qipauthcallout**.
- In the **[QIPAuthCallout]** section of the *qip.pcy* file, set the **AuthenticationCLI** policy to the path of a custom CLI command. See "Authentication CLI command" on page 189 for more information on the CLI command.

Login success and failure messages are written to the system log, and administrator passwords are authenticated by the configured CLI command.

System log messages

The *QipAuthCallout.dll* library logs the messages to the system log. The severity of these messages is set using the **SuccessLogLevel** and **FailureLogLevel** policies, as described in “Authentication callout policies”, on page 185. The following messages are logged:

```
“<administrator> authenticated against <product>/<database> from <1.2.3.4>”
“<administrator> failed authentication against <product>/<database> from <1.2.3.4>”
“<administrator> is trying to authenticate against <product>/<database> from <1.2.3.4>”
```

Where <administrator> is replaced by the user name of the administrator, <product> is either QIP or Audit, <database> is the name of the database that the administrator is trying to connect to, and <1.2.3.4> is the IP address of the host that the user is logging in from.

The following is an example message logged when an authentication CLI command cannot be executed; this message is generated at the Error log level:

```
“Can not execute authentication CLI name when authenticating administrator against
  product/database from 1.2.3.4”
```

☐

Authentication CLI command

An authentication CLI command is invoked when the **AuthenticationCLI** policy is set in the **[QIPAuthCallout]** section of the *qip.pcy* file. Sample authentication CLI commands are located in *QIPHOME\examples\qipauthcallout.bat* (Windows) and *QIPHOME/examples/qipauthcallout* (UNIX).

The authentication CLI command can be written to restrict authentications based on password, user name, time of day, or a client's IP address. The CLI command can report an authentication failure.

Authentication CLI command parameters

The command line parameters are passed as individual parameters to the authentication CLI command. The parameters are as follows:

```
<Authentication CLI command> <User name> <QIP|Audit> <Database> <IP address>
```

Table 38 describes the authentication CLI command parameters.

Table 38 Authentication CLI command parameters

Value	Description
User name	The administrator's user name.
QIP|Audit	The name of the product that the administrator is logging in to. The following can be specified: **QIP** - VitalQIP or Network Allocator **Audit** - Audit Manager
Database	The name of the database to which the administrator is trying to connect.
IP address	The IP address of the host that created the login request presented in dotted decimal form. **Important!** When using the VitalQIP Web Interface, the address is the IP address of the host running the web applications and not the address of the administrator's browser.

Important! To secure the administrator's password, the password is not passed on the command line. Instead, the Login Service sends the administrator's password to the authentication CLI command using the $LUCENT_PASSWORD environment variable.

The following is an example:

```
./qipauthcallout ”qipman” ”QIP” ”QIPDB” ”1.2.3.4”
```

Return values

The authentication CLI command must return one of the following values:

- 0 to indicate that the administrator successfully authenticated
- 1 to defer password validation to VitalQIP or Audit Manager

- 2 to indicate that the administrator did not successfully authenticate.

☐

Custom administrator authentication callout

The *qipauthcallout* library may not be the best way to integrate administrator authentication into VitalQIP. This may be the case when a third party authentication routine:

- Has a large startup cost, such as connecting to a database or through SSL
- Is provided by an API and there is no command line tool available
- Returns custom failure messages based on the reason for failure
- Returns custom failure messages based on administrator name
- Maps individual domain administrator name to an individual VitalQIP administrator
- Maps multiple domain administrator names to a single VitalQIP administrator

In these cases, VitalQIP allows you to create a custom dynamic library that can implement auditing and authentication functions.

A *QIPAuthCallout.h* header file and an example implementation file is located in the *QIPHOME/examples* directory where *qipauthcallout* is installed.

☐

Create a custom authentication callout

A custom authentication callout must be in a dynamic library with the symbol QIPAuthCallout defined as below. On Windows, QIPAuthCallout must be exported from the dynamic library. The authentication callout name QIPAuthCallout must conform to C language bindings.

Important! The authentication callout implementation must not have connectivity to the VitalQIP, Network Allocator, and Audit Manager services as well as databases. In particular, the VitalQIP API Toolkit is not supported by an authentication callout.

The following is a function prototype:

```
#if defined(__cplusplus)
extern "C" {
#endif

#if defined(WINNT)
__declspec( dllexport )
#endif
int QIPAuthCallout ( const char * admin_name_in,
const char * product_name,
const char * database_name,
const char * admin_password,
const char * host_address,
int * failure_delay,
char admin_name_out[31]
char failure_message[253] );

#if defined(__cplusplus)
}
#endif
```

Table 39 describes the arguments that invoke the callout.

Table 39 Authentication callout arguments

Argument	Type	Pointer	Value	Description
admin_name_in	Input	const char *	Null terminated string of not more than 30 characters.	The name that the user typed into the user interface. This name need not exist in the VitalQIP database.
product_name	Input	const char *	Null terminated string of not more than 253 characters.	The name of the product that the administrator is logging into: • QIP - VitalQIP or Network Allocator product • Audit - Audit Manager product

Argument	Type	Pointer	Value	Description
database_name	Input	const char *	Null terminated string of not more than 253 characters.	The name of the database instance that the administrator is logging in to.
admin_password	Input	const char *	Null terminated string of not more than 253 characters.	The plain text password that the administrator typed in.
host_address	Input	const char *	Null terminated string of not more than 253 characters.	The IP address of the host that created the login request or the IP address of the web server if authenticating in the web interface.
failure_delay	Output	int *	Pointer to integer.	Number of seconds that the client program is delayed before telling the administrator that they cannot log in. Delaying the response on a failed authentication attempt is a good security practice. The delay time is spent on the client and not the server.
admin_name_out	Output	char[31]	A buffer of 31 characters	The name of a VitalQIP administrator to be passed back to the user interface. This name must exist in the VitalQIP database and determines the user's permissions. The callout only needs to change this argument when the name that the user typed is different than the name in VitalQIP. This argument is not relevant in the case of authentication failure.
failure_message	Output	char[253]	A buffer of 253 characters	A message to be displayed to the administrator in case of authentication failure. The callout should place a zero terminated string that is no longer than 253 bytes.

The authentication callout must return the following values:

- **0** - indicates that the administrator can log in.
- **1**- indicates that the administrator can log in if the password matches the one stored in the VitalQIP database. Returning 1 allows the callout to be used as an auditing or filtering mechanism.
- **2** - indicates the administrator cannot log in.

☐

6 Secure message routes

Overview

Purpose

This chapter describes how VitalQIP uses tunneling and SSL to provide security for all message routes, as well as to enable dynamic DNS updates of Microsoft DNS servers.

Contents

This information presents the following topics.

☐

Introduction to secure message routes

Beginning with Release 6.2, VitalQIP provides increased security for message routes, and supports :

- Tunneling of all messages through the VitalQIP Message Service
- Use of Secure Sockets Layer (SSL) to provide message encryption and authentication of VitalQIP components
- Ability to make dynamic DNS updates to a secure zone hosted on a Microsoft DNS server

Important! SSL messaging is optional and is turned off by default.

☐

Port tunneling

The only well known port that VitalQIP requires is the **qip-msgd** port for the Message Service. All other services bind to an ephemeral port assigned by the operating system at startup, and register that port with the Message Service. When the Message Service receives traffic destined for one of those registered services, it forwards the traffic to the appropriate ephemeral port. This is the default configuration, and is shown in Figure 17.

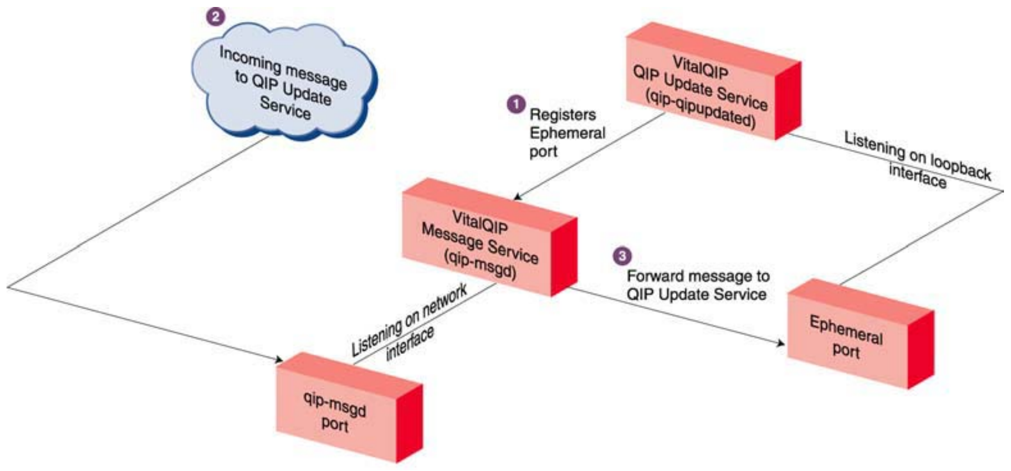

2
Incoming message to QIP Update Service
VitalQIP QIP Update Service (qip-qipupdated)
1
Registers Ephemeral port
Listening on loopback interface
VitalQIP Message Service (qip-msgd)
3
Forward message to QIP Update Service
Listening on network interface
Ephemeral port
qip-msgd port

Figure 17 Default non-secure message flow

After VitalQIP 6.2 is installed, message routes are modified to state the service name instead of the port number of destination services. For example, instead of the old message route format,

```
MessageRoute=DHCP:A:0:QIP Update Service (DHCP):qip-qdhcp:127.0.0.1
```

the route looks like this:

```
MessageRoute=DHCP:A:0:QIP Update Service (DHCP):VitalQIP QIP Update Service:127.0.0.1
```

The bold text indicates the change from port name (or number) to service name. The current allowable service names are:

- VitalQIP Audit Update Service
- VitalQIP DNS Update Service
- VitalQIP QIP Update Service
- VitalQIP Message Service

Backward compatibility of remote servers

After the enterprise server is upgraded to use ephemeral ports, all message routes on remote servers will fail to work because they are still trying to connect using the old port numbers. To avoid this, edit the enterprise server *qip.pcy* and configure the **ListenPort** policy in the VitalQIP QIP Update Service and DNS Update Service sections to be its respective legacy port name or number, as listed inTable 40 (for example, `ListenPort=2490` or `ListenPort=qip-qdhcp`).

> **Important!** Additionally, when an enterprise server is upgraded from VitalQIP 6.1, the **AllowConnectionList** policy needs to be set to allow connections from remotes to the VitalQIP QIP Update Service and DNS Update Service.

If you also have Audit Manager installed on remotes, you need to modify the **ListenPort** policy in the Audit Update Service section of *qip.pcy*. Refer to the *Audit Manager User's Guide* for more information on **qip-auditupdated**.

Table 40 Legacy port names and numbers

Service name	Port name	Port number
VitalQIP QIP Update Service	qip-qdhcp	2490
VitalQIP DNS Update Service	qip-dns	3119
VitalQIP Audit Update Service	qip-audup	2765

After modifying the **ListenPort** policies in the enterprise server *qip.pcy*, the services listen on their old port numbers (instead of ephemeral ports) so that the pre-6.2 message routes continue to work. The services also register their service names with the Message Service, so that VitalQIP 6.2 systems with new message route formats work.

After all systems are upgraded to 6.2 and 6.2-style message routes, the services on the enterprise server should be set to listen on ephemeral ports.

> **Important!** SSL ***cannot*** be enabled for remote servers that are using legacy ports.

Secure message transport

Non-secure message routes

Making all messages secure has resulted in some changes in the way VitalQIP services are deployed and messages are routed. Prior to VitalQIP 6.2, messages went directly from the source (for example, the VitalQIP GUI) to a Message Service at the destination, and from there to the destination service. With VitalQIP 6.2, all messages are tunneled through a local Message Service on the sending machine, as shown in Figure 18.

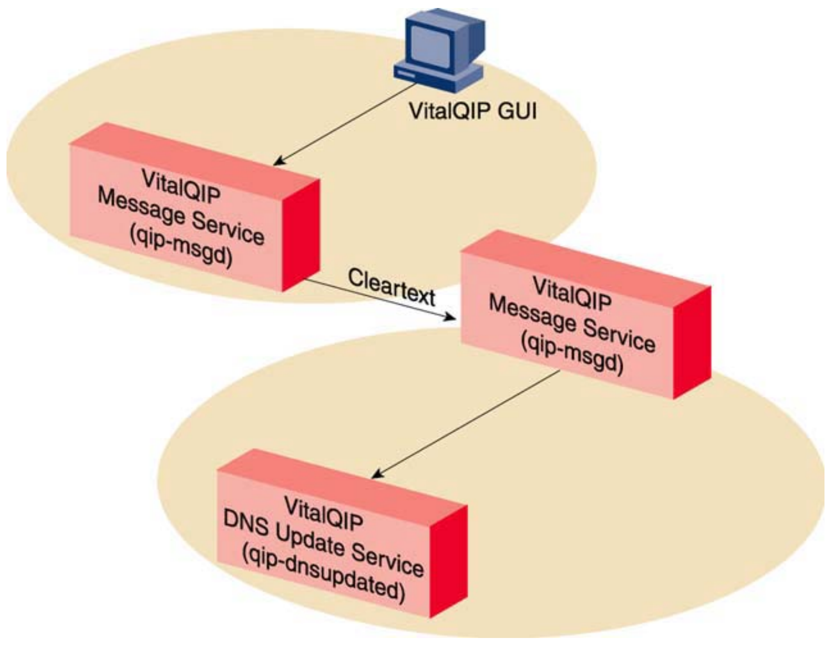

VitalQIP GUI
VitalQIP Message Service (qip-msgd)
Cleartext
VitalQIP Message Service (qip-msgd)
VitalQIP DNS Update Service (qip-dnsupdated)

Figure 18 Non-secure message transport tunneled through Message Service

In Figure 18, unsecured cleartext messages using the standard MessageRoute policy are sent from the VitalQIP GUI to the DNS Update Service:

```
MessageRoute = <id>:<flags>:<ACK_timeout>:<desc>:<port>:<Server_IP>:…<*#>
```

This policy is described in detail on page 127.

SSL-enabled message routes

If SSL is enabled, VitalQIP requires an SSL tunnel on both the recipient (receiving) and initiator (sending) machines, as shown in Figure 19.

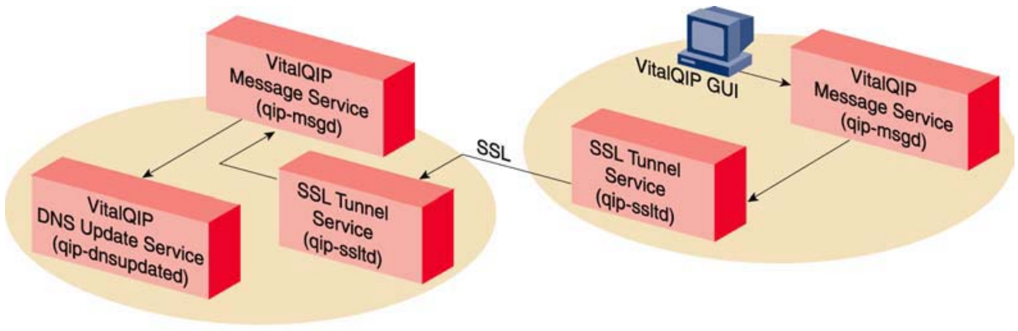

VitalQIP Message Service (qip-msgd)
VitalQIP DNS Update Service (qip-dnsupdated)
SSL Tunnel Service (qip-ssltd)
SSL
VitalQIP GUI
SSL Tunnel Service (qip-ssltd)
VitalQIP Message Service (qip-msgd)

Figure 19 Secure message transport tunneled through SSL Tunnel and Message Service

In Figure 19, secure encrypted messages using the new SecureMessageRoute policy are sent from the VitalQIP GUI to the DNS Update Service:

```
SecureMessageRoute = <id>:<flags>:<ACK_timeout>:<desc>:<port>:<Server_IP>:…<*#>
```

This policy uses the same settings as the MessageRoute policy.

Message transport without co-located Message Service

In situations where you do not wish to have the Message Service running on all GUI clients, or on platforms that cannot run services, such as Windows 2000 Professional and Windows XP, you need to set the QIPMESSAGESERVICE environment variable to the IP address of a machine that is running the Message Service. In this case, SSL ***cannot*** be used on the connection from the GUI to the machine running the Message Service, as shown in Figure 20.

QIPMESSAGESERVICE may also be a comma-separated list of IP addresses. In that case, the GUI will try the first in the list and fail over to the following addresses if the previous addresses do not respond.

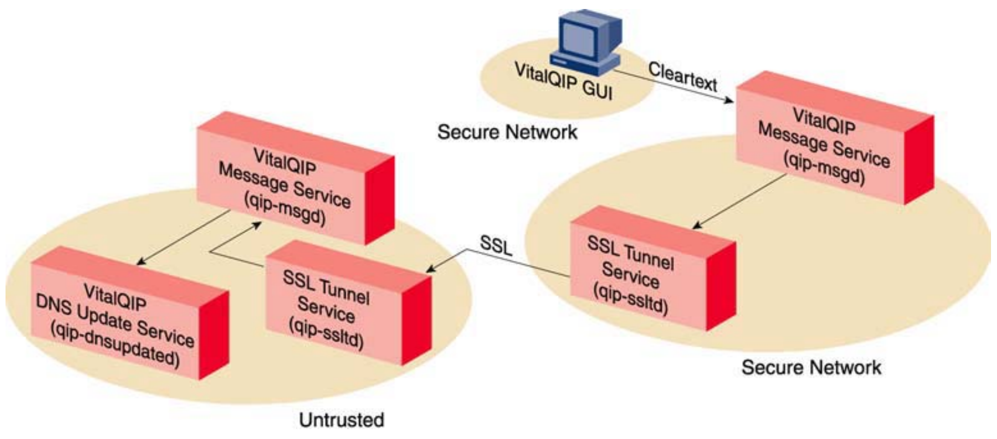

VitalQIP GUI
Cleartext
Secure Network
VitalQIP Message Service (qip-msgd)
SSL Tunnel Service (qip-ssltd)
Secure Network
SSL
VitalQIP Message Service (qip-msgd)
VitalQIP DNS Update Service (qip-dnsupdated)
SSL Tunnel Service (qip-ssltd)
Untrusted

Figure 20 GUI deployment without co-located Message Service

You can also secure only parts of the VitalQIP deployment. This is desirable if, for example, part of the deployment is behind a firewall on a trusted part of the network, but other parts are in a potentially hostile environment. This deployment scenario uses proxies and is described in .“Remote proxies”, on page 205.

□

Configuring SSL

When to use

SSL is configured if selected during the installation of VitalQIP. The following information is included if you need to configure SSL manually.

To manage the keys and certificates that SSL needs, you need to use Java's **keytool** key management utility. **keytool** is part of the J2EE SDK that is packaged with VitalQIP. It is used to create a key pair and a self-signed certificate, which should be placed on all VitalQIP GUI, remote, and enterprise server machines.

Before you begin

- http://java.sun.com/j2se/1.4.2/docs/tooldocs/windows/keytool.html has complete instructions on creating and managing a keys file.
- Lucent recommends that only one key certificate be used, but it is possible to use multiple certificates. For example, each remote service and client could have a unique certificate that would be signed by the enterprise server's self-signed certificate.
- The VitalQIP 6.2 installation allows you to create a self-signed certificate. If you already created one during installation, securely transfer the certificate file (*vitalqip.cer*) and key store (*qipkeystore*) to $QIPHOME on any other machines on which you wish to run the SSL Tunnel Service. During the VitalQIP 6.2 remote server installation, you are prompted for the location of these files.

Procedure

On all machines where SSL is enabled, use the following steps to configure SSL:

1 Create a set of keys for VitalQIP. You will be prompted for a keystore password.

Important! Be sure to enter the password value and keystore file name in the PassPhrase and KeysFile policies under the SSL Tunnel Service section of *qip.pcy.*

In the following example, the certificate will be valid for ten years.

```
$QIPHOME/jre/bin/keytool -genkey -keyalg RSA -alias vitalqip -validity 3652 -keystore
  $QIPHOME/qipkeystore
```

2 Create a self-signed certificate file for VitalQIP.

```
$QIPHOME/jre/bin/keytool -export -alias vitalqip -keystore $QIPHOME/qipkeystore -file
  $QIPHOME/vitalqip.cer
```

3 Import the self-signed certificate into the **cacerts** keystore (which already exists).

```
$QIPHOME/jre/bin/keytool -import -alias vitalqip -keystore
  $QIPHOME/jre/lib/security/cacerts -file $QIPHOME/vitalqip.cer -storepass changeit
```

4 Securely transfer the certificate file (*vitalqip.cer*) and key store (*qipkeystore*) to $QIPHOME on any other machines running the SSL Tunnel Service.

5 Repeat step 3 and import the self-signed certificate into the server's **cacerts** keystore and use the same Private Key Encryption Password of the enterprise server for each of these other machines.

6 Ensure that VitalQIP only accepts SSL-enabled communication by setting

```
SecureIncomingMessageServiceConnections = true
```

in the Global Section of *qip.pcy* (refer to "Global section", on page 102).

This informs the Message Service to stop listening for cleartext traffic on the network interfaces, and to only accept communication from local services. It also informs the SSL Tunnel Service to listen on the Message Service's port on all network interfaces. The SSL Tunnel Service only accepts SSL-enabled traffic where the initiator can authenticate itself. Once authenticated, the SSL Tunnel Service unencrypts the traffic and passes it to the Message Service for routing.

END OF STEPS

Message Service communication

After SSL is enabled, all non-message route based communication is secured: that is, all traffic for the Login Service, Active Lease Service, IBM DHCP Monitor Service, MS DNS Update Service, as well as push requests to the Remote Service.

Per route security

Message route traffic can be secured on a per route basis. If you wish to have a specific message route secured, simply replace the `MessageRoute` in *qip.pcy* with `SecureMessageRoute.` This informs the Message Service to forward all traffic for that route to the SSL Tunnel Service for encryption and authentication.

☐

Remote proxies

If you are using proxies, either in the Server Profile or in the **Remote Server Proxy** global policy, to push to remote servers, you need to configure them to fit the security model of your network. You may need to create cleartext, secure, promotion, or demotion proxies as needed.

When VitalQIP sends a push request, it first checks the **QIPSECUREPROXY** environment variable. If this variable is True, it sends a secure request. If it is False, it sends a cleartext request. If it is not set, it first tries to send a cleartext request and only if that fails will it try to send the request over a secure connection. Because VitalQIP may need to push to both secure and cleartext remotes, it needs to attempt both types of communication to determine which will work. If the cleartext connection fails, a debug message is logged unless the secure connection also fails. In that case an error is recorded.

> **Important!** If you know that your proxies/remotes will always be secure (or conversely non-secure), you should set the **QIPSECUREPROXY** environment variable so that VitalQIP will not perform unnecessary work for every connection.

Remote Server Proxy Description sub-parameter

The Remote Server Proxy parameter in the Server Profile contains a **Description** sub-parameter where you can enter a comment (255 characters). For example, you could indicate that a proxy is configured to accept secure connections. You can enter a comment in this sub-parameter whether or not there is a value entered for Remote Server Proxy.

□

Cleartext proxy

When you install the Message Service to act as a proxy, it is a cleartext proxy by default.

The Message Service listens on all network interfaces and the loopback on its well-known port and accepts only insecure incoming connections. To configure this on the proxy, set the **SecureIncomingMessageServiceConnections** policy in the Global Section of *qip.pcy* to False.

This proxy would be used for push requests if the GUI that is generating the push request did not have SSL enabled and this proxy was listed in the Server Profile or in the Remote Server Proxy global policy. The remote could be secured with SSL (if **SecureOutgoingProxyConnections** is True in the Message Service section of *qip.pcy*) or not, or it could be a VitalQIP system on an older release such as 6.1 (if **SecureOutgoingProxyConnections** were False).

Important! Using a proxy to promote cleartext incoming connections to secure outgoing connections could be dangerous. If you need to do this, it is a good idea to lock down the **AllowConnectionsList** policy to be the minimum set necessary.

□

Secure proxy

If your whole network is SSL-enabled, you need to configure your proxies to be secure proxies.

To do this, set the **SecureIncomingMessageServiceConnections** policy in the Global Section of *qip.pcy* on the proxy to True. This ensures that the proxy only accepts secure connections. The **SecureOutgoingProxyConnections** policy in the Message Service section of *qip.pcy* on the proxy also needs to be set to True. This causes the proxy to use SSL to complete all outgoing proxy connections, as shown in Figure 21.

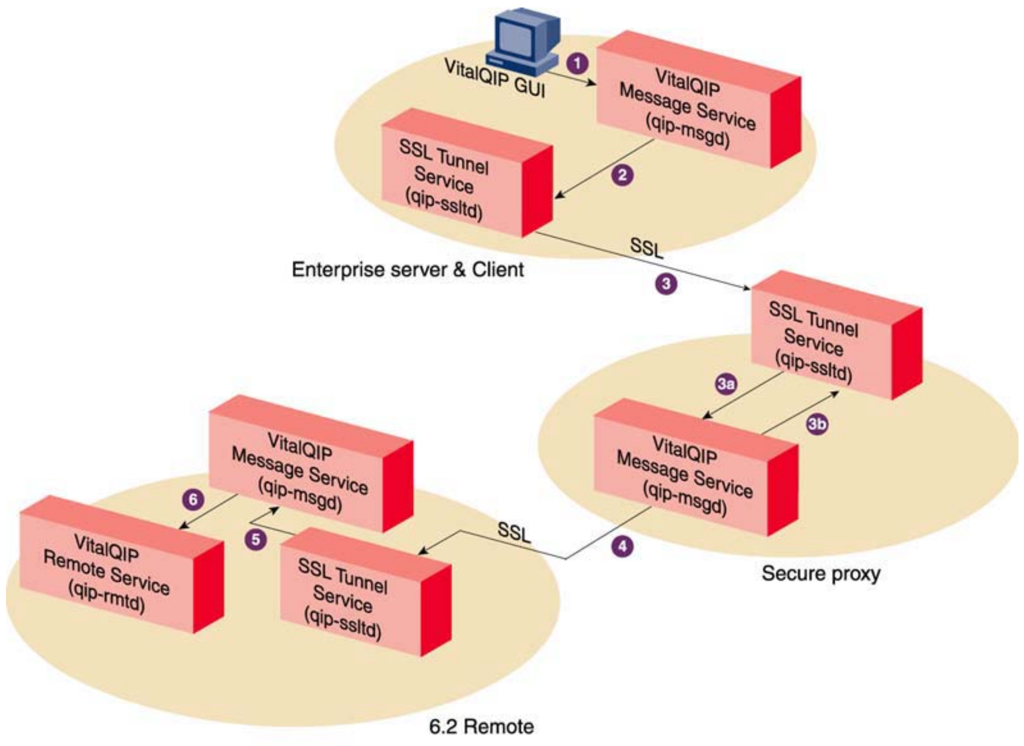

VitalQIP GUI
1
VitalQIP Message Service (qip-msgd)
2
SSL Tunnel Service (qip-ssltd)
SSL
3
Enterprise server & Client
SSL Tunnel Service (qip-ssltd)
3a
3b
VitalQIP Message Service (qip-msgd)
4
SSL
Secure proxy
VitalQIP Message Service (qip-msgd)
6
5
VitalQIP Remote Service (qip-rmtd)
SSL Tunnel Service (qip-ssltd)
6.2 Remote

Figure 21 Routing of push request through secure proxy to the remote server

The following message flow occurs:

1. The GUI needs to send a push request to a remote, and sends a conduit connection request (with the IP address of the remote server, and the address of the proxy) to the local Message Service. The Message Service is bound to the Message Service port on the loopback interface from which it only accepts cleartext TCP communication.
2. The local Message Service makes a TCP connection to the local SSL Tunnel Service. The Message Service knows where to communicate with the SSL Tunnel, because the tunnel is registered with the local Message Service. The SSL Tunnel Service only accepts cleartext communication on the loopback interface.
3. The local SSL tunnel makes a connection to the proxy SSL tunnel. The proxy SSL Tunnel Service is running on the Message Service port on the proxy machine's network interface.
   - 3a. The proxy SSL Tunnel Service decrypts the conduit request and hands it over to the proxy Message Service for processing. The SSL Tunnel Service sends the cleartext TCP message to the Message Service running on the proxy's loopback interface.
   - 3b. The proxy Message Service realizes that the connection request does not terminate locally and forwards the cleartext TCP message to the proxy SSL tunnel on the ephemeral port it registered with the proxy Message Service.
4. The proxy SSL Tunnel Service realizes that the connection request does not terminate locally, and proxies the request to the remote SSL Tunnel Service. Encrypted and authenticated TCP communication occurs where both the initiator and recipient must be able to authenticate each other's credentials (certificates). If they cannot, the connection attempt fails.
5. The remote SSL Tunnel Service removes the encryption, and sends the conduit request to the remote's Message Service running on the loopback interface and the Message Service port. The Message Service only accepts TCP cleartext communication from the loopback interface.
6. The remote Message Service looks in its service registry, finds the Remote Service, and makes a TCP cleartext connection to it on the loopback interface.

□

Promotion proxy

If you have VitalQIP GUIs on a safe part of the network, but have remote servers located in a hostile environment, you need to deploy a promotion proxy that takes cleartext traffic from the GUI and secures it before forwarding it to the remote.

To do this, set the **SecureIncomingMessageServiceConnections** policy in the Global Section of *qip.pcy* on the proxy to False (the default). This allows the proxy to accept the cleartext GUI communication. The **SecureOutgoingProxyConnections** policy in the Message Service section of *qip.pcy* needs to be set to True, so that the communication out to the remote server is secure.

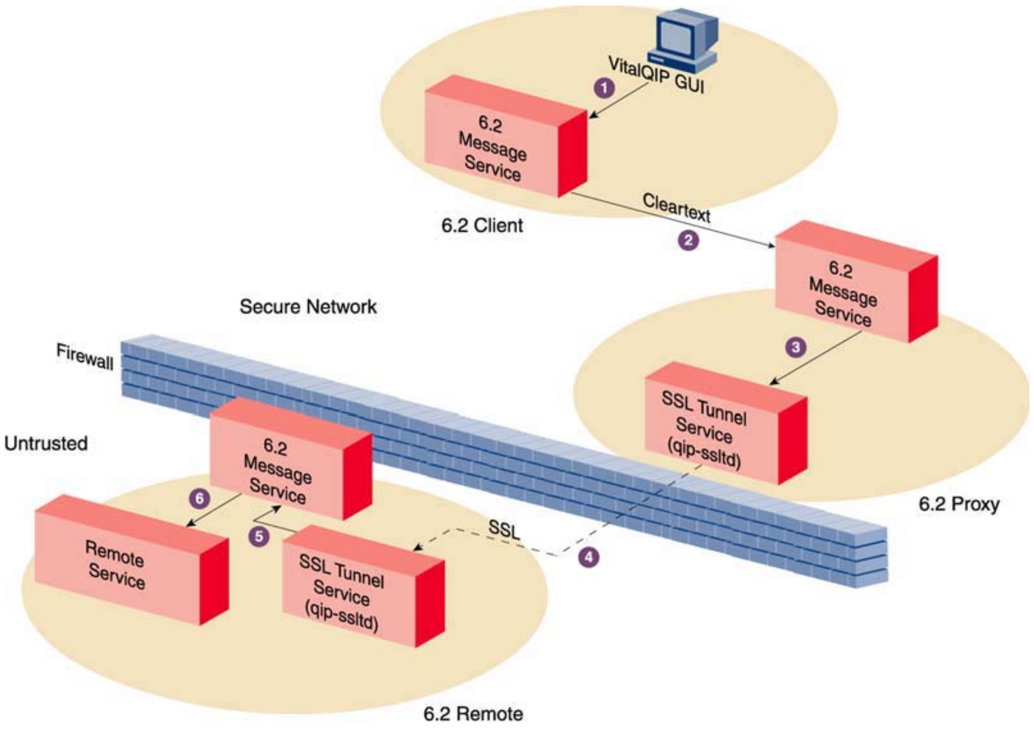


Figure 22 Routing of push request through promotion proxy

The following message flow occurs:

1. The GUI needs to send a push request to a remote, and sends a conduit connection request (with the IP address of the remote server, and the address of the proxy) and data to the local Message Service. The Message Service is bound to the Message Service port on the loopback interface. The Message Service only accepts TCP cleartext communication from the loopback interface.
2. The client Message Service makes a cleartext connection to the proxy Message Service. (The Message Service does so because the **SecureOutgoingProxyConnections** policy is set to False.)
3. The proxy Message Service realizes that the connection request does not terminate locally, and proxies the request to the proxy SSL tunnel. The Message Service knows where to communicate with the SSL tunnel, because the tunnel is registered with the local message service and the **SecureOutgingProxyConnections** policy is set to True. The SSL Tunnel only accepts TCP cleartext communication on the loopback interface.
4. The proxy SSL tunnel makes a connection to the remote SSL tunnel. The remote SSL Tunnel is running on the Message Service port on the remote machine's network interface. Encrypted and authenticated TCP communication occur where both the initiator and recipient must be able to authenticate each other's credentials (certificates). If they cannot, the connection attempt fails.
5. The remote SSL tunnel removes the encryption, and sends the conduit request to the local Message Service. The Message Service knows where to communicate with the SSL tunnel, because the tunnel is registered with the local Message Service. The SSL Tunnel only accepts TCP cleartext communication on the loopback interface.
6. The remote Message Service looks in its service registry, finds the Remote Service, and makes a TCP cleartext connection to the Remote Service on the loopback interface.

□

Demotion proxy

If you are traversing across a firewall in the other direction, you may want a demotion proxy. For example, if one of your administrators is in an insecure section of the network and is pushing to a remote server nestled safely within the firewall, you need to take the secure incoming traffic and demote it to clear text.

To do this, set the **SecureIncomingMessageServiceConnections** policy in the Global Section of *qip.pcy* on the proxy to True to accept only secure traffic. The **SecureOutgoingProxyConnections** policy in the Message Service section of *qip.pcy* needs to be set to False, to demote the secure traffic to cleartext.

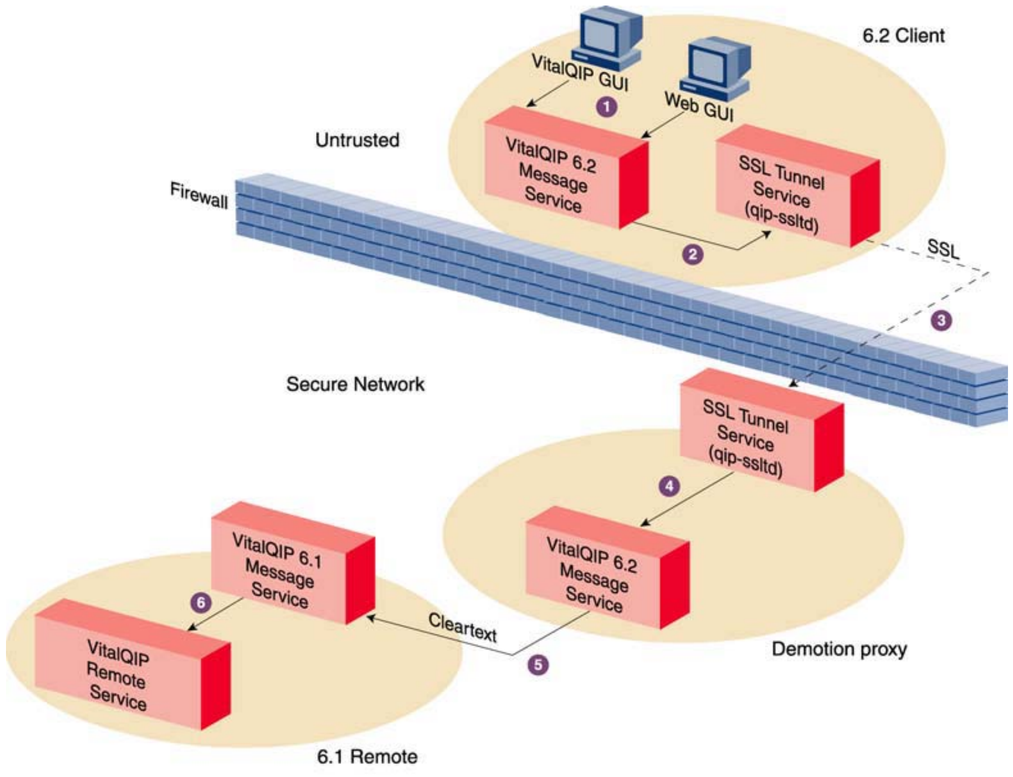

6.2 Client
VitalQIP GUI
Web GUI
1
Untrusted
VitalQIP 6.2 Message Service
SSL Tunnel Service (qip-ssltd)
Firewall
2
SSL
3
Secure Network
SSL Tunnel Service (qip-ssltd)
4
VitalQIP 6.1 Message Service
VitalQIP 6.2 Message Service
6
Cleartext
VitalQIP Remote Service
5
Demotion proxy
6.1 Remote

Figure 23 Routing of push request through demotion proxy

The following message flow occurs:

1. VitalQIP needs to send a push request to a remote, and sends a conduit connection request (with the IP address of the remote server, and the address of the proxy) to the client Message Service. The Message Service is bound to the Message Service port on the loopback interface, where it only accepts TCP cleartext communication.
2. The client Message Service makes a TCP connection to the local SSL Tunnel. The Message Service knows where to communicate with the SSL Tunnel, because the tunnel has registered with it. The SSL Tunnel only accepts TCP cleartext communication on the loopback interface.
3. Because SecureOutgoingProxyConnections is True, the client SSL tunnel makes a connection to the proxy SSL Tunnel. The proxy SSL Tunnel is running on the Message Service port on the proxy machine's network interface. Encrypted and authenticated TCP communication occur, where both the initiator and recipient must be able to authenticate each other's credentials (certificates). If they cannot, the connection attempt fails.
4. The proxy SSL Tunnel removes the encryption, and sends the conduit request to the proxy's Message Service. The proxy Message Service only accepts TCP cleartext communication on the loopback interface because the **SecureOutgoingProxyConnections** policy is set to False.
5. The proxy Message Service realizes that the connection request does not terminate locally, and proxies the request to the remote Message Service, which is running on the loopback interface and the Message service port. The proxy Message Service only accepts TCP cleartext communication on the loopback interface because the **SecureOutgoingProxyConnections** policy is set to False.
6. The remote Message Service looks in its service registry, finds the Remote Service, and makes a connection to it. The remote service only accepts TCP cleartext communication on any interface.

□

Login proxies

It is also possible that you have enabled SSL on your enterprise server, but have not yet deployed it on all of the administrative clients. In this state, your login server is only accepting secure traffic, but not all your clients are capable of sending it. In this case, the insecure clients need to have the **QIPLOGINPROXY** environment variable set. This environment variable informs the VitalQIP GUI and the CLIs to forward login requests through a proxy.

The proxy through which such requests are forwarded is set up as a promotion proxy. The proxy take the insecure login request from the client and promotes it to be a secure request to the Login Service, as shown in Figure 24.

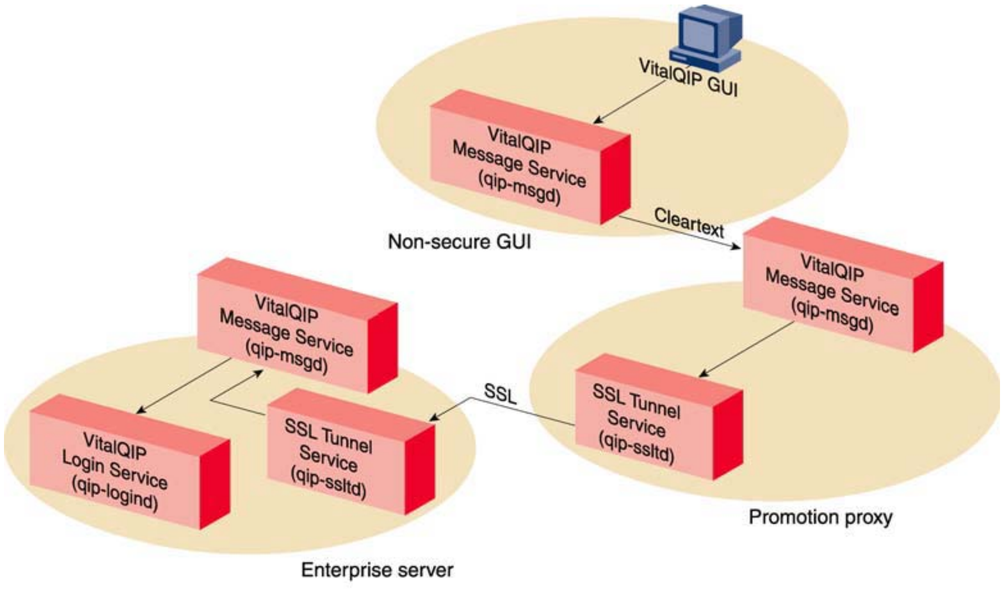

VitalQIP GUI
VitalQIP Message Service (qip-msgd)
Cleartext
Non-secure GUI
VitalQIP Message Service (qip-msgd)
VitalQIP Message Service (qip-msgd)
SSL
SSL Tunnel Service (qip-ssltd)
VitalQIP Login Service (qip-logind)
SSL Tunnel Service (qip-ssltd)
Promotion proxy
Enterprise server

Figure 24 Routing of login request through promotion proxy

Dynamic updates to secure MS DNS zones

With VitalQIP support of SSL-enabled secure message routes, dynamic updates can be made of objects located in a secure Windows 2000 zone. These updates can be sent securely via SSL to the remote server on which the DNS server is running. They can then be applied to the DNS server by the local MS DNS Update Service.

Use DNS Update Service policy

The path that the message takes to get to the remote server varies, depending upon the setting of the **Use DNS Update Service** global policy. If this is set to False, VitalQIP sends the message (via the Message Service and/or SSL Tunnel Service) directly to the remote server. If there are proxies specified for the server, VitalQIP attempts to route the message through those proxies.

If the **Use DNS Update Service policy** is set to True, VitalQIP asks the local Message Service to route the message based on its message routes to the DNS Update Service. The DNS Update Service forwards the message to the appropriate remote servers depending on the contents of the *ddns.conf* file.

The DNS Update Service honors the proxies defined in the Server Profile and in the **Remote Server Proxy** global policy and routes messages through those proxies.

> **Important!** The update process for a Microsoft DNS server is somewhat different from the process used for other types of DNS server. For other DNS server types, the DNS Update Service makes the RFC 2136 dynamic update directly to the target DNS server. For Microsoft DNS servers, however, the DNS Update Service forwards the VitalQIP message to the MS DNS Update Service on the remote server. The MS DNS Update Service makes the actual dynamic update.

☐

Dynamic update messaging without DNS Update Service

If the **Use DNS Update Service** global policy is set to False, the update information is forwarded from the GUI to the MS DNS Update Service installed on the remote server as shown in Figure 25. The MS DNS Update Service applies the update securely to the DNS server.

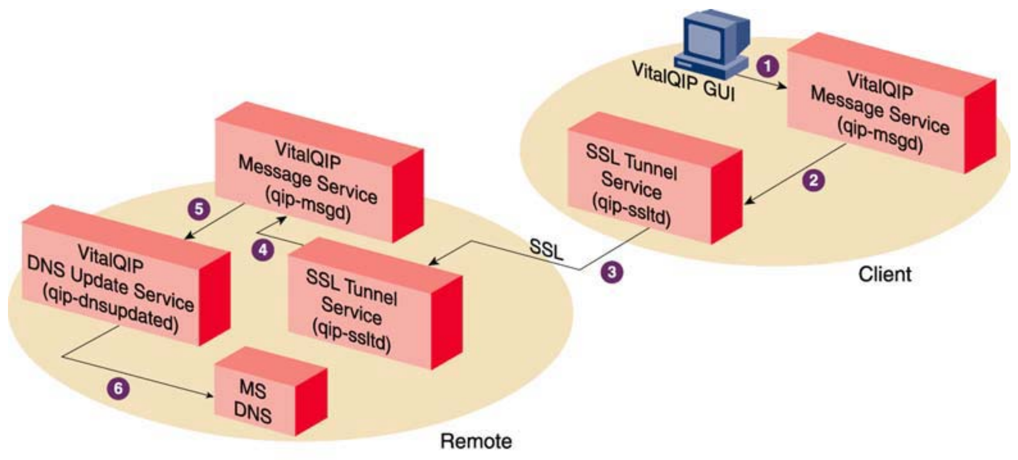

VitalQIP GUI
1
VitalQIP Message Service (qip-msgd)
2
SSL Tunnel Service (qip-ssltd)
SSL
3
Client
VitalQIP Message Service (qip-msgd)
5
4
VitalQIP DNS Update Service (qip-dnsupdated)
SSL Tunnel Service (qip-ssltd)
6
MS DNS
Remote

Figure 25 Dynamic update message flow without DNS Update Service

The following message flow occurs:

1. The VitalQIP GUI needs to send an update to a Windows secure DNS zone, and sends a conduit connection request (with the IP address of the Microsoft DNS server to update) and data to the local Message Service. The Message Service is bound to the Message Service port on the loopback interface where it only accepts TCP cleartext communication.
2. The local Message Service makes a conduit connection to the local SSL Tunnel. The Message Service knows where to communicate with the SSL Tunnel, because the tunnel is registered with it. The SSL Tunnel only accepts TCP cleartext communication on the loopback interface.
3. The local SSL Tunnel makes a connection to the remote SSL Tunnel. The remote SSL Tunnel is running on the Message Service port on the remote machine's network interface. Encrypted and authenticated TCP communication occur, where both the initiator and recipient must be able to authenticate each other's credentials (certificates). If they cannot, the connection attempt fails.
4. The remote SSL Tunnel removes the encryption, and sends the conduit request to the remote's Message Service running on the loopback interface and the Message Service port. The Message Service only accepts TCP cleartext communication from the loopback interface.
5. The remote Message Service looks in its service registry, finds the MS DNS Update Service, and makes a TCP cleartext connection to it. The MS DNS Update Service only accepts TCP communication on the loopback interface.
6. The MS DNS Update Service updates the MS DNS server using GSS-TSIG or **dnscmd** commands. GSS-TSIG updates occur using the principals configured in VitalQIP and stored on the remote during a push.

   Important! The information is passed along as a VitalQIP message until it reaches the MS DNS Update Service. This service uses the GSS-TSIG update protocol to apply the update to the DNS server. To completely secure the update to the DNS server, it is recommended that SSL be enabled for communication between the VitalQIP components. Otherwise, update messages to the MS DNS Update Service could be spoofed.

□

Dynamic update messaging using the DNS Update Service

If the **Use DNS Update Service** global policy is set toTrue, update information is passed through the DNS Update Service. The DNS Update Service forwards the information to the MS DNS Update Service to be applied to the Microsoft DNS server.

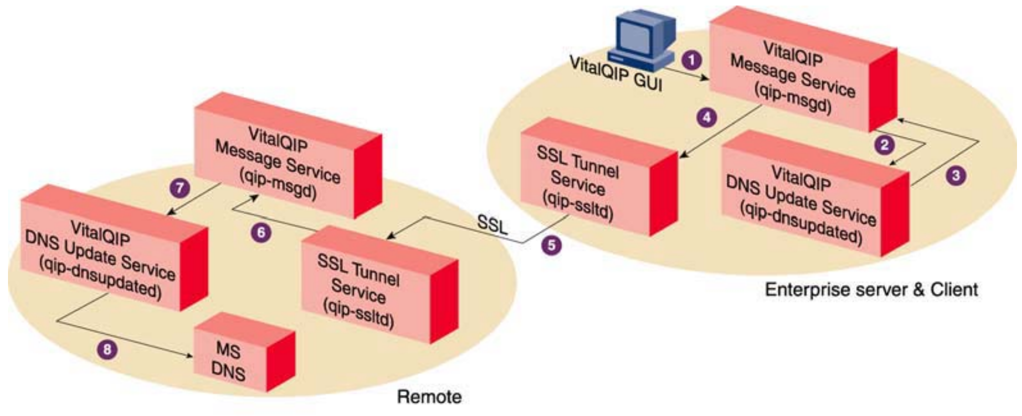

VitalQIP GUI
1
VitalQIP Message Service (qip-msgd)
4
2
3
SSL Tunnel Service (qip-ssltd)
VitalQIP DNS Update Service (qip-dnsupdated)
SSL
5
Enterprise server & Client
VitalQIP Message Service (qip-msgd)
7
6
VitalQIP DNS Update Service (qip-dnsupdated)
SSL Tunnel Service (qip-ssltd)
8
MS DNS
Remote

Figure 26 Dynamic update message flow using the DNS Update Service

> **Important!** SSL is recommended for the VitalQIP communication to ensure a secure zone.

The following steps occur:

1. VitalQIP notices the **Use DNS Update Service** policy is set to True and sends a UDP message (of either DNSUpdateObject or DNSUpdateRR message type) to the loopback interface.
2. The Message Service sends the message to the DNS Update Service.
3. The DNS Update Service needs to send an update to a Windows secure DNS zone, and sends a conduit connection request (with the IP address of the MS DNS server to update) and data to the local Message Service. The Message Service is bound to the Message Service port on the loopback interface where it only accepts TCP cleartext communication.
4. The local Message Service makes a conduit connection to the local SSL Tunnel. The Message Service knows where to communicate with the SSL Tunnel, because the tunnel is registered with it. The SSL Tunnel only accepts TCP cleartext communication from the loopback interface.
5. The local SSL Tunnel makes a connection to the remote SSL Tunnel. The remote SSL Tunnel is running on the Message Service port on the remote machine's network interface. Encrypted and authenticated TCP communication occur where both the initiator and recipient must be able to authenticate each other's credentials (certificates). If they cannot, the connection attempt fails.

6. The remote SSL Tunnel removes the encryption, and sends the conduit request to the remote's Message Service running on the loopback interface and the Message Service port. The Message Service only accepts TCP cleartext communication from the loopback interface.
7. The remote Message Service looks in its service registry, finds the MS DNS Update Service, and makes a connection to it. The MS DNS Update Service only accepts TCP cleartext communication on the loopback interface.
8. The MS DNS Update Service updates the MS DNS server using GSS-TSIG or **dnscmd** commands. GSS-TSIG updates occur using the principals configured in VitalQIP and stored on the remote during a push.

☐

QIP Update Service policy to support secure zone updates

DHCP updates that come from the DHCP server to the QIP Update Service for secure zones on an MS DNS server do not contain information to determine which principal (weak or strong) to use for the secure update.

The QIP Update Service looks up this information in the VitalQIP database before passing the update along to the DNS Update Service. The process involves checking the **Allow DHCP Clients to Modify Dynamic Resource Records** policy on the object, on the subnet, and the global policy. If this process is slowing down the QIP Update Service, you can set the **DNSUpdatePrincipal** policy in the QIP Update Service section of *qip.pcy* to be either weak or strong, rather than the default value of derived. If it is set to weak or strong, the QIP Update Service always uses that principal for the updates. If it is set to derived (the default), the value continues to be looked up in the VitalQIP database. For more information on QIP Update Service policies, refer to “VitalQIP QIP Update Service policies”, on page 104.

□

7 Advanced DNS configurations

Overview

Purpose

This chapter covers advanced configuration of DNS servers. DNS configuration instructions are included for IBM, Windows, and Lucent DNS servers.

Contents

This information presents the following topics.

VitalQIP BIND support

For server class DNS, the **Server Type** field in the Server Profile window allows you to select support for either Lucent DNS, Windows 2000, BIND 8.x, or BIND 9.

Before assigning DNS server types in the Server Profile, it is important for you to know which DNS server type is installed on your server. There is no way for VitalQIP to automatically alert you. You must research this yourself.

Lucent DNS 3.x, Lucent DNS 4.x, BIND 8.x, and BIND 9 provide a configuration file called *named.conf*. The data files created by BIND have not changed between the BIND 8.x and 9.x versions (with the exception BIND 9-x requires a "$TTL" statement for every data file). There are entirely new areas of configuration in the configuration files, such as access control lists and categorized logging. Many options that previously applied to all zones can now be used selectively. You must ensure the DNS server version in the Server Profile matches the version installed on the server in question. If not, the DNS server will not function properly.

You can add extensions to *named.conf* (BIND 8.x and 9.x) file through the VitalQIP client in the DNS **Server Profile|Corporate Extension** fields. For more information on setting up the Corporate Extension, see "Organizations" in Chapter 3 of the *VitalQIP User's Guide*.

Important! For additional information on BIND, see *DNS and BIND, Fourth Edition* by Paul Albitz & Cricket Liu, published by O'Reilly & Associates.

☐

Lucent DNS directives

In Lucent DNS 3.1 and above, you can specify Lucent-specific directives in a `qddns` block inside the `options` directive. Most directives in the `qddns` block are supported in Lucent DNS 3.1. In Lucent DNS 4.0, some directives are supported or supported on a limited basis. Those directives will be identified as such.

The `qddns` block cannot be empty. If there is not at least one directive in it, a syntax error occurs at startup and appears in the system log.

The valid directives for the `qddns` block are shown:

```
options {
   qddns {
       notify-after-load yes_or_no;
       allow-axfr-during-dynamic-update yes_or_no;
       axfr-client-timeout number
       axfr-serial-diff-tolerance number;
       loopback-reload yes_or_no;
       reverse-wins yes_or_no;
       close-debug-stream-at-notrace yes_or_no;
       allow-edit-dynamic-zone yes_or_no;
       roundrobin yes_or_no;
rrset-order yes_or_no;
 allow-secondary-update yes_or_no;
retry-tcp-on-truncate yes_or_no;
remove-cname-on-cname-and-other-data-error yes_or_no;
      };
```

Important! Lucent DNS 4.0 does not support the `close-debug-at-notrace`, `axfr-client-timeout`, `axfr-serial-diff-tolerance`, `reverse-wins`, `loopback-reload`, `allow-edit-dynamic-zone`, `allow-axfr-during-dynamic-update`, use-ixfr *yes_or_no;* and `roundrobin` directives.

Important! The `rrset-order`, `allow-secondary-updates`, and `remove-cname-on-cname-and-other-data-error` directives are only supported in Lucent DNS 4.0.

Each directive shown in the above example is described as follows:

notify-after-load

In Lucent DNS 3.1, BIND 8.2.2 Patch 7 notifies all the slave servers after loading the zones. The Lucent DNS default is to keep notification off. To make the master server compliant with BIND 8.2.2 Patch 7, specify `notify-after-load yes` in the `qddns` block in the `options` section. If `notify-after-load no` is specified, notifies are suppressed after load.

In Lucent DNS 4.0, this directive only allows the possibility of notifies occurring (for instance, notifies can be turned off at the zone level) if `notify-after-load yes` is specified.

allow-axfr-during-dynamic-update

This directive is only supported in Lucent DNS 3.1. According to BIND 8.2.2 Patch 5, the master server shuts down an `axfr tcp` connection immediately if a dynamic update arrives for that zone while the master server is serving an outgoing `axfr` to a slave server. (The error message "premature EOF, fetching <*zone_name*>" appears in *syslog* or *eventlog* on the slave server.) At sites with a lot of dynamic update activity, there may be a problem getting zone transfers to the slave server. If you want the master server to allow outgoing axfr while a dynamic update is in progress, specify `allow-axfr-during-dynamic-update yes` in the `qddns` block of the `options` section (the default is "No").

> **Important!** This directive is significant only for a master server serving an outbound AXFR. If you specify this directive on a master server, you should also set the directive `axfr-serial-diff-tolerance` on the slave server. Otherwise, the slave server may never load the zone file. If this directive is turned "on", the server may crash.

axfr-client-timeout

This directive is only supported in Lucent DNS 3.1. When a slave server performs a zone transfer via AXFR from a master server, the hard-coded read timeout value is 120 seconds. In some situations, if the link is very slow, 120 seconds may not be enough. With this directive, you can specify a read timeout value (in seconds) of up to 2,147,483,647.

> **Important!** This directive is significant only in `client-side axfr` (where a slave zone is getting pulled from a master server).

axfr-serial-diff-tolerance

This directive is only supported in Lucent DNS 3.1. When dynamic updates occur on a master server at the same time as a slave server is performing a zone transfer (for example, over a slow link), the serial numbers in AXFR first and last SOA become different. Under such circumstances, the slave server deletes the *tmp* file where the zone is transferred and tries again; subsequent tries fail for the same reason, thereby making the zone unavailable to the slave server.

On the slave server, the `axfr-serial-diff-tolerance` directive can be used to ignore or set a tolerance for the difference in first and last SOA serial numbers. The directive takes an integer as its value (the maximum value is 2,147,483,647). If the difference between the serial numbers in AXFR first and last SOA is less than or equal to the tolerance, the difference is ignored, the transferred zone in *tmp* file is renamed to the zone file, and the slave zone is loaded. A message is also written to the log file if the serial numbers differ.

> **Important!** This directive is significant only in `client-side axfr` (where the slave zone is getting pulled from a primary). Furthermore, this directive is meaningful if `allow-axfr-during-dynamic-update` is also set to "yes" on the primary server.

loopback-reload (Windows only)

This directive is only supported in Lucent DNS 3.1. Setting the value to "yes" causes the following: when the service thread receives the reload command, instead of calling the routine **ns_reload()** directly, it creates a TCP connection to the control socket on which the server is listening (in worker thread) and issues the reload command. It is similar to calling **ndc** with the **-c** option. This option *must* be accompanied with the `controls` directive:

```
controls { inet 127.0.0.1 port (any_port_number) allow { 127.0.0.1; };
```

This directive is a workaround for crashes caused by memory corruption during a reload. The default value is "no".

reverse-wins

This directive is only supported in Lucent DNS 3.1. Setting the value to "yes" (default) allows the DNS server, after a successful WINS query, to insert A and PTR RR in DNS memory cache (not to be confused with DNS cache), which overwrites any existing PTR record. If the `wins-ttl` directive is 0, any existing PTR will be removed. A value of "no" disables reverse WINS query altogether.

allow-edit-dynamic-zone

This directive is only supported in Lucent DNS 3.1. If a dynamic zone is modified externally (rather than by dynamic update), at reload, **named** dumps that zone to make sure that external modification is overwritten. Since this creates a problem with QIP Push, the default behavior is changed to not dump the zone (the default value is "yes"). If BIND compatibility is desired, use the directive `allow-edit-dynamic-zone no`. If `allow-edit-dynamic-zone` is set to "no", the message "dumping the dynamic zone: <*zone*>" appears in *syslog* (if the zone is modified and is getting overwritten).

close-debug-stream-at-notrace

This directive is only supported in Lucent DNS 3.1. If "yes" is specified for this directive, and if debug is turned off, the associated debug *log* file will be released from use by **named**. The message `closing debug stream` is written to *named.run* (or whatever channel is configured for debug). If the value is "no" (the default), the debug file continues to be marked as "in use by a program" and the file cannot be modified until **named** is stopped.

roundrobin

In Lucent DNS versions prior to Lucent DNS 4.0, the `roundrobin` directive with a value of "yes" indicates round robin is used for this server.

use-ixfr

This directive is only supported in Lucent DNS 3.1. The `use-ixfr` directive with a value of "yes" allows the server to perform incremental zone transfers.

retry-tcp-on-truncate

This directive is supported in Lucent DNS 3.1 and 4.x. `The retry-tcp-on-truncate` can be added to the `options` section in `qddns` block. If set to "no" (default is "yes"), **named** does not make the TCP connection. However, it replies with the partially received response.

Lucent DNS 4.0 directives

rrset-order

rrset ordering is configured by adding the `rrset-order` directive to the `qddns` block of the `options` block. If `rrset-order yes` is specified rrset ordering is turned on (default). If `rrset-order no` is specified, no rrset ordering is done. Otherwise, the default BIND 9 style rrset-ordering is done, which is called "cyclic random" in the BIND 9 documentation.

In this operation, a resource record is chosen at random from the rrset at the first query for that rrset, the subsequent one is returned on the next query, and so on, with wraparound.

allow-secondary-update

If `allow-secondary-update` is set to "yes", the server accepts dynamic updates to secondary zones. If this server is set to "no", the server does not accept dynamic updates to secondary zones. The default is yes.

Important! Lucent DNS 4.0 cannot be configured to accept dynamic updates to secondary zones via command line parameters.

remove-cname-on-cname-and-other-data-error

The `remove-cname-on-cname-and-other-data-error yes` directive can be added to the `options` section inside `qddns` block to remove a CNAME entry if "CNAME and other data" error is detected in the zone, and the zone is loaded. If SNMP is active, a SNMP trap is also triggered.

☐

BIND 9.x support

VitalQIP supports BIND 9.x by providing DNS server type, BIND 9 in the Server Profile. The BIND 9 server type is similar to the BIND 8.x server type except for the additional parameters **RNDC Key**, **RNDC Path**, and **Generate rndc.conf** in the Server Profile. The RNDC Key parameter's key value is base 64 encoded and placed in the *named.conf* file, so **rndc** can communicate with the BIND 9.x server. The **RNDC Path** parameter defines the pathname for the **rndc** executable.

Important! When BIND 9 configuration files are pushed on UNIX, the *named.conf* file is placed in the same directory as the zone files by default.

Important! On Windows, the *named.conf* and zone files are pushed to separate directories. The *named.conf* file is pushed to the *<Default_Directory>\etc*, and the zone files are pushed to the **<Default Directory>**. In order to ensure the BIND server receives its data, the **Default Directory** parameter in the BIND 9 Server Profile must match the following registry key or the BIND 9.x server does not receive its data**:**

```
KEY_LOCAL_MACHINE\SOFTWARE\ISC\BIND\InstallDir
```

Important! The **Generate rndc.conf** parameter defaults to False on existing Bind 9 servers that have been upgraded from previous versions of VitalQIP.

VitalQIP handles BIND 9.x transfers differently than BIND 8.x. BIND 9.x transfers are handled in the following ways:

- A TTL value required for every record in a zone. If that $TTL statement is supplied in the **Prefix Domain Extensions** of the Domain or Reverse Zone Profile, the value is used. If the directive is not specified, VitalQIP uses the Default TTL value from the Domain Profile and adds the $TTL statement to the beginning of the zone file.
- If a `control` statement is not specified, VitalQIP automatically adds the following lines when **Network Services|DNS Generation** is used:

```
controls {
 inet 127.0.0.1 allow { localhost; } keys { rndc-key;};
 };
   key rndc-key {
 algorithm "hmac-md5";
 secret "<RNDC Key Parameter Value from Server Profile base 64
 encoded>";
};
```

Important! For BIND 9.x servers only, you must have a user exit to remove *db.<zone>.jnl* files if the server uses dynamic zones. This prevents errors when a zone is loaded.

- The *named.conf* file for BIND 9 servers are only read and written by root (mode 600). The `rndc key` is written to the *named.conf* file. This file needs to be protected. (In contrast, BIND 8.x server *named.conf* files are read _everyone and write _root.)

- The push to a BIND 9 server invokes an **rndc** reload so the server rereads its conf and zone files. In contrast, BIND 8 servers are sent the **kill -HUP** signal. The push to Lucent DNS 4.0 invokes `rndc qddns-push` to remove *db.<zone>.jnl* files.

☐

Windows Internet Naming Service gateway support

The Windows Internet Naming Service (WINS) is used to resolve NetBIOS names using the name and address in the zone file. Standard DNS does not perform any lookups within a WINS server. By using the WINS gateway, any query not resolved in the DNS Service is optionally sent to a WINS server. The WINS server then returns the IP address to the requestor as shown in Figure 27.

Important! The WINS directives are supported only on Lucent DNS 3.X servers. The WINS directives are not supported on Lucent DNS BIND 4.x.

☐

The forward record lookup

Figure 27 shows how forward record lookup works.

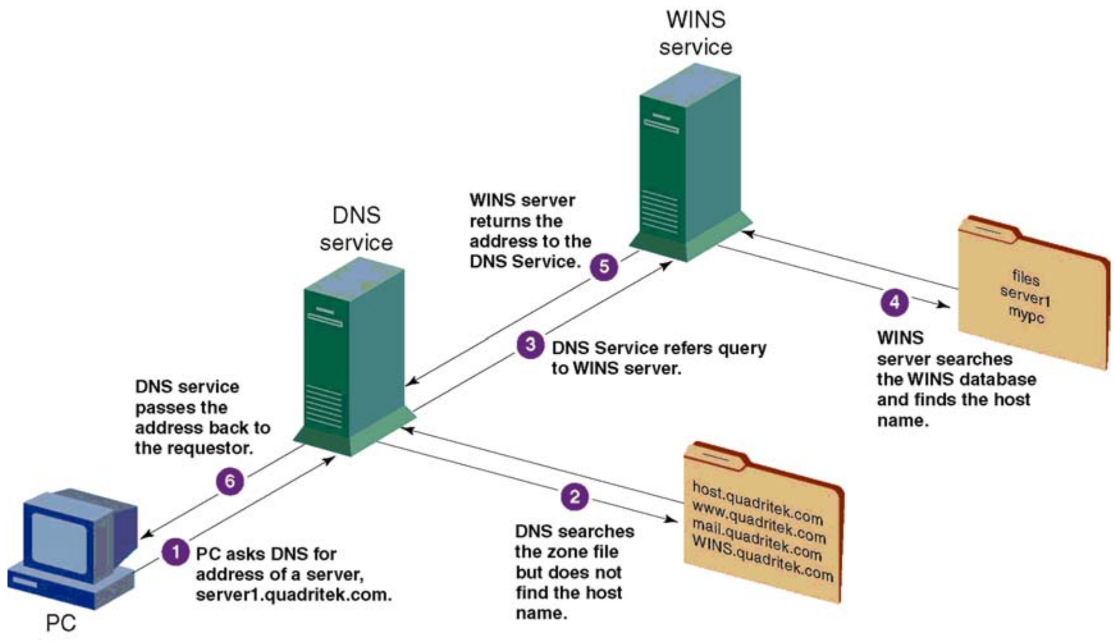

WINS service
DNS service
WINS server returns the address to the DNS Service.
5
files
server1
mypc
4
WINS server searches the WINS database and finds the host name.
3
DNS Service refers query to WINS server.
DNS service passes the address back to the requestor.
6
2
host.quadritek.com
www.quadritek.com
mail.quadritek.com
WINS.quadritek.com
DNS searches the zone file but does not find the host name.
1
PC asks DNS for address of a server, server1.quadritek.com.
PC

Figure 27 Forward record lookup process

The following steps occur:

1. A requestor (shown as a PC) requests an address from the DNS server.
2. DNS searches the zone file to resolve the address to a hostname (#2).

3. A switching mechanism is added to determine if a DNS query has failed, the query is sent to the WINS server.
4. A routine is then called that formulates and sends the WINS packet (forward or reverse). The new routine creates a WINS packet, and sends the packet to the correct WINS server.
5. It then waits for a response from the WINS server.
6. The response from the WINS server is converted into a standard DNS packet and sent back to the requestor. Results from the WINS query can be stored in the DNS Service cache enabling a faster subsequent lookup (see “wins-ttl”, on page 234).

The reverse record lookup

Figure 28 shows how reverse record lookup works.

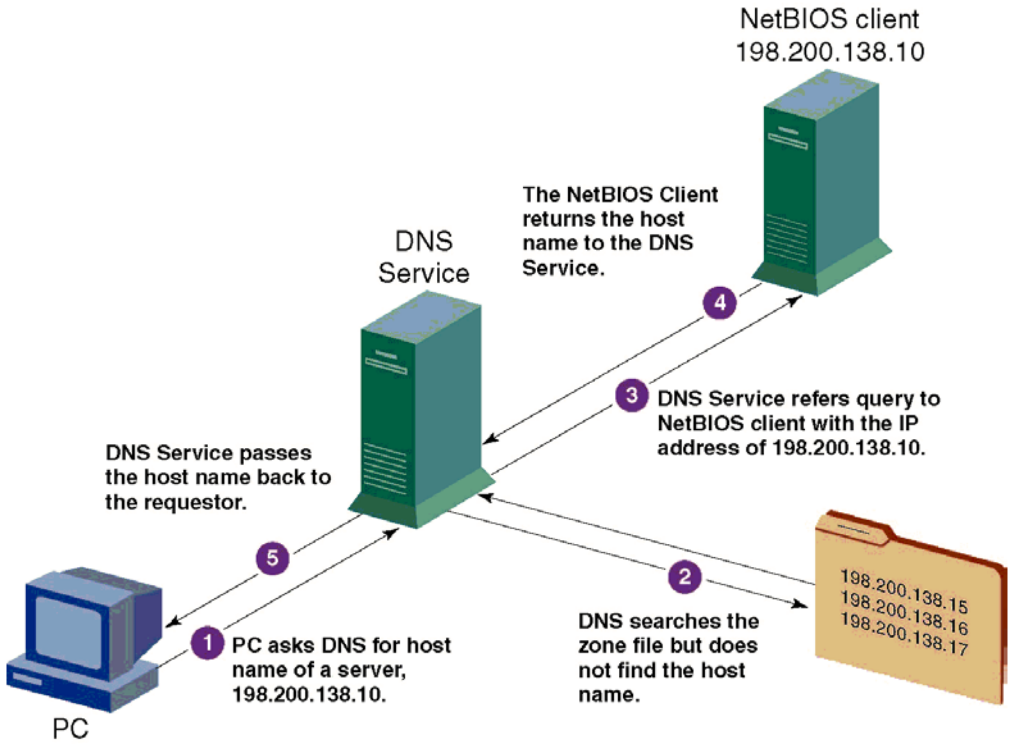

NetBIOS client
198.200.138.10
DNS
Service
The NetBIOS Client returns the host name to the DNS Service.
4
3
DNS Service refers query to NetBIOS client with the IP address of 198.200.138.10.
DNS Service passes the host name back to the requestor.
5
2
198.200.138.15
198.200.138.16
198.200.138.17
1
PC asks DNS for host name of a server, 198.200.138.10.
DNS searches the zone file but does not find the host name.
PC

Figure 28 Reverse record lookup process

The following steps occur:

1. A requestor requests a hostname from the DNS server.
2. DNS searches the zone file to resolve the address to the hostname - the reverse of the lookup.
3. A switching mechanism to determine if a DNS query has failed, the query is sent to the NetBIOS client to resolve the IP address to a hostname.
4. DNS waits for a response from the NetBIOS Client.
5. The response from the WINS server is converted into a standard DNS packet and sent back to the requestor. Results from the WINS query can be stored in the DNS Service cache enabling a faster subsequent lookup (see “wins-ttl”, on page 234).

□

Configure the WINS Gateway on Lucent DNS 3.x servers

In Lucent DNS 3.1, the WINS gateway can be configured on a per zone basis, by specifying the WINS parameters in the `qddns` block inside the `zone` directive. You can place the directive manually in the *named.conf* file using the **qipdnscnfuserexit** or through the VitalQIP client. If you are using the client, access the **Zone Options** tab in the Domain Profile and edit the **zone block of named.conf** field.

For each forward zone, you can specify a different WINS server. You can also specify multiple WINS servers per zone. A WINS query works for both master and slave zone types, provided the server is authoritative for the zones.

The `valid` directives for the `qddns` block inside the `zone` directive are shown as follows:

```
qddns
{
   /*
    * maximum 16 IP addresses for the WINS servers can be provided. At
    * least one WINS server must be present for forward zone.
    */
    wins-servers { 198.200.138.200; 198.200.138.247; };

    wins-domain "foo.com"; /* must be present for reverse zone */
                           /* meaningless for forward zone */

    wins-ttl 10;                  /* optional */
    wins-bind-to-netbios-port no; /* optional for reverse zone */
};
```

Important! The `qddns` block cannot be empty. If there is not at least one directive in it, a syntax error occurs at startup and appears in the system log.

The descriptions for each directive follow:

wins-servers

This directive specifies IP addresses of WINS servers (separated by semicolons inside left and right curly braces). This directive is only meaningful for forward zones. A maximum of 16 WINS servers can be specified with this directive and at least one WINS server must be specified for a forward zone. If it does not exist in the `qddns` block for a forward zone, **named** might be authoritative for the zone but any unresolved name is not forwarded to any WINS server. If a query is not resolved, each WINS server is contacted in sequence until either the query succeeds or the list is exhausted.

wins-domain

This directive is only meaningful for reverse zones. For a reverse query, **named** queries the machine with the IP address (the NetBIOS client) to get the host name. As WINS has no concept of a domain, the specified domain is appended at the end of the name that it received from the NetBIOS client. The value for this directive is a double quoted string. This directive must be present in the reverse zone. If it does not exist in the `qddns` block for a reverse zone, then **named** might be authoritative for the zone but an unresolved IP address will not be forwarded to any NetBIOS client.

wins-ttl

The value for this optional directive is a number. The minimum value is 0 (default) and the maximum value is 2,147,483,647. If the value of this directive is greater than 0, it has two effects:

1. The master server adds the RR in the internal record cache, which means that any subsequent query for that RR will not query the WINS server, but rather return an answer from memory. If a secondary server performs a zone transfer of the zone, the resource records appear in the zone. The resource records are only cleared from the master server's memory after a restart or a reload. However, if you perform a dynamic update to the WINS zone, the cache is written to the zone file at the maintenance time of the zone or at the clean shutdown of **named**.
2. If a caching name server gets the RR from this server, it expires the RR from its memory after the amount of seconds specified in `wins-ttl`. If the value of this directive is 0, the master server does not add it in memory, and each query is forwarded to the concerned WINS server.

wins-bind-to-netbios-port

Before sending a WINS reverse query to a NetBIOS client, **named** always binds to a NetBIOS port (because some old Windows 95/98 clients always send the packet to the port where they receive the request). However, in some versions of **Winsock** (on the machine where **named.exe** is running), the WINS reverse query does not work if **named** binds to the NetBIOS port. A new Boolean (yes or no) directive `wins-bind-to-netbios-port` in the `qddns` section of the zone has been added for this purpose.

Important! By default **named** always binds to the port. If you do not want **named** to bind to the NetBIOS port, specify “no” for this directive. Use “no” for this directive only if WINS reverse query does not work at all. However if you do so, a reverse query to Windows 95 machines using **Winsock**, version 1 does not work. This directive is significant for reverse zones.

Sample Directives

The following are sample directives:

```
/*
 * Forward Zone.
 * Note: no wins-domain is present. It is valid to have wins-domain but
 * it is meaningless for forward zone. For forward zone, the zone name,
 * in this case “foo.com” is used as domain. This is only used by the
 * name server to identify the WINS server to send the query to, WINS
 * has no concept of zone.
 */
zone “foo.com” in
{
    type master;
    file “db.foo.com”;
    qddns
    {
        wins-servers { 198.200.138.200; 198.200.138.116; 1.2.3.4; };
        wins-ttl 10;
    };
};

/*
 * Reverse Zone.
 * Note: no wins-servers is present. It is valid to have wins-servers but
 * it is meaningless for reverse zone as named talks to the machine
 * with the IP directly to get the name. As WINS server has no concept
 * of domain, you must specify one. named will add this domain with the
 * host name it gets from the WINS server. You can specify anything as
 * wins-domain, but you should use the valid domain for consistency.
 */
zone “138.200.198.in-addr.arpa” in
{
    type master;
    file “db.198.200.138”;
    qddns
    {
        wins-domain “foo.com”;
        wins-ttl 10;
        wins-bind-to-netbios-port no; //qddns 3.0 build 16+
    };
};
```

Now you must create minimal and valid zone files *db.foo.com* and *db.198.200.138*. An example of *db.foo.com* is shown:

```
$TTL 100
foo.com. IN SOA ns.foo.com. foo.foo.com. (
        45          ; serial
        10800       ; refresh 3 hrs
        3600        ; retry 1 hr
        604800      ; expire 1 week
        86400 )     ; min ttl 1 day
foo.com.    IN  NS  ns.foo.com.
ns.foo.com. IN  A   127.0.0.1
```

And the reverse zone file *db.198.200.138* might look like:

```
$TTL 1000
138.200.198.in-addr.arpa. IN SOA ns.foo.com. muquit.foo.com. (
        10          ; serial
        10800       ; refresh 3 hrs
        3600        ; retry 1 hr
        604800      ; expire
        86400)      ; min ttl 1 day
138.200.198.in-addr.arpa.   IN  NS  ns.foo.com.
```

If there is any error in the zone file, the name server will not load the file. It will therefore not be authoritative for the zone and will not forward the query to the WINS server.

□

IBM AIX DNS support

VitalQIP provides dynamic IP addressing solutions in conjunction with IBM's DNS servers running on AIX. Additionally, IBM provides a connection between their DNS services running on the IBM Risc/6000 platform.

☐

IBM AIX 5.3 DNS server support

The AIX 5.3 operating system provides the */usr/sbin/named8* file. To use a different *named* file, create a link for */usr/sbin/named* to the *named8* file and modify the *$QIPHOME/userexits/qipdnsuserexit* script to use the correct version.

The user exit, **qipdnsuserexit**, is provided to check the version of BIND you are using. This user exit updates the *named.boot* or *named.conf* files when the DNS update to the server is executed via the **Network Services|DNS Generation**. The **qipdnsuserexit** user exit must be checked to ensure the correct version of BIND is commented out to match the version of */usr/sbin/named*. The following example is what the user exit adds to the configuration file.

For BIND 8.x, the line "update-security controlled" is added to the zone stanza. For example,

```
zone "austin.ibm.com" in {
   type master;
   file "austin.com";
   allow-update {any;};
   update-security controlled;
```

"Controlled" specifies the level of dynamic update security. "Controlled" means that the new records are added to the zone without the need of a proper signature, but any changes to existing domain name entries that contain a KEY record are verified with the update signature. For more information, see *DNS and BIND, Fourth Edition* by Paul Albitz & Cricket Liu, published by O'Reilly & Associates.

☐

Manage Windows 2000 DNS servers

Microsoft supplies a DNS server with its Windows 2000 operating system, and a simple configuration tool that can be used to configure a single DNS server. VitalQIP can be used as a replacement to this configuration interface, to manage single or multiple Windows 2000 DNS servers from a centralized platform. VitalQIP can also manage an active directory integrated or file-based Windows 2000 DNS server (both secure and non-secure zones).

Important! VitalQIP also supports Windows 2003 DNS servers in the same way as it does Windows 2000 DNS servers.

In addition to storing their zone information on disk, Windows 2000 DNS servers can store this information in Active Directory. When VitalQIP manages Active Directory-based Windows 2000 DNS servers, it stores the zone information in the directory server. Standard disk-based primaries save generated boot and zone files to disk by setting registry entries through VitalQIP (even for file-based VitalQIP servers).

Important! After you install the Windows 2000 DNS service, it is strongly recommended that you apply or reapply the latest Windows 2000 Service Pack and any patches that Microsoft has shipped for the DNS Service. You also need to restart your computer after you install the patches.

VitalQIP Remote Service

The Remote Service is used to generate the Microsoft DNS configuration files, and to restart the DNS server after a configuration file is changed. You must install the VitalQIP Remote Service on the Windows 2000 DNS server. See the instructions in the *VitalQIP Installation Guide* on how to install these components.

Zone information

The VitalQIP client allows Windows 2000 DNS zone options to be set in the Domain Profile, the Reverse Zone Profile, and the Non-Managed DNS Profile. For more information on Windows 2000 DNS zone options, refer to “Domains” in Chapter 3 of the *VitalQIP User’s Guide*.

Services

If the server is set to boot from disk, the boot file and zone files are generated in the same fashion as they are generated for Windows 2000. However, all server and zone options are set through **dnscmd**. See the Microsoft documentation for more information on the **dnscmd** utility.

☐

Configure Windows 2000 DNS server in VitalQIP

When to use

This describes how to configure a Windows 2000 DNS server in VitalQIP.

Procedure

To configure the Windows 2000 DNS server within VitalQIP, follow these steps:

1. Select **Infrastructure|Server**. The Server Profile Option window opens.

2. Click **OK** (since **Add New Server** is the default). The Server Profile: Add window opens.

3. Select **DNS** from the **Server Class** drop-down list.

4. Select **Windows 2000 DNS** from the **Server Type** drop-down list.

5. In the **Host Name** field, either select a server name from the existing host list, or enter a new server host name. You can enter a maximum of 63 alphanumeric characters beginning with an alphanumeric. The host name may include the dash (-) and underscore (_) characters, but cannot contain plus signs (+) or exclamation marks (!).

6. Enter a domain in the **Domain Name** field by clicking on **...** to open the Domain Option: Select window and select a domain from the **Existing Domain List**.

7. For the parameters, the server parameters can be defined based on your functional needs. Refer to "DNS servers" in Chapter 3 of the *VitalQIP User's Guide* for more information.
   - The **Default Directory** is a read-only field and is automatically set to *%systemroot%\system32\dns*.
   - **Prefix** and **Suffix Corporate Extensions** - These are free form text fields. The value is always expected in **dnscmd** format. See the Microsoft documentation for more information on the **dnscmd** utility.
   - Windows 2000 DNS Directory based servers must be fully managed by setting the **Fully Managed** option in the Server Profile for DNS servers to "True".
   - The **Create "db.127.0.0"** option should always be set to "False". This zone is automatically created by Windows 2000.
   - **Boot Method** - Valid values are "File" based or Active "Directory" based. If Boot Method is set to Directory, Secure DNS updates for Windows 2000 can be enabled with the **Secure DNS Updates** parameter. "Windows 2000 DNS secure zones support", on page 241 for details.

8 Define the rest of your infrastructure including domains, networks, subnets, and IP objects. Be sure to attach domains to this DNS server.

9 WINS forwarding support for forward zones is enabled by placing the following Microsoft-specific Resource Record into the **Postfix of zone db file** field of the **Domain Profile|Zone Options** tab, under **Extensions**:

```
@  0  IN  WINS  <IP_address_of_WINS_server> <IP_address_of_next_WINS_server>
```

Examples:

```
@     0    IN    WINS    123.123.123.123
@     0    IN    WINS    123.123.123.123  123.123.123.234
```

10 WINS forwarding support for reverse zones is enabled by placing the following Microsoft specific Resource Record into the **Postfix of zone db file** field of the **Reverse Zone Profile|Zone Options** tab, under **Extensions**:

```
@  0 IN  WINS-R  <domain_name_to_be_appended_to_the_name_returned>
```

Example:

```
@     0    IN    WINS-R    quadritek.com
```

11 Use **Network Services|DNS Generation** to push data to this DNS server.

12 The DNS server should automatically start. If it does not, use **Control Panel|Services** to start the Windows 2000 DNS server.

END OF STEPS

Windows 2000 DNS secure zones support

VitalQIP can manage and update secure zones on a Windows 2000 DNS server. This includes all zones, all records, changed zones only, and changed resource records.

Important! VitalQIP also supports secure zones with Windows 2003 in the same way as it does Windows 2000.

Important! Lucent does not recommend you manage a zone with both Windows 2000 and Lucent DNS servers as primaries. Lucent and Microsoft implement multi-mastered zones differently. Microsoft DNS servers can only replicate their changes among other Microsoft Active Directory integrated DNS servers unless the **qip-syncexternal** command is run in a scheduled fashion and that information is sent to the Lucent DNS servers. This volume of transfers is not recommended.

Secure zone updates

With VitalQIP support of SSL-enabled secure message routes, dynamic updates can be made of objects located in a secure Windows 2000 zone. These updates can be sent securely via SSL to the remote server on which the DNS server is running. They can then be applied to the DNS server by the locally running MS DNS Update Service.

The path that the message takes to get to the remote server varies, depending upon the setting of the **Use DNS Update Service** global policy. If this is set to False, VitalQIP sends the message (via the Message Service and/or SSL Tunnel Service) directly to the remote server. If there are proxies specified for the server, VitalQIP attempts to route the message through those proxies. For more information on secure message routes and setting up SSL, refer to Chapter 6.

□

Background

Before you set up secure zones with Window 2000 DNS, you should understand what Microsoft means by a secure zone and what is entailed in the management of secure zones.

A zone can be marked as secure only if it is “directory integrated”. Non-directory integrated zones cannot be secured.

When a secure dynamic update is made to a secure zone, the security verification happens in two stages. First, the GSS-TSIG protocol is used to verify the identity of the updater. Second, the Windows 2000 DNS server receives the update and uses the user’s security context to update Active Directory with the new information. At this stage, Active Directory’s security mechanism is invoked.

Active Directory maintains access control information with each entry in the Active Directory. This access control information specifies who is allowed to access it. For example, if UserA adds entries to DNS, all authenticated users can read the information. However, only administrators and UserA are allowed to modify the information.

If the access control information does not prohibit the user from making changes to the Active Directory entry that it is trying to modify, the update succeeds. If the entry had no security or did not previously exist, the access control information for the entry is updated such that only the updater (and administrators) are allowed to make changes to the entry.

There is an exception. If the user is a member of a special security group called "DNSUpdateProxy", objects created by members of the DNSUpdateProxy group have no security. Any authenticated user can take ownership of the objects.

□

Configure VitalQIP to manage Windows 2000 DNS secure zones

When to use

To configure VitalQIP to use secure zones with Windows 2000 DNS, there are several steps you must complete.

Set up Window 2000 users

To use secure zones with a Windows 2000 DNS server, you need to set up two users on your Windows 2000 Domain Controller. One user should be a normal user (referred to as the "strong user"). This user controls static objects for which VitalQIP is authoritative. The other user (referred to as the "proxy user") should be assigned to the DNSUpdateProxy security group. The proxy user allows external entities to update objects in DNS. For information about setting up these users, refer to your Windows 2000 DNS server documentation.

Once you have added the users, write down the name and password of these users. You will need them to set up your Windows 2000 DNS server.

Define a Windows 2000 DNS server with secure zone capabilities

To define a Windows 2000 DNS server with secure zone capabilities, follow these steps:

1. Select **Server** from the **Infrastructure** menu. The Server Profile Option window opens.

2. If you are modifying an existing Window 2000 DNS server, select the server from the **Existing Server List** and click **OK**. If you are adding a new Windows 2000 DNS server, select **Add New Server** and click **OK**. The Server Profile windows opens.

3. Follow the procedures discussed in "Servers" in Chapter 3 of the *VitalQIP User's Guide* for adding or modifying a server. Ensure you set the following parameters:
   - **Boot Method** is set to **Directory**.
   - **Secure DNS Updates** is set to **True**.
   - **Strong Kerberos Principal Name** is the user logon name of the strong user.
   - **Strong Kerberos Principal Password** is the password of the strong user.
   - **Proxy Kerberos Principal Name** is user logon name of the proxy user.
   - **Proxy Kerberos Principal Password** is the password of the proxy user.
   - **Use DNS domain as Active Directory domain** is typically set to **True**. Set to False if the Active Directory domain is different.
   - **Active Directory Domain** is the domain for the Active Directory, if different from the DNS domain.

4 Click **OK**.

END OF STEPS

Configure the secure domains in VitalQIP

To configure domains in VitalQIP to be secure zones, follow these steps:

1 Select **Domain** from the **Infrastructure** menu, . The Domain Option window opens.

2 Either select a domain or select **Add New Domain**.

3 Click **OK**. The Domain Profile window opens.

4 Click the **Primary/Secondary Servers** tab.

5 Select the Windows DNS 2000 server configured for secure zones from the **Existing DNS Server(s) List**.

6 Click **Add Primary** or **Add Secondary**. Server appears in the **Selected DNS Server(s) List**.

7 Expand the Windows DNS 2000 server.

8 Select **Secure DNS Updates** and select **True**.

9 Click **OK** when you have completed adding or modifying the domain.

END OF STEPS

Configure the secure reverse zones in VitalQIP

To configure reverse zones in VitalQIP to be secure zones, follow these steps:

1 Select **Network/Reverse Zone** from the **Infrastructure** menu. The Network/Reverse Zone Option window opens.

2 Either select a network or select **Add New Network.**

3 Click **OK**. The Network Profile window opens.

4 If needed, define or modify your network as directed in “Networks and subnets” in Chapter 3 of the *VitalQIP User’s Guide*.

5 Click the **Reverse Zones** tab.

6 Select a reverse zone and click **Properties**. The Reverse Zone Profile window opens.

7 Click the **Primary/Secondary Servers** tab.

8 Select the Windows DNS 2000 server configured for secure zones from the **Network/Reverse Zone**.

9 Click **Add Primary** or **Add Secondary**. The server appears in the **Selected DNS Server(s) List**.

10 Expand the Windows DNS 2000 server.

11 Select **Secure DNS Updates** and select **True**.

12 Click **OK** when have completed adding or modifying the domain.

END OF STEPS

☐

Set secure zone policies

VitalQIP allows you to determine whether dynamic objects are created in Active Directory using the strong user or the proxy user. If VitalQIP creates dynamic objects in Active Directory with the strong user, only VitalQIP is allowed to update those resource records. If VitalQIP creates dynamic objects in Active Directory with the proxy user, Windows clients can delete or update dynamic object resource records with GSS-TSIG dynamic DNS updates.

This section describes how to set this policy for all dynamic objects in a domain, all dynamic objects in a subnet, and an individual dynamic object.

☐

Set secure zone policies at global level

When to use

You can set a Global Policy that allows DHCP clients to take ownership of dynamic object resource records for all dynamic objects in a domain.

Procedure

To set the Global Policy, follow these steps:

1 From the **Policies** menu, select **Global Policies**. The Global Policies window opens.

2 Click **Dynamic DNS** to expand it.

3 Click **Windows 2000 DNS Secure Update Polices.**

4 Select **Allow DHCP Client to Modify Dynamic Object Resource Record** and set the policy to True. This enables control to be maintained at a global level if the DNS updates for DHCP clients are to be made.

END OF STEPS

□

Set secure zone policies at subnet level

When to use

You can set Windows 2000 DNS secure zone policies that allows DHCP clients to take ownership of dynamic object resource records for all dynamic objects in a subnet.

Procedure

To set the secure zone policies, follow these steps:

1. If the **QIP** tab is not displayed, select **QIP Hierarchy** from the **Management** menu. The **QIP** tab is displayed.
2. Click **Subnets** to expand the tree under it.
3. Right-click on a subnet and select **Properties**. The Subnet Profile window opens.
4. Click the **Policies** tab.
5. Select **Subnet Level** and set the policy to one of the following:
   - **False** - prevents the allow of DHCP clients to modify dynamic objects at the subnet level (that is, assigns from the strong user).
   - **Same As in Global Policies** - uses the Global Policy value to manage secure zones with Windows 2000 DNS.
   - **True** - enables the allow of DHCP clients to modify dynamic objects at the subnet level (that is, assigns from the strong user).

   Important! The **Global Policy Level** is a read only policy that shows the current Global Policy setting.
6. Change or add any necessary information.
7. Click **OK**.

END OF STEPS

Set secure zone policies at object level

When to use

You can set an object level policy that allows DHCP clients to take ownership of dynamic object resource records for an object. The **Policies** tab exists in the Object Properties, Object Profile, and Dynamic Configuration: DHCP Setup windows.

Procedure

To set the secure zone policies at an object level, follow these steps:

1 Access the Object Properties, Object Profile, and the Dynamic Configuration: DHCP Setup windows:

- Object Properties window: Expand **Subnet** in the **QIP Hierarchy** tab, right-click on a subnet, select **Object Management**, select an object, select **Object Properties** from the **Edit** menu, and click the **Policies** tab.
- Object Profile window: Expand **Subnet** in the **QIP Hierarchy** tab, right-click on a subnet, select **Object Management**, double-click on an object, and click the **Policies** tab.
- Dynamic Configuration DHCP Setup window: Expand **Subnet** in the **QIP** tab, right-click on a subnet, select **Object Management**, select an object, select **Add|Dynamic** from the **Edit** menu, change **Dynamic Configuration** to **Dynamic DHCP**, click **...**, and click the **Policies** tab.

2 Select **Object Level** and set the policy to one of the following:

- **False** - prevents DHCP clients from modifying dynamic objects at the subnet level (that is, assigns from the strong user).
- **Same As in Subnet Profile** - uses the Subnet Profile value to manage secure zones with Windows 2000 DNS.
- **True** - enables DHCP clients to modify dynamic objects at the subnet level (that is, assigns from the strong user).

□

Generate secure zones

When to use

Once you have set up your secure zone with Windows 2000 DNS, you can push the secure zones to the server.

Procedure

To push secure zones to a Windows 2000 DNS server, follow these steps:

1 From the **Network Services** menu, click **DNS Generation**. The DNS Generation window opens.

2 From the **DNS Server List**, select the Windows 2000 DNS secure zone server. The options in the Type section change.

3 Select **All Records** to generate the Windows 2000 DNS configuration and data files, including incrementing the SOA Serial Number, or select **Changed Records Only** to generate files for only changed records. If a zone has not been “pushed” previously, the file generation fails. In other words, you must perform an **All Records** push at least once before any changed records can be pushed to that zone. An error message will be displayed explaining the possible problems.

4 Select **All Zones** to update all zones or **Changed Zones Only** to update only zones that have changed.

5 Select to view the results on screen (**Screen**), push to your client machine (**Local**), or to a server (**Server**).

6 If you select **Local** or **Server**, **Default Directory** becomes active to allow you to changes where the files are generated.

7 Click **OK**. Configuration and data files for this DNS server appear in the DNS Generation Results window. If desired, you can search, copy, print, or email them.

8 Click **Exit** to close the window.

☐

ddns.conf with proxies for Windows 2000 secure servers

The *ddns.conf* format has changed to support a potential list of proxies for each Windows 2000 DNS server. The existing *ddns.conf* format will continue to be generated for existing remotes, since VitalQIP supports pushing to downlevel remotes that may have their own DNS Update Service installed. The new format of *ddns.conf* will be generated for 6.2 (and above) remotes.

DDNS.conf format changes

The following is an example of the new XML file format. It contains two servers for the reverse zone 10.in-addr-arpa. The first server is a Lucent DNS server accepting signed updates. The second is a Microsoft DNS server accepting secure updates through one proxy.

```
<ddns-conf >
<org-id>28</org-id>
     <org-name>Malvern Organization</org-name>
     <generated-by>VitalQIP Common Libraries Version 9.0 Build 32
 [PreRelease - Service Pack 1]Windows)</generated-by>
     <ddns-conf-internal-list>
        <ddns-zone  zone-name="10.in-addr.arpa">
           <is-reverse-zone>True</is-reverse-zone>
           <reverse-zone-start>10.0.0.0</reverse-zone-start>
           <reverse-zone-end>10.255.255.255</reverse-zone-end>
           <reverse-zone-length>8</reverse-zone-length>
           <ddns-server-list>
              <ddns-server  server-address="10.100.30.7">
                   <signed-update>True</signed-update>
                   <proxy-list/>
              </ddns-server>
              <ddns-server  server-address="10.100.30.248">
                   <is-ms-dns>True</is-ms-dns>
                   <signed-update>True</signed-update>
                   <proxy-list>
                        <ip-address  v4-addr="10.100.30.12">
                        </ip-address>
                   </proxy-list>
              </ddns-server>
           </ddns-server-list>
        </ddns-zone>
</ddns_conf>
```

The Microsoft signed zones do not need the principal name in this file since that information is stored on the remote server and does not need to be sent on every update.

□

Secure dynamic updates support

VitalQIP can send secure dynamic updates (GSS-TSIG) to the Lucent DNS Service using Kerberos authentication. Lucent DNS Service (version 3.1, build 8 or higher) can:

- Provide secure dynamic DNS support in a Windows 2000 Kerberos environment.
- Support secure dynamic DNS (GSS-TSIG). The Lucent DNS Service can handle secure dynamic updates sent from a Windows 2000 client, Windows 2000 domain controller, VitalQIP client, and the DNS Update Service.

  Important! VitalQIP also supports secure dynamic updates with Windows 2003 Kerberos environment in the same as it does with Windows 2000 Kerberos environment.

  Important! The term “Lucent DNS server” refers to a server running the Lucent DNS Service.

  Important! Ensure you have set up a Lucent DNS server. Refer to “DNS servers” in Chapter 3 of the *VitalQIP User’s Guide* for instructions on setting up a DNS server.

About Kerberos

Kerberos, version 5 is an authentication system developed at MIT. Kerberos is named for the three-headed watchdog from Greek mythology, which guarded the entrance to the underworld. Much of the following text describing Kerberos can be found at the Kerberos website at MIT: *http://web.mit.edu/kerberos/www/*.

Here, for example is a brief description of Kerberos complete with an explanation of the acronyms associated with the system.

“Under Kerberos, a client (generally either a user or a service) sends a request for a ticket to the Key Distribution Center (KDC). The KDC creates a ticket-granting ticket (TGT) for the client, encrypts it using the client’s password as the key, and sends the encrypted TGT back to the client. The client then attempts to decrypt the TGT, using its password. If the client successfully decrypts the TGT (that is, if the client gave the correct password), it keeps the decrypted TGT, which indicates proof of the client’s identity.

The TGT, which expires at a specified time, permits the client to obtain additional tickets, which give permission for specific services. The requesting and granting of these additional tickets is user-transparent.”

Configure Lucent DNS

Lucent DNS can be configured for secure dynamic updates and involves preparing the VitalQIP client, the DNS Update Service, as well as the Lucent DNS server. Sections “Set up a VitalQIP client for secure DNS updates”, on page 254 and “Set up the Lucent DNS server for secure dynamic updates”, on page 259 must be completed before you can use secure dynamic updates.

Secure dynamic update notes

The software expects to see server and client principal names in certain formats. Use the following rules when specifying principal names:

- On the Lucent DNS Server in the *named.conf* file, the server principal name must be in ***DNS@<server name>.<domain>*** format.

  Important! In the *named.conf* file, if you replace the server principal name with “yes”, it causes the Lucent DNS server to determine its own principal name.

- On Windows 2000, the operating system handles the client principal name format and is not required to be specified by the user.
- On UNIX, the VitalQIP client specifies its own principal name. For example the Kerberos Principal policy in **Policies|General Policies|Dynamic DNS** or in the QIP_PRINCIPAL_NAME environment variable as:

```
<client_principal_name_defined_in_Active_directory>/<host>.<domain>@<REALM>
```

The Lucent DNS server cannot resolve names by using itself when starting. It must either use another DNS server through a *resolv.conf* file or rely on entries in */etc/hosts* files. Typically, */etc/nsswitch* file needs both “file” and “dns” specified.

If you need to add the IP address of the Lucent DNS server and Windows KDC in the */etc/hosts* file, follow these steps:

- In */etc/krb5.conf* file set the default realm to the realm you are testing.
- If your DNS principal name is fully qualified when it is entered into the *keytab* file, the first host entry in the line for the Lucent DNS server’s IP address must also be fully qualified (for example, what comes back from gethostbyaddr() must match the DNS principal name, as specified in the *keytab* file).
- If you have either a VitalQIP client or VitalQIP server running on Windows 2000 machines, ensure those machines can resolve the SRV records in the `_domains` for the realm of the KDC.

□

Set up a VitalQIP client for secure DNS updates

When to use

This section describes the steps you need to follow to set up a VitalQIP client for secure DNS updates on both UNIX and Windows platforms.

Before you begin

Set up policies depending on whether the VitalQIP client updates DNS through the DNS Update Service or not.

VitalQIP client updates DNS through DNS Update Service

Set up the VitalQIP client policies as follows:

1 In the VitalQIP client, access **Policies|Global Policies**. The Global Policies window opens.

2 In the Global Policies window, click **Dynamic DNS**. A list of policies is displayed beneath Dynamic DNS.

3 Set the **Static DDNS Updates** and **Use DNS Update Service** policies to “True”.

4 Click **OK**.

5 ***UNIX only***. Ensure the following policies are set in the *qip.pcy* file in the **[VitalQIP DNS Update Service]** section:

- **KerberosPrincipal** policy to *<DNS_client_principal>/<domain>@<REALM>*. This can be overridden by the VitalQIP client with the QIP_PRINCIPAL_NAME environment variable.
- Set the **DoKinit** policy to “True”. This can be overridden by the VitalQIP client with the QIP_DO_KINIT environment variable.
- Set the **KeyTabPath** policy to the directory where *krb5.keytab* file is stored. The policy can be overridden by the VitalQIP client with the QIP_KEYTAB_PATH environment variable.

OR

VitalQIP client updates DNS but not through DNS Update Service

Set up the VitalQIP client policies as follows:

1 In the VitalQIP client, access **Policies|Global Policies**. The Global Policies window opens.

2 In the Global Policies window, click **Dynamic DNS**. A list of policies is displayed beneath Dynamic DNS.

3 Set the **Static DDNS Updates** policy to "True" and **Use DNS Update Service** policy to "False".

4 ***UNIX only***. Ensure the following policies are set:

- Set the **Principal Name (UNIX only)** policy to <DNS_client_principal>/<domain>@<REALM>. This can be overridden by the VitalQIP client with the QIP_PRINCIPAL_NAME environment variable.
- Set the **Perform Kinit (UNIX only)** to "True". This can be overridden by the VitalQIP client with the QIP_DO_KINIT environment variable.
- Set the **Kerberos Keytab Path (UNIX only)** to the directory where *krb5.keytab* is stored. This can be overridden by the VitalQIP client with the QIP_KEYTAB_PATH environment variable.

5 Click **OK**.

END OF STEPS

Configure the Message Service if Client updates DNS via DNS Update Service

The Message Service environment variable, QIPMESSAGESERVICE, must be configured if the Message Service is not installed on the same machine as the VitalQIP client. Two MessageRoutes must be added to the *qip.pcy* file (or *qip-msgd.pcy* file if your network uses individual policy file for the services). To configure the Message Service, follow these steps:

1 Use a text editor to open the *QIPHOME/qip.pcy* file (or the *qip-msgd.pcy* file if appropriate) of the machine running the Message Service.

2 In the policy file, locate the **[VitalQIP Message Service]** section and add the following line:

```
MessageRoute=DNSUpdateObject:A:0:DNS Update Service:VitalQIP DNS Update Service:<ip_address_of_qip_update_service>
MessageRoute=DNSUpdateRR:A:0:DNS Update Service:VitalQIP DNS Update Service:<ip_address_of_qip_update_service>
```

3 Save the policy file.

4 Ensure the QIPMESSAGESERVICE environment variable is set to the IP address of a Message Service configured with a route to the DNS Update Service.

END OF STEPS

Assign domain to the Lucent DNS client *(Windows Only)*

If the VitalQIP client or DNS Update Service is running on Windows 2000, ensure the server running the client is a member of the same Windows 2000 domain as the Windows 2000 KDC, and can resolve the SRV records for this domain. See your Windows 2000 administrator for assistance.

Important! Ensure the user is logged into the Windows 2000 domain and not the local machine.

Set up the *krb5.conf* file *(UNIX Only)*

To set up the *krb5.conf* file, follow these steps:

1 Open the */etc/krb5.conf* file with a text editor.

2 Set up a realm in the local *krb5.conf* file that points to the Microsoft realm and Microsoft KDC. Add or modify the following lines:

```
[realms]
<REALM> =
{
   kdc = <IP_address_of_Windows_2000_KDC>:88
   admin_server = <IP_address_of_Windows_2000_KDC>:749
   default_domain = <DNS_zone_of_local_machine_or_other_zone_if_desired>
}

[libdefaults]
default_realm = <REALM>
default_tkt_enctypes = des-cbc-md5
default_tgs_enctypes = des-cbc-md5
[domain_realm]
.<DNS_zone> = <REALM>
```

Important! Encryptions types must be "des-cbc-md5" when using a Windows 2000 KDC. The **[domain_realm]** heading lists DNS zones and the REALM to associated with that zone. Typing a period (.) before the DNS zone name means to use the REALM for all subdomains of that domain.

3 Save the file.

4 Ensure the following:

- The specified realm must resolve on the host's DNS server. Typically, you must edit */etc/resolv.conf* to point to the DNS server that can resolve the domain correctly.
- A reverse lookup on the host of the server's IP address must either not resolve or resolve to a host or host/domain in the target domain. Typically, */etc/hosts* file is edited to look for the server's IP address. Fully-qualified name must be listed first in the */etc/hosts* file before the shortname.

- The IP address of the KDC must also not resolve in */etc/hosts*, or resolve to the proper domain. This step and the one before it must be done before the DNS server is started if it is on that UNIX server. See “Secure dynamic update notes” on page 253 for more information about resolving names.

END OF STEPS

Define a new user *(UNIX Only)*

For a VitalQIP client that updates DNS and the DNS Update Service on UNIX, define a user enabled to perform secure dynamic updates as follow:

1 On the Windows 2000 KDC, add a new user through the Active Directory Users and Computers mmc snap-in. The user can use an arbitrary name, but cannot use DNS.

2 Create the *keytab* file for that user as follows:

```
ktpass -princ <DNS_Updater>/<host>.<domain>@<REALM> -mapuser
<DNS_Updater> -pass <password> -crypto DES-CBC-MD5
-out  <DNS_Updater>.keytab
```

Table 41 describes the generic values used in the above example and used for the remainder of this section.

Table 41 Description of generic values

Value	Description
<DNS_Updater>	User who updates DNS.
<host>	The host running the VitalQIP client. If this value is a generic principal name to be used on multiple hosts, the *<host>* is not needed. The principal name would look like *<DNS_Updater>/<domain>@<REALM>*.
<domain>	The domain to which the *<DNS_Updater>* is assigned.
<REALM>	The Windows 2000 domain of the KDC in capital letters.
<password>	The password that was defined for that user in AD Users and computers mmc snap in.

3 On the machine running the VitalQIP client, add the user (in the following example “qipclient”) from the *.keytab* file to the *krb5.conf keytab* file by running:

```
ktutil: rkt qipclient.keytab
ktutil: wkt /etc/krb5.keytab
ktutil: exit
```

4 Ensure the user is added to the *krb5.keytab* file by running:

```
ktutil: rkt /etc/krb5.keytab
ktutil: list
```

END OF STEPS

Set up the Lucent DNS server for secure dynamic updates

When to use

This section describes the steps you need to follow to enable secure dynamic updates on a Lucent DNS server.

Enable the Lucent DNS server for secure dynamic updates

To enable the Lucent DNS server for secure dynamic updates, follow these steps:

1 In the VitalQIP client, access **Infrastructure|Server**. The Server Profile Option window opens.

2 In the Server Profile Option window, select the Lucent DNS server from the Existing Server list. Refer to “Servers” in Chapter 3 of the *VitalQIP User’s Guide* for information on adding servers.

3 Select **Modify Server**.

4 Click **OK**. The Server Profile window opens.

5 In the Server Profile window, select **Secure DNS Updates** parameter in the Parameters/Values list. Values for the options appear.

6 Select **True**.

7 Click **OK**.

END OF STEPS

Assign the Lucent DNS server to a domain

To assign the Lucent DNS Server to a domain, follow these steps:

1 In the VitalQIP client, access **Infrastructure|Domain**. The Domain Option window opens.

2 In the Domain Option window, select the domain from the Existing Domain list.

3 Select **Modify Domain**.

4 Click **OK**. The Domain Profile window opens.

5 In the Domain Profile window, click the **Primary/Secondary Servers** tab.

6 Select the Lucent DNS server.

7 Click **Add Primary** or **Add Secondary**. The server is added to the Selected DNS Server list with a secure status of “False”.

8 In the Selected DNS Server list, select the Lucent DNS server.

9 Check **Send Secure Updates**. The Secure status of the Lucent DNS server is changed from “False” to “True”.

10 Click **OK**.

END OF STEPS

Assign the Lucent DNS server to a reverse zone

To assign the Lucent DNS Server to a reverse zone, follow these steps:

1 In the VitalQIP client, access **Infrastructure|Network/Reverse Zone**. The Network/Reverse Zone Option window opens.

2 In the Network/Reverse Zone Option window, select the network from the Existing Network list.

3 Select **Modify Domain**.

4 Click **OK**. The Network window opens.

5 In the Network window, click the **Reverse Zone** tab and select the reverse zone to send secure dynamic updates.

6 Click **Properties**. The Reverse Zone Properties window opens.

7 In the Reverse Zone Properties window, click the **Primary/Secondary Servers** tab and select the Lucent DNS server.

8 Click **Add Primary** or **Add Secondary**. The server is added to the Selected DNS Server list with a secure status of “False”.

9 In the Selected DNS Server list, select the Lucent DNS server.

10 Check **Send Secure Updates**. The Secure status of the Lucent DNS server is changed from “False” to “True”.

11 Click **OK**.

END OF STEPS

Generate DNS files

To generate the DNS files, follow these steps:

1 In the VitalQIP client, access Network **Services|DNS Generation**. The DNS Generation window opens.

2 In the DNS Generation window, select the appropriate options. Refer to “Generate DNS configuration and data files” in Chapter 5 of the *VitalQIP User’s Guide* for information on these options.

3 Click **OK**. The following lines are written to the *named.conf* file:

In the `options` section:

```
qddns
{
   gss-principal "DNS@<host>.<domain>";
};
```

In the `zone` section:

```
qddns
{
  secure-updates yes;
};
```

END OF STEPS

Steps for Lucent DNS server on Windows 2000

If the DNS server is running on Windows 2000, the following items must be completed:

1 The Lucent DNS server must be part of the same Windows 2000 domain that the Windows 2000 KDC is managing. It must also be able to resolve the SRV records for this domain.

2 The Lucent DNS server must be run by the Service Controller as local system; it will not work if it is run from the console.

3 The DNS server must be able to resolve its own name without using itself. In other words, when the Lucent DNS server starts, the network must be configured in such a way that the DNS server can resolve its principal name without using itself.

END OF STEPS

Steps for Lucent DNS server on UNIX

If Lucent DNS server is running on UNIX, follow these steps:

1 Add an A record for the host in the appropriate domain or realm on the Lucent DNS server.

2 On the Windows 2000 KDC, use the Active Directory Users and Computers MMC snap in to add a new user with the same name as the host on which the Lucent DNS server is running.

3 After adding the user, create the *keytab* file by running the following command from a command interface:

```
ktpass -princ host/<hostname>.<host's_domain>@<Windows_2000_KDC_domain>
-mapuser <hostname> -pass <AD_User's_password> -crypto DES-CBC-MD5
-out <hostname>.keytab
```

4 Add an user called "DNS" to Active Directory Users and computers in the same way and create its *keytab* file as follows:

```
ktpass -princ DNS/<hostname>.<host's_domain>@<Windows_2000_KDC_domain>
-mapuser DNS -pass <AD_User's_password> -crypto DES-CBC-MD5 -out DNS.keytab
```

5 Move the two *.keytab* files to the UNIX host.

6 Merge the *.keytab* files into the */etc/krb5.keytab* file by running:

```
ktutil: rkt <hostname>.keytab
ktutil: rkt DNS.keytab
ktutil: wkt /etc/krb5.keytab
```

```
ktutil: exit
```

7 Ensure the user is added to the *krb5.keytab* file by running:

```
ktutil: rkt /etc/krb5.keytab
ktutil: list
```

END OF STEPS

☐

External objects and resource records support

VitalQIP is capable of managing all DNS name space, even if that space is modified by another product. VitalQIP can represent resource records that are added to the DNS name space by other products as external objects or external resource records.

□

About external objects and resource records

An external object or resource record is created by dynamic DNS updates (defined by RFC 2136) from some product to a Lucent DNS server managed by VitalQIP. For example, Microsoft operating systems, such as Windows 2000, perform dynamic DNS updates whenever a host boots. The Lucent DNS server is then updated with the host's name, IP address, and hosted services. Other Microsoft hosts query the DNS server to locate necessary services.

When a Lucent DNS server managed by VitalQIP processes this dynamic update, it sends the update to VitalQIP. VitalQIP represents this record as an external resource record or an external object. These external resource records or external objects are deleted by VitalQIP when the record is deleted from DNS. VitalQIP can optionally send the external update to all DNS servers that manage the affected zone.

Partially managed object

VitalQIP supports another type of external object called partially managed object. A partially managed object is created and deleted by a VitalQIP administrator. However, VitalQIP and Windows 2000 administrators can modify a partially managed object. Windows 2000 administrator can modify partially managed objects by changing the Windows machine hostname. VitalQIP is then updated through dynamic DNS updates received by the Lucent DNS server configured to propagate DNS updates to VitalQIP.

When partially managed objects are created, modified, or deleted, the audit records are sent to a VitalQIP add-on product called Audit Manager if the product is used. The partially managed objects appear in reports.

Important! See your sale representative for more information on purchasing Audit Manager. See the ***Audit Manager User's Guide*** for how Audit Manager works.

Partially managed objects are created by accessing or creating an Object Profile, clicking **Object** tab, and setting the **Object Class** field to Partially_Managed. See "Add a static object" in Chapter 4 of the *VitalQIP User's Guide* for more information on this field.

□

How VitalQIP imports external object and resource records

VitalQIP supports two models for collecting external objects or resource objects - continuous and polling models. External objects and resource records imported through the continuous model are indistinguishable from external objects or resource records created by the polling model. The two models can update the same zone with no data loss.

Continuous model

In the continuous model, VitalQIP is continuously being updated with external objects and resource records. The continuous model is not restricted to capturing objects and resource records with a particular naming convention. The continuous model only supports RFC 2136 dynamic DNS updates that specify a single resource record. External objects and resource records are not “tombstoned” if the RFC 2136 update specifies an entire resource record set. See “About tombstoned objects and resource records”, on page 269 for information about tombstoned resource records.

Important! Currently, only the Lucent DNS Service (version 3.1 or above) supports the continuous model.

Figure 29 shows how the continuous model works.

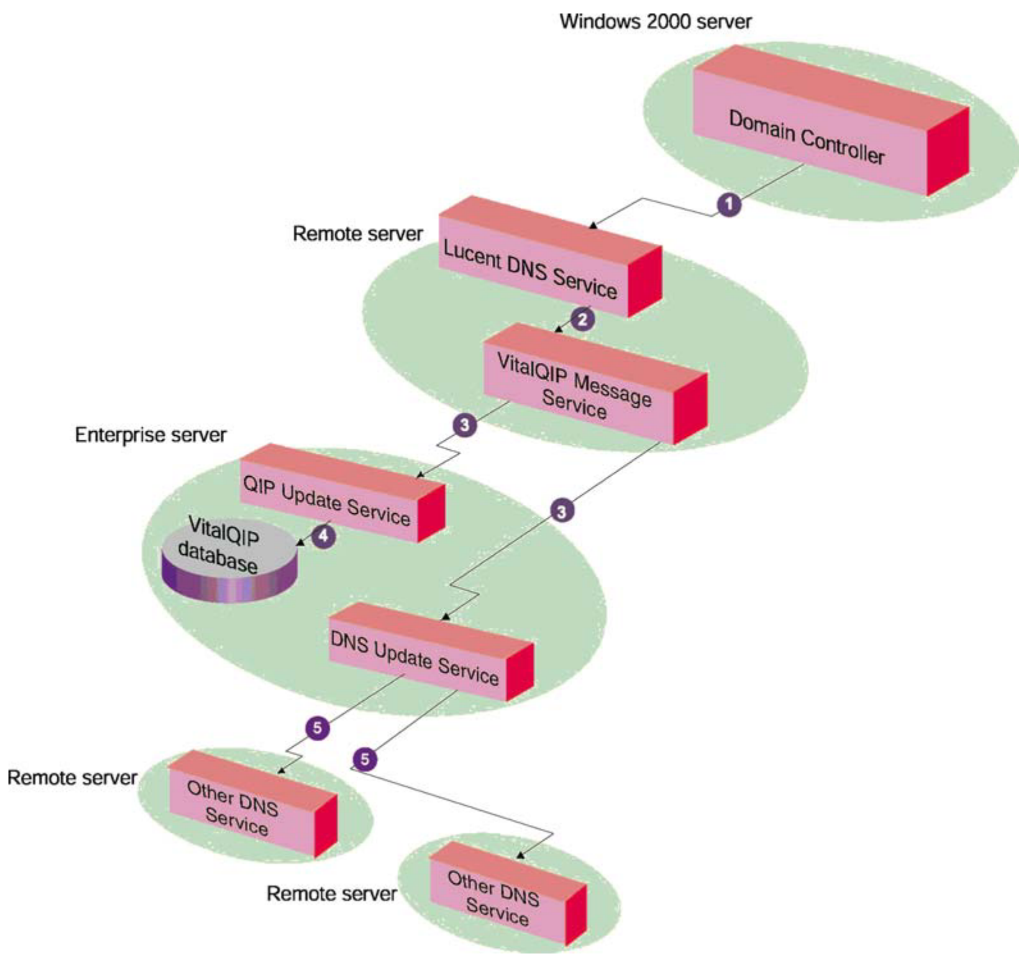


Figure 29 Continuous Model

The following steps occur in Continuous Model:

1. A DNS client, such as a domain controller on a Windows 2000 server, sends a RFC 2136 dynamic DNS update to a Lucent DNS Service managed by a VitalQIP remote server.
2. The Lucent DNS Service converts the RFC 2136 update to a DNSUpdateRR message and sends the message to the local Message Service.

   Important! Only DNSUpdateRR messages sent from Lucent DNS servers that are primary for a zone are supported.

3. The Message Service forwards these updates to the DNS Update Service and the QIP Update Service on the VitalQIP enterprise server.
4. The VitalQIP Update Service updates the VitalQIP database with the external update. Resource records and objects are created in VitalQIP.

5. The DNS Update Service forwards the update to other DNS Services on remote servers based on the contents of the *x.ddns.conf* files. Embedded within the message is the IP address of the originating DNS Service. The DNS Update Service uses the IP address to ensure it does not forward the update back to the originating DNS Service.

Polling Model

The polling model depends on the periodic execution of the **qip-syncexternal** CLI command. The polling model is restricted to capturing A, PTR, TXT, AAAA, CNAME and SRV resource records that are added to DNS by a Windows operating system. This model supports all managed DNS servers that implement the AXFR DNS protocol.

Important! The **qip-syncexternal** CLI is not executed by the Schedule Service, but it can be run by an external scheduler, such as cron on UNIX or **Start|Settings|Control Panel|Scheduled Task** on Windows.

Figure 30 shows how the polling model works.

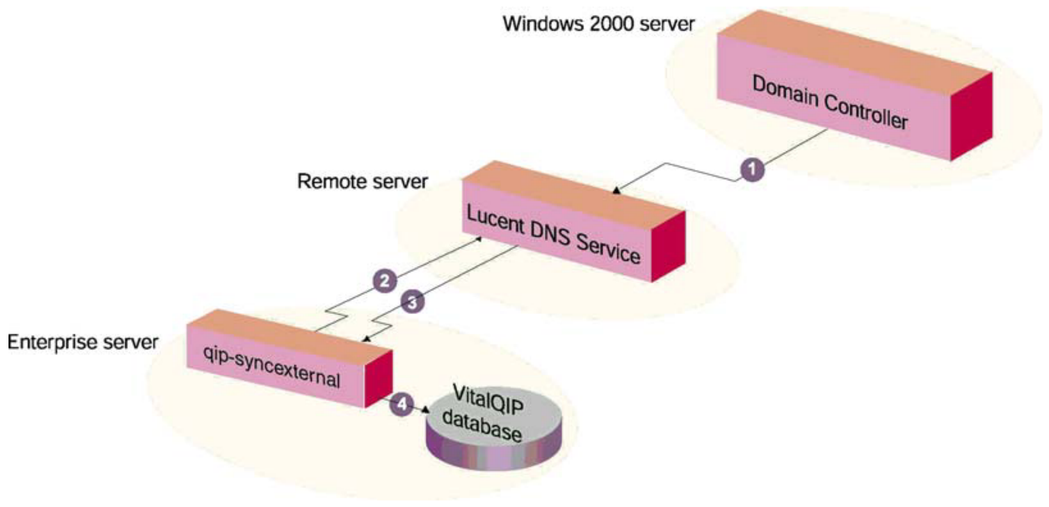


Figure 30 Polling Model

The following steps occur in Figure 30:

1. A DNS client, such as a domain controller on a Windows 2000 server, sends RFC 2136 dynamic DNS updates to a Lucent DNS Service managed by a VitalQIP remote server.
2. The **qip-syncexternal** CLI initiates a zone transfer (AXFR) from the Lucent DNS Service that received the 2136 DNS update.
3. The DNS Service responds to the zone transfer request with the zone content. Contents include resource records that have been dynamically updated by the Windows 2000 controller and resource records VitalQIP knows about.

4. The **qip-syncexternal** CLI compares records from the zone transfer with the contents of the VitalQIP database. Based on the comparison, the **qip-syncexternal** CLI may create or “tombstone” external objects or resource records.

□

How VitalQIP handles external objects and resource records

VitalQIP encodes resource records from external sources as external objects or resource records. The "External" Object Class in the Object Management window identifies external objects. External resource records are identified by the **Managed by External Updates** checkbox in the Resource Record tab of the Domain Profile and Reverse Zone Profile windows.

Important! External objects cannot be created through the VitalQIP GUI.

VitalQIP handles external objects or resource records in the following ways:

- When processing an A or PTR resource record, VitalQIP adds the resource record as an external object. If VitalQIP cannot represent the A or PTR record as an object, the external resource record is attached to a domain or reverse zone.
- When processing a TXT, AAAA, CNAME, or SRV resource record, VitalQIP adds an external resource record to the appropriate domain or reverse zone.
- When processing a dynamic add for a "tombstoned" external object or resource record, VitalQIP "un-tombstones" the object or resource record.

About tombstoned objects and resource records

"Tombstoned" objects are created when VitalQIP notices that an external object or resource record is deleted. VitalQIP uses tombstoned objects and resource records to resolve DNSUpdateRR message conflicts. Conflicts can occur when two DNSUpdateRR messages take different message routes to reach the QIP Update Service.

For example, consider a Windows 2000 server that reboots frequently. As Windows 2000 starts up and shuts down, it adds and removes itself from DNS many times. If the DNSUpdateRR messages do not arrive at VitalQIP in the correct order, VitalQIP does not represent the Windows 2000 server when it should or will represent the Windows 2000 server when it should not.

Tombstoned object or records contain a times stamp, which resolves conflicts such as the one described above. The Lucent DNS Service puts the current time in the DNSUpdateRR messages when it generates the message. The QIP Update Service only updates the VitalQIP database if the time stamp in the message is more recent than the tombstoned object's or resource record's time stamp.

When a delete message is received for an A or PTR record that is a part of an external object, the appropriate resource record type becomes unchecked in the Name Service portion of the Object Profile. When neither A or PTR Name Services are checked, the object is "tombstoned". Tombstoned objects are indicated in the VitalQIP client by a "Tombstoned" status in the Object Profile window.

When a delete message is received for an A or a PTR record that are not part of an external object or when a delete update comes in for a TXT, AAAA, CNAME or SRV resource record, VitalQIP tombstones the resource records attached to the domain or reverse zone. Tombstoned resource records are not visible in the VitalQIP client but are kept in the VitalQIP database.

Tombstoned objects and resource records are deleted from VitalQIP periodically by the **qip-tombstonepurge** CLI if the **Tombstone Purge** policy is set in **Policies|Global Policies|Dynamic DNS**. This CLI is run by the VitalQIP Schedule Service. The CLI deletes any tombstoned resource records or objects that are older than the value specified in **Tombstone Max File** policy set in the **Policies|Global Policies|Dynamic DNS**. External objects and resource records can be deleted in the VitalQIP client at any time.

See the *VitalQIP Command Line Interface User's Guide* for more information on the **qip-tombstonepurge** CLI. For more information on setting up the **Tombstone Purge** and **Tombstone Max Life** policies, see the General policies table in Chapter 2 of the *VitalQIP User's Guide.*

□

Modify external objects and resource records

External objects that belong to VitalQIP cannot be modified by the VitalQIP client except to change the Object Class from “External” to something else, such as “Workstation”.

External resources records belonging to VitalQIP cannot be modified except to make them not external resource records. To make a resource record not external, uncheck the Managed by External Updates checkbox in the Resource Record tab of the Domain Profile and Reverse Zone Profile windows. For more information on these profiles, see “Domains” in Chapter 3 of the *VitalQIP User’s Guide.*

Once an object or resource record is no longer classified as an external object or resource record, it can be modified in the same fashion as any other VitalQIP object or resource record. VitalQIP cannot tombstone or delete an object or resource record once it is no longer classified as an external object or resource record.

□

About external object updates to DNS

The network, rather thanVitalQIP, is considered authoritative for external objects and resource records. Although the VitalQIP QIP Update Service performs some basic checking to ensure the update does not clash with VitalQIP authoritative objects, extensive name management is not carried out, as it is for regular DHCP updates. For this reason, the VitalQIP QIP Update Service (**qip-qipupdated**) only adds external updates to the VitalQIP database, and does not forward them to the VitalQIP DNS Update Service (**qip-dnsupdated**).

You should send the updates to other DNS servers via **qip-dnsupdated**, so that the zone is not in an inconsistent state. Set up the following MessageRoute on the Remote Server that is acting as the primary DNS server configured to support External Dynamic Update Propagation (EDUP) :

```
MessageRoute=DNSUpdateRR:A:0:QIP Update Service (DHCP):VitalQIP DNS Update Service:<IP
  Address of Server running qip-dnsupdated>
```

One of the reasons for not performing name management (FIRST-IN/LAST-IN, and so on) on the external resource records is that, in addition to being a VitalQIP update mechanism, EDUP is a multi-master mechanism. If VitalQIP were not to propagate these updates as-is, it would break the multi-master functionality.

Clients that are not trusted to update DNS correctly should not have privileges to make the updates at all. DHCP clients should go through the standard DHCP server->QIP Update Service->DNS Update Service route because DHCP clients are generally not trustworthy enough to be allowed uncontrolled access to DNS records. EDUP should be reserved for trusted machines, such as servers or domain controllers, that need to have DNS information published without manual interference.

□

Configure VitalQIP to collect external objects and resource records

When to use

Three classes of VitalQIP infrastructure affect the creation of external objects, Lucent DNS Services, forward zones, and reverse zone.

Before you begin

Lucent Technologies recommends the following for the optimal processing of external objects:

- Deploy a network time service (NTP) on the VitalQIP enterprise server and all server machines running the Lucent DNS Service. A time service synchronizes the various clocks and allows VitalQIP to resolve time stamp conflicts.
- Create zones for the various subdomains (_msdcs, _sites, _tcp, _udp) used by Windows domain controllers, and only allow creation of external objects and resource records in these zones. These zones are smaller, faster to transfer, and easier to manage than a single zone, which contain these subdomains.
- Install the DNS Update Service on the VitalQIP enterprise server or some other central location using the Distributed Services installation. This makes it easier to deploy firewalls between DNS Services and the DNS Update Service. For more information on installing the QIP Update Service, see the ***VitalQIP Installation Guide***.
- In Windows 2000 environments, only domain controllers and VitalQIP sources should have update permission (through ACLs) for the Lucent DNS server. Lucent does not recommend allowing end users' Windows 2000 clients to update DNS (which would update VitalQIP). Updates from end user's Windows 2000 clients pose a security risk. End users can obtain the names associated with Static Objects in this situation.

Procedure

To configure VitalQIP to handle external objects and resource records, follow these steps:

Configure the Message Service

Configure the Message Service to process external objects and resource records. Follow these steps:

1 Log on to the remote server.

2 Go to the *QIPHOME* directory.

3 Open the *qip.pcy* file (or the *qip-msgd.pcy* if you keep separate policy files) with a text editor.

4 In the policy file, find the **[VitalQIP Message Service]** section.

5 Add the following line:

```
MessageRoute=DNSUpdateRR:A:0:QIP Update Service (Update RR):VitalQIP QIP Update
  Service:<QIP_Update_Service_Address>
```

6 Add the following line on the ***remote server*** to send DNS updates to the DNS Update Service (refer to "About external object updates to DNS", on page 272 for more information on EDUP updates to DNS).

```
MessageRoute=DNSUpdateRR:A:0:DNS Update Service (Update RR):VitalQIP DNS Update
  Service:<DNS_Update_Service's_address>
```

7 Save the policy file.

Configure the Lucent DNS Service

Currently, only Lucent DNS Service 3.1 and higher supports forwarding external objects and resource records to VitalQIP. To forward external updates to VitalQIP, you must use a Lucent DNS Service.

A Lucent DNS server forwards external objects and resource records to VitalQIP according to the configuration of its forward and reverse zones. For information about setting up a server running the Lucent DNS Service, see "Servers" in Chapter 3 of the *VitalQIP User's Guide*.

Configure forward zones

To configure a forward zone to forward external updates to VitalQIP, follow these steps:

1 In the VitalQIP administrative client, select **Infrastructure|Domain**. The Domain Option window opens.

2 Select the domain to be modified.

3 Select **Modify Domain**.

4 Click **OK**. The Domain Profile window opens.

5 Click the **Zone Options** tab.

6 Open up the appropriate server type (Lucent 3.X or Lucent 4.X) and open up **Import External Updates**. Once Import External Updates is selected, the Resource Record Import Types are enabled.

7 Select which resource records are to be forwarded to VitalQIP. The following resource records types may be selected:

- A (Host IPv4)
- AAAA (Host IPv6)
- CNAME (Canonical Name)
- PTR (Pointer)
- SRV (Server Resource Record)
- TXT (Text)

Click **OK**. The Lucent DNS server that hosts this zone does not look at the new values until the next configuration push.

Configure reverse zones

To configure a reverse zone to forward external updates, follow these steps:

1 In the VitalQIP administrative client, select **Infrastructure|Network/Reverse Zone**. The Network/Reverse Zone Profile Option window opens.

2 Select the network to be modified.

3 Select **Modify Network**.

4 Click **OK**. The Network Profile window opens.

5 Click the **Reverse Zones** tab.

6 Select the reverse zone to be modified.

7 Click **Properties**. The Reverse Zone Properties window opens.

8 Click the **Zone Options** tab.

9 Open up the appropriate server type (Lucent 3.X or Lucent 4.X) and open up **Import External Updates**. Once Import External Updates is selected, the Resource Record Import Types are enabled.

10 Select which resource records are to be forwarded to VitalQIP. The following resource records types may be selected:

- A (Host IPv4)
- AAAA (IPv6)
- CNAME (Canonical Name)
- PTR (Pointer)
- SRV (Server Resource Record)
- TXT (Text)

Click **OK**. The Lucent DNS server that hosts this reverse zone does not look at the new values until the next configuration push.

END OF STEPS

□

8 Advanced DHCP configurations

Overview

Purpose

This chapter covers configuring DHCP servers. DHCP configuration instructions are included for IBM, Windows, and Lucent DHCP servers.

Contents

This information presents the following topics.

Support for IBM AIX DHCP servers

VitalQIP provides dynamic IP addressing solutions in conjunction with IBM's DHCP servers running on AIX. Additionally, IBM provides a connection between their DHCP server running on the IBM Risc/6000 platform.

Configuration/setup for IBM AIX 5.2 DHCP

Although the Lucent DHCP servers are supported on AIX platforms, VitalQIP supports the native DHCP services. For IBM's AIX 5.2, Lucent supplies a version of **dhcpaction** and **dhcpremove** during the VitalQIP installation process.

Whenever VitalQIP generates DHCP configuration files, it creates the */etc/dhcpsd.cnf* file. If you need to add extensions, put them in the */etc/dhcpsd.cnf.corp* file. They are automatically appended to the */etc/dhcpsd.cnf* file. The IBM AIX DHCP server is located in */usr/sbin/dhcpsd*. When setting up an AIX DHCP server within VitalQIP, set the default directory to */etc*.

Table 42 describes the files that are automatically copied or placed during the VitalQIP installation process.

Table 42 Files Copied During the VitalQIP Installation

Copied From	Copied To	Description
$QIPHOME/usr/bin/dhcpaction.aix	*/usr/sbin/dhcpaction*	The AIX script runs every time a client releases its lease, and then dynamically updates DNS and VitalQIP databases.
$QIPHOME/usr/bin/dhcpremove.aix	*/usr/sbin/dhcpremove*	The AIX script runs every time a client updates its lease, and then dynamically updates DNS and VitalQIP databases.
$QIPHOME/usr/bin/dhcpsd.cnf.corp	*/etc/dhcpsd.cnf.corp*	Portion of the **dhcpsd** configuration file that includes basic settings. The default location for *dhcpsd.cnf* is in the */etc* directory. If you are using a different directory, copy this file into that directory.

Copied From	Copied To	Description
	$QIPHOME/usr/bin/upQIP	The VitalQIP program that sends an update request to VitalQIP to modify the hostname and MAC address.
	$QIPHOME/usr/bin/ rmQIP	The VitalQIP program that sends an update request to VitalQIP to remove the hostname and MAC address.
$QIPHOME/userexits/qipdnsuserexit.aix	*$QIPHOME/userexits /qipdnsuserexit*	The VitalQIP script is executed after the Lucent DNS push. This script modifies the *named.conf* file for BIND 8.x to support dynamic updates for DNS.
	$QIPHOME/usr/bin/check-named	The VitalQIP utility is used to modify the *named.conf* files with additional settings.

□

VitalQIP Interface with IBM DHCP Server

VitalQIP packages its interface to IBM's DHCP server with its VitalQIP remote server package. The package provides extensions to **dhcpsd**, **dhcpaction**, and **dhcpremove**. VitalQIP adds modules that update the VitalQIP server. Two daemons, **qip-rmtd** and **qip-netd** provide communication between the DHCP server and the VitalQIP server. Two routines, **upQIP** and **rmQIP** update the VitalQIP database for lease renewals and releases respectively.

It is important to note that the IBM DHCP Service does not allow VitalQIP to be notified when a lease expires. This is a known IBM DHCP service/VitalQIP interface limitation. Lease information is added to the VitalQIP database, but it cannot be automatically removed as part of the lease expiration process.

Additionally, IBM provides a command that is executed on the AIX system running their DHCP server to display current leases. The format is:

```
/usr/bin/lssrc -s dhcpsd -l
```

You must be the root user in order to run this command.

Starting and Stopping the DHCP Service

To start the DHCP daemon, run the following script:

```
startsrc -s dhcpsd
```

To stop the DHCP daemon, run the following script:

```
stopsrc -s dhcpsd
```

To tell the **dhcpsd** service to re-read its configuration, use

```
refresh -s dhcpsd
```

Important! See the operating system’s Man Pages for additional information regarding **dhcpsd**, **dhcpaction**, and **dhcpremove**.

Organizations and IBM AIX DHCP Servers

Currently, the IBM AIX DHCP server does not support the **Organization** feature implemented by the VitalQIP software. In order to update DHCP lease information within the VitalQIP database an “organization ID” must be passed to the two utilities (**upQIP** and **rmQIP**) used to forward information to the VitalQIP database.

upQIP and **rmQIP** are currently invoked by the shell scripts **dhcpaction** and **dhcpremove**, respectively. By default, these scripts are configured to update the default VitalQIP organization. If other organizations are created, **dhcpaction** and **dhcpremove** must be modified to reflect the appropriate organization of the DHCP server on that machine.

Procedure

To do this, follow these steps:

1 From a command line interface, find theVitalQIP Organization ID by running the following script:

– For Sybase:

```
isql -U qipadmin -P <password> -S <dataserver>
> select * from organizations
> go
```

– For Oracle:

```
sqlplus qipadmin/<password>@dataserver
> select * from organizations;
```

2 Write down this number; it is the organization ID.

3 Use a text editor, such as vi, to open the */usr/bin/dhcpaction* file.

4 In the */usr/bin/dhcpaction* file, find the following line:

```
%QIPHOME%/usr/bin/upQIP $hostname $domainname $ipaddr $leasetime 1-0x$clientid
```

(not the loopback address)

5 Add two additional parameters, one to identify the IP address of the local machine and the second to identify the VitalQIP Organization ID. Add the text in bold:

```
%QIPHOME%/usr/bin/upQIP $hostname $domainname $ipaddr $leasetime $clientid 127.10.1.0.
  <VitalQIP_organization_ID_from_Step_2>
```

6 Make the same modification to the *%QIPHOME%/usr/bin/dhcpremove* file.

7 When the modifications are complete, copy the *dhcpaction* and *dhcpremove* files to the */usr/sbin* directory.

Support for Windows 2000 DHCP Servers

Windows 2000 contains many enhancements for the management of network entities, including DHCP servers. VitalQIP takes advantage of the features that Microsoft has in their services. These features include:

- The ability to manage a Windows 2000 DHCP server
- The ability to manage multicast addresses in the VitalQIP GUI and push those out to the Windows 2000 DHCP server

  Important! VitalQIP also supports Windows 2003 DHCP servers in the same way as it does Windows 2000 DHCP servers.

Managing Windows 2000 DHCP Servers

VitalQIP supports management of Microsoft's Windows 2000 DHCP server. This includes creation of configuration information, viewing of active leases, and having lease information forwarded to VitalQIP.

Support for the Microsoft DHCP server running Windows 2000 is available with VitalQIP and supports the following:

- Ability to import existing Microsoft DHCP server reserved addresses, scope definitions, and DHCP option definitions into VitalQIP
- Ability to import components of the existing active lease information into VitalQIP
- Ability to display active lease information in VitalQIP

The management of Windows 2000 DHCP is accomplished through a DHCP server type in VitalQIP called **Windows 2000 DHCP**.

The Windows 2000 DHCP servers query Active Directory to determine if they are authorized to service their scopes. VitalQIP adds an authorization record into Active Directory when it performs a push to the server with additional information captured for the new server type.

A policy called **Active Directory Server** is also associated with the server type. This is the IP address of the domain controller where the Active Directory information for the DHCP server resides. You can also select a domain controller defined in VitalQIP if you set the policy to "Domain Controller List". Additional information you wish to establish for the Windows 2000 DHCP server can be specified in the **Additional Policies** policy. Anything placed in the **Additional Policies** field must be in **netsh dhcp** format. For more information on this format, access your Microsoft 2000 help files.

Additionally, three new Microsoft options can be defined in the DHCP template options window. In the **DHCP/Bootp Template** options, under the **Application and Services Template Class**, you see the following three **Available Parameters** that you can define for Windows 2000:

- Microsoft Disable NetBios
- Microsoft Release DHCP Lease on Shutdown
- Microsoft Default Router Metric Base

For more information about this DHCP server type and values, refer to "Servers" in Chapter 3 of the *VitalQIP User's Guide.*

VitalQIP Interface with Windows 2000 Limitations

If you are managing Microsoft DHCP 2000 servers, you must be aware that there are limitations in the VitalQIP interface to the Microsoft 2000 DHCP server and VitalQIP infrastructure. The limitations are the result of configuration differences in supported feature functionality of the Microsoft DHCP server itself. To accommodate these limitations:

- You must have Remote Service running on the Microsoft DHCP 2000 server to configure the Microsoft 2000 DHCP server, even if the VitalQIP client is running on the Microsoft 2000 DHCP server.
- You are not able to delete Active Leases from the active lease display.
- Only Dynamic-DHCP and Manual-DHCP object types are supported in VitalQIP for the Microsoft 2000 DHCP server. However, Microsoft automatically shares these pools between DHCP and Bootp devices as needed, based on the Microsoft DHCP server support. In other words, when you are defining objects within VitalQIP, you should define them as D-DHCP or M-DHCP. Microsoft 2000 DHCP treats them as either Bootp or DHCP as needed.
- Microsoft 2000 DHCP server does not support more than one DHCP template defined per subnet.
- Automatic-DHCP, Manual-Bootp, and Automatic-BOOTP objects are excluded addresses in the Microsoft DHCP 2000 configuration.
- When specifying additional policies in the Microsoft Windows 2000 DHCP Server Profile, each policy must begin with “netsh dhcp server”.

Importing Existing Reserved Addresses, Scope Definitions, and DHCP Options into VitalQIP

Information that exists within a Microsoft 2000 DHCP server can be imported into the VitalQIP system to facilitate the management of Microsoft 2000 DHCP servers, or to assist in the conversion of DHCP servers (for example, Microsoft 2000 DHCP to Lucent DHCP). However, User Class DHCP Options, Multi-Class scopes, and Vendor Class DHCP Options cannot be imported.

Use the following steps to import existing Microsoft 2000 DHCP information into VitalQIP:

1 Install the VitalQIP remote server, including support for the Microsoft 2000 DHCP server, on the system that is currently running the Microsoft 2000 DHCP server.

2 Install the VitalQIP Command Line Interface on the system that is currently running the Microsoft 2000 DHCP server.

3 Create a temporary directory to store temporary import information (for example, *\temp).*

Important! If a *temp* directory is not used when running **qip-msextract**, the files are, by default, placed in the directory specified by QIPHOME (for example, qip-msextract -d c:\temp).

4 Make sure that the Microsoft 2000 DHCP server is running.

5 Run the **qip-msextract** utility program from the command line. The **qip-msextract** utility is located in the *%QIPHOME%\cli* directory. (For detailed information on **qip-msextract**, see the *VitalQIP Command Line Interface User’s Guide*.) Run **qip-msextract**, as follows:

```
qip-msextract -d <temporary_drive:path>
```

6 The following files are created:

- *ms_templates.txt* - contains all DHCP option information used to create Lucent DHCP templates
- *ms_scopes.txt* - contains all DHCP scope information
- *ms_setdynobjects.txt* - contains all Microsoft 2000 Dynamic DHCP dynamic clients (D-DHCP)
- *ms_setmanobjects.txt* - contains all Microsoft 2000 DHCP reserved clients (M-DHCP)
- *mstoqip.bat* - a batch file that can be used to load the data from the files listed above
- *ms_setsubnets.bat* - a batch file that is used to link shared subnets together and provides subnet names

7 Define the domain of the DHCP server within VitalQIP.

8 Define your Microsoft 2000 DHCP server within VitalQIP. Note the server name is case sensitive and must match exactly.

9 Define all networks that Microsoft 2000 DHCP scopes reference within VitalQIP.

10 Define all subnets that Microsoft 2000 DHCP scopes reference within VitalQIP.

11 Run the *mstoqip.bat* file, or run the individual import routines manually by following the instructions below. This file runs on the enterprise or remote server and defines dynamic objects in the VitalQIP database. The object class defaults to PC and overwrites existing addresses, and new DHCP templates are created.

To run the batch file:

```
<temporary_directory>\mstoqip
```

To run them manually:

```
%QIPHOME%\cli\qip-template -u qipman -p <password> -f ms_templates.txt
%QIPHOME%\cli\qip-scope -u qipman -p <password> -f ms_scopes.txt
%QIPHOME%\cli\qip-setobject -u qipman -p <password> -d ms_setdynobjects.txt
%QIPHOME%\cli\qip-setobject -u qipman -p <password> -d ms_setmanobjects.txt
  <temporary_directory>\mstoqip
```

Important! Check the reject files (*<filename>.rej*) for any records that are not accepted.

Default Entries for the qip.pcy Policy File

You can also enable logging for the MS DHCP Monitor Service. Refer to “Sample qip.pcy file”, on page 155 for more information.

Updating VitalQIP with Information from the Microsoft 2000 DHCP Server

VitalQIP provides new services that allow information written to the Microsoft 2000 DHCP log file to be captured and sent to the VitalQIP enterprise server. This information includes hostname, IP address, and MAC address. The VitalQIP enterprise server is updated with this information based on "Sleep time" as the Microsoft 2000 DHCP server gives out new leases, renews leases, and deletes leases.

> **Important!** The VitalQIP **MSDHCPMonitorService** starts the MS DHCP server automatically if it is not already running.

The Microsoft 2000 DHCP server must be configured with logging turned "on". Optionally, logging can be turned "on" via the registry; however, the VitalQIP Microsoft 2000 DHCP Monitor Service (**MSDHCPMonitorService**) turns on this option automatically when started.

If the MonitorService detects that logging is disabled, it enables it and automatically stops and restarts the DHCP server in order to re-read the configuration information. Logging must be turned "on", and working correctly in order for the VitalQIP Microsoft 2000 DHCP Monitor service to work properly. You can test to make sure that logging is working correctly by looking at the *%system32%\dhcp\DhcpSrvLog.<day>* file. As leases are granted or deleted, a message is written to the log file.

Note that there is increased latency between the time that a lease is granted, or deleted, and when the update to VitalQIP actually occurs. This is due to the time interval that it takes the Monitor Service to process the log file. It processes the log file on a timed interval rather than in real time as changes are made.

When a **Network Services|DHCP Generation** (a push to the server) is done to an MS DHCP server, the DHCP Directory field is not displayed because the information is written to an inaccessible file (*dhcp.mdb*). This is a limitation of the implementation.

> **Important!** You can use the **qip-dhcpsync** CLI to import/synchronize your MS DHCP Active Lease information with the VitalQIP database. **qip-dhcpsync** does not work when updating VitalQIP with client-released leases. This is because MS DHCP does not store released leases in its active lease database. It does store active and expired leases. See the *VitalQIP Command Line Interface User's Guide* for more information about **qip-dhcpsync.**

Displaying Active Lease Information in VitalQIP

To display Active Lease information from the Microsoft 2000 DHCP server, you must have Active Lease Service running on the Microsoft 2000 DHCP server.

You cannot delete Active Leases on the Microsoft 2000 DHCP server.

CLI Commands and Utilities

The **enterdhcpsvr** and **exportdhcpsvr** CLI commands support Window 2000 DHCP.

Multicast Addressing

The previous versions of Microsoft's DHCP server did not support multicast scopes, user classes, and vendor classes.

The "multicast" template allows users to define the **Scope Life** and **Multicast TTL** for multicast scope in the Application Service Template Class. For more information on these options, see "DHCP template classes and options" in Chapter 2 of the *VitalQIP User's Guide.*

The VitalQIP client allows networks and subnets to be managed in the multicast address range. You can define networks and subnets in the multicast address space of 224.0.0.0 through 239.255.255.255 through the Network Profile. The default mask for these address ranges is 255.0.0.0. These addresses behave identically to Class A networks. It is recommended that CIDR be used to define a more realistic network mask or that subnets be created using the starting and ending address of the subnet to avoid performance issues that come with large numbers of networks.

Support for DHCP Multi-NIC

VitalQIP supports multiple NIC cards on the Lucent DHCP server as the default action of the DHCP server. This gives the added ability of managing multiple networks. The support for multi-NIC is available on Windows 2000 and UNIX (Solaris) platforms.

On Solaris and HP-UX platforms, multi-NIC support is provided at the Data Link Protocol Interface (DLPI) layer. Typically, this requires additional processing by the server to create and parse packets.

A Lucent DHCP Server Profile has two policies, **DHCPServer** and **DHCPSocketAddr**, that can be used to influence the multi-NIC support. See "Additional policies for the Lucent DHCP server" on page 296 for more information about these policies.

A few things to keep in mind

– When configuring the multicast 239 network, each subnet must have at least 256 addresses in its address range. If any 239 subnets have less than 256 addresses, the configuration of the DHCP server fails.

– Active multicast leases are not viewable in VitalQIP.

– The **Policies|General Policies|Allow Dotted Hostnames**, must be set to “True” in order to modify the Object Profile of multicast active leases. This is due to multicast lease object hostnames being stored within VitalQIP as a combination of IP address of the original requesting device to the original VitalQIP default hostname. Refer to “Global policies” in Chapter 2 of the *VitalQIP User’s Guide* for details about the **Allow Dotted Hostnames** policy.

Support for the DHCP Failover Server

VitalQIP provides support for many-to-one DHCP failover servers, which provides a high level of redundancy for dynamic IP environments. Using a DHCP failover server allows an organization to design a DHCP server network through parameters that can provide uninterrupted service for DHCP clients. DHCP failover policies can be defined in the *dhcpd.pcy* file on the primary server and the failover servers and thereby allow you to create a failover environment specific to your needs. (See "Configuring the Primary DHCP Server" on page 292 to learn how to define the options through the VitalQIP GUI.)

Although VitalQIP does not limit the number of primary DHCP servers that are assigned to handle a single failover DHCP server, there are practical limits for the many-to-one configuration. From the outset, it has always been recommended that the ratio not exceed the 1-to-5 ratio.

The 1-to-5 ratio is recommended for several reasons. In the event of catastrophic failure, such as all primary servers are down or the failover server cannot communicate with the primary servers, there may not be enough processing power for the secondary server to reliably service all primary servers. For each primary server that is backed up by a failover server, a separate thread is created to manage communication between servers. The failover server manages approximately 17 threads under normal standalone operation. Additional threads for each failover channel causes additional context switching by the operational system. Performance can be severely hampered. In some cases, massive context switching can degrade the server to inoperable levels.

You can back up primary servers with DHCP many-to-one failover server support, and you can specify failover policies for each primary. This allows the failover server to “poll” the primaries at different intervals and maintain different options for retries, synchronization, and so on.

A few things to keep in mind

The following are DHCP failover server limitations:

- When a DHCP client performs an explicit release of an address, the DHCP protocol requires that the DHCPRelease message be unicast to the server that issued the lease originally. If the client obtains a lease from a primary server, which then goes down, this causes the secondary to take over. If the client then issues a release, the packet is unicast to the primary (which is not up), and is not seen at all by the secondary. Thus, it is not reflected in the lease database of the secondary. This “released” address is not reflected as released in either server until the original lease time expires.
- A secondary server cannot serve as a failover for two primary servers that service the same subnet. For example, assume primary A server is managing address 1-100 on the 10.200.50.0/255.255.255.0 subnet, and primary B server is managing addresses 101-200 on the same subnet (10.200.50.0/255.255.255.0). A secondary server cannot serve as failover for both primary A and B servers. Typically, two primary servers are configured in this way to back up each other. This configuration is known as **split scopes**. A secondary server usually is not required for this subnet.

Configuring the Primary DHCP Server

The *dhcpd.pcy* file, containing the policies applicable to the primary DHCP server, is created automatically on the primary DHCP server when you establish the server as a Lucent DHCP Primary server and perform a “push” to the server (as described in “About DHCP”, on page 16).

Additional parameters *applicable to DHCP failover* must be specified in the *dhcpd.pcy* file in order for the failover to work properly. The **SecondaryIpAddr** option is added by VitalQIP during the push when the Failover Server Type parameter is set to 'Standalone/Primary' in the Server Profile and the Failover Server is specified as described in the **User Failover Server** parameter (see the following table for a description of this parameter).

> **Important!** When configuring the primary server to support only registered clients in a many-to-one failover environment, MAC address pools must be configured at the subnet level, not globally.

The DHCP server parameters applicable to the Primary server type are described in Table 43.

Table 43 Failover Parameters for the Primary DHCP Server

Parameter name	Values type	Description
CtlReqRetryMax	Numeric Default: 3	If no CtlRet message is received from the failover server in the time specified by WaitCtlRetSecs, attempt to contact the failover by resending the CtlReq message this many times. If this maximum is reached without receiving a CtlRet response, the primary assumes that the failover is inoperable, and continues on to operate as a DHCP server. It continues to send all binding changes to the server identified in the SecondaryIpAddr parameter, even though that server may not be up. The recommended maximum value is 10. **Note:** If both the CtrlReqRetryMax and WaitCtlRetSecs parameters are specified, while the DHCP server is waiting, no addresses are given out. Large values could cause delays in offering leases following a server startup.

Parameter name	Values type	Description
PollDelay	Numeric in seconds Default: 60	The amount of time between each poll/reply sequence, specified in seconds. Upon startup, after synchronization (if specified), the failover sends a Poll message to the primary server. It waits for the amount of time specified by the WaitPollRplSecs parameter for a reply. If a reply is received, the failover server “sleeps” for the amount of time specified by this parameter before sending another Poll message to the primary server. The maximum value is 86400 (1 day).
SyncBindRetryMax	Numeric Default: 3	The number of times the server should resend a binding update if an Ack is not received within WaitSyncBindAckSecs.
SyncBindingBufSize	Numeric in bytes Default: 1024	Size of the “options” area for synchronizing the binding information between the primary and secondary servers. **Note:** This value *must* be the same on *both* servers. The minimum value is 64. The maximum value is 4096.
Use Failover Server	True | False Default: False	If set to True, the Failover Server parameter appears, where you must select the fully qualified host name of the failover server to be used by the primary server. If set to False, no failover server is associated with the primary server.
WaitCtlRetSecs	Numeric in seconds Default: 5	The number of seconds that this primary server should wait for a response to the CtlReq (request for control) message sent to the failover server at start-up. The maximum number of seconds is 3600 (1 hour). If a value of zero is supplied, the default is used. **Note:** If both the CtlReqRetryMax and WaitCtlRetSecs parameters are specified, while the DHCP server is waiting, no addresses are given out.
WaitSyncBindAckSecs	Numeric Default: 5	The number of seconds the server waits for an Ack to a binding update packet when operating on the sending side of synchronizing with the other server.
WaitSyncBindUpdateSecs	Numeric Default: 15	The number of seconds that the server should wait for subsequent binding update packets when operating on the receiving side of synchronizing with the other server.

Configuring the failover DHCP server

Policies applicable to the failover DHCP server must be specified in the *dhcpd.pcy* file if the failover is to work properly. These options are added automatically by VitalQIP during the push when the Failover Server Type is set to 'Failover/Secondary' in the Server Profile.

The *dhcpd.pcy* file on the failover DHCP server can contain *any* policy (as discussed in "DHCP Server policies" in Chapter 2 of the *VitalQIP User's Guide*) except the failover policies that are specific to the primary server.

> **Important!** Please remember to include the regular *dhcpd.pcy* policies, along with the policies specific to the failover server. Otherwise, the failover server does not know its basic parameters of operation.

The DHCP server parameters applicable to the failover/secondary server are described in Table 44.

Table 44 Failover parameters for the secondary DHCP server

Parameter name	Value type	Description
PollRetryMax	Numeric Default: 3	If no reply to the Poll message is received from the primary, the failover retries sending the Poll message and waiting for a reply for this number of times before assuming that the primary has crashed and becomes active in handling DHCP client requests. The maximum is 10.
SyncBindingBufSize	Numeric (in bytes) Default: 1024	Size of the "options" area for synchronizing the binding information between the primary and secondary servers. **Note:** This value ***must*** be the same on ***both*** servers. The minimum value is 64. The maximum value is 4096.
SyncBindings	True | False Default: True	If this policy is True, this failover server requests all current binding information from the primary at startup.
SyncBindRetryMax	Numeric Default: 3	The number of times the server should resend a binding update if an Ack is not received within **WaitSyncBindAckSecs**.
SyncFailCritical	True | False Default: False	If True, the failover server terminates upon failure to synchronize bindings with the primary server. Note that this option only applies if the **SyncBindings** option is True.

Parameter name	**Value type**	**Description**
SyncReqRetryMax	Numeric Allowed: 3	If no SyncStart message is received from the primary server in the time specified by **WaitSyncStartSecs**, attempt to contact the primary by resending the SyncReq message this many times. If this maximum is reached without receiving a SyncStart response, this failover checks the value of the **SyncFailCritical** policy. If that policy is True, this secondary server terminates. Otherwise, this secondary server initiates polling of the primary server. The maximum value is 10.
WaitPollRplSecs	Numeric (in seconds) Default: 5	The number of seconds that the failover should wait for a Poll reply from the primary. Note that if the failover receives a binding update from the primary during this period, the primary is assumed operational, and the binding update message is taken as a response to the poll. The maximum is 3600 (1 hour).
WaitSyncBindAckSecs	Numeric Default: 5	The number of seconds the server waits for an Ack to a binding update packet when operating on the sending side of synchronizing with the other server.
WaitSyncBindUpdateSecs	Numeric Default: 15	The number of seconds that the server should wait for subsequent binding update packets when operating on the receiving side of synchronizing with the other server.
WaitSyncStartSecs	Numeric (in seconds) Default: 5	The number of seconds that the failover server should wait for a response to the SyncReq (request for bindings) message sent to the primary server at start-up. A value of zero causes the server to wait indefinitely. The maximum value is 3600 (1 hour).

The *dhcpd.pcy* policy file must exist for each failover server. Each primary server is defined with a sequential number following the parameter, as indicated by the following example:

```
PrimaryIpAddr1=198.200.138.207
PrimaryIpAddr2=198.200.138.243
PrimaryIpAddr3=198.200.138.205
```

Additional policies for the Lucent DHCP server

Additional policies for the Lucent DHCP 5.x server are available to configure the server's behavior. Additional policies can be added by using **Infrastructure|Server** in the VitalQIP client and overriding those specified as server parameters. You can also edit the *dhcpd.pcy* file to include these policies. Other Lucent DHCP policies are available. Refer to "Servers" in Chapter 3 of the *VitalQIP User's Guide.*

Important! If you are editing the *dhcpd.pcy* file, do not change policies that are not documented. They are intended for internal use by Lucent Technologies only.

Important! In VitalQIP 5.x, all non-default DHCP server policies were specified as Additional Policies. If your data was imported from VitalQIP 5.x, there may be unnecessary policies listed in Additional Policies. If this situation occurs, these policies can be specified in the Server Profile or Server Policy Template. The ones appearing in Additional Policies should be deleted.

Adding additional Lucent DHCP policies

When to use

This section describes how to add additional Lucent DHCP policies.

Procedure

To add additional policies through the VitalQIP client, follow these steps:

1 In the VitalQIP administrative client, access **Infrastructure|Server**. The Server Profile Option window opens.

2 In the Server Profile Option window, select your Lucent DHCP server from the **Existing Server** list.

3 Select **Modify Server**.

4 Click **OK**. The Server Profile window opens.

5 In the Server Profile window, select **Additional Policies** from the Parameters/Values list. A **Value** field appears.

6 In the **Value** field, type the policy in Parameter=Value format. Table 45 describes the additional policies that are available.

Table 45 Additional Lucent DHCP Server Policies

Policy	Values	Descriptions
DHCPServer	IP address Default: None	See "DHCPServer and DHCPSocketAddr policies" for more information about this policy.
DHCPSocketAddr	IP address Default: None	See "DHCPServer and DHCPSocketAddr policies" on page 299 for more information about this policy.
SiAddr	IP address|Hostname Default: Dotted decimal IP address	This policy specifies the IP address or hostname

Policy	Values	Descriptions
NumberofThreads	Numeric Default: 15	This policy establishes the number of threads created by the service to process DHCP client requests.
ListenOnLoopback	1 (True)| 0 (False) Default: 0	This policy causes the server to create a socket and listen on port 67 of the loopback interface 127.0.0.1. This policy is only required when used with the Services Manager product, specifically when the DHCP Probe is placed on the same machine with the Lucent DHCP Service.

7 Click **Apply**. The policy and value appear in the Value column next to Additional Policies.

8 Click **OK**.

END OF STEPS

☐

DHCPServer and DHCPSocketAddr policies

The **DHCPServer** policy controls the server's "local" subnet. By default, this policy is not set, allowing for support of multiple "local" subnets, one for each active network interface. If this policy is set, the Lucent DHCP Service has only one "local" subnet, regardless of which NIC a broadcasted packet received. For example, a service has three NICs with the following configured addresses:

```
NIC1 - 10.100.200.1 mask 255.255.255.0
NIC2 - 20.100.200.1 mask 255.255.255.0
NIC3 - 30.100.200.1 mask 255.255.255.0
```

In this example, the service has three "local" subnets when the DHCPServer policy is not set. The "local" subnets are 10.100.200.0, 20.100.200.0, and 30.100.200.0, respectively. DHCP clients can be on the same segment with the Lucent DHCP Service on each of these subnets. Therefore, if the service receives a broadcasted DHCP Discover packet from a client on NIC1, it offers an address from the ranges (if any) that are configured in the 10.100.200.0 subnet. Likewise, a broadcast packet on NIC2 or NIC3 causes the service to offer an address from the 20.100.200.0 or 30.100.200.0 subnet, respectively, if available.

If the **DHCPServer** policy is set, then the Lucent DHCP Service only has one "local" subnet regardless of which NIC a broadcasted packet is received. Thus, in the above example if the following line appeared in the dhcpd.pcy file:

```
DHCPServer=10.100.200.1
```

The service always offers addresses from the 10.100.200.0 subnet for any broadcasted packets that are received locally, regardless of which NIC the packet received.

In all cases, the service honors the "GiAddr" field that is set when a packet is forwarded through a DHCP relay agent. If this field in the Bootp/DHCP packet is non-zero, the service always offers an address (from ranges defend for the subnet) that matches the address in the "GiAddr" field, if any address is configured and available.

The **DHCPSocketAddr** policy is used to configure which network interfaces (NICs) the service should listen on. By default, this policy is not set allowing the service to listen on all NICs. In order to specifically tell the Lucent DHCP Service to ignore an interface, each desired interface must be set using this policy, and if specified, the service binds to port 67 on the NIC that has this address. Otherwise, the service binds to all interfaces. For example, using the configuration above, setting the following in the dhcpd.pcy file:

```
DHCPSocketAddr=10.100.200.1
DHCPSocketAddr=20.100.200.1
```

Causes the service to listen on NICs 1 and 2; the service ignores any Bootp/DHCP packets received on NIC3.

Important! On machines in which support for all DHCP clients are separated from the Lucent DHCP server by a router or DHCP relay, performance is improved by instructing the server to operate at the socket interface level. To do so, set the **DHCPServer** policy to the server's IP address and set the **DHCPSocketAddress** policy to 255.255.255.255. When the **DHCPSocketAddr** policy is set to 255.255.255.255, the server must be stopped and restarted. Do not use the **HUP** command.

□

Unique hostname resolution options

Two options are available which provide unique hostname resolution options - FIRST-IN and LAST-IN.

FIRST-IN/LAST-IN methodology

Once DHCP clients are deployed in an enterprise network, it is difficult to enforce the DHCP client hostname. Users can too easily change the name that is assigned without knowing the ramifications. To help manage this problem, VitalQIP has implemented two methods of configurable unique hostname resolution policies. They are defined as **FIRST-IN** and **LAST-IN**. FIRST-IN and LAST-IN are configured via the **Policies|Global Policies** (Policy Class - General) function in the VitalQIP client.

The two methods are described as follows.

FIRST-IN

Using FIRST-IN, if a new dynamic object is being added and it has a duplicate hostname, the existing object hostname remains the same and the new object is assigned a new hostname. The new object's new hostname is a combination of hostname and defaultname or hostname and lastusedname (see Example 1).

Example 1

1. The existing hostname is shown in Table 46.

Table 46 Existing hostname

IP Address	Hostname	Status	Type
192.200.138.1 192.200.138.2	mydhcppc pc0002pc	used unused	dynamic dynamic

2. A new dynamic object with the existing hostname of **mydhcppc** is added at IP address 192.200.138.2. It is a duplicate of an existing hostname.
3. The hostname of the new object is changed to a combination of **hostname-defaultname** or **hostname-lastusedname**. The original name (the first one in Table 47) stays the same.

Table 47 FIRST-IN hostname changes

IP Address	Hostname	Status	Type
192.200.138.1 192.200.138.2	mydhcppc mydhcppc-pc0002pc	used used	Dynamic dynamic

LAST-IN

Using LAST-IN, when a new dynamic object is being added and a duplicate hostname is found, the object hostname that was added first (for example, the existing object) is changed to a combination of **hostname** and **defaultname** or **hostname** and **lastusedname** (see Example 2).

Example 2:

1. The existing hostname is shown in Table 48.

Table 48 LAST-IN existing hostname

IP address	Hostname	Status	Type
192.200.138.1 192.200.138.2	mydhcppc pc0002pc	Used unused	dynamic dynamic

2. A new dynamic object with the existing hostname of **mydhcppc** is added at IP address 192.200.138.2. It is a duplicate of an existing hostname.
3. The hostname of the existing object is changed to a combination of **hostname-defaultname** or **hostname-lastusedname**. The new object (the last one in) receives the requested hostname, as Table 49 shows.

Table 49 LAST-IN object receives requested name

IP Address	Hostname	Status	Type
192.200.138.1 192.200.138.2	mydhcppc-pc0001pc mydhcppc	used used	dynamic dynamic

Important! Dynamic objects never override a static object hostname or static object alias name. This protects static objects within your network and does not allow a dynamic object to ever obtain the same name as a previously defined static object. If this were to occur, DNS could inadvertently be updated, and you would not be able to resolve your static object hostname or static object alias name. This could cause problems if your static object is a router or server. See Example 3.

Example 3:

1. The existing hostname is shown in Table 50.

Table 50 LAST-IN Existing Hostname

IP Address	Hostname	Status	Type
192.200.138.1 192.200.138.2	myserver pc0002pc	used unused	static dynamic

2. A new dynamic object with the existing hostname of **myserver** is added at IP address 192.200.138.2. It is a duplicate of an existing hostname, as shown in Step #1.

3. Because the new object is a dynamic object, the hostname of the new object is changed to a combination of **hostname-defaultname** or **hostname-lastusedname** in Table 51. The original static name stays the same.

Table 51 LAST-IN original static Nname remains the same

IP Address	Hostname	Status	Type
192.200.138.1	myserver	used	static
192.200.138.2	myserver-pc0002pc	used	dynamic

□

QIP Update Service processing of DHCP lease updates

This section outlines the current database logic for the QIP Update Service processing of DHCP Lease updates into the VitalQIP database. The global policies that are used by the update process are listed again (they are also described in Chapter 2 of the *VitalQIP User's Guide*, and then the business logic is outlined.

Important! The term "incoming" in this section describes the QIP DHCP object that is being passed into the VitalQIP database by the QIP Update Service.

Global policies

Accept Client Names

This policy is set for each DHCP server that issues DHCP leases and is set to true or false. If the policy is set to true, then the DHCP objects in the QIP database are updated with the fully qualified domain names that are generated by the DHCP clients themselves. If the policy is set to false, then the QIP database will use the generated default name for the DHCP object name.

Generate Simple Unique Names

This global policy determines how hostnames are generated within the QIP database if there is some form of fully qualified domain name collision. If this policy is set to true, then a unique generated hostname will be the requested hostname and it's IP Address concatenated. The IP Address is zero filled from the left for each octet. For example, *foobar.lucent.com* with IP Address of 10.100.30.1 conflicts with *foobar.lucent.com* at IP Address of 10.200.120.5 in the database. Therefore, the unique hostname will be renamed to foobar-010100030001. If this policy is set to false, the unique hostname will be generated using the VitalQIP naming policy that contains prefix, suffix and numeric values. If this policy is set to true, there is less database overhead with generating a unique hostname, and an increase in performance may be seen with processing DHCP lease updates via the QIP Update Service.

FirstIn-LastIn

This global policy determines which QIP object is allowed to keep its requested hostname in the case of dynamic fully qualified domain name collisions within the DHCP Update lease process in the QIP database. If this policy is set to FirstIn, the dynamic object that currently exists in the database that has the same fully qualified domain name as an incoming DHCP lease has, would get to keep its name. The incoming DHCP lease update hostname would get a new generated hostname. If this policy is set to LastIn, the incoming DHCP lease update hostname will get to keep its requested name, and the existing dynamic object in the database that has the same fully qualified domain name will be renamed with a unique hostname.

Protect MDHCP/Bootp Objects

This global policy determines whether or not the DHCP lease update process modifies Manual DHCP/Bootp object hostnames. If the policy is set to true, Manual DHCP/Bootp object hostnames will not be modified during the DHCP lease update process if the MDHCP/Bootp object's fully qualified hostname conflicts with another dynamic object's fully qualified hostname. The dynamic object's hostname will be renamed to a unique hostname. If this policy is set to false, then the Manual DHCP/Bootp object will be treated like any other dynamic object, and depending on what the FirstIn-LastIn policy is set to, will determine which dynamic object is allowed to retain its requested hostname.

Database business logic

MDHCP/BOOTP Objects

The Update lease process will first look for any other Manual DHCP/Bootp objects that may have the same MAC Address that an incoming lease update for a Manual DHCP/Bootp lease has. If the process finds an MDHCP/Bootp object that has the same MAC Address on the same subnet as the incoming lease, then an exception will be thrown, and the update lease process will fail for that incoming lease. Basically, the MDHCP/BOOTP object that is already in the database with the same MAC Address will be honored, and the incoming Manual DHCP/Bootp lease update will be rejected. If the same Manual DHCP/Bootp object with the same MAC Address is found on another subnet, then the update lease process will continue.

DHCP Objects changing domains

If an incoming lease for a DHCP object changes from one domain to another, then as long as the domain is a valid domain on the subnet, the update process will honor the domain change and notify the DNS Update Service of the change. If the domain specified for the DHCP object is not a valid domain on the subnet, or if the domain is null, then the default domain on the subnet will be assigned to the DHCP object.

Fully qualified domain name collision checking

Each DHCP lease update coming into the database via the QIP Update service will be checked against other DHCP objects that exist in the database already for the same fully qualified domain name. This check will only take place if the incoming DHCP lease update has indicated that it's hostname and/or domain name has changed or if the DHCP lease is a new lease. If nothing has changed on the DHCP lease update other than the lease grant/expiration dates and/or the MAC address, then the update lease process will simply update the lease grant/expirations dates and MAC Address for the DHCP object. If there is an indication of hostname and/or domain name change, then the name collision checking needs to be processed.

The first check is to determine if the incoming DHCP lease update fully qualified domain name conflicts with an alias. If there is an alias that has the same FQDN, then the alias is treated as a static object, and it's hostname will not be modified. However, the incoming DHCP lease hostname will be modified. A generated unique hostname will be assigned.

The second check is to determine if the incoming DHCP lease update fully qualified domain name conflicts with a static or dynamic object. If a duplicate FQDN exists in the database, then additional processing is required and outlined below:

- If a static object has the same FQDN, and the object is not a tombstoned object, then the static object will get to keep it's hostname, and the incoming DHCP lease hostname will be modified to a unique generated hostname.
- If the static object is a tombstoned object, then the tombstoned object hostname will be modified to a unique generated hostname, and the DHCP lease update object will be allowed to keep it's requested FQDN.
- If a MDHCP/Bootp object has the same FQDN, and the Protect MDHCP/Bootp object policy is set to true, then the MDCHP/Bootp object is treated like a static object, and will be allowed to keep it's fully qualified domain name. The incoming DHCP lease update hostname will have its requested name changed to a unique generated hostname.
- If an MDHCP/Bootp Object has the same FQDN, and the Protect MDHCP/Bootp object policy is set to false, then the MDHCP/Bootp object is treated like a dynamic object. Then based on what the value of FirstIn-LastIn policy is set to, will determine which object (the MDHCP/Bootp object or the incoming DHCP lease object) retains its hostname, and which object's hostname will be changed to a unique generated hostname.
- If an incoming DHCP lease update has the same FQDN as an existing dynamic object in the database, and they both have the same MAC address, then we will assume that both leases are related to the same DHCP client. The DHCP client is roaming on subnets without the QIP DHCP lease update process first deleting the lease information from the database via a delete lease action. If the DHCP client is roaming, then the incoming DHCP lease update fully qualified domain name will be updated at the IP address it's requesting, and the existing dynamic object in the database with the same FQDN will be reverted back to it's original default name in the database, and the MAC address and lease dates will be removed.
- If an incoming DHCP lease update has the same FQDN as an existing dynamic object in the database, but they both have different MAC addresses, then depending on the value of the FirstIn-LastIn policy, will determine which dynamic object is allowed to keep its fully qualified domain name, and which object will have a unique generated hostname.

If the check determines that the incoming DHCP lease update fully qualified domain name does not conflict with any other object’s fully qualified domain name within the database, then the update occurs in the database with no additional checking.

In the all cases, a dynamic DNS update will take place for the incoming DHCP lease update as well as a dynamic DNS update for an existing object in the database if that object’s hostname was modified via the process described above.

☐

9 Database administration

Overview

Purpose

This chapter discusses administration of the VitalQIP database. Most sections in this chapter are specifically for Sybase database users. However, some sections can be used by Oracle database users and are indicated accordingly. For Oracle database administration beyond this chapter, see your Oracle database administrator.

Important! ***Only experienced database administrators should perform database administrative tasks***.

For this chapter only, notes are provided for some command line examples. Italicized text after // is a note and is not part of the command.

Contents

This chapter presents the following topics.

Running Sybase

This section is intended for users running the VitalQIP server with a Sybase database. For Oracle users, ensure that your Oracle database is started on the machine where it is located. See your Oracle database administrator for assistance.

Starting Sybase on Windows

Sybase users can start Sybase using the Microsoft Windows Services Controller.

Starting Sybase on UNIX

Sybase users can start Sybase by typing the following at a command line:

```
# cd  $SYBASE/ASE-12_0/install
# RUN_<SQL_Server_Name> <DSQUERY_value>
```

☐

Finding VitalQIP version numbers with vercheck

To obtain the version number of every VitalQIP program in a specific directory (and its subdirectories), run the **vercheck** utility. The resulting information is displayed in the window or sent to a file The returned information includes the filename, file size, date/time stamp, file type, version number, and the file's **checksum**. This utility helps you maintain consistency between upgrades. Use the following format to run the utility from the command line:

```
vercheck [-d directory] [-m field_mask] [-h] [-v] [-c] [-e]
 [-z] [-5] [filename]
```

The following parameters can be used with **vercheck**:

`-d` *directory*	Specifies the directory for which version information is to be obtained. The default is the current working directory.
`-m` *field_mask*	Identifies the fields that are to be displayed by **vercheck**. The fields are identified using a field mask consisting of zeros (0) and ones (1). One indicates that the field should be displayed, and a zero indicates that the field should not be displayed. The fields, in order of specification, are as follows: - File Name - File Size - File Owner - File Permissions - File Creation Date - File Modification Date - File Type - File Version - File Checksum
`-h`	Provides the syntax of the command.
`-v`	Provides the version information of the **vercheck** utility.
`-c`	Outputs the information in CSV format.
`-e`	Provides VitalQIP environment information.
`-z`	Provides only the filenames and product version numbers.
`-5`	Searches only for VitalQIP 5.x versions. If this parameter is omitted, **vercheck** searches for all VitalQIP versions.
filename	Provides information about the specified file. If this parameter is omitted, all files in the current directory or the directory specified by the **-d** option are processed. The subdirectories are also processed.

□

Backing up the VitalQIP Sybase database

You first must decide if you are using VitalQIP's scripts to do the backups or using your own backup method. There are separate sections for each described here.

Three different backups exist to complete a total Sybase backup. They are backing up the VitalQIP database, backing up the transaction log and backing up the master database.

Backing up the VitalQIP database should be done frequently. In the event of a power outage, loss of disk drives, or any other loss, you can retrieve the latest backup version of your VitalQIP database.

Backing up the transaction log is used for “near point in time” recovery. You can recover to the last database backup, and then apply the transaction logs up to the point of failure. If you chose to back up your transaction logs, change the **‘truncate log on checkpoint’** to “False”. The VitalQIP installation process sets this option to “True”. If you set the **‘truncate log on checkpoint’** to “False”, Sybase writes transactions to the transaction log. When the log fills up, all modifications to the database fail immediately. It is important that the log file be dumped frequently to ensure “near point in time” recovery.

The Master database should be backed up after increasing the database or log size, changing configuration parameters using **sp-configure** (for example, user connections and memory).

> **Important!** It is strongly suggested that you back up the entire VitalQIP database ***daily*** by running **backup_qip_dat**, and if near-point-in-time recovery is required, backup the VitalQIP transaction log ***several times a day*** by running **backup_qip_log**. The frequency of transaction log backups depends upon how fast the transaction log fills up. It is necessary to use Sybase's utilities to back up the database. ***A file system backup is not sufficient.*** You should also back up the database and transaction log immediately after you import a large file. For information about **backup_qip_dat** and **backup_qip_log**, see “Backing up the VitalQIP database with VitalQIP scripts” on page 314.
>
> If you set **sp_dboption QIP, ‘trunc log on chkpt’, true**, Sybase truncates the log automatically. You do not need to backup the VitalQIP log.

All backups can be done via tape or hard drive (disk file).

☐

Preparing to back up the transaction log

If you choose to back up the VitalQIP transaction for “near point in time” recovery, set the **sp_dboption truncate** log on checkpoint to “False” by running from a command line:

```
isql -U sa -P <sa_password>
1> sp_dboption QIP, 'trunc log on chkpt',false
2> go
1> use QIP
2> go
1> checkpoint
2> go
1> quitUsing the Backup Server
```

□

Using the backup server

The backup server must be running prior to performing a backup. To determine whether the server is running, run the following from a command line:

```
ps -ef | grep backupserver
```

To start the backup server, run the following from a command line:

```
$SYBASE/ASE-12_0/install/RUN_QIPSYBASE_BACKUP  //assuming the name of
  the backup server is QIPSYBASE_BACKUP
```

☐

Backing up the VitalQIP database with VitalQIP scripts

VitalQIP comes with two scripts to back up the VitalQIP database and VitalQIP transaction log on a hard drive: **backup_qip_dat** and **backup_qip_log**. The **backup_qip_dat script** backs up the VitalQIP database on a hard drive. The **backup_qip_log** backs up the VitalQIP transaction log by dumping the VitalQIP transaction log.

Script for the VitalQIP database

Before you back up the VitalQIP database, make sure there is enough space available on the hard drive. To back up the VitalQIP database, run the following from a command line:

```
$QIPHOME/script/backup_qip_dat
```

Script for the VitalQIP transaction log

To dump the VitalQIP transaction log, run the following from a command line:

```
$QIPHOME/script/backup_qip_log
```

After dumping the transaction log, move it to a new location before the next dump occurs. The database recovery using the transaction log requires applying all logs in the order they are dumped. If one is missing, you can only recover to your last database backup.

□

Manual backup

If you create your own scripts for backing up your database, you must create your dump devices first. You can only create these devices one time. If you change those dump devices, you must first dump them. For instructions about dumping those devices, see “Changing the backup medium” on page 317.

Creating the database dump devices for the VitalQIP database

Dump devices only need to be created one time. The instructions covered in this section are used in preparation for backing up the VitalQIP database.

The examples presented in this section use arbitrary names for the device. The device’s name may vary in different UNIX environments. ***The text not bolded and in quotes is an example only.*** Consult your system administrator for the proper device name.

Creating a database dump device for a tape

If the database is backed up to a tape device and the maximum capacity of the tape device is 1GB (1000MB), run:

```
1> sp_addumpdevice "tape", "qip_dump_dat", "/dev/rmt/0m", 6, skip, 1000
2> go
```

Creating a Database Dump Device for a Hard Disk

If the database is backed up to a hard disk, run the following from a command line:

```
1> sp_addumpdevice "disk", "qip_dump_dat", "/opt/qip/qip_dump_dat"
2> go
```

Once the dump device is created, proceed to the “Backing up the VitalQIP Sybase database”, on page 311”.

Creating the VitalQIP transaction log dump devices

Dump devices only need to be created one time. The instructions covered in this section are used in preparation for backing up the VitalQIP transaction log.

The examples presented in this section use arbitrary names for the device. The device’s name may vary in different UNIX environments. ***The text not bolded and in quotes is an example only.*** Consult your system administrator for the proper device name.

Creating a transaction log dump device for a tape

The examples presented that follow use arbitrary names for the device. The device’s name may vary in different UNIX environments. ***The text not bolded and in quotes is an example only.*** Consult your system administrator for the proper device name.

If the database is backed up to a tape device and the maximum capacity of the tape device is 1 GB, run the following from a command line:

```
1> sp_addumpdevice "tape", "qip_dump_log", "/dev/rmt/0m", 6, skip, 1000
2> go
```

Creating a database dump device for a hard disk

If the database is backed up to a hard disk, run from a command line:

```
1> sp_addumpdevice "disk", "qip_dump_log", "/opt/qip/qip_dump_log", 2
2> go
```

Once the dump device is created, proceed to the "Backing up the VitalQIP Sybase database" on page 311.

Backing up the VitalQIP database and transaction log

To back up the VitalQIP database to a tape or file system, run from a command line:

```
1> dump database QIP to qip_dump_dat
2> go
```

To back up the VitalQIP transaction log to a tape or file system, run from a command line:

```
1> dump tran QIP to qip_dump_log
2> go
```

Creating the master database dump device

A master database dump device is only created once. The examples presented in this section use arbitrary names for the device. The device's name may vary in different UNIX environments. ***The text not bolded and in quotes is an example only.*** Consult your system administrator for the proper device name.

```
1> sp_addumpdevice "disk","master_dump","/opt/qip/master.dump"
2> go
```

The following commands are run on a regular basis:

```
1> dump database master to master_dump
2> go
```

□

Changing the backup medium

If you have performed a backup before and you want to change the backup medium, dump the devices first. To dump the devices, run the following from a command line:

```
isql -U sa -P <sa_password>
1> sp_dropdevice qip_dump_dat
2> go
```

After dumping the devices, follow the procedure in the previous section, “Backing up the VitalQIP database and transaction log”.

☐

Changing the procedure cache size and total memory size

The following scripts show how to change the procedure cache size and the total memory size of your Sybase database. These scripts are for Sybase:

```
sp_configure “total memory”, 7500 //the default in 2k block
sp_configure “procedure cache percent”, 20//the default is 20% of
  memory
```

□

Encrypting your password

You may encrypt your “qipadmin” password on Sybase and Oracle by running the **qip-crypt** utility. **qip-crypt** takes the password as the first argument and sends an encrypted password to the screen (also known as STDOUT), unless you specify a file name in the CLI command. To run **qip-crypt**, run:

```
qip-crypt <your_password>
```

The newly encrypted password must be placed in the Global section of the *qip.pcy* file and set as password=<*encrypted_password*>. To find the newly encrypted password, access the filename you defined in the utility or see STDOUT. Refer to “Manage the VitalQIP policy file”, on page 95 for more information on the *qip.pcy* file.

For additional information about **qip-crypt**, see the *VitalQIP Command Line Interface User’s Guide.*

☐

Re-initializing your database

VitalQIP has a CLI, **qip-dbinit**, which can be used to re-initialize your database under specific circumstances. The CLI can be used for Oracle and Sybase.

Once your VitalQIP database and Master database are in place, you can begin building your infrastructure, setting up your domains, setting up your DHCP and Bootp servers, and all the other preliminary processes required for the use of VitalQIP. If at any point you need to rebuild the infrastructure, and begin again from a "clean slate" so to speak, the **qip-dbinit** procedure can be used.

Important! The **qip-dbinit** process is automatically run during an upgrade of VitalQIP.

□

qip-dbinit

The **qip-dbinit** CLI command clears the data from the database, re-initializes the database, and reinstalls the triggers, stored procedures, tables, and indexes. It is run as part of a new installation of VitalQIP. It can also be run to re-initialize your VitalQIP database (for example, erase a test system). This CLI can be used for Audit Manager and VitalQIP database as well as a Sybase or Oracle database.

Synopsis

```
qip-dbinit [-t qip_dbase] -s qip_dataserver -u username -p password
 [-l log_file] [-q output_file] [-i script_path][-b db_name] [-a] [-k]
```

Parameters

qip-dbinit recognizes the following parameters:

`-t` *`qip_dbase`*	Specifies the database server type: either **Oracle** or **Sybase**.
`-s` *`qip_dataserver`*	Specifies the name of the database server. The database server name must match the Sybase server name or the Oracle database alias name. This parameter is optional if the $QIPDATASERVER environment variable is set. The command line argument overrides the environment variable. For Audit Manager, the Audit Manager database server name must be specified.
`-u` *`username`*	Specifies the VitalQIP administrator account to be used in establishing the database connection.
`-p` *`password`*	Specifies the password for the associated administrator account.
`-l` *`log_file`*	Specifies the name of the log file.
`-q` *`output_file`*	Quiet Mode. If this parameter is not used, the output is sent to STDOUT by default.
`-i` *`script_path`*	Specifies the directory where the SQL scripts are located.
`-b` *`database_name`*	Specifies the database name: **QIP** or **LAM**. If this parameter is omitted, the default (QIP) is used. You cannot specify QIP and LAM at the same time.
`-a`	Appends to the log file. The default is to overwrite the log file.
`-k`	Skips the prompt.

☐

Performing database administrative tasks with qip-util

qip-util is a CLI used to perform specific database functions that can be helpful for a database administrator. The **qip-util** CLI can perform functions, such as determine who can connect to the database and to which file data is written. This CLI can be used on the VitalQIP or Audit Manager database as well as a Sybase or Oracle database.

The following sections provide a synopsis and description of the parameters used by **qip-util**. Function values shows the values that can be passed as parameters to **qip-util**.

Synopsis

```
qip-util [-t qip_dbase] [-s qip_dataserver] [-u username]
 [-p password] [-l log_file] [-q output_file] [-i script_path]
 -b db_name] [-a] FUNCTION VALUES…
```

Parameters

qip-util recognizes the following options:

`-t` *qip_dbase*	Specifies the database server type: either **Oracle** or **Sybase**.
`-s` *qip_dataserver*	Specifies the name of the database server. The database server name must match the Sybase server name or the Oracle database alias name. This parameter is optional if the $QIPDATASERVER environment variable is set. The command line argument overrides the environment variable. For Audit Manager, the Audit Manager database server name must be specified.
`-u` *username*	Specifies the VitalQIP administrator account to be used in establishing the database connection.
`-p` *password*	Specifies the password for the associated administrator account.
`-l` *log_file*	Specifies the name of the log file.
`-q` *output_file*	Quiet Mode. If this parameter is not used, the output is sent to STDOUT by default.
`-i` *script_path*	Specifies the directory where the SQL scripts are located.
`-b` *database_name*	Specifies the database name; **QIP** or **LAM** (Audit Manager). This parameter defaults to the VitalQIP database if it is not specified. QIP and LAM databases cannot be specified at the same time.
`-a`	Appends to the log file. The default is to overwrite the log file.

Table 52 shows the function values that can be used.

Table 52 Function values

Function values	**Description**	**Database**	**Product**
CalculateQIPSize	Estimates the size of the VitalQIP database by using <*number_of_objects*|*number_of_subnets*>.	Sybase/ Oracle	VitalQIP
CalculateLAMSize	Estimates the size of the LAM (Audit Manager) database based on the following: <*number_of_dhcp_clients*> <*number_of_static_objects*> <*number_of_nt_objects*>	Sybase/ Oracle	Audit Manager
CheckDatabaseLogin	Checks to see if the connection to Sybase/Oracle is OK.	Sybase/ Oracle	VitalQIP/Audit Manager
CheckDBProcesses	Shows the number of processes that are currently connected to the Sybase/Oracle database by using <*database_name*>. The default is VitalQIP.	Sybase/ Oracle	VitalQIP/Audit Manager
CheckVersionFromDatabase	Checks the version information from the qip_version table in the Sybase/Oracle database by using <*database_name*>. The default is VitalQIP.	Sybase/ Oracle	VitalQIP/Audit Manager
CheckVersionFromData	Checks the version information from the **qip_version** (*qef*) file by using <*export_path*>.	Not available	VitalQIP/Audit Manager
CheckVersionFromScript	Checks the version information from the **table.sql** script.	Sybase/ Oracle	VitalQIP/Audit Manager
CheckSybaseDevice	Checks to see if a Sybase device exists by using <*logical_device_name*>.	Sybase	VitalQIP/Audit Manager
CheckSybaseDatabase	Checks to see if a Sybase database exists by using <*database_name*>. The default is VitalQIP.	Sybase	VitalQIP/Audit Manager
CheckUserExist	Checks to see if the database login user exists by using <*login_name*>.	Oracle	VitalQIP

Function values	Description	Database	Product
ClearAdmin	Removes all logins and users, which are assigned to the database with the related roles or groups by using <*database_name*>. The default is **VitalQIP**.	Sybase/ Oracle	VitalQIP/Audit Manager
ClearData	Truncates all data from all user tables by using <*database_name*>. The default is **VitalQIP**. For Sybase, drops all tables with type equal to "U" (user tables, not system tables). For Oracle, drops all tables owned by the user running **qip-util**.	Sybase/ Oracle	VitalQIP/Audit Manager
CreateAccess	Calls **create_access.sql**. This function should only be called by the installation.	Sybase/ Oracle	VitalQIP/Audit Manager
CreateSybaseDatabase	Creates a Sybase database by using <*data_device_name*><*data_size*> <*log_device_name*><*log_size*> <*database_name*>.	Sybase	VitalQIP/Audit Manager
CreateSybaseDevice	Creates a Sybase device by using <*logical_device_name*| *physical_name size*|*device_size*>.	Sybase	VitalQIP/Audit Manager
DropIndex	Drops all indexes on all user indexes in the database by using: <*database_name*>. The default is **VitalQIP**. For Sybase, drops all indexes on all user tables. For Oracle, drops all indexes owned by the user running **qip-util**.	Sybase/ Oracle	VitalQIP/Audit Manager
DropSP	Drops all stored procedures in the database by using: <*database_name*>. The default is VitalQIP. For Sybase, drops all stored procedures in the database. For Oracle, drops all stored procedures owned by the user running **qip-util**.	Sybase/ Oracle	VitalQIP/Audit Manager
DropTable	Drops all user tables in the database by using: <*database_name*>. The default is **VitalQIP**. For Sybase, drops all user tables in the database. For Oracle, drops all tables owned by the user running **qip-util**.	Sybase/ Oracle	VitalQIP/Audit Manager

Function values	Description	Database	Product
DropTrigger	Drops all triggers on all user tables in the database by using <*database_name*>. The default is VitalQIP. For Sybase, drops all triggers for all user tables in the database. For Oracle, drops all triggers for all user tables owned by the user running **qip-util**.	Sybase/ Oracle	VitalQIP/Audit Manager
EditTextFile	Finds the text file name and the first line containing the text you want to delete. You can establish what to add in its place with the “add_line” (optional). Use <*text_file_name*><*delete_line*> <*add_line*>.	Not available	Not available
EstimateRequiredSpace	Estimates the minimum disk space used during **qip-import** using <*database_name*>.	Sybase/ Oracle	VitalQIP/Audit Manager
GetAndSetSybaseDBOption	Gets and sets the value of the Sybase dboption "trunc.Log on chkpt” by using <*option_value*><*database_name*>. The default is VitalQIP.	Sybase	
GetDatabaseSize	Gets and sets the size of the database by using <*database_name*>. The default is VitalQIP.	Sybase	VitalQIP/Audit Manager
OracleReCompile	Recompiles all stored procedures and triggers owned by the user running **qip-util** by using <*database_name*>. The default is VitalQIP.	Oracle	VitalQIP/Audit Manager
RemoveInvalidChars	Removes all invalid (unprintable) characters from all files in the specified directory by using <*export_path|file_name*>.	Not available	VitalQIP
RemoveSpaceFields	Goes through all the string fields on all user tables and removes all space-only fields by using <*table_name*>.	Sybase	VitalQIP

Function values	Description	Database	Product
SearchReplace	Searches some special characters, and replaces them with proper characters by using <*search_char*> <*replace_char*>. By default, changes "\n" to a single space, a double quote to a single quote, and "^" to a single space.	Sybase	VitalQIP
SetSybaseConfigure	Sets Sybase configuration values based on the file <*config_file*>. QIP and LAM (Audit Manager) databases require the following settings: • Procedure cache percent=22 • Total memory=21577 • Number of locks=100000	Sybase	VitalQIP/Audit Manager
SybaseUpdateStatistics	Runs "update statistics" on all tables by using <*database_name*>.	Sybase	VitalQIP/Audit Manager
OracleUpdateStatistics	Runs an analysis on all objects within the database.	Oracle	VitalQIP/Audit Manager.

Tracking stored procedures and triggers

At times, technical support may ask you to identify the version of a discreet component. A tracking system is in place, which maintains the current version of all VitalQIP stored procedures and triggers. The tracking system is maintained in the VitalQIP database. To display this table, run the following from a command line:

```
isql -U qipman -P <qipman_password>
1> select * from qip_version
2> go
```

The returned information provides version information for all stored procedures and triggers in the system.

□

Managing the VitalQIP data space

When to use

Your system must be periodically checked to ensure enough free space is available to run the VitalQIP system.

Procedure

To check the amount of data space available, follow these steps:

1 Find the total allocation of the VitalQIP database. Run the following from a command line:

```
#isql -U sa -P <sa_password>
1> sp_helpdb QIP
2> go
```

The output looks like this:

```
name                          db_size       owner                   dbid
        created
        status
----------------------- ------------- ----------------------- ------
VitalQIP                      60.0 MB       qipman                  5
        Nov 29, 2003
        select into/bulkcopy/pllsort, trunc log on chkpt

1 row affected)
device_fragments                size          usage                 free kbytes
------------------------------ ------------- -------------------- ----------
qip_dat                         45.0 MB       data only                  25936
qip_log                         15.0 MB       log only                   15328
return status = 0)
```

2 If there are **multiple device_fragments** with a usage of “data only”, add the number in the Size column to determine the “total data allocation” of the VitalQIP database.

3 Find the amount of free space in the data allocation portion of the VitalQIP database. Run the following from a command line:

```
#isql -U sa -P <sa_password>
1> use QIP
2> go
1> sp_spaceused
2> go
```

The output looks like similar to the following:

```
database_namedatabase_size
--------------------------------
QIP65.0 MB
reserveddataindex_sizeunused
---------  -----------------------------------
30330 KB27420 KB456 KB2454 KB
```

4 Determine the amount of free space by subtracting the number in the Reserved column from total data allocation sum determined in Step 2 (for example, 45 MB - 30330 KB is approximately 15 MB).

5 If there is less than 1 MB of free space (1 MB is recommended by Sybase for reliable operation), you must enlarge your data space. Run the following from a command line:

```
#isql -U sa -P <sa_password>
1>sp_helpdevice //*to check the vdevno number used*
2>go
>Use master;
>go
1>disk init  // create a device for QIP data
2>name="newqip_log or dat", physname="<full_path_to_physical_location
  of new device>",
  //the physical location's path must contain "newqip_log or dat"
3>vdevno=9, size=4096//assume 9 is the next unused device number and
  4096 2-KB blocks=8MB
4>go
1>alter database QIP  //This adds an additional 8MB of space for data
2>on newqip_dat=8 //use only as much from the new device as needed, it
  can be extended later using the first approach.  Assume 8 is the
  megabytes you want to add to the database.
3>go
1>quit
```

Important! If **qip_dat** is full, you may use **qip-clear** to remove some or all audit data. For information about **qip-clear**, see "Truncating audit data", on page 339.

☐

Managing the VitalQIP transaction log space

Managing VitalQIP transaction log space involves monitoring disk space utilization, dumping the transaction log, and enlarging the transaction. Instructions are given in this section to monitor, dump, and enlarge the transaction log.

Monitoring transaction log disk space

To monitor the transaction log disk space utilization, run the following from a command line:

```
#isql -U sa -P <sa_password>
1>use QIP
2>go
1>dbcc checktable (syslogs)
2>go
```

The system displays the percentage of the transaction log currently in use.

Dumping the transaction log without un-installing Sybase

To dump the transaction log without uninstalling your database (when your log disk space is full, and there is no space for the truncate-only option), run:

```
# isql -U sa -P <sa_password>
1> dump tran QIP with no_log
2> go
```

Once you complete this process, all the log information in the *syslogs* is destroyed. A dump database process has to be done immediately because the database cannot be recovered with the previous database dump since all logs are destroyed.

After you dump your transaction log, it is recommended that you make one (or both) of the following changes:

- Enlarge the log device
- Reduce the time between transaction log backups

Enlarging the transaction log space

Before enlarging the transaction log space, be aware that if you are adding a new device, make the device big enough for future use. Device numbers are limited.

To enlarge the transaction log space, run the following from a command line:

```
#isql -U sa -P <sa_password>
1>sp_helpdevice //*to check the vdevno number used
2>go
1>disk init // create a device for VitalQIP log
2>name="newqip_log or dat",
  physname="<full_path_to_physical_location>", //The path must contain
  "newqip_log or dat" in the path.
3>vdevno=8, size=1024 //assume 8 is the next unused device number and
  1024 2-KB blocks=2MB (for example, 20MB=10240, 1MB=512, and so on)*
```

```
4>go
1>alter database QIP //This adds additional 2MB space for VitalQIP log
2>log on newqip_log=2 //use only as much from the new device as needed,
 it can be extended later using the first approach
3>go
1>sp_logdevice QIP, newqip_log
2>go
1>quit
```

Sybase Database Recovery

In the event the VitalQIP system goes down or you have lost the master device, steps are provided to recover them. If your system goes down, see the next section, “Recovering the VitalQIP database” to recover the VitalQIP database. If the master device is lost, see the next section, “Recovering the master database”. For Oracle users, see your Oracle database administrator for assistance.

Important! To provide up-to-the-minute recovery, the database option “trunc” must be off.

□

Recovering the VitalQIP database

This section is organized into steps by which the VitalQIP database is recovered. Each step provides in-depth information on how to complete the step. Once a step is complete, proceed to the next step.

To recover from a system crash, use the following procedure:

1. Dump the current transaction log

If you chose to log transactions in your database, determine the VitalQIP transaction log should be dumped by running the following from a command line:

```
1> sp_helpdb QIP
2> go
```

If the status column is set to “`trunc. Log on chkpt.`”, you are not logging transactions. Otherwise, you are logging transactions. If you are logging transactions, run the following from a command line:

```
isql -U sa -P <sa_password>
1> dump tran QIP to qip_dump_log with no_truncate //if running in
log mode, dump the log file
2> go
```

2. Drop the existing database

To drop the existing database, run the following from a command line:

```
1> dbcc dbrepair (QIP, dropdb) // delete the current database
2> go
```

or

```
1> drop database QIP
2> go
```

3. Drop the existing devices

To drop the existing devices, follow these steps:

a. Run the following from a command line:

```
1> sp_helpdevice
2> go
```

b. Write the size, location, and **vdevno** of **qip_dat** and **qip_log**. The database and device sizes are used later. The devices must be at least the same size as they were before they failed.

c. Drop the existing devices by running the following from a command line:

```
1> sp_dropdevice qip_dat// delete the devices for VitalQIP
2> go
1> sp_dropdevice qip_log
2> go
```

4. Shut down and restart the Sybase database

Stop and restart the Sybase database; stop and restart the backup server first.

5. Delete qip_data and qip_log

Delete **qip_dat** and **qip_log** from the operating system. They are located as specified in “3. Drop the existing devices” on page 333 using **sp_helpdevice**.

6. Recreate the database

To recreate the database, follow these steps:

a. Recreate the VitalQIP database by running the following from a command line:

```
isql -U sa -P <sa password>
1> disk init // recreate the device for VitalQIP data
2> name="qip_dat", physname=
"<full_path_to_physical_location>",
3> vdevno= <qip_dat's vdevno>, size= <number_of_blocks>
4> go
```

Important! **vdevno**, location, size were determined in “3. Drop the existing devices“ on page 333 using **sp_helpdevice**. The number of blocks equals the size of **qip_dat** (in MB) x 512 (for example, if **qip_log** equals 30 MB, the number of blocks equals 30 multiplied by 512 for a total of 15360).

b. Recreate the VitalQIP transaction log by running the following from a command line:

```
1> disk init// create a device for VitalQIP log
2> name="qip_log", physname= "<full_path_to_physical_
location>",
3> vdevno= <qip_log's vdevno>, size=<number_of_blocks>
4> go
```

Important! **vdevno**, location, size were determined in Step #3 by using **sp_helpdevice**. The number of blocks equals the size of **qip_log** (in MB) x 512 (for example, if **qip_log** equals 10 MB, the number of blocks equals 10 multiplied by 512 for a total of 5120).

7. Create the VitalQIP database

To create the VitalQIP database, run the following from a command line:

```
1> create database QIP // create a database for VitalQIP
2> on qip_dat =<size in MB> // size = 30 in above example
3> log on qip_log =<size_in_MB> // size = 10 in above example
4> for load
5> go
```

8. Reload the VitalQIP database

Reload the database from the dump database by running the following from a command line:

```
1> load database QIP from qip_dump_dat //for recent database dump
2> go
```

If transactions are logged (this was determined in “1. Dump the current transaction log” on page 332), reload the transaction log. Reloading the VitalQIP transaction log must follow FIFO (FIRST-IN/FIRST-OUT) methodology. In other words, the first dumped file must be the first that you reapply, the second dumped file must be the second that you reapply, and so on.

```
1> load tran QIP from qip_dump_log // for all recent log dumps
2> go // after the recent database dump
```

9. Ensuring a successful recovery

To ensure a successful recovery, follow these steps:

a. To set the server to single user mode, run the following from a command line:

```
> sp_dboption QIP, 'single user',true
> go
> use QIP
> go
> checkpoint
> go
```

b. Create the *dbcc* file by running the following from a command line:

```
> dbcc checkdb(QIP)
> go
> dbcc checkalloc(QIP)
> go
> exit
```

c. Check if the data is loaded correctly by running the following from a command line:

```
isql -U sa -P <password> -i dbcc > dbcc.out
```

d. Review the *dbcc.out* file for errors. If errors are displayed, call technical support.

10. Bring the VitalQIP database online

To bring the VitalQIP database online, run:

```
isql -U sa -P <sa_password>
1> online database QIP
2> go
```

11. Reset administrator access

To reset administrator access, run:

```
cd $QIPHOME script
isql -U sa -P <password> -i create_access.sql
```

12. Change serial number of all primary DNS servers (if necessary)

Restoring the database causes secondary DNS zone transfers to cease because the secondary DNS server's serial number is higher than the primary DNS server's serial number. If this is the case, find the serial number of the current primary DNS server and change it to a higher value.

To find the serial numbers of all primary DNS servers, run the following from a command line:

```
select prim_svr, ser_num from prim_svr
```

To change the serial number of the specified server to any 10-digit number, run:

```
update zone_servers set ser_num=<newnumber> where zone_id=(select
  domn_id from domain where domn_name="<domainname>" and org_id=(select
  org_id from organizations where org_name="<orgname>") ) and
  dns_svr_id=(select server_id from srvrs where domn_id=(select domn_id
  from domain where domn_name="<domainname>" and org_id=(select org_id
  from organizations where org_name="<orgname>") ) )
```

□

Recovering the master database

This section is organized into steps by which the master device is recovered. Each step provides in-depth information on how to complete the step. Once a step is complete, proceed to the next step.

You are referred to the *VitalQIP Installation Guide* for several steps. It is recommended you have the guide handy while you complete these steps.

To recover the master device, follow these steps:

1. Reinstall the Sybase server

To reinstall the Sybase server, refer to Appendix A of the *VitalQIP Installation Guide*.

2. Start the Sybase backup server

See the “Running Sybase“ on page 309 for information about starting the Sybase server.

3. Recreate the database

To recreate the database, follow these steps:

a. Recreate the VitalQIP database. Run the following from a command line:

```
isql -U sa -P <sa_password>
1> disk init // recreate the device for VitalQIP data
2> name=”qip_dat", physname= "<full_path_to_physical_location>",
3> vdevno= <qip_dat's vdevno>, size= <number_of_blocks>
4> go
```

Important! **vdevno**, location, size were determined in “3. Drop the existing devices” on page 333 using **sp_helpdevice**. The number of blocks equals the size of **qip_dat** (in MB) multiplied by 512 (for example, if **qip_log** equals 30 MB, the number of blocks equals 30 multiplied by 512 for a total of 15360).

b. Recreate the VitalQIP transaction log. Run the following from a command line:

```
1> disk init // create a device for VitalQIP transaction log
2> name="qip_log", physname= "<full_path_to_physical_location>",
3> vdevno= <qip_log's vdevno>, size=<number_of_blocks>
4> go
```

Important! **vdevno**, location, size were determined in “3. Drop the existing devices” on page 333 using **sp_helpdevice**. The number of blocks equals the size of **qip_log** (in MB) multiplied by 512 (for example, if **qip_log** equals 10 MB, the number of blocks equals 10 multiplied 512 equals 5120).

4. Create the VitalQIP database

To create the VitalQIP database, run the following from a command line:

```
1> create database QIP // create a database for VitalQIP
2> on qip_dat = <size in MB> // size equals 30 in above example
3> log on qip_log = <size_in_MB> // size equals 10 in above example
4> for load
5> go
```

5. Preparing for a load

To prepare for a load, follow these steps:

a. Check that the status for the VitalQIP database has “trunc” log on checkpoint set to “true” by running:

```
1> sp_helpdb QIP
2> go
```

b. If the “trunc” logon is not set to “true”, set it to “true” by running:

```
1> sp_dboption QIP, 'trunc. log on chkpt.', true
2> go
1> use QIP
2> go
1> checkpoint
2> go
1> use master
2> go
```

6. Reload the VitalQIP database

Reload the database from the dump database by running the following from a command line:

```
1> load database QIP from qip_dump_dat //for recent database dump
2> go
```

If transactions are logged (this was determined in “1. Dump the current transaction log” on page 332), reload the transaction log. Reloading the VitalQIP transaction log must follow FIFO (FIRST-IN/FIRST-OUT) methodology. In other words, the first dumped file must be the first that you reapply, the second dumped file must be the second that you reapply, and so on.

```
1> load tran QIP from qip_dump_log // for all recent log dumps
2> go // after the recent database dump
```

7. Ensuring a successful recovery

To ensure a successful recovery, follow these steps:

a. Using a text editor, type the following in the lines in the *dbcc* file:

```
dbcc checkdb(QIP)
go
dbcc checkalloc(QIP)
go
```

b. Check if the data loaded correctly by running the following from a command line:

```
isql -U sa -P <password> -idbcc > dbcc.out
```

c. Review the *dbcc.out* file for errors. If errors are displayed, call technical support.

Important! If the master database is not restarted along with VitalQIP, run **create_access.sql**. If is not run, the **syslogins** link is broken in the master database, and VitalQIP data cannot be accessed.

8. Reset administrator access

To reset administrator access, run:

```
cd $QIPHOME script
isql -U sa -P <password> -i create_access.sql
```

□

Truncating audit data

The audit data in the VitalQIP database should be truncated periodically. This can be accomplished by executing **qip-clear** on a periodic basis or by setting up a job on your system. This can be performed by entering the following command:

```
qip-clear -u qipman -p <your_qipman_password> -d preferred_date
```

For example:

```
qip-clear -u qipman -p qipman -d 02/01/1996
```

The system truncates all auditing data older than 02/01/1996.

For more information about **qip-clear**, see the *VitalQIP Command Line Interface User's Guide*.

☐

Index Statistics Maintenance (Sybase only)

When you create an index after a table is loaded, a data distribution table is created for that index. The distribution page is not automatically maintained. The database owner must issue an "update statistics" command to ensure that the index statistics are current. Whenever you import large amounts of data, an "update statistics" command must be run. Failure to update statistics can severely hurt performance.

Run the **qip-util** command with the **SybaseUpdateStatistics** function as described in "Performing database administrative tasks with qip-util" on page 322.

Important! **qip-import** runs this script automatically.

☐

Part III: Troubleshooting

Overview

Purpose

In Part III, you will find help information for troubleshooting VitalQIP, DNS, DHCP and error messages. Refer to Part III when you encounter problems with VitalQIP, DNS, DHCP, and error messages.

Contents

Part III contains the following chapters:

10 Troubleshooting VitalQIP

Overview

Purpose

This chapter provides tips and possible resolutions to issues you may encounter with VitalQIP. Many of the common problems and questions related to VitalQIP operations are included in this chapter. Carefully review this chapter before calling technical support.

Contents

This chapter presents the following topics.

□

Before calling technical support

Before calling technical support, certain information and files must be collected in preparation for your call. The following items will assist in the troubleshooting process. Technical support will ask for the information and files when you call. Review Table 53 before calling technical support.

Table 53 Required troubleshooting information

Problem type	Required files or information
Installation or upgrade from previous VitalQIP version	Include the following items: • The *qsi-load.log* file and *qsi-load.template* files • VitalQIP data files with (.*qef* extension) from export if you encountered a problem during the import process • Exact steps to reproduce the problem
Importing or exporting VitalQIP data	Include the following items: • The *qip-import.log* or *qip-export.log* and *qip-import.check* files (if available) • If Oracle, the *sqlload.log* file (located in the export file directory) • VitalQIP data files from export (.*qef* extension) • Exact steps to reproduce the problem
DHCP	For all DHCP primary and failover servers involved include: • Platform, version, and build number of all DHCP servers • The *dhcpd.pcy*, *dhcpd.conf*, and *dhcpd.log* files • *Syslog* or Event Viewer message log • Core file (UNIX) or Dr. Watson log (Windows), if available • DHCP User Exit file, if used • Server configuration information (for example, one failover to four primary servers) • Number of NICs configured on system(s) • Related VitalQIP policy values • Exact steps to reproduce the problem

Problem type	**Required files or information**
DNS	For all DNS Primary (master) and secondary (slave) servers involved include: • Platform, version, and build number of all DNS servers • The *named.conf* or *named.boot* and *named.run* files • Zone files and dynamic DNS zone log files • *Syslog* or Event Viewer message log • Core file (UNIX) or Dr. Watson log (Windows), if available • DNS user exit file, if used • Server configuration information (for instance, one primary server to two secondary servers) • Time interval of automatic scheduled updates, if configured • Related VitalQIP policy values • Exact steps to reproduce the problem
Other VitalQIP services	Include the following items: • Platform, version, and build number • The *qip.pcy* or *service-specific.pcy* files • The *<service>.log* file • *Syslog* or Event Viewer message log • Core file (UNIX) or Dr. Watson log (Windows), if available • Related VitalQIP policy values • Exact steps to reproduce the problem
VitalQIP client (on Windows or UNIX)	Include the following items: • Platform, version, and build number • Operating system version including service packs or patches applied • The *ipmanage.log* file if available • Related VitalQIP policy values • Administrator type and privileges defined • Exact steps to reproduce the problem • VitalQIP data files (with *.qef* extensions) from the export may be required to reproduce the problem

Verifying Your Environment

Before beginning any troubleshooting, verify that your VitalQIP environment variables are set properly.

How do I check my environment variables?

To check your environment variable on UNIX, run:

```
env
```

To check your environment variable on Windows 2000, run:

```
set | more
```

How do I “source” or set my environment variables?

To set your environment variables for UNIX, run **shrc** (if using Bourne shell) or **cshrc** (if using C shell) file from the *$QIPHOME/etc* directory. For example:

```
cd $QIPHOME/etc
. ./shrc OR source cshrc
```

Important! If any values need to be changed, update **shrc** before executing it.

For Windows 2000, change environment variables via **Control Panel|System|Advanced|Environment Variables+**.

How do I check the version of VitalQIP I am on?

From the VitalQIP client (**ipmanage**), select **About VitalQIP** from the **Help** menu, or run **qip-patchid** (UNIX only) to determine the VitalQIP version.

VitalQIP also provides an option (**-v**) that shows you the version and build level of the file. To check the version of the file, run the file in question with a **-v** option from a command prompt. For example:

- Windows 2000:

```
c:\qip\qip-scheduleservice.exe -v
```

- UNIX:

```
# $QIPHOME/usr/bin/dhcpd -v
```

In addition, the **vercheck** utility shows the full version information for files provided with VitalQIP. To run **vercheck**, run the following script:

- Windows 2000:

```
c:\QIP>vercheck <qip_filename>
```

- UNIX:

```
# $QIPHOME/usr/bin/vercheck <qip_filename>
```

For example:

```
vercheck ipmanage.exe
```

Output:

```
FILE=ipmanage.exe
SIZE=3186688
OWNER=(0,0)
PERMS=-rwxrwxrwx
CDATE=Oct-30-2000 10:48
MDATE=Sep-09-2000 08:45
TYPE=Executable
VERSION=5.2.20
CHECKSUM=a3be94119391542945aaab3d6252528a
```

The **vercheck** utility can also list the versions for the contents of a directory if using the **-d** option (for example, `c:\QIP\vercheck -d lib`).

Alternatively, you can list just the file name and version information with the **-z** option (for example, `f:\QIP>vercheck -d lib -z`). See "Finding VitalQIP version numbers with vercheck", on page 310 for more information about this CLI command.

Where is my license key file located?

The license key is located in *%QIPHOME%\.Lic* (Windows 2000) or *$QIPHOME/.Lic* (UNIX).

☐

System Logs

Error conditions within VitalQIP (and other running applications) are recorded in the *syslog* file (UNIX) or the Event Viewer (Windows 2000) System/Application logs. For UNIX systems, the */etc/syslog.conf* file must be configured properly, and the **syslogd** daemon must be running for messages to be sent to the *syslog* file. See the system "man" pages for more information on *syslog*. The *syslog* or Event Viewer system/application files are useful for diagnosing system and VitalQIP problems. The *syslog* and system/application log files should be sent to technical support when reporting a problem.

Table 54 describes common messages in System Log or *<server>.log* file.

Table 54 Common Messages in System Log or <*server*>.log Files

Error Message	Message Source	Probable Cause of Problem
/etc/named.conf:10:syntax error near "movie.com"	DNS server	Indicates the name server encountered a syntax error on or around line 10 in the *named.conf* file.
Bad packet received on DHCP server port	DHCP server	The DHCP server received a packet from a client that it was unable to decode. A more detailed analysis of the information sent in the DHCP packet is required.
BOOTP Request Failed	DHCP server	A Bootp client on your network is attempting to get an address. Your DHCP server does not have this device's MAC address. The DHCP server configured as a manual Bootp object, or no automatic Bootp addresses are available. You can prevent these messages by configuring M-BOOTP or A-BOOTP addresses within VitalQIP, reconfiguring this device as a DHCP client, or removing it from the network.
Cannot open QDHCP configuration file	DHCP server	The server was not able to find the *dhcpd.conf* file. Perform a **Network Services|DHCP Generation** to the server to generate this file.
Can't change directory to /var/named: no such file or directory	DNS server	Indicates the working directory you set in the *named.conf* file does not exist.
Can't open 'F:\WINNT\System32\drivers\etc\named.conf'	DNS server	The server was not able to find the *named.conf* file. Use **Network Services|DNS Generation** to generate this file.
CNAME and other data invalid.	DNS server	Indicates you have a domain name in the zone that owns both a CNAME record and another record type.
DHCP Inform Failed	DHCP server	A client, for whom the DHCP server does not have a lease, requested additional configuration information via a DHCP Inform packet.

Error Message	Message Source	Probable Cause of Problem
DHCP Request Failed	DHCP server	A client tried to request an address, which the server did not have.
Duplicate MAC found in Manual Subnet	DHCP server	The MAC address specified was defined twice for a manual DHCP object in the same subnet. Duplicate MAC addresses are not allowed. Delete one of the duplicates from the VitalQIP configuration.
Err/To getting serial # for "domain.com"	DNS server	Indicates the name server encountered server syntax errors or illegal characters while trying to load the zone. The zone is rejected and is not loaded.
Load initial policy file failed	DHCP server	The server was not able to find the *dhcpd.pcy* file. Perform a **Network Services|DHCP Generation** to the server to generate this file.
Master Zone "domain.com" (IN) rejected due to errors	DNS server	Indicates the name server encountered server syntax errors or illegal characters while trying to load the zone. The zone was rejected and was not loaded.
No DHCP lease is available to offer from subnet	DHCP server	The subnet specified has no more leases to hand out.
Owner name "hosta.domain.com" IN (primary) is invalid-rejected	DNS server	Indicates this Resource Record contains invalid characters or is illegal. The record is not loaded.
Unable to locate failover server record	DHCP server	The failover DHCP server received a failover packet from a server for which it is not a failover server. Contact technical support for assistance in determining problem in your DHCP and the failover server configuration(s).
Unable to open socket	DHCP server	There is another process using port 67 (possibly another DHCP server already running), or there is a problem connecting to that port.
Unapproved AXFR (IXFR) frm [1.2.3.4].54 for domain.com	DNS server	Indicates the DNS server denied a zone transfer for domain.com to IP address 1.2.3.4 because of a local access list.
Unapproved update from [198.138.121.3].1848 for domain.com	DNS server	Indicates your name server refused a dynamic update to the zone "domain.com" from the host 198.138.121.3.

Enabling Debugging in VitalQIP

When to use

The VitalQIP services and the VitalQIP interface can have debugging enabled to assist in troubleshooting a problem. If you experience a problem from a function within VitalQIP and if it is easily reproducible, run **ipmanage** with debug enabled and try the operation again. A log file is generated, which can be sent to technical support to assist in resolving the issue.

How do I turn on debugging for the VitalQIP client (ipmanage)?

To turn on debugging for Windows 2000, run the following script:

```
ipmanage.exe -d
```

Important! The log file is located in *$QIPHOME/log/ipmanage.log*.

To turn on debugging for UNIX, run the following script:

```
ip-manage -d &
```

Important! The log file is located in *$ROOT/QIPManage.log*.

If you do not want the *QIPManage.log* in the root directory, run:

```
ip-manage -d  -f /<path_to_directory>/QIPManage.log &
```

If the problem requires debug information for one VitalQIP service, the debug setting must be enabled before restarting the service for debugging to take affect. Reproduce the problem and send the resulting logs to technical support for further assistance.

How do I turn on debugging for all the VitalQIP services?

Setting debug policy to “All” enables debugging at the maximum level. Debugging occurs for the VitalQIP services (excluding DHCP and DNS). The “FeatureBackup” setting provides a backup (*.bak_1.log*) debug file to prevent overwriting of log information if the service is restarted. For more information on other debug policies, see “The debug policy”, on page 97.

To turn on debugging, follow these steps:

1 Use a text editor to open the *QIPHOME/qip.pcy* file.

2 In the **[Global Section]** of the *qip.pcy* file (excluding DHCP and DNS Services), change the debug setting from **debug=nothing** to **debug=All FeatureBackup**.

Important! If the debug policy is set to “all” in the **[Global Section]** of the *qip.pcy* file, debug logging occurs for all CLI commands as well as VitalQIP services. Debug information for CLI commands is written to the *$QIPHOME/log/<CLI_name>.log*.

3 Stop and restart each service/daemon to enable debugging. The log files are located in *$QIPHOME/log* by default.

END OF STEPS

☐

VitalQIP interface -trouble logging in

When to use

If you encounter difficulty logging into the VitalQIP interface, review this section before calling technical support.

How do I change qipman's password when I don't know it?

To change qipman's password when qipman's current password is unknown, follow these steps:

1 Obtain the encrypted qipman password by executing:

- For UNIX:

```
$QIPHOME/usr/bin/qip-crypt <new_password>
```

- For Windows:

```
%QIPHOME%\CLI\qip-crypt <new_password>
```

2 Modify qipman's password by executing:

- For Sybase:

```
isql -Uqipadmin -P<qipadmin's_password> -S<database_server>
1> UPDATE admins SET pass_word = '<encrypted_password>' WHERE login_name='qipman'
2> go
```

- For Oracle:

```
sqlplus qipadmin/<qipadmin's_password>@<database_server>
SQL> UPDATE admins SET pass_word = '<encrypted_password>' WHERE login_name='qipman'
```

3 Modify qipman's password in *qip.pcy* file if necessary.

END OF STEPS

Login error message

Table 55 describes error messages you may encounter while logging into VitalQIP.

Table 55 Login error messages

Error message	Description	Resolution
[SERVERERROR] Error from Server: DBconnect(): olog () failed user: qipman, service: <ORACLE_SID>" followed by "Error 5: Failed to connect to database	This error indicates you cannot connect to the database with the current user name, password and server specified.	Verify that the ORACLE_HOME environment variable is set to the correct home directory for Oracle. For example, *c:\oracle\ora81*. The ORACLE_HOME environment variable must be set manually following the Oracle client installation.
[SERVERERROR] Error from Server: Login failed	This error indicates you cannot connect to the database with the current user name, password and server specified.	Take the following actions: • Verify that the username and password specified are correct (remember these are case-sensitive). • Verify that you can connect to the database using other methods, such as using DSEDIT to ping the server or logging into the database with the current username and password using **isql** (Sybase) or **sqlplus** (Oracle).
SERVERERROR] Error from Server:DB-Library Error: Net-Lib protocol driver call to connect two endpoints failed. DBPROCESS is dead. Operating System Error: Failed to connect to the server	This error indicates you cannot connect to the database or database is not up and running.	Verify that the database is running.

VitalQIP server failures

If a remote server or the VitalQIP server fails, it is important to resynchronize the VitalQIP server with the DHCP servers and/or DNS servers after the failed server is restored. Synchronization reads the database on each server and updates it, if necessary.

Synchronization is particularly important because Lucent’s Dynamic DNS (DDNS) provides real-time updating of DHCP information to DNS servers.

Use VitalQIP’s **Network Services** function to synchronize DNS with the VitalQIP server. Refer to “Generate DNS configuration and data files” in Chapter 5 of the *VitalQIP User’s Guide.*

☐

Automatic synchronization

The **Scheduled Automatic Updates** field within the Server Profile defines the interval at which VitalQIP compiles the DNS data for a domain and updates the primary DNS servers. This allows updates to be scheduled every 1 to 24 hours. Refer to “Servers” in Chapter 2 of the *VitalQIP User’s Guide*.

□

Importing and exporting data

If you encounter difficulties importing or exporting VitalQIP data, review this section before calling technical support.

VitalQIP import is failing, what can I do?

Any errors that occur during the import can be found in the *qip-import.log* located in *QIPHOME/log*.

For Oracle imports, an additional file, *sqlload.log*, (located in the import directory with the *.qef* files) is necessary to determine the cause of the import failure. Contact technical support with the logs and data files being imported to assist in determining the cause of your import failure.

□

VitalQIP services/daemons

Before reading this section, verify that your environment variables are set correctly. See "Verifying Your Environment" on page 346 for more information.

Troubleshooting services/daemons on UNIX

For UNIX platforms only, there are additional ways to assist in troubleshooting problems. Signal handling, *syslog*, and statistics are useful tools to assist in troubleshooting problems.

Signal handling

Signals, which are not specified, have the default behavior for the applicable operating system. A signal can be sent to a process by running:

```
kill <signal_type> <process_ID>
```

Schedule Service signal types

Table 56 describes signal types for the VitalQIP Schedule Service.

Table 56 Schedule Service signal types

Signal type	Action
SIGTERM	Used to stop the daemon.
SIGUSR1	Dumps statistical information to *syslog*.
SIGBUS	***Critical Program Error*** logged to *syslog*, followed by default action.
SIGSEGV	***Critical Program Error*** logged to *syslog*, followed by default action.
SIGINT	If debugging was enabled on the command line, this signal terminates the application (**Ctrl-C**). Otherwise, this signal is ignored.
SIGQUIT	Ignored.
SIGHUP	Ignored.

Login Service signal types

Table 57 describes signal types for the Login Service.

Table 57 Login Service signal types

Signal type	Action
SIGTERM	Used to stop the daemon.
SIGUSR1	Dumps statistical information to *syslog*.
SIGBUS	***Critical Program Error*** logged to *syslog*, followed by default action.
SIGSEGV	***Critical Program Error*** logged to *syslog*, followed by default action.
SIGINT	If debugging was enabled on the command line, this signal terminates the application (**Ctrl-C**). Otherwise, this signal is ignored.
SIGQUIT	Ignored.
SIGHUP	Reads the policy file again.

Message Service signal types

Table 58 describes the signal types for the Message Service.

Table 58 Message Service signal types

Signal type	Action
SIGTERM	Used to stop the daemon.
SIGUSR1	Dumps statistical information to *syslog*.
SIGBUS	***Critical Program Error*** logged to *syslog*, followed by default action.
SIGSEGV	***Critical Program Error*** logged to *syslog*, followed by default action.
SIGINT	If debugging was enabled on the command line, this signal terminates the application (**Ctrl-C**). Otherwise, this signal is ignored.
SIGQUIT	Ignored.
SIGHUP	Ignored.
SIGPIPE	Handled.

QIP Update Service signal types

Table 59 describes the signal types for the QIP Update Service.

Table 59 QIP Update Service signal types

Signal type	Action
SIGTERM	Used to stop the daemon.

Signal type	Action
SIGUSR1	Dumps statistical information to *syslog*.
SIGBUS	***Critical Program Error*** logged to *syslog*, followed by default action.
SIGSEGV	***Critical Program Error*** logged to *syslog*, followed by default action.
SIGINT	If debugging was enabled on the command line, this signal terminates the application (**Ctrl-C**). Otherwise, this signal is ignored.
SIGQUIT	Ignored.
SIGHUP	Re-initiates DNS.
SIGPIPE	Handled.

DNS Update Service signal types

Table 60 describes the signal types for the DNS Update Service.

Table 60 DNS Update Service signal types

Signal Type	Action
SIGTERM	Used to stop the daemon.
SIGUSR1	Dumps statistical information to *syslog*.
SIGBUS	***Critical Program Error*** logged to *syslog*, followed by default action.
SIGSEGV	***Critical Program Error*** logged to *syslog*, followed by default action.
SIGINT	If debugging was enabled on the command line, this signal terminates the application (**Ctrl-C**). Otherwise, this signal is ignored.
SIGQUIT	Ignored.
SIGHUP	Re-initiates DNS.
SIGPIPE	Handled.

Active Lease Service signal types

Table 61 describes the signal types for the Active Lease Service.

Table 61 Active Lease Service signal types

Signal type	Action
SIGTERM	Used to stop the daemon.

Signal type	Action
SIGUSR1	Dumps statistical information to *syslog*.
SIGBUS	***Critical Program Error*** logged to *syslog*, followed by default action.
SIGSEGV	***Critical Program Error*** logged to *syslog*, followed by default action.
SIGINT	If debugging was enabled on the command line, this signal terminates the application (**Ctrl-C**). Otherwise, this signal is ignored.
SIGQUIT	Ignored.
SIGHUP	Ignored.
SIGPIPE	Handled.

Remote Service signal types

Table 62 describes the signal types for the Remote Service.

Table 62 Remote Service signal types

Signal type	Action
SIGTERM	Used to stop the daemon.
SIGBUS	***Critical Program Error*** logged to *syslog*, followed by default action.
SIGSEGV	***Critical Program Error*** logged to *syslog*, followed by default action.
SIGINT	If debugging was enabled on the command line, this signal terminates the application (**Ctrl-C**). Otherwise, this signal is ignored.
SIGQUIT	Ignored.
SIGHUP	Ignored.
SIGPIPE	Handled.

RMI Scheduler Service signal types

Table 63 describes the signal types for the RMI Scheduler Service.

Table 63 Remote Service signal types

Signal type	Action
SIGTERM	Used to stop the daemon.
SIGINT	If debugging was enabled on the command line, this signal terminates the application **Ctrl-C**. Otherwise, this signal is ignored.

RMI QAPI Service signal types

There are no signal types for the RMI QAPI Service.

MS DNS Update Service signal types

There are no signal type for the RMI QAPI Service.

Syslog file

Informational, warning, and error messages are logged by all services to the */etc/syslog.conf* file. To review messages, the */etc/syslog.conf* file must be properly configured for user.info messages. For example, in the *syslog.conf* file, you would have the following entry for user.info:

```
user.info/var/log/mylog
```

The user.warning and user.error messages are logged in */var/log/mylog* because they are more severe messages than info-type messages.

The */var/log/mylog* must exist before the daemon (**syslogd**) writes to the file. After you have edited the file, you can run **kill -1** on the process ID of **syslogd**. The **syslogd** daemon is restarted.

If you configured *syslog.conf* correctly, your *syslog* file contents look similar to the following:

```
Jul 17 05:56:25  enterprise1  qipd[20819]:  Started
Jul 17 05:56:25  enterprise1  qipd[20819]:  Using Default Policies
Jul 17 05:56:27  enterprise1  qipd[20819]:  License Key Update Interval=1800 seconds
Jul 17 05:56:27  enterprise1  qipd[20819]:  Initialization complete
```

The number following the daemon (in this case **qipd**) is the process ID. This daemon can be killed by issuing the command **kill 20819**.

The following tables provide typical messages written to the *syslog.conf* file. Other messages may be logged.

Schedule Service Message Types

Table 64 shows a subset of messages generated by the Schedule Service.

Table 64 Schedule Service message types

Message	Severity	Description
Service already running as <*pid number*>	ERROR	According to the *$QIPHOME/etc/qipd.pid* file, the process is already running.
Critical Program Error	ERROR	Program received SIGSEGV or SIGBUS. A stack trace is generated on Solaris.
Using Default Policies	INFO	There is no Policy file.
Started	INFO	The service is running and can log messages.

Message	Severity	Description
License Key Update Interval = <*number of seconds*>	INFO	The service updates the License Key every <*number of seconds*>.
Initialization complete	INFO	The database initialization is complete and the service is running.
Stopped	INFO	A stop of the service is imminent.
Miscellaneous Stats	INFO	SIGUSR1 was sent. See "Schedule Service statistics" on page page 371 for a list of Statistics.

Login Service message types

Table 65 shows a subset of messages generated by the Login Service message types.

Table 65 Login Service message types

Message	Severity	Description
Service is already running as PID #	ERROR	According to the file *$QIPHOME/etc/qip-logind.pid*, the process is already running.
Critical Program Error	ERROR	Program received a SIGSEGV or a SIGBUS. A stack trace is only generated on Solaris.
Could not determine <*service*> port number	ERROR	There was no entry for the service in */etc/services.*
Set Socket Option Failure	ERROR	There was an internal socket error.
Accepting connections	INFO	The service has initialized and is accepting connections.
Stopped	INFO	A stop of the service is imminent.
Miscellaneous Stats	INFO	A SIGUSR1 was sent. See "Login Service statistics" on page 372 for a list of Statistics.

Message Service message types

Table 66 describes the messages types for the Message Service.

Table 66 Message Service messages types

Message	Severity	Description
Service already running as <*pid_number*>	ERROR	According to the file *$QIPHOME/etc/qip-msgd.pid*, the process is already running.

Message	Severity	Description
Critical Program Error	ERROR	Program received SIGSEGV or SIGBUS. A stack trace is generated on Solaris.
Could not determine <*service*> port number	ERROR	There was no entry for the service found in */etc/services*.
Could not resolve server <*server_name*>	ERROR	DNS had no entry for <*server_name*>.
Socket Option Set Failure	ERROR	There was an internal socket error.
Corrupt message received from *source*	ERROR	*source* sent a message which did not conform to the VitalQIP protocol.
Communications problems with *peer*	ERROR	The connections to *peer* have been abruptly terminated.
Could not queue, out of memory	ERROR	The Message Service could not allocate enough memory to queue the message. Try configuring MessageQueue policy for appropriate message types or increasing available memory for this process.
Could not increase connection count	WARNING	MaxConnections policy was set to a value higher than allowed by the operating system.
Accepting connections	INFO	Initialization is complete, waiting for connections.
<*number_of_messages*> messages are being discarded	INFO	The service was shut down while messages were still in the queue, and the messages were lost.
Queue Length has reached <*number_of_elements*> elements.	INFO	The queue length is longer than expected. This error is usually due to a database or Update Service failure. Messages start appearing when the value in the QueueLengthWarningLimit field is reached.
Expected message *x* but received message *y.*	INFO	*y-x* messages were lost. Messages may be lost if the Message Service is too busy process incoming messages.
Using Default Policies	INFO	There is no Policy file.
Started	INFO	The service is running and can log messages.
Stopped	INFO	A stop of the service is imminent.
MaxConnections set to #	INFO	Indicates number of permitted connections for this server has changed.
<*Miscellaneous Statistics*>	INFO	SIGUSR1 was sent. See "Message Service statistics" on page 372 for a list of Statistics.

MS DNS Update Service message types

Table 67 describes the messages for the MS DNS Update Service.

Table 67 MS DNS Update Service message types

Message	Severity	Description
Exiting service	INFO	The service has exited .
Unable to locate qip-msgd in services file.	ERROR	The **qip-msgd** service name should be added to the */etc/services* file. The MS DNS Update Service is unable to process requests.
Unable to register with Message Service (Failed to connect).	ERROR	The service could not connect to the Message Service. Ensure the Message Service is running on the local host on the **qip-msgd** port. The MS DNS Update Service is unable to process requests.
Create socket failed	ERROR	The service could not create a socket to communicate with the Message Service. The MS DNS Update Service is unable to process requests.
Set socket option failed!	ERROR	The service was unable to make the communication socket reusable. The MS DNS Update Service is unable to process requests.
Bind to socket failed	ERROR	The service was unable to bind to the ListenPort. The MS DNS Update Service is unable to process requests.
Listen on socket failed	ERROR	The service was unable to listen to the ListenPort. The MS DNS Update Service is unable to process requests.
Failed to get socket name	ERROR	The service was unable to determine the bound local address of the communication socket. The MS DNS Update Service is unable to process requests.
Accepting Connections	INFO	The MS DNS Update Service has started processing requests.
Communication problems with socket while handling request.	INFO	The MS DNS Update Service lost connectivity with the Message Service. The MS DNS Update Service is unable to process requests until it reconnects with the Message Service.
Can not accept connection from IP Address <*address*>	WARNING	A connection was denied because of the access control policies.
Accepted client connection from <*address*>	INFO	A connection was allowed because of the access control policies.

Message	Severity	Description
Lost connection to message service	ERROR	The MS DNS Update Service lost connectivity with the Message Service. The MS DNS Update Service is unable to process requests until it reconnects with the Message Service.

QIP Update Service message types

Table 68 describes the messages for the QIP Update Service.

Table 68 QIP Update Service messages types

Message	Severity	Description
Service already running as pid <*pid_number*>	ERROR	According to the *$QIPHOME/etc/qip-qipupdated.pid* file, the process is running.
Critical Program Error	ERROR	Program received SIGSEGV or SIGBUS. A stack trace is generated on Solaris.
Could not determine <*service*> port number	ERROR	There was no entry for the service found in */etc/services*.
Version of service is obsolete, update.	ERROR	According to the database, there is a newer version of the QIP Update Service available.
Could not obtain database credentials	ERROR	The QIP database password and username need to be specified on the command line or in the *qip.pcy* file.
Accept Socket Failed	ERROR	The process is unable to accept any more connections, the file limit may is reached.
Communication problems with <*hostname*>	ERROR	There are problems with the Message Service on <*hostname*>. The remote server may have gone down or there may be other network issues.
Message <*message_number*> from <*hostname*> is out of sequence.	ERROR	This typically indicates problems with the TCP/IP stack on the server running the QIP Update Service.
DNS Initialize failed (DNS Update Disabled).	ERROR	DNS could not be initialized. Enable debugging to determine the problem.
Could not determine current resource limits	WARNING	MaxConnections policy was specified but the QIP Update Service was unable to determine the current connection limit.
Could not increase connection count	WARNING	The system is preventing service from setting connection count to MaxConnections policy.
Could not connect to Database: <*errnum*>	WARNING	An error occurred (errnum) while trying to connect to the database.

Message	**Severity**	**Description**
Unknown Message id <*msgid*>	WARNING	The QIP Update Service received a message it did not know how to process.
Could not create ddns.conf (Using old file).	WARNING	The process may not have sufficient permission to create this file. Enable debugging.
Established Connection to <*server_name*>	INFO	The Message Service running on <*server_name*> has connected.
Closing Connection to <*server_name*>	INFO	The Message Service running on <*server_name*> was stopped, or a network problem was encountered.
Database Update Failed: status = <*lease information*>	INFO	Could not update lease information in the VitalQIP database.
Lost Connection to Database	INFO	The database and its network path to it are in an inconsistent state.
Re-Established Connection to Database	INFO	The database is backed up.
DNS Re-Initialize Complete	INFO	DNS has successfully re-initialized.
Configuration Error: DNS Previously Updated by DHCP Message Service	INFO	Both the Message Service and the QIP Update Service are configured to update DNS.
Network Config Error, Duplicate Name: <*hostname*> on <*IP_address*>	INFO	The QIP Update Service is configured to update DNS and has found a duplicate name. The duplicate would replace a static object (server) if allowed to continue.
Using Default Policies	INFO	There is no policy file.
Started	INFO	The service is running and can log messages.
Accepting connections	INFO	Initialization is completed, waiting for connections from the Message Service.
Ignoring connection attempts until Database Re-Init	INFO	Since the database is down, no connections from the Message Service is allowed. Once the database is back up, connections proceed.
Stopped	INFO	A shutdown of the service is imminent.
Miscellaneous Stats	INFO	SIGUSR1 was sent. See "QIP Update Service statistics" on page 372 for a list of Statistics .

DNS Update Service message types

Table 69 describes the messages for the DNS Update Service.

Table 69 DNS Update Service message types

Message	Severity	Description
Service already running as pid <*pid_number*>	ERROR	According to the *$QIPHOME/etc/qip-dnsupdated.pid* file, the process is already running.
Critical Program Error	ERROR	Program received SIGSEGV or SIGBUS. A stack trace is generated on Solaris.
Could not determine <*service*> port number	ERROR	There was no entry for the service found in */etc/services*.
Accept Socket Failed	ERROR	The process is unable to accept any more connections, the file limit has been reached.
Communication problems with <*hostname*>	ERROR	There are problems with the Message Service on <*hostname*>. The remote server may have gone down or there may be other network issues.
Message <*message_number*> from <*hostname*> is out of sequence.	ERROR	This typically indicates problems with the TCP/IP stack on the server running the DNS Update Service.
DNS Initialize failed (DNS Update Disabled).	ERROR	DNS could not be initialized. Enable debugging to determine the problem.
Could not determine current resource limits	WARNING	MaxConnections policy was specified but the DNS Update Service was unable to determine the current connection limit.
Could not increase connection count	WARNING	The system is preventing service from setting connection count to MaxConnections policy.
Unknown Message id <*msgid*>	WARNING	The DNS Update Service received a message it did not know how to process.
Could not create ddns.conf (Using old file).	WARNING	The process may not have sufficient permission to create this file. Enable debugging.
Established Connection to <*server_name*>	INFO	The Message Service running on <*server_name*> has connected.
Closing Connection to <*server_name*>	INFO	The Message Service running on <*server_name*> was stopped or a network problem was encountered.
DNS Re-Initialize Complete	INFO	DNS has successfully re-initialized.

Message	Severity	Description
Using Default Policies	INFO	There is no policy file.
Started	INFO	The service is running and can log messages.
Accepting connections	INFO	Initialization is completed, waiting for connections from the Message Service.
Stopped	INFO	A shutdown of the service is imminent.
Miscellaneous Stats	INFO	SIGUSR1 was sent. See “DNS Update Service statistics“ on page 372 for a list of Statistics.

Active Lease Service message types

Table 70 describes the messages for the Active Lease Service.

Table 70 Active Lease Service message types

Message	Severity	Description
Exiting service	INFO	The Service has exited
Unable to locate qip-msgd in services file.	ERROR	The **qip-msgd** service name should be added to the */etc/services* file. The Active Lease Service is unable to process requests.
Unable to register with Message Service (Failed to connect).	ERROR	The service could not connect to the Message Service. Make sure the Message Service is running on the local host on the **qip-msgd** port. The Active Lease Service is unable to process requests.
Create socket failed	ERROR	The service could not create a socket to communicate with the Message Service. The Active Lease Service is unable to process requests.
Set socket option failed!	ERROR	The service was unable to make the communication socket reusable. The Active Lease Service is unable to process requests.
Bind to socket failed	ERROR	The service was unable to bind to the ListenPort. The Active Lease Service is unable to process requests.
Listen on socket failed	ERROR	The service was unable to listen to the ListenPort. The Active Lease Service is unable to process requests.
Failed to get socket name	ERROR	The service was unable to determine the bound local address of the communication socket. The Active Lease Service is unable to process requests.
Send to socket failed!	ERROR	The service was unable to send a message to a remote client. The remote client may no longer be running.

Message	Severity	Description
No Reply. Stop Sending	ERROR	The service was unable to receive a message from a remote client. The remote client may no longer be running.
The Service is unable to open the file requested by the client. The file may not exist or there may be a permission problem.	INFO, File *filename* open error.	The file could not be opened by the client.
Environment QDHCPCONFIG was not set	INFO	The service cannot locate the Lucent DHCP server installation.
GetHostByName for *hostname* failed	ERROR	The service was not able to determine an IP address for the local host.
Unsupported version of AIX: *NN.NN* Please Update	ERROR	The service is not supported on this operating system release.
DeleteLease on Microsoft DHCP is not available!'	WARNING	A client sent a request for an unsupported operation to the service.
DeleteLease on AIX is not available yet!	WARNING	A client sent a request for an unsupported operation to the service.
Recv from socket failed	ERROR	The service expected, but did not receive a message from a remote client.
Unknown message received	WARNING	A client sent a request for an unsupported operation to the service.
Unknown LDAP server message received	WARNING	A client sent a request for an unsupported operation to the service.
Delete Lease failed	ERROR	The service was unable to delete a lease.
Accepting Connections	INFO	The Active Lease Service has started processing requests.
Can not accept connection from IP Address *address*	WARNING	A connection was denied because of the access control policies.
Accepted client connection from *address*	INFOR	A connection was allowed because of the access control policies.
Lost connection to message service	ERROR	The Active Lease Service lost connectivity with the Message Service. The Active Lease Service is unable to process requests until it reconnects with the Message Service.

Message	Severity	Description
Exiting service	INFOR	The service is exiting.

Remote Service message types

Table 71 describes the message types for the Remote Service.

Table 71 Remote Service messages types

Message	Severity	Description
Service already running as <*pid_number*>	ERROR	According to the *$QIPHOME/etc/qip-rmtd.pid* file, the process is already running.
Critical Program Error	ERROR	Program received SIGSEGV or SIGBUS. A stack trace is generated on Solaris.
Could not determine <*service*> port number	ERROR	There was no entry for the service found in */etc/services*.
Could not resolve server <*server_name*>	ERROR	DNS had no entry for <*server_name*>.
Socket Option Set Failure	ERROR	There was an internal socket error.
Accepting connections	INFO	Initialization is complete, waiting for connections.
Using Default Policies	INFO	There is no policy file.
Started	INFO	The service is running and can log messages.
Stopped	INFO	A shutdown of the service is imminent.
<*Miscellaneous Statistics*>	INFO	SIGUSR1 was sent. No list of Statistics is maintained.

RMI Scheduler Service message types

Table 72 describes the message types for the RMI Scheduler Service.

Table 72 RMI Scheduler Service messages types

Message	Severity	Description
Service already running as <*pid_number*>	ERROR	According to the *$QIPHOME/etc/qip-rmisched.pid* file, the process is already running.
Critical Program Error	ERROR	Program received SIGSEGV or SIGBUS. A stack trace is generated on Solaris.
Could not determine <*service*> port number	ERROR	There was no entry for the service found in */etc/services*.

Message	**Severity**	**Description**
Could not resolve server <*server_name*>	ERROR	DNS had no entry for <*server_name*>.
Socket Option Set Failure	ERROR	There was an internal socket error.
Accepting connections	INFO	Initialization is complete, waiting for connections.
Using Default Policies	INFO	There is no policy file.
Started	INFO	The service is running and can log messages.
Stopped	INFO	A shutdown of the service is imminent.
<*Miscellaneous Statistics*>	INFO	SIGUSR1 was sent. No list of Statistics is maintained.

RMI QAPI Service message types

This service produces no message types.

Important! To enable logging for the QAPI Service, turn on **DebugLevel=All** in the Global Policies section of the *qip.pcy* file. Any messages are written to the *QIPHOME/log/<QAPI_number>.log* file.

Statistics

Statistics can provide more information about a signal. Statistics are sent to the *syslog* file. For example, you can run **kill -usr1 20819** to obtain statistics. All statistics are dumped to the *syslog* file when the process received is **SIGUSR1**.

Schedule Service statistics

The following statistics are generated by the Schedule Service:

- Subnets Reclaimed
- DHCP Lease Updates
- DHCP Server Updates
- Host Tables Generated
- Bootp Tables Generated
- DNS Primary Updates
- Secondary Server Updates
- Total Cycles Completed
- Minimum Cycle Time
- Maximum Cycle Time
- Average Cycle Time

Login Service statistics

The following statistics are generated by the Login Service:

- Connected clients
- Total requests received

Message Service statistics

The following statistics are generated by the Message Service:

- Messages Received
- Messages Lost
- Memory Queue for message type <*number*>:<*number*>
- Disk Queue for message type <*number*>:<*number*>

QIP Update Service statistics

The following statistics are generated by the QIP Update Service:

- Subnets Reclaimed
- DHCP Lease Updates
- DHCP Server Updates
- Host Tables Generated
- Bootp Tables Generated
- DNS Primary Updates
- Domain Controller Updates
- Tombstone Purges
- Total Cycle Time
- Minimum Cycle Time
- Maximum Cycle Time
- Average Cycle Time
- Standard Deviation

DNS Update Service statistics

The following statistics are generated by the DNS Update Service:

- Number of connected Message Services
- Total number of messages received
- Number of connected Message Services
- Total number of messages received

Services that do not produce statistics

The following services do not product statistics:

- Active Lease Service
- RMI Scheduler Service
- Remote Service RMI QAPI Service

☐

Troubleshooting starting and stopping services/daemons

When to use

If you encounter problems starting and stopping services/daemons, review this section.

How do I start the VitalQIP services?

To start VitalQIP services, take one of the following actions:

- To start Schedule Service (**qipd**), QIP Update Service (**qip-qipupdated**), DNS Update Service (**qip-dnsupdated**) for a VitalQIP enterprise server on UNIX, run:

```
$QIPHOME/etc/qip-es-startup
```

- To start Remote Service (**qip-rmtd**), Message Service (**qip-msgd**), Active Lease Service (**qip-netd**), Lucent DNS Service (**named**), and Lucent DHCP Service (**dhcpd**) as well as the DNS Update Service (**qip-dnsupdated**) if it is installed for a VitalQIP remote server on UNIX, run:

```
$QIPHOME/etc/qip-rs-startup
```

- For Windows 2000, open the VitalQIP Service Controller via **Start|Programs|Lucent VitalQIP|VitalQIP Service Controller**, select the service, and click **Start**.

How do I stop the VitalQIP services/daemons on UNIX?

To stop VitalQIP services, follow these steps:

1 List all currently running VitalQIP processes by running:

```
ps -ef | grep qip.
```

2 Run:

```
kill <process_ID_of_VitalQIP_service>
```

Important! Do not use the **kill** command with Sybase or Oracle processes, as it can cause database corruption.

END OF STEPS

How does the kill command terminate a process?

When **kill** <*process_ID*> is run, the signal 15 (SIGTERM) is sent to the process. SIGTERM allows the process to properly shut down; sockets are closed, buffers are flushed, and a message is logged that the process is about to terminate. When the first SIGTERM (default signal for **kill**) is received, the process sets a flag that tells itself to terminate as soon as possible. The process invokes its shutdown function as soon as it can. When the second SIGTERM is received, the process invokes its shutdown function immediately. Subsequent SIGTERMs are caught and ignored.

When a program specifies a signal handler (a function to be called when the specified signal is received by the process), the kernel stops the process from what it is doing and calls the specified function. Once the function is completed, the kernel has the process continue with what it was previously doing when the signal is received. The set of functions that are safe to call from within a signal handler are limited. Rather than terminating the process immediately, the signal handler sets a flag for the process to terminate when it returns to the main loop. If the process is too busy to return to the main loop, the signal handler for the VitalQIP services calls the shutdown function directly when the second SIGTERM is received. Subsequent SIGTERM signals are ignored and can interfere with a shutting down the process properly.

Some applications can implement their own signal handlers and can ignore signals. A fail-safe way is need to terminate a process. To this end, a process cannot catch a signal 9 (SIGKILL). When **kill - 9** is run, the kernel unloads the process; thus, files are closed without flushing buffers. This command removes the process from memory abruptly and without the process' knowledge. As a result, data can be corrupted or lost.

In some rare occasions, **kill -9** <*process_ID*> does not terminate a process. There are rare occasions with some operating systems when a process makes a call into kernel space, and the kernel is unable to fulfill the request. For example, a process attempts to write to a NFS mounted partition, and the NFS server is down. **kill -9** <*process_ID*> does not stop the process.

Why can't I start Schedule Service (qipd)?

You may not be able to start the Schedule Service (**qipd**) for one of the following reasons:

- Database is not running
- Login Service is not running
- License file does not contain a valid license
- Maximum objects, subnets exceeded

To rectify the situation, take the following actions:

- Verify the user name and password that the service is using to connect to the database. User name and password is located in the *qip.pcy* file.
- Verify that the *.lic* file is not corrupted. Contact technical support for assistance in verifying the correct license.
- Reduce the number of objects, subnets or obtain a new license key allowing greater values.

Why can't I start the DHCP Service (dhcpd)?

Refer to “DHCP error messages”, on page 397.

Why can’t I start the DNS Service (named)?

Refer to “DNS Error Messages”, on page 388.

Why can't I start the Remote Service (qip-rmtd)?

It is recommended you enable debugging to determine the cause of the problem. To enable debugging, follow these steps:

1 Update the *qip.pcy* file by setting **Debug=AllFeature Backup** in **[VitalQIP Remote Service]** section.

2 Restart Remote Service (**qip-rmtd**). A *qip-rmtd.log* file is generated, which can be sent to technical support to assist in resolving the issue.

END OF STEPS

Why can't I start the Active Lease Service (qip-netd)?

It is recommended you enable debugging to determine the cause of the problem. To enable debugging, follow these steps:

1 Update *qip.pcy* file by setting **Debug=AllFeatureBackup** in the **[VitalQIP Active Lease Service]** section.

2 Restart Active Lease Service (**qip-netd**). A *qip-netd.log* file is generated, which can be sent to technical support to assist in resolving the issue.

Why can't I start the Message Service (qip-msgd)?

It is recommended you enable debugging to determine the cause of the problem. To enable debugging, follow these steps:

3 Update *qip.pcy* file by setting **Debug=AllFeatureBackup** in the **[VitalQIP Message Service]** section.

4 Restart the Message Service (**qip-msgd**). A *qip-msgd.log* file is generated that can be sent to technical support to assist in resolving the issue.

END OF STEPS

Why can't I start the QIP Update Service (qip-qipupdated)?

It is recommended you enable debugging to determine the cause of the problem. To enable debugging, follow these steps:

1 Update *qip.pcy* file by setting **Debug=AllFeatureBackup** in the **[VitalQIP QIP Update Service]** section.

2 Restart QIP Update Service (**qip-qipupdated**). A *qip-qipupdated.log* file is generated, which can be sent to technical support to assist in resolving the issue.

END OF STEPS

□

11 Troubleshooting DNS

Overview

Purpose

This chapter provides tips and possible resolutions to issues you may encounter with DNS. Many of the common problems and questions related to VitalQIP operations are included in this chapter. Carefully review this chapter before calling technical support.

Before reading this chapter, ensure you have verified that your environment is properly set up and you have the proper files before calling Technical Support. Read “Before calling technical support”, on page 344 and “Verifying Your Environment”, on page 346 for more information.

Contents

This chapter presents the following topics.

Common DNS Problems and Questions

This section contains common problems and questions you may run into. The possible causes and resolutions are listed after each question.

Why can't I start the DNS Service?

To determine why you cannot start the DNS service, follow these steps:

- Verify that the *named.conf* (BIND 8.x) file is in place. For UNIX, the */etc/named.boot* or */etc/named.conf* file has a symbolic link to the **Default Directory** specified in the **DNS Server Profile** in VitalQIP. For Windows 2000, the *named.conf* resides in *\WINNT\System32\drivers\etc*, and for BIND 4. x resides in *\WINNT*. For example, run the following to establish a link in UNIX:

```
ln -s /etc/named.conf /<QIPHOME>/<DNS_directory>.named.conf
```

- Verify that the correct version of *named.exe* file is installed (BIND 8) in VitalQIP. Use named **-v** command to determine the version by typing `named -v`.
- Enable debugging to determine the problem's cause. See the next question, "How do I enable debugging for DNS (named)?".

  Important! In Windows, certain registry keys must be populated before **named** is started. To do so, install **named** by running:

```
named -install
```

How do I enable debugging for DNS (named)?

To enable DNS (**named**) debugging for Windows 2000, use the following methods.

For Windows 2000, follow these steps:

1 Open the VitalQIP Service Controller by selecting **Start|Programs|Lucent VitalQIP|VitalQIP Service Controller**.

2 In the Service Controller window, click the **Services** tab.

3 In the Services tab, select **Lucent DNS Service**.

4 Click **Options**. The DNS Controller window opens.

5 Click **Debugging On/Increase Level**. (To increase debug levels, continue clicking on **Debugging on/Increase Level**. For example, for debug level 1, click once. For debug level 10, click ten times).

6 Click **Exit**. The Services Controller window opens.

7 Click **Exit**.

END OF STEPS

For UNIX, run:

```
named -d <debug_level>
```

For example, `named -d 10`. Level 11 is the highest debug level.

Important! The *named.run* file is located in the configured DNS directory.

- Run `ndc` to turn DNS debugging on or off (for example, **ndc trace**). (You can increase the debug level by running **ndc trace** again.) Run **ndc notrace** to turn off tracing.
- Use signaling within UNIX to increase the trace levels for DNS (for example, `kill -USR1 <process_ID>`). You can increase the debug level by running `kill -USR1 <process_ID>` again.) Run **kill -USR2** *<process_ID>* once to turn off tracing. See "Signal handling", on page 357 for more information about signal handling.

For additional information on troubleshooting DNS, see *DNS and BIND, Fourth Edition* by Paul Albitz and Cricket Liu. Be aware that setting the debug level to 10 generates a very large log file over time. Be sure you have sufficient disk space to accommodate the file.

How do I enable roundrobin in DNS?

To enable roundrobin for Lucent DNS 4.9.x, start **named** process with the **-o** option (for example, `named -o`), or type the following directive into the *named.boot* file of the Server Definition Area in the **Corporate Extensions** field:

```
roundrobin
```

To enable roundrobin for Lucent DNS, follow these steps:

1 Open the *named.conf* file with a text editor.

2 In the *named.conf* file, type the following directive in the options block of the Server Definition Area in the **Suffix Corporate Extension** field:

```
options
{
roundrobin yes;
};
```

Why am I unable to resolve a short name?

To resolve a short name, try one of the following methods:

- For UNIX, make sure your */etc/resolv.conf* is configured properly.
- For Windows 2000, make sure the *\Winnt\resolv.ini* file or the TCP/IP stack is configured properly.

- For Windows 2000, verify the correct DNS server is listed as the primary server via the **Network Properties**.
- If you are running in a Windows 2000 environment, ensure that your domain name, name server, and search order are properly specified within the **DNS** tab under TCP/IP configuration. (Access the **DNS** tab via **Start|Settings|Control Panel|Network|Configuration|TCP|IP|Properties**.)

The *resolv.conf* or *resolv.ini* file looks as follows:

```
Domain mydomain.com
Search mydomain.com
Nameserver <IP_address_of_DNS_Server>
```

The domain and search directive cause the domain to be appended to the host lookup.

Why do I need a forwarder, and how do I configure one in DNS?

Forwarders define one or more DNS servers to pass requests to if they do not have the answer. Forwarders provide another way of altering the path by which non-authoritative names are resolved. For example, you may need a forwarder if you want only one name server in your network to have Internet access. All other name servers can forward Internet requests to that single name server. For more information on forwarding, see *DNS and BIND, Fourth Edition* by Paul Albitz and Cricket Liu.

To configure a forwarder for BIND 8.x, add a forwarder statement *named.conf* file similar to the following:

```
Options
{
forwarders {123.123.123.132; 123.123.123.123;};
};
```

Each IP address in the forwarders option is separated by a semicolon (;).

You must also populate the *db.cache* file for the forwarders statements to work. If you do not use a *db.cache* file, you must add a slave statement to the forwarders' directives similar (BIND 8.x):

```
Options
{
forwarders {123.123.123.123; 234.234.234.234;};
forward only;
};
```

Why aren't zone transfers occurring?

Several reasons exist as to why zone transfers are not occurring:

- The primary server is not running.
- The primary server is not configured to allow transfers from a secondary.
- The secondary server is not configured as a slave for the primary zone.
- There are syntax or other errors preventing the zone from loading correctly on the primary server.

To rectify the situation, take the following actions:

1 Verify that the primary server is running.

2 Verify the primary server's *named.conf* file is allowing transfers for the secondary from the `allow-transfer` statement in the `zone` statement or in the `options block if no allow-transfer` statement appears in the `zone` statement.

3 Verify within the *named.conf* or *named.boot* on the secondary server that the zone is properly defined.

4 Verify there are no errors when loading the primary zone; otherwise, the secondary server also fails to load the zone. Check the *syslog* (UNIX) or the Event Viewer (Windows 2000) to determine if any DNS error messages were received and that the zone was loaded properly. Refer to "Service errors and informational messages", on page 413 for an error message list.

END OF STEPS

How do I pull zones from a non-VitalQIP integrated DNS server?

For instructions on pulling zones from non-VitalQIP integrated DNS servers, see "Non-managed DNS servers" in Chapter 3 of the *VitalQIP User's Guide*.

When I use NSLOOKUP on an object, I am seeing a non-authoritative response. Why?

The responding DNS server has non-authoritative responses in its cache. The DNS server is not a primary or a secondary server for the zone in which the record resides. The record may exist in cache from a previous query and the name server responds to the query with information that may be "stale". It responds with a "non-authoritative" response to let the user know the answer may no longer be accurate.

Another reason that you may be receiving non-authoritative responses for queries is that the zone for your authoritative name server did not load properly. Check the *syslog* (UNIX) or the Event Viewer (Windows 2000) to determine if any DNS error messages were received, and the zone loaded properly. Refer to "Service errors and informational messages", on page 413 for an error message list.

Why aren't dynamic updates to DNS for my DHCP clients occurring?

There are several reasons why dynamic updates to the DNS server may not be occurring:

- DNS server is not running on the remote server.
- DNS configuration is incorrect.
- The Message Routes for the VitalQIP enterprise and the VitalQIP remote servers are not configured properly.

To rectify the situation, take the following the actions:

1 Enable debugging to determine the cause of the problem.

2 Verify **named** is running.

3 Verify the DNS configuration is correct.

4 Verify that the proper domain is configured in the DHCP scope.

5 Verify that the proper DNS server is configured to be authoritative for that domain.

6 If the updates from DNS are being handled by the Message Service (default configuration), then the *qip.pcy* file on the Remote Service should have the following route statements in the **[VitalQIP Message Service]** section:

```
MessageRoute=DHCP:A:0:VitalQIP QIP Update Service:VitalQIP QIP Update
  Service:<QIP_Update_Service's_IP_address>
MessageRoute=DHCP:A:0:VitalQIP DNS Update Service:VitalQIP DNS Update
  Service:<DNS_Update_Service's_IP_address>
```

7 If the updates from DNS are being handled by the QIP Update Service (recommended method to avoid dynamic name collisions with static objects) the QIP Update Service message route statement is required, but the DNS Update Service message route statement must be commented out. In addition, the *qip.pcy* file must have UpdateDNS = True (set in the **[VitalQIP QIP Update Service]** section). Refer to “VitalQIP QIP Update Service policies”, on page 104 for more information on configuring the DNS updates properly.

8 If your name servers are “split”, meaning one name server is primary for the reverse zone, ensure the “Independent Zones” policy is set to “True”.

9 Use the *dhcpd.log*, *qip-msgd.log*, or the *qip-qipupdated.log* (depending on if you have the Message Service or QIP Update Service configured to do the DNS updates) to see if the dynamic updates are being sent.

END OF STEPS

Why does Network Services|DNS Generation fail on Windows 2000?

The push may have failed because Sybase is “holding on” to port 1099. Check the *qip-rmisched.log* file for an error message similar to the following:

```
Thu, Jan  3 21:51:15.085: Creating RMIScheduler
Thu, Jan  3 21:51:17.008: Using previously created registry
Thu, Jan  3 21:51:17.118: busy Executors = 0
```

```
Thu, Jan  3 21:51:17.118: Remote Scheduler created
Thu, Jan  3 21:51:17.118: Binding QAPI_Scheduler
Thu, Jan  3 21:51:19.010: Binding failed.  Is rmiregistry running?
Thu, Jan  3 21:51:19.010: Error starting service
Thu, Jan  3 21:51:19.010: java.rmi.ConnectException: Connection refused to
host: 198.200.138.243; nested exception is:
java.net.ConnectException: Connection refused: connect
Thu, Jan  3 21:51:19.010: ------------------  Generating stack
race  ------------------java.rmi.ConnectException: Connection refused to
host: 198.200.138.243; nested exception is:
java.net.ConnectException: Connection refused:
connectjava.net.ConnectException: Connection refused: connectat
java.net.PlainSocketImpl.socketConnect(Native Method)at
java.net.PlainSocketImpl.doConnect(Unknown Source)at
java.net.PlainSocketImpl.connectToAddress(Unknown Source)at
java.net.PlainSocketImpl.connect(Unknown Source)at
java.net.Socket.<init>(Unknown Source)at java.net.Socket.<init>(Unknown
Source)at
sun.rmi.transport.proxy.RMIDirectSocketFactory.createSocket(Unknown Source)
at sun.rmi.transport.proxy.RMIMasterSocketFactory.createSocket(Unknown
Source)at sun.rmi.transport.tcp.TCPEndpoint.newSocket(Unknown Source)at
sun.rmi.transport.tcp.TCPChannel.createConnection(Unknown Source)at
sun.rmi.transport.tcp.TCPChannel.newConnection(Unknown Source)at
sun.rmi.server.UnicastRef.newCall(Unknown Source)at
sun.rmi.registry.RegistryImpl_Stub.rebind(Unknown Source)at
com.lucent.qtek.qip.qapi.RMISchedulerServer.<init>(RMISchedulerServer.java:2
97)at
com.lucent.qtek.qip.qapi.RMISchedulerServer.main(RMISchedulerServer.java:218
)Thu, Jan  3 21:51:19.020: ------------------   Stack trace
mplete   ------------------Thu, Jan 03 21:51:19.030:
com.lucent.qtek.qip.qapi.RMISchedulerServer::main() returned
Thu, Jan 03 21:51:19.030: Waiting for termination
```

If you find an error message similar to the on above, stop and restart the RMI Scheduler Service and Sybase database. Port 1099 will be freed, and you can use **Network Services|DNS Generation** to push data to a server.

Is there a way to ensure that file permissions and ownership are retained when pushing files from the FGS to remotes?

Because VitalQIP pushes to heterogeneous environments (for example, from Windows to Unix), file ownership attributes are lost when the files are sent to the remote. Only the file name and contents are copied from the FGS.

You can use the **qipdnsuserexitfgs** user exit to create a separate file on the FGS that contains attributes of each file in the push directory. Then, on the remote, have the user exit use this file to restore attributes after the files have been created.

How do I shut off Incremental Zone Transfers (IXFRs)?

To shut off incremental zone transfers, follow these steps:

1 In the VitalQIP client’s main window, select **Infrastructure|Server**. The Server Profile Option window opens.

2 Select **Modify Server**.

3 Select the server.

4 Click **OK**. The Server Profile window opens.

5 Click **Corporate Extensions|Modify**.

6 Type:

```
options {
use-ixfr no;
};
```

7 Click **OK** to save the changes. You are returned to the VitalQIP main window.

8 Select **Network Services|DNS Generation** to “push” this information to the server receiving the error.

END OF STEPS

How do I get my Windows 2000 SRV records in VitalQIP?

VitalQIP can be set up to import Window 2000 SRV. For more information about importing SRV records, see “External objects and resource records support”, on page 264.

How do I reinstall a service?

If you tried all troubleshooting resolutions in this chapter and continue to experience problems with the service, try reinstalling the service through a command line interface.

Table 73 shows where the executables for VitalQIP services are stored.

Table 73 Location of VitalQIP services executables

Service	Location
VitalQIP Schedule Service	%QIPHOME%\ScheduleService.exe
VitalQIP QIP Update Service	%QIPHOME%\UpdateService.exe

Service	**Location**
VitalQIP Remote Service	%QIPHOME%\Remote\RemoteService.exe
Lucent DNS Service	%SYSTEMROOT%\System32\named.exe
Lucent DHCP Service	%QIPHOME%\DHCP\DHCPService.exe
VitalQIP Active Lease Service	%QIPHOME%\DHCP\ActiveService.exe
VitalQIP Message Service	%QIPHOME%\DHCP\DHCPMessageService.exe
MS 4.0 DHCP Monitor Service	%QIPHOME%\DHCP\MS4DHCPMonitorService.exe
IBM DHCP Service	%QIPHOME%\DHCP\IBMDHCPMonitorService.exe

To reinstall a service, follow these steps:

1. To uninstall a VitalQIP service, type <*service_name*> **-remove** in the appropriate directory.
2. To install a VitalQIP service, type **<***service_name***> -install** in the appropriate directory.
3. To get the version number of the executable for a VitalQIP service, type <*service_name*> **-v** in the appropriate directory.

END OF STEPS

DNS Error Messages

Table 74 describes error messages for DNS.

Table 74 DNS Error Messages

Error Message	Description	Resolution
[VitalQIP DomainNameService check_hints: no A records for D.ROOT-SERVERS.NET class 1 in hints]	The root server is not listed in your *db.cache* file. When this server is queried, it returns the root servers, but they are not defined in your cache.	Update your *db.cache* file to include all the necessary Internet root servers.

Error Message: [AXFR for (<domain> restarting, first and last serial differs:3619,3620]	The error message appears if a secondary DNS server is pulling a zone from the primary server while the primary server is getting dynamic updates, and the secondary server fails to load the zone. At the beginning of a zone transfer, the secondary server checks the primary server's SOA record for the serial number. If dynamic updates are coming to the primary server, its serial number is incremented.
	At the end of a zone transfer, the secondary server checks this serial number, it differs from the serial number at the beginning of the zone transfer. If the serial numbers at the beginning of a zone transfer differ, the secondary server deletes the *tmp* file to which the zone is transferred, and tries again. The second attempt also fails for the same reason. Thus, the zone is unavailable to the secondary.
		For sites with many dynamic updates, the secondary server might have problems pulling zones from the primary server. If you want the primary server to allow outgoing AXFR while dynamic updates occur, go to the Server Profile of the primary server and add the following to the Corporate Extensions on the slave server:
		`options`
		`{ directory "c:\\qip\\dns";`
		`qddns`
		`{ axfr-serial-diff-tolerance 1000; }; };`
		After the lines are added, push this information to the server receiving the error by using **Network Services|DNS Generation**.

Error Message: [VitalQIP DomainNameService i_sysop: finds error (NXDOMAIN) on <host.domain>]	<host.domain> is listed as a nameserver for a domain, but the IP address for <host.domain> cannot be resolved.	Ensure the DNS server can resolve the NS resource record of the zones fir which it is authoritative.

Error Message	Description	Resolution

Error Message: [VitalQIP DomainNameService premature EOF, fetching "seg4.qa.quadritek.com"	The primary server immediately shuts down any AXFR TCP connection for a zone when it receives a dynamic update for that zone. The slave server writes an error message (premature EOF, fetching "zone name") to the *syslog* file (UNIX) or *eventlog* file (Windows 2000). This only occurs for BIND 8.x servers.	For sites with many dynamic updates, the secondary server might have problems pulling zones from the primary server. If you want the primary server to allow outgoing AXFR while dynamic updates occur, go to the Server Profile of the primary server and add the following to the Corporate Extensions:
		`options`
		`  { directory "c:\\qip\\dns";`
		`    qddns`
		`      allow-axfr-during-dynamic-update yes; };`
		After the lines are added, push this information to the server receiving the error
		Important! This option is only for a primary server serving an outbound AXFR.by using **Network Services|DNS**.

Error: Could not connect to <Server Name>. DNS File Generation failed. Error Code= -1	This error message occurs while doing a DNS push. The Remote Service is not running.	To rectify the situation, start Remote Service (**qip-rmtd**). If the service does not start, see the "VitalQIP services/daemons", on page 357 for information on enabling debugging.

□

12 Troubleshoot DHCP

Overview

Purpose

This chapter provides tips and possible resolutions to issues you may encounter with DHCP. Many of the common problems and questions related to VitalQIP operations are included in this chapter. Carefully review this chapter before calling technical support.

Before reading this chapter, ensure you have verified that your environment is properly set up and you have the proper files before calling Technical Support. Read “Before calling technical support”, on page 344 and “Verifying Your Environment”, on page 346 for more information.

Contents

This chapter presents the following topics.

☐

Common DHCP Problems and Questions

When to use

This section contains common problems and questions you may run into. The possible causes and resolutions are listed after each question.

Why can't I start the DHCP Service (dhcpd)?

If you cannot start the DHCP Service (**dhcpd**), take the following actions:

1. Verify that *dhcpd.conf* and *dhcpd.pcy* files are in place and not corrupted.
2. Verify that port 67 is not being used by another application. If there is another DHCP or Bootp server running, stop it. There can be only one DHCP or Bootp server (using Port 67) running on a box at a time. Use the **netstat** command to determine active TCP/UDP connections.
3. Enable debugging to determine the problem's cause. See the next question, How do I enable debugging for DHCP (dhcpd)?.

END OF STEPS

How do I enable debugging for DHCP (dhcpd)?

From within the VitalQIP client, the DHCP server must be configured to enable debugging. Debugging can be enabled through the **Debug Information** policy of the DHCP Server Profile, and using **Network Services|DHCP Generation**. For information on the Server Profile, see “Servers” in Chapter 3 of the*VitalQIP User's Guide*.

After debugging is enabled, use **Network Services|DHCP Generation**. As the DHCP is restarted, a *dhcpd.log* file is generated in the *$QIPHOME/log* directory. Contact technical support with this log handy to assist in resolving DHCP server issues. For information on the Network Services, see Chapter 5 of the *VitalQIP User's Guide*.

Why is my DHCP server not handing out IP addresses?

There are several possibilities why a DHCP server may not be handing out leases. The more common reasons are:

- The act of creating a new DHCP scope in the GUI is not enough for the DHCP server to give new addresses. Whenever a new DHCP scope is created, a “push” to the server must be done using **Network Services|DHCP Generation** before the DHCP server can start giving out new addresses.
- The **Infrastructure|Server|RegisteredClientsOnly** parameter is set to “True” in the Server Profile. The DHCP server only gives IP addresses to clients with MAC addresses defined in global or subnet MAC address pools. With DHCP 5.2 and later versions,

Manual DHCP objects do not need to be defined in global or subnet MAC pools. The DHCP server also gives IP addresses to clients whose MAC address is defined in Global Include MAC pools, but it cannot give IP addresses to clients defined in Global Exclude MAC pools. If there are no MAC address pools defined within VitalQIP, the DHCP server cannot give any IP addresses.

- Routers in your environment may not be configured properly. Verify with your router administrators that "IP Helper" addresses are configured on the routers. A *dhcpd.log* showing no traffic originating from clients in the affected subnet indicates a router is not configured correctly.

  Important! "IP Helpers" must be configured for failover servers and primary servers. When a failover DHCP server assumes the role of the primary server, it can only hand out addresses to DHCP Discovers and Requests that it receives.

- Networks, subnets, routers, or DHCP servers that are assigned to the ranges of addresses may not be configured correctly in VitalQIP. Enable debugging to determine the cause of the problem. If you need to contact technical support, send a copy of *dhcpd.conf*, *dhcpd.pcy*, and a *dhcpd.log.* The *dhcpd.log* file should contain a client attempting, but failing to get an IP address. Specify the MAC address of the DHCP client in question when contacting technical support.
- Shared subnetting within VitalQIP may not be configured properly. See the next question, "How do I set up Shared Subnetting (Secondary Addressing or Virtual Addressing?".

How do I set up shared subnetting (secondary addressing or virtual addressing)?

Having multiple "virtual" subnets configured on one physical local area network segment is known as "multi-netting" or secondary addressing. The multiple subnets have the same physical router interface for all the shared subnets. When the DHCP addresses on the primary interface are used, the DHCP server hands out leases from the next configured shared subnet, and so on. The router defined on the primary interface is the gateway used for DHCP ("GIAddr" field in DHCP packets). To enable secondary addressing in VitalQIP, follow these steps:

1. Enter a shared network name in the **Shared Network** field of the Subnet Profile for the primary subnet to be shared. This enables the **Primary Interface**.
2. Check **Primary Interface**.
3. Select the same shared network name in the **Shared Network** field of the Subnet Profile for each of the other subnets to be shared.
4. Define at least one dynamic address (D-DHCP, M-DHCP or A-DHCP) on the subnet that is defined as the primary interface.
5. Use **Network Services|DHCP Generation** push information to the screen.

6 Review the changes made to the *dhcpd.conf* file. Each of the subnets designated as a “Shared” subnet is contained in a shared network within the *dhcpd.conf* file.

For example:

```
DHCP Server: dhcpsvr1.lucent.com
server-identifier dhcpsvr1.lucent.com;

# Name: Share
   shared-network _200_200_200_0 {
      subnet 200.200.200.0 netmask 255.255.255.0 {
         dynamic-dhcp range 200.200.200.9 200.200.200.11 {
            option subnet-mask 255.255.255.0;
            option domain-name “lucent.com”;
            option domain-name-servers 200.200.200.3;
            option dhcp-lease-time 7776000;
         }
      }
      subnet 200.220.220.0 netmask 255.255.255.0 {
         dynamic-dhcp range 200.220.220.4 200.220.220.6 {
            option subnet-mask 255.255.255.0;
            option domain-name “lucent.com”;
            option domain-name-servers 200.200.200.3;
            option dhcp-lease-time 7776000;
         }
      }
   }
```

END OF STEPS

How do I setup custom DHCP Server Templates?

Refer to “DHCP template classes and options” in Chapter 2 of the *VitalQIP User’s Guide*.

Why does Network Services|DHCP Generation fail on Windows 2000?

The push may have failed because Sybase is “holding on” to port 1099. Check the *qip-rmisched.log* file for an error message similar to the following:

```
Thu, Jan  3 21:51:15.085: Creating RMIScheduler
Thu, Jan  3 21:51:17.008: Using previously created registry
Thu, Jan  3 21:51:17.118: busy Executors = 0
Thu, Jan  3 21:51:17.118: Remote Scheduler created
Thu, Jan  3 21:51:17.118: Binding QAPI_Scheduler
Thu, Jan  3 21:51:19.010: Binding failed.  Is rmiregistry running?
Thu, Jan  3 21:51:19.010: Error starting service
Thu, Jan  3 21:51:19.010: java.rmi.ConnectException: Connection refused to
host: 198.200.138.243; nested exception is:
java.net.ConnectException: Connection refused: connect
Thu, Jan  3 21:51:19.010: ------------------  Generating stack
race  ------------------java.rmi.ConnectException: Connection refused to
host: 198.200.138.243; nested exception is:
```

```
java.net.ConnectException: Connection refused:
connectjava.net.ConnectException: Connection refused: connectat
java.net.PlainSocketImpl.socketConnect(Native Method)at
java.net.PlainSocketImpl.doConnect(Unknown Source)at
java.net.PlainSocketImpl.connectToAddress(Unknown Source)at
java.net.PlainSocketImpl.connect(Unknown Source)at
java.net.Socket.<init>(Unknown Source)at java.net.Socket.<init>(Unknown
Source)at
sun.rmi.transport.proxy.RMIDirectSocketFactory.createSocket(Unknown Source)
at sun.rmi.transport.proxy.RMIMasterSocketFactory.createSocket(Unknown
Source)at sun.rmi.transport.tcp.TCPEndpoint.newSocket(Unknown Source)at
sun.rmi.transport.tcp.TCPChannel.createConnection(Unknown Source)at
sun.rmi.transport.tcp.TCPChannel.newConnection(Unknown Source)at
sun.rmi.server.UnicastRef.newCall(Unknown Source)at
sun.rmi.registry.RegistryImpl_Stub.rebind(Unknown Source)at
com.lucent.qtek.qip.qapi.RMISchedulerServer.<init>(RMISchedulerServer.java:2
97)at
com.lucent.qtek.qip.qapi.RMISchedulerServer.main(RMISchedulerServer.java:218
)Thu, Jan  3 21:51:19.020: ------------------   Stack trace
mplete   ------------------Thu, Jan 03 21:51:19.030:
com.lucent.qtek.qip.qapi.RMISchedulerServer::main() returned
Thu, Jan 03 21:51:19.030: Waiting for termination
```

If you find an error message similar to the on above, stop and restart the RMI Scheduler Service and Sybase database. Port 1099 will be freed, and you can use **Network Services|DHCP Generation** to push data to a server.

After a lease has been granted to an object, why don't I see object information updated in the VitalQIP Object Management window?

Several things can cause the update to not appear:

- The QIP Update Service (**qip-qipdupdate**) is not running on the VitalQIP enterprise server.
- The Message Service (**qip-msgd**) is not running on the VitalQIP remote server.
- The Message Routes for the VitalQIP enterprise server and the VitalQIP remote servers are not configured properly.

To rectify the situation, take the following action:

- Start the QIP Update Service (**qip-qipupdated**) on the VitalQIP enterprise server. Refer to “VitalQIP services/daemons”, on page 357 for information on enabling debugging if the service does not start.
- Start the Message Service (**qip-msgd**) on the VitalQIP remote server(s).
- If the updates from DNS are being handled by the Message Service (default configuration), then the *qip.pcy* file on the VitalQIP remote server must have the following route statements in the **[VitalQIP Message Service]** section:

```
MessageRoute=DHCP:A:0:Lucent DHCP Update Service:VitalQIP QIP Update
  Service:<QIP_Update_Service's_IP_address>
MessageRoute=DHCP:A:0:Lucent DNS Update Service:VitalQIP DNS Update
  Service:<DNS_Update_Service's_IP_addres>
```

If the updates from DNS are being handled by the QIP Update Service (recommended method to avoid dynamic name collisions with static objects) the QIP Update Service message route statement mentioned above is required, but the DNS Update Service MessageRoute statement must be commented out. In addition, the *qip.pcy* file on the VitalQIP enterprise server must have **UpdateDNS = True** set in the **[VitalQIP QIP Update Service]** section. Refer to “VitalQIP QIP Update Service policies”, on page 104 for more information on configuring the DNS updates properly.

□

DHCP error messages

Table 75 describes error messages for DHCP.

Table 75 DHCP error messages

Error message	Description	Resolution
Error: Could not connect to <*Server Name*>	This error message occurs when doing a DHCP push. Remote Service (**qip-rmtd**) is not running.	To rectify the situation, start the Remote Service (**qip-rmtd**). Refer to "VitalQIP services/daemons", on page 357 for information on enabling debugging if the service does not start.
Error: DHCP Generation Failed	Your failover server might be configured, but the administrator does not have privileges to push to the failover server.	If this is the cause, the *qip-rmtd.log* file contains the following error: DHCP push status: 3 Leaving function DbdhcpConfigGenerate Action status = 3 DHCP Update: <server name> failed - path="<pathname>" The VitalQIP administrator (qipman) or other privileged user must change the current Administrator Profile to include the failover DHCP server in the managed list. Refer to "Administrators" in Chapter 3 of the *VitalQIP User's Guide* for more information.
Error: No available lease information	This error message occurs when viewing active leases. There are no leases recorded in the lease file (*dhcp.db*) for the subnet(s) selected.	None.
Error: Could not connect to <*server name*>	The Active Lease Service (**qip-netd**) must be running to view the current active lease file.	Ensure that Active Lease Service (**qip-netd**) is running on your DHCP server system. If it is not, start the Active Lease Service (**qip-netd**). Refer to "VitalQIP services/daemons", on page 357 for information on enabling debugging if the service does not start.

□

13 Other messages

Overview

Purpose

This chapter describes messages you may encounters. Many common messages are included in this chapter. Carefully review this chapter before calling technical support.

Before reading this chapter, ensure you have verified that your environment is properly set up and you have the proper files before calling Technical Support. Read “Before calling technical support”, on page 344 and “Verifying Your Environment”, on page 346 for more information.

Contents

This information presents the following topics.

☐

Database error messages

Table 76 describes database errors and their possible resolutions.

Important! The gap in sequence numbers is due to error messages that are no longer valid.

Table 76 Database errors

Database error	Description	Resolution
1 Invalid subnet address	The selected subnet is not managed by VitalQIP.	Verify the subnet exists in VitalQIP. If not, add the subnet.
2 Start or end address is not correct	The selected start and/or end address is outside the legal values of the subnet.	The start/end address must be valid for the subnet based on the subnet mask. For more information on subnetting, see “Subnetting”, on page 26.
3 Privilege Denied	Permission to do this function is not granted. User does not have write privileges.	If allowed, assign write permissions to this administrator for this function.
5 There are no more ID’s available	There are no object IDs available.	All possible object IDs are configured. Contact technical support.
16 No IP Addresses available	There are no IP addresses available.	All possible IP addresses are configured. Contact technical support.
19 No hub port available	There is no hub or port available.	All possible hubs/ports are configured. Contact technical support.
21 This domain [name] exists already	This domain is already managed by VitalQIP.	Duplicate domain names are not allowed; choose another name.
22 The name [name] is the same as a domain name	The selected name is the same as a domain that is managed by VitalQIP.	Choose another name.
23 The object [name] does not exist	This object name is not defined in VitalQIP.	Verify the object name and address. If it does not exist, define the object in VitalQIP.
25 This server [name] does not exist	An IP address (object) has not been assigned to this server.	Define this server as an object in the Object Management window.

Database error	Description	Resolution
29 This domain [name] does not exist	The domain had not been defined in VitalQIP.	Verify that the domain name has been typed in correctly (may be case sensitive). If it does not exist, define the domain in VitalQIP.
31 The name [name] is the same as a subnet name	The subnet name is already defined for another subnet.	Duplicate subnet names are not allowed. Choose another name.
32 This subnet does not exist	This subnet has not been defined in VitalQIP.	Verify that the subnet name has been typed in correctly (may be case sensitive). If it does not exist, define the subnet in VitalQIP.
34 The hub [name] does not exist	The selected hub is not managed by VitalQIP.	Verify that the hub name has been typed in correctly (may be case sensitive). If it does not exist, define the hub in VitalQIP.
46 This subnet group [name] does not exist	The specified subnet group name has not been defined in VitalQIP.	Verify that the subnet group name has been typed in correctly (may be case sensitive) If it does not exist, define the subnet group in VitalQIP.
47 This network does not exist	The specified network has not been defined in VitalQIP.	Verify that the network name has been typed in correctly (may be case sensitive). If it does not exist, define the network in VitalQIP.
48 This subnet does not exist	The specified subnet has not been defined in VitalQIP.	Verify that the subnet name has been typed in correctly (may be case sensitive). If it does not exist, define the subnet in VitalQIP.
49 This administrator [name] exists already.	The specified administrator is already defined in VitalQIP.	Duplicate administrator names are not allowed; choose another name, or modify an existing administrator.
52 The forwarder [name] does not exist	The mail forwarder host does not exist.	Verify that the forwarder name has been typed in correctly (may be case sensitive). If it does not exist, define the forwarder in VitalQIP.
53 This mail host [name] does not exist	The selected mail host is not currently managed by VitalQIP.	Verify that the mail host name has been typed in correctly (may be case sensitive). If it does not exist, define the mail host in VitalQIP.
54 This forwarder [name] does not exist	The selected mail forwarder is not currently managed by VitalQIP.	Verify that the forwarder name has been typed in correctly (may be case sensitive). If it does not exist, define the forwarder in VitalQIP.

Database error	Description	Resolution
56 DECNet address information is missing	All DECNet address information is required.	Add the required DECNet address information.
61 The name [name] is the same as an object name	The selected name is the same as an already existing object in VitalQIP.	Duplicate object names are not allowed in the same subnet, choose another name.
62 No free subnet available	There is not a free subnet available. For the networks defined, there is no undefined address space where a subnet can be configured.	Define a new network to have more subnets available.
63 This object does not exist	The specified object is not defined in VitalQIP.	Verify that the object name and address has been typed in correctly (may be case sensitive). If it does not exist, define the object in VitalQIP.
64 IPX information is missing.	IPX information is incomplete.	Add the missing IPX information.
67 This application [name] does not exist	The specified application is not defined in VitalQIP.	Verify that the application name has been typed in correctly (may be case sensitive). If it does not exist, define the application in VitalQIP.
69 This manufacturer [name] with the object class does not exist	This manufacturer is not associated with this object class.	Create and associate the manufacturer with the object class, or choose another manufacturer.
70 This manufacturer with the object class does not exist	This manufacturer is not associated with this object class.	Create and associate the manufacturer with the object class, or choose another manufacturer.
71 This manufacturer does not exist	The selected manufacturer is not a registered manufacturer within VitalQIP.	Verify that the manufacturer has been typed in correctly (may be case sensitive). If it does not exist, define the manufacturer in VitalQIP.
72 This manufacturer with the object class exists already.	The selected manufacturer is already assigned to this object class.	Cancel the operation or choose another manufacturer/class.
74 This object class is invalid	An invalid object class is defined for this object.	Choose one of the existing object classes defined in VitalQIP.
76 This IP address is not in the subnet	The selected IP address is not within the selected subnet.	Verify that the IP address was typed in correctly, or assign an object to this IP address.

Database error	Description	Resolution
77 This IP address is unused	An object has not been defined for this IP address.	Verify that the IP address was typed in correctly or assign an object to this IP address.
78 This hub slot/port is not available	There are no slot or ports defined for the selected hub, or they are not available.	Define slots and ports for this hub.
79 No big-name available	Unable to create a unique object name.	All possible object names for this object class have been created.
80 A DHCP server is required	DHCP servers must be specified when creating dynamic objects.	Assign a DHCP server to this object.
88 This administrator [name] does not exist	The specified administrator has not been defined in VitalQIP.	Verify that the administrator name was typed in correctly. If it does not exist, define the administrator in VitalQIP.
90 Some objects are connected to this hub	This hub has existing objects connected, so it cannot be deleted.	Delete or re-assign the existing objects from the hub before deleting the hub.
92 This IP address or name is invalid	An illegal IP address or name is being selected for the object.	Verify that address and name were typed in correctly.
93 This address or name is invalid	An illegal IP address or name is being selected for the object.	Verify that address and name were typed in correctly.
94 This action is invalid	The selection made is not a valid one.	Select another item or cancel the operation.
96 More than one subnet found	Multiple subnets were found.	Narrow the search.
101 The whole subnet has been set to move to another subnet already	This subnet has already been scheduled to move to another subnet.	Cancel or modify the scheduled move.
102 This mac address is in use already	Duplicate MAC addresses are not allowed in VitalQIP.	Choose another MAC address, or change the ALLOWDUPMACADDR global policy if you want duplicate MAC addresses.
103 The name [name] is the same as a mac address.	The selected name is the same as that of a MAC address.	Choose another name.
107 This alias name [name] is the same as one object in this domain	Alias and object names cannot be the same within the same domain.	Change the alias to a name that is not the same as an object within this domain.

Database error	Description	Resolution
108 This alias name [name] exists already in this domain	Alias names must be unique within domains.	This alias is already defined within this domain; choose another alias name.
109 Please use “MOVE” to modify IP Address	IP addresses cannot be modified to an address in another subnet.	If you want to change the IP address to one in another subnet, use the “Move” function to move the object.
113 The object name is missing	There is no name assigned to this object.	Assign a name to this object.
121 The name [name] is the same as an OSPF area name	The name specified is the same as an existing OSPF area name.	Choose another name.
124 This OSPF area [name] does not exist	The specified OSPF area has not been defined in VitalQIP.	Verify that the OSPF name has been typed in correctly (may be case sensitive). If it does not exist, define the OSPF Area in VitalQIP.
133 Duplicate netbios name	The selected NetBIOS name is already in use.	Choose another NetBIOS name.
134 The object name is the same as an alias name under this domain	Alias and object names cannot be the same within the same domain.	Change the alias to a name that is not the same as an object within this domain.
135 The whole subnet is scheduled to move to another subnet	This subnet has already been scheduled to move to another subnet.	Cancel or modify the scheduled move.
137 This DNS server [name] does not exist	An IP address (object) has not been assigned to this DNS server.	Create a server object with this DNS server name.
138 This time server [name] does not exist	An IP address (object) has not been assigned to this time server.	Create a server object with this time server name.
139 This DECNet area is invalid	The DECNet area specified is outside the valid range of 1 - 63.	Specify an area from 1 to 63.
140 No DECNet address available at this time	All addresses in this DECNet area have been allocated.	Specify another DECNet area.
141 This DECNet address is already in use	The specified DECNet address is already assigned to another object.	Choose a different DECNet address.

Database error	Description	Resolution
142 This DECNet area is invalid.	The DECNet area specified is outside the valid range of 1 - 63.	Specify an area from 1 to 63.
146 This IP address is used or selected already	The selected IP address is assigned or is currently selected for another object.	The IP address has already been assigned to an object, choose another object, or modify existing object.
154 This IP address is not in the same subnet	The specified IP address does not fall in the specified subnet.	Verify that the IP address was typed in correctly, or choose another subnet.
158 Space in the router group name [name] is not allowed	Spaces are not allowed in router group names.	Remove the spaces from the router group name. Use underscores or dashes.
159 Space in the object name [name] is not allowed	Spaces are not allowed in object names.	Remove the spaces from the object name. Use underscores or dashes.
160 Object name is required.	The object name is required in order to proceed.	Specify an object name.
162 Class is required.	The object class is required in order to proceed.	Specify an object class.
163 This manufacturer [name] has been used.	The specified manufacturer name is already in use.	Specify another manufacturer name.
170 This tag [name] does not exist in the generic template table	The specified tag value for this DHCP template parameter is not part of the specified template.	Add the tag value to the DHCP template parameter.
171 This tag [name] exists already	The specified tag value for this DHCP template parameter exists in the template. It cannot be re-used.	Choose another tag value for this parameter.
172 This code is in use	The selected template is in use by another user.	Wait until the user has finished and reassessed.
174 This address is reserved. It cannot be used	The selected address is reserved for another use.	Choose another address, or delete the reserved object.
177 Class [name] exists already	This object with this class name already exists.	Verify the object name has been typed in correctly (may be case sensitive).

Database error	Description	Resolution
180 DHCP server [name] does not exist	The specified DHCP server is not defined in VitalQIP.	Verify that the DHCP server has been typed in correctly (may be case sensitive). If it does not exist, define the DHCP server in VitalQIP
182 Template [name] exists already.	A template with this name is already defined in VitalQIP.	Duplicate template names are not allowed; choose another name.
183 Template [name] does not exist	This template is not defined within VitalQIP.	Verify that the template name has been typed in correctly (may be case sensitive). If it does not exist, define the template in VitalQIP.
184 Template [name] cannot be deleted. It is attached to 1 or more objects, subnets, or dhcp servers	A template cannot be deleted if it is in use.	Assign another template or remove the association to the objects, subnets, and/or DHCP server, then delete the template.
186 DHCP template [name] does not exist	The specified DHCP template has not been defined within VitalQIP.	Verify that the DHCP template has been typed in correctly (may be case sensitive). If it does not exist, define the DHCP template in VitalQIP.
192 Bootp server [name] exists already.	The specified Bootp server is already defined within VitalQIP.	Duplicate server names are not allowed; choose another name.
193 NIS or Local Server [name] exists already	The specified NIS or Local server is already defined with VitalQIP.	Duplicate server names are not allowed; choose another name.
194 DHCP or Bootp Server [name] exists already	The specified DHCP or Bootp server is already defined within VitalQIP.	Duplicate server names are not allowed; choose another name.
195 Bootp server [name] does not exist	The specified Bootp server has not been defined within VitalQIP.	Verify that the Bootp server has been typed in correctly (may be case sensitive). If it does not exist, define the Bootp server in VitalQIP.
197 DHCP server [name] does not exist	The specified DHCP server has not been defined within VitalQIP.	Verify that the DHCP server has been typed in correctly (may be case sensitive). If it does not exist, define the DHCP server in VitalQIP.
198 Bootp/DHCP server [name] does not exist	The specified Bootp/DHCP server has not been defined within VitalQIP.	Verify that the Bootp/DHCP server has been typed in correctly (may be case sensitive). If it does not exist, define the Bootp/DHCP server in VitalQIP.

Database error	Description	Resolution
200 This address does not exist in the database.	This object is not managed by VitalQIP or is not part of any subnet managed by VitalQIP.	Verify that the address was typed in correctly.
202 VitalQIP does not contain any info for this object	The specified object is not defined within VitalQIP.	Verify that the object was typed in correctly. If it does not exist, define the object in VitalQIP.
205 Class [name] does not exist	The specified class is not defined within VitalQIP.	Verify that the class has been typed in correctly.
206 There is no field named [name] for class [name]	The specified field for this class has not been defined within VitalQIP.	Verify that the field name has been typed in correctly (may be case sensitive). If it does not exist, define the field for the class in VitalQIP.
210 At least one alias is required	You must specify at least one alias.	Specify an alias.
211 This server is the only primary for at least 1 zone	This server cannot be deleted because it is the only primary server for at least 1 zone.	Assign another server to the zone, and then delete the server.
212 This subnet overlaps an OSPF range.	Cannot create a subnet that straddles an existing OSPF Area.	Change the subnet boundaries or delete the existing OSPF Area.
213 This subnet overlaps an existing subnet	Cannot create a subnet that overlaps an existing subnet.	Change the subnet boundaries by modifying the subnet mask.
214 Network [name] exists already	This network name is in use for another network.	Duplicate network names are not allowed; choose another name.
215 This server cannot be deleted while it has a defined scope	Servers cannot be deleted if they have a defined scope. There are dynamic objects associated with this server.	Delete the scope attached to this server, or assign another server to these dynamic objects, then delete the server.
216 Object name cannot be changed because it is attached to 1 or more servers	Server name has to be changed before changing the name on the object.	Change the name of the server before changing the name on the object.
218 Organization [name] exists already	Duplicate organizations are not allowed.	Choose another name for the organization.
220 OSPF range [name] overlaps an existing range in OSPF [name]	Each range is an OSPF area has to be unique.	Change the range boundaries, so that it does not overlap an existing OSPF area.
221 This reverse zone must have a parent	The reverse zone does not have a parent.	Define a parent for the reverse zone.

Database error	Description	Resolution
222 This reverse zone's parent must have a server assigned and have an 8, 16, or 24 bit mask	The reverse zone's parent does not have a server assigned and/or the reverse zone's parent does not have an 8, 16, or 24 bit mask.	Assign a server for the reverse zone's parent. Ensure the parent has an 8, 16, or 24 bit mask.
223 You cannot delete the base reverse zone for a network.	The reverse you are trying to delete is assigned to a network.	Operation is illegal.
224 The hierarchy class is not allowed here	You are trying to add a hierarchy class to node where it is not allowed.	Define the hierarchy class to another parent node.
225 Column name [name] is invalid	The column name is not valid.	Call technical support.
226 Out of application memory	Available memory has been used.	Reboot.
227 You cannot delete the default organization	At least one organization must be defined within VitalQIP.	You may change the name on the default organization, but you cannot delete it.
228 DHCP key group [name] exists already	Cannot add a duplicate DHCP template class.	Choose another template class.
229 There is already a network named [name]	There is a network defined with this name, duplicates are not allowed.	Choose another network name.
230 A non-compliant OSPF cannot be changed to compliant	You are trying to change a non-compliant OSPF area to compliant.	Create a compliant OSPF area.
231 Reverse zone [name] already exists	There is already a reverse zone defined with this name, duplicates are not allowed.	Choose another reverse zone name.
232 This action will delete [name] subnets	This split or join deleted objects.	If you do not want the objects to be deleted, change the boundaries on the split or join.
233 Address [name] is not within subnet [name]	The specified address is outside the scope of the specified subnet.	Verify that the IP address has been typed in correctly or change the specified subnet.
234 Server [name] is not associated with an IP address.	An IP address (object) has not been assigned to this server.	Define a server object with this server name.

Database error	Description	Resolution
235 There is already a subnet organization name [name].	There is already a subnet organization defined with this name; duplicates are not allowed.	Choose another subnet organization name.
236 Objects cannot be moved within the same subnet using this function. Use the Object Profile to modify the IP Address	Objects can only be moved from one subnet to another, not within the same subnet.	Modify the IP address within the Object Profile, or move the object to another subnet.
237 A DHCP template is required	Must specify a DHCP template.	Specify an existing DHCP template.
239 Domain [name] not valid for this subnet	This domain is not a valid domain for this subnet.	Add the domain to the subnet/network properties, or choose another domain.
240 Value "[value]" contains invalid characters	The value has invalid characters.	Enter valid characters.
241 User defined field [name] already exists	This user-defined field is defined within VitalQIP.	Duplicate user-define field names are not allowed; choose another name.
242 This module is not supported by the current license.	Need to obtain a new license key to run this module.	Contact your sales representative to purchase additional modules, or contact technical support to verify the license key.
243 Mail host and forwarder [name] cannot be the same.	The mail host and mail forwarder specified must be different.	Choose a different mail host or mail forwarder.
244 Administrator must be given privileges to associate users with objects.	This administrator does not have permission to associate users with objects.	If allowed, change the permissions in the Administrator Profile to allow the creation of users.
245 Address Range already exists.	The specified address range is already defined within this organization.	Use the existing range, or define a new one.
246 Address Range does not exist	The specified address range is not defined within this organization.	Verify that the address range has been typed in correctly (may be case sensitive). If it does not exist, define the address range in VitalQIP.
247 This name cannot be blank	A name must be specified.	Specify a valid name.
248 You cannot delete yourself	Unable to delete the currently logged in administrator.	A privileged administrator or qipman may be able to delete the administrator.

Database error	Description	Resolution
249 The Managed Item [name] already exists in the list.	Duplicate managed items are not allowed.	The managed item is already specified; choose another one.
250 Parent Reverse Zones cannot be deleted.	Cannot delete the parent reverse zone.	Do not delete the reverse zone parent.
251 A scheduled reclaim exists for this subnet, please remove reclaim and then delete subnet.	Unable to delete a subnet that has a scheduled reclaim associated with it.	Remove the scheduled reclaim before deleting the subnet.
252 This Non-Managed Secondary/Managed Primary Server combination already exists.	Duplicate combinations are not allowed.	Choose a different non-managed secondary or managed primary server combination.
253 Alias name, [name], cannot be that of an existing domain.	An alias name cannot be the same as an existing domain.	Choose another name for this alias that is not the same as an existing domain.
254 It is invalid to have two objects in the same subnet with the same MAC address.	MAC addresses must be unique within the same subnet.	Duplicate MAC addresses are only allowed in different subnets if ALLOWDUPMACADDR general policy is set to “True”.
255 Duplicate OSPF Area ID.	The OSPF area ID is already defined within VitalQIP.	Choose a different OSPF area ID or use the existing OSPF area.
256 DNS Server [name] exists already.	The DNS server specified already exists in this organization. Duplicates are not allowed.	Choose a different DNS server name or use the existing DNS server.
257 IPX Network and Node exists already	Duplicate combinations are not allowed.	Choose a different IPX Network and Node, or use the existing one.
258 E-Mail address [name] is formatted incorrectly	Invalid email address specified.	Need to specify a valid email address format (for example, user@company.com).
259 A scheduled move or planned use exists for this subnet, please remove scheduled move and then delete subnet.	Unable to delete a subnet that has a scheduled move or planned use.	Cancel the scheduled move or planned use, then delete the subnet.
260 Parent Folder does not exist.	The parent folder specified has not been defined within VitalQIP.	Verify that you have the correct folder name, or define the folder.
261 Folder, [name], already exists under this parent	The specified folder name already exists under this parent. Duplicates are not allowed within parent folders.	Choose another folder name or use the existing folder.

Database error	Description	Resolution
262 Folder must be empty	Cannot delete a folder that contains domains.	Remove the domain(s) from the folder before deleting the folder.
263 Folder does not exist	The folder specified has not been defined within VitalQIP.	Verify that the folder name has been typed in correctly (may be case sensitive). If it does not exist, define the folder in VitalQIP.
264 Server must be fully managed to boot from active directory	This server is not fully managed.	Verify the server name, and choose “Fully Managed” when setting up the server, add a fully managed server, or modify server to be fully managed.
265 Duplicate Zone. User Defined Zone with same name already exists.	Unable to have duplicates.	Use the existing zone, or choose another zone name.
266 The name [name] does not contain a valid domain.	The domain specified within the name has not been defined within VitalQIP.	Verify that the domain is correct or define the domain within VitalQIP.
267 Unable to modify administrator. Administrator may currently be connected to VitalQIP	Cannot modify an administrator that is currently logged into VitalQIP.	Have administrator log out in order to modify.
268 The zone [name] associated with the mx/cname is not part of the subnet domain	If administrator privileges are set to CNAME and MX record restrictions, the record CNAME and MX created must contain a domain that is valid for that subnet.	Verify that the domain is valid for subnet. If needed, assign the domain to the subnet properties. If the CNAME and MX record restrictions are not needed, modify the administrator privileges.
400 This is a demo copy. You are limited to 200 objects	Demo copies are limited to 200 objects.	None.
500 Policy requires Usage Billing Info for this object	This object must have billing information defined based on the policy settings.	Define billing information for the object.
501 Invalid Billing Usage Type.	The Billing Type entered is not valid.	Verify that the Usage Billing Type has been typed incorrectly (may be case sensitive).
502 Billing Location, User Group and Object Class are mandatory when billing usage info is specified for an object	This object must have billing information defined based on the policy settings.	Add billing information to the object.
503 This Usage Billing item already exists	The usage billing item specified is already defined within this organization.	Use the existing billing item, or choose another name for this item.

Database error	Description	Resolution
504 [name] is a VitalQIP defined Object Class	The Billing Object Class specified is defined as a VitalQIP Billing Object Class.	Use the existing Billing Object Class.
505 Cannot delete a domain, which contains a server.	The domain has a server attached to it.	Rename the server.
506 Unable to delete server	The server cannot be deleted.	Contact technical support.
507 Unable to fetch server	The server cannot be retrieved from the database.	Contact technical support.
508 Unable to fetch the template	The template cannot be retrieved from the database.	Contact technical support.
509 Cannot modify Root Zone name	The Root Zone name cannot be modified.	Illegal operation.
510 Cannot modify domain name, [name], to be that of a Root Zone	The domain name is the same as the Root Zone.	Contact technical support.
511 Non-Managed Server/Managed Server Combination exists in database as a deleted record. Perform a push to the managed server and retry	The managed server definition needs to be pushed to remove it from the database.	Push the data to the server and try performing the functions again.
512 Cannot assign Root Zone to a [server]	The Root Zone cannot be assigned to the server.	Contact technical support.
513 Unable to fetch administrator	The administrator cannot be retrieved from the database.	Contact technical support.
514 Unable to fetch administrator's organization	The administrator's organization cannot be retrieved from the database.	Contact technical support.
514 Unable to delete managed range values	The managed range values cannot be deleted from the database.	Contact technical support.

Service errors and informational messages

Non-database errors do not have a text string that displays, but an error code. Table 77 describes non-database error codes, the source of the problem, and the action to resolve the problem.

Table 77 Non-database error codes

Error code	Description	Resolution
2000	A VitalQIP function was called with invalid input.	Contact technical support.
2100	System is out of memory.	Install more memory.
2101	A new object cannot be created and possibly out of memory.	Install more memory.
2102	Could not open a socket. Process or system cannot allocate any more file descriptors	Reduce the number of processes running on the system.
2103	Process or system cannot allocate any more file descriptors.	Reduce the number of processes running on the system.
2104	Could not write to a file. The disk, device, or partition is full.	Erase some unnecessary files on the disk, device, or partition.
2200	QIPHOME is not defined. The QIPHOME environment variable is not set.	Set the QIPHOME environment variable.
2201	QIPHOME is not valid.	Correct the value of QIPHOME.
2202	Could not open the specified file.	Copy the file into the expected directory.
2203	Attempted an operation on an invalid process (pid).	Check the *.pid* files.
2204	Cannot determine database credentials (for example, username and password).	Check the *qip.pcy* file.
2205	Could not connect to the database.	Ensure the database is running.
2206	Do not have permission to perform a specified action.	Correct the permissions on the directory or file.
2207	Server address (of VitalQIP service) needs to be configured.	Check the policy files.
2208	Ping Service was not installed properly	Check for the existence of **qip_pingd**.
2209	Ping Service would not start.	Ensure set-uid bit is set (rwsr-xr-x).

Error code	Description	Resolution
2210	Could not open a dynamic library.	First, check that the environment variables are set, then check the libraries in *$QIPHOME/usr/lib*.
2211	DNS could not be initialized.	Ensure the *<x>.ddns.conf* file exists.
2300	Protocol could not be determined.	Check */etc/protocols*.
2301	Socket port could not be determined.	Ensure service name is found in */etc/services* directory.
2302	Could not resolve the hostname.	Hostname should be in the */etc/hosts* file.
2303	Could not set socket options.	Process or service may need to run as root or administrator.
2304	Could not set the listen buffer.	System resources may be consumed. End any unnecessary processes.
2305	Could not accept any new connections.	System resources may be consumed. End any unnecessary processes.
2306	Could not connect to a remote service.	Ensure the service is running on the specified host.
2307	Invalid IP address was specified.	Correct the IP address specified in the policy file.
2308	A time-out occurred while waiting for a response.	Check the network connectivity to ensure no services were killed.
2310	Send on socket failed.	System resources may be consumed. End any unnecessary processes.
2311	Receive on socket failed.	System resources may be consumed. End any unnecessary processes.
2312	Receive on socket failed.	System resources may be consumed. End any unnecessary processes.
2313	Communication with remote service abruptly terminated.	Do not kill services with a **kill -9**.
2314	Bind of socket failed.	Service may need to be run as root, or there is a service using the desired port.
2315	Authoritative response. Host not found.	Specified hostname needs to be put into DNS.

Error code	Description	Resolution
2316	Non-authoritative response. Host not found.	Host not in secondary DNS and primary is probably down. Bring the primary back online.
2317	Non-recoverable error. Host not found.	Resolver library is generating incorrect requestor bind server is refusing connections. Use network monitor to validate packages.
2318	No data for specified host.	The hostname is in DNS but does not have an associated IP address. Add appropriate information.
2400	Communication with the remote service is garbled in the header.	Check the network. Avoid mixing releases of VitalQIP.
2401	Communication with the remote service is garbled in the data.	Check the network. Avoid mixing releases of VitalQIP.
2402	Communication with the remote service is garbled. You are running mixed versions of VitalQIP services.	Avoid mixing releases of VitalQIP.
2403	A reply is pending from a previous ping when the service attempted another ping.	Obtain a new version of the service.
2404	Service did not perform a ping.	Obtain a new version of the service.
2500	A DNS or Corporate Extension string is greater than 255 characters.	Shorten the string.
2501	An object name has failed.	Correct the hostname of the offending machine.
2502	A “look” was unsuccessful.	None.
2507	Failed to generate NIS report.	None.
2600	The license file is wrong or the database entry is out of date.	Contact technical support.
2601	Could not determine the server host type.	The *.lic* file may be old. Obtain a new *.lic* file from technical support.
2602	License manager was unable to determine the next license key.	The Schedule Service (**qipd**) may not be able to update the database. Check to see if **qipd** is running.
2603	License manager was unable to read the license file.	Ensure the *.Lic* file is present in the QIPHOME directory.

Error code	Description	Resolution
2604	License manager unable to retrieve the server host ID.	In Windows 2000, ensure the server has an IP address.
9999	The application has received a fatal signal (either SIGSEGV or SIGBUS)	Contact technical support.

☐

Error and information messages generated by VitalQIP services

Both informational and error messages are logged by the following services:

- Schedule Service
- Message Service
- QIP Update Service

The messages are placed in the Event Log on Windows 2000. Table 78 describes informational and error messages logged by the above services.

Table 78 Informational and error messages

Message	Severity	Resolution
Using Default Policies	INFO	No policy file exists for this service.
Started	INFO	The service is running and can log messages.
License Key Update Interval = <*number_of_seconds*>	INFO	The service updates the license key every <*number_of_seconds*>.
Initialization complete	INFO	The database initialization has been completed.
Stopped	INFO	A service shutdown is imminent.
Could not determine <*service*> port number	ERROR	No entry for this service was found.
Could not resolve server <*server name*>	ERROR	DNS found no entry for <*server name*>.
I don't know where Update Service is	ERROR	The QIP Update Service was not specified in the policy file *(enterprise server)* or in the command line arguments.
Cannot connect to VitalQIP QIP Update Service	INFO	The QIP Update Service is not running.
Connection to Update Service Closed	INFO	The QIP Update Service was stopped.
Database Update Failed: <*lease_information*>	INFO	Lease information could not be updated in the VitalQIP database.
Lost Connection to Database	INFO	The database, its server, or the network path to it, is in an inconsistent state.
Re-Established Connection to Database	INFO	The database is available again.

Other messages

Message	**Severity**	**Resolution**
Configuration Error: DNS Previously Updated by DHCP Message Service	INFO	Both the Message Service and the Update Service are configured to update DNS.
Network Config Error, Duplicate Name: <*hostname*> on <*IP_address*>	INFO	The QIP Update Service is configured to update DNS, and has found a duplicate name. The duplicate would replace a static object (server) if allowed to continue.
Accepting connections	INFO	Initialization has been completed, waiting for connections from the Message Service.
Ignoring connection attempts until Database Re-Init	INFO	Since the database is down, no connections from the Message Services are allowed. Once the database is available again, connections can proceed.

☐

Glossary

A **authentication**

Authentication is the verification of the identity of a user or host. When you communicate over a network, authentication ensures that you are communicating with the intended party and not an imposter.

C **certificate**

A certificate is a file that contains information about a browser, server, proxy, or other network entity. It includes identifying information, the entity public key, and a signature made by a Certification Authority.

E **encryption**

Encryption is the encoding of data such that only the intended recipient can read it. Encryption hides the data from eavesdroppers and ensures that only the intended recipient can read it.

ephemeral port

An ephemeral port is a temporary port assigned by a machine's IP stack, and assigned from a designated range of ports for this purpose.

K **key pair**

A key pair consists of a matching public and private key.

S **SSL**

An acronym for Secure Socket Layer. SSL is a protocol for authentication and data encryption between a Web server and a Web browser. It permits private documents to be encrypted and safely transmitted over the Internet.

T **tunneling**

A way to channel communications between a server and a remote user that assists in simplifying firewall configurations. Secure methods of tunneling involve encryption.

Index