VitalQIP® DNS/DHCP & IP Management Software

VITALQIP® (QIP) | RELEASE 7.2

ADMINISTRATOR REFERENCE MANUAL

190-409-042R7.2
JULY 2009
ISSUE 5

Trademarks

All trademarks and service marks specified herein are owned by their respective companies.

Licenses

Refer to Appendix D, “Third party software license statements” in the *VitalQIP Installation Guide* (190-409-043R7.2) for a complete description of all software licenses used to develop this product.

Contents

About this document

Purpose xix

Reason for reissue xx

How to use this information product xxiii

Conventions used xxvi

Related information xxvi

Product Training Support xxvii

Technical support xxvii

How to order xxviii

How to comment xxviii

Part I: System planning

1 Plan and configure your network

VitalQIP components and planning

VitalQIP components 1-2

VitalQIP services 1-3

VitalQIP service configurations 1-9

Plan your network 1-11

Installation configurations

Single server scenario 1-12

Single enterprise and remote server scenario 1-12

Single enterprise, remote, and distributed server scenario 1-13

Single enterprise server, multiple remote servers, and multiple clients scenario 1-14

Single enterprise server, multiple remote servers, multiple distributed servers, and clients scenario 1-16

Part II: VitalQIP services administration

2 Manage VitalQIP services

VitalQIP services on Windows

Services checklist for Windows 2-3

VitalQIP Service Controller 2-6
Run services with the VitalQIP Service Controller 2-7
Configure the VitalQIP Service Controller 2-8
Lucent DNS Service options 2-11
Diagnose problems with the Event Viewer tab 2-13
Use Filtered Services 2-15
To start or stop the Tomcat server on Windows 2-16

VitalQIP services on UNIX

Services checklist for UNIX 2-17
VitalQIP startup scripts 2-20
Start DNS services manually 2-21
To start or stop the Tomcat server on UNIX 2-22
Start and stop daemons 2-22
qipd - VitalQIP Schedule Service daemon 2-23
qip-qipupdated - VitalQIP QIP Update Service daemon 2-24
qip-msgd - VitalQIP Message Service daemon 2-25
qip-logind - VitalQIP Login Service daemon 2-26
qip-dnsupdated - VitalQIP DNS Update Service daemon 2-27
qip-netd - VitalQIP Active Lease Service daemon 2-28
qip-rmtd - VitalQIP Remote Service daemon 2-29
qip-rmisched - VitalQIP RMI Scheduler Service daemon 2-30
qapi - VitalQIP RMI QAPI Service daemon 2-31
qip-ssltd - VitalQIP SSL Tunnel Service daemon 2-32
named - Lucent DNS Service daemon 2-33
dhcpd - Lucent DHCP Service daemon 2-34

3 qip.pcy file management

Manage the VitalQIP policy file

VitalQIP services policies 3-3
Sample qip.pcy file 3-3

Common policies

The debug policy 3-44
Java debug policies 3-47
Global section 3-49

4 VitalQIP QIP Update Service policies

Management of QIP Update Service policies

The qip-qipupdated.pcy file versus the qip.pcy file 4-2

VitalQIP QIP Update Service policies 4-2

Avoid dynamic name collisions 4-9

5 VitalQIP DNS Update Service policies

Management of DNS Update Service policies

The dnsupdated.pcy file versus the qip.pcy file 5-2

VitalQIP DNS Update Service policies 5-2

6 VitalQIP MS DNS Update Service policies

Management of MS DNS Update Service policies

The qip-msdnsupdated.pcy file versus qip.pcy file 6-2

VitalQIP MS DNS Update Service policies 6-2

7 VitalQIP Message Service policies

Management of the Message Service policies

Message Service behavior 7-2

The qip-msgd.pcy versus the qip.pcy file 7-3

VitalQIP Message Service policies 7-3

MessageQueue policy 7-9

MessageRoute policy 7-10

8 VitalQIP Login Service policies

Management of Login Service policies

The qip-logind.pcy versus the qip.pcy file 8-2

VitalQIP Login Service policies 8-2

9 VitalQIP Schedule Service policies

Management of Schedule Service policies

The qipd.pcy file versus the qip.pcy file 9-2

VitalQIP Schedule Service policies 9-2

Redundant Schedule Service 9-5

10 VitalQIP Active Lease Service policies

Management of Active Lease Service policies

The qip-netd.pcy file versus the qip.pcy file 10-2

VitalQIP Active Lease Service policies 10-2

11 VitalQIP MS DHCP Monitor Service policies

Management of MS DHCP Monitor Service policies

The MSDHCPMonitorServiced.pcy file veruses the qip.pcy file 11-2

VitalQIP MS DHCP Monitor Service policies 11-2

12 VitalQIP Remote Service policies

Management of Remote Service Policies

The qip-rmtd.pcy file versus qip.pcy file 12-2

VitalQIP Remote Service policies 12-2

13 VitalQIP RMI QAPI Service policies

Management of RMI QAPI Service policies

The qip-qapid.pcy veruses the qip.pcy file 13-2

VitalQIP RMI QAPI Service policies 13-2

14 VitalQIP RMI Scheduler Service policies

Management of RMI Scheduler Service policies

The qip-rmisched.pcy versus the qip.pcy file 14-2

VitalQIP RMI Scheduler Service policies 14-2

15 VitalQIP SSL Tunnel Service policies

Management of SSL Tunnel Service policies

The qip-ssltd.pcy file versus the qip.pcy file 15-2

VitalQIP SSL Tunnel Service policies 15-2

16 VitalQIP Access Control policies

Management of Access Control policies

VitalQIP Access Control policies 16-2

17 Web client related policies

Management of web client related policies

VitalQIP Service Layer policies 17-2

Address Allocator Service policies 17-3

VitalQIP Web GUI policies 17-4

MyView policies 17-5

Job Scheduler policies 17-6

Part III: VitalQIP system administration

18 Web client configuration

Character set and language configuration

Overview 18-3

To install localization files 18-4

To prepare the translation catalog 18-5

To create directories for localization 18-9

To update graphics with the new language 18-13

To make localized directories available 18-14

To customize specific labels 18-15

To migrate customized or localized labels and messages 18-17

To set linguistic sorting 18-20

Web client services configuration

Overview 18-22

Tomcat server configuration 18-22

Web Service configuration 18-23

Web client logging 18-24

log4j message levels 18-24

Sample log4j.properties file 18-26

qip.properties file 18-33

Sample qip.properties file 18-47

To configure session time-outs 18-65

To enable authentication via the Login Service 18-67

To enable compatibility with VitalQIP 7.0/7.1 Web Service 18-69

UDA callout

UDA callout configuration 18-70

Java callout classes 18-71

User-defined object naming policy

To configure a user-defined object naming policy for the web client 18-73

To create a UDA callout 18-75

Access external applications from the web client toolbar

How it works 18-78

To add an external application to the web client 18-80

19 Database and server administration

Database administration

Run Sybase 19-3

Find VitalQIP version numbers with vercheck 19-3

VitalQIP Sybase database backup 19-4

Transaction log backup 19-6

Server backup 19-6

Manual backup 19-6

Change the backup medium 19-8

Change the procedure cache size and total memory size 19-8

Password encryption 19-9

Database re-initialization 19-9

qip-dbinit 19-9

Database administrative tasks using qip-util

Track stored procedures and triggers 19-15

VitalQIP data space management 19-16

VitalQIP transaction log space management 19-18

Sybase database recovery

VitalQIP database recovery 19-21

Truncate audit data 19-24

Index statistics maintenance (Sybase only)

Server maintenance

Stop and start servers 19-26

To stop the VitalQIP enterprise server for maintenance 19-27

To start the VitalQIP enterprise server 19-29

To stop the VitalQIP remote server for maintenance 19-30

To start the VitalQIP remote server 19-31

20 Troubleshoot VitalQIP

General VitalQIP troubleshooting

Before calling technical support 20-2

Environment verification 20-4

System logs 20-5

Enable debugging in VitalQIP 20-8

Problem with logging into VitalQIP interface 20-10
Login error message 20-10
VitalQIP server failures 20-11
Automatic synchronization 20-12
Importing and exporting data 20-12

VitalQIP services/daemons

Troubleshooting services/daemons on UNIX 20-13
Troubleshooting starting and stopping services/daemons 20-30
Communication problems for VitalQIP service 20-33

21 General informational and error messages

Database error messages 21-1
Service errors and informational messages 21-15
Error and information messages generated by VitalQIP services 21-18

Part IV: Security administration

22 Authenticate administrators

Administrator authentication tool

VitalQIP Authentication Callout overview 22-2
Invoke the Authentication Callout function 22-4
Sample administrator authentication callout 22-6
Authentication callout policies 22-7
Use the authentication callout as a logging tool 22-10
Use the authentication callout as authentication tool 22-11
Authentication CLI command 22-12
Custom administrator authentication callout 22-14
Create a custom authentication callout 22-14
Enable the authentication callout for the Web Service 22-18

LDAP authentication tool

LDAP authentication tool overview 22-20
Invoke the LDAP command as an authentication tool 22-22
Enable the qip-ldapauth authentication callout for Tomcat 22-28
The qip-ldapauth command 22-29
The qip-ldapauth.pcy file 22-31

23 Secure message routes

Introduction to secure message routes 23-2

Port tunneling

Backward compatibility of remote servers 23-4

Secure message transport

Configuring SSL 23-8

Remote proxies

Cleartext proxy 23-11

Secure proxy 23-12

Promotion proxy 23-13

Demotion proxy 23-15

Login proxies

Dynamic updates to secure MS DNS zones

Dynamic update messaging without DNS Update Service 23-19

Dynamic update messaging using the DNS Update Service 23-21

QIP Update Service policy to support secure zone updates 23-22

Part V: Advanced DNS administration

24 Advanced DNS configurations

VitalQIP BIND support

Lucent DNS directives 24-3

BIND 9.x support 24-3

Manage Windows 2003 DNS servers

Configure Windows 2003 DNS server in VitalQIP 24-7

Windows 2003 DNS secure zones support

Background 24-10

Configure VitalQIP to manage Windows 2003 DNS secure zones 24-12

Set secure zone policies

Set secure zone policies at global level 24-17

Set secure zone policies at subnet level 24-18

Set secure zone policies at object level 24-19

ddns.conf with proxies for Windows 2003 secure servers 24-20

Secure dynamic updates 24-21

Lucent DNS server setup

Windows 2003 DC as KDC and Lucent DNS running on UNIX 24-26

Windows 2003 DC as KDC and Lucent DNS running on Windows 24-34

DNS client setup

VitalQIP DNS Update Service setup on UNIX 24-46

VitalQIP DNS Update Service setup on Windows 24-48

VitalQIP client setup on UNIX 24-48

VitalQIP client setup on Windows 24-50

Setup required if Lucent DNS Server is primary for the Windows Domain zone 24-51

Configure DNSSEC Support

About DNSSEC 24-56

DNSSEC userexit 24-57

Configure DNSSEC 24-58

External objects and resource records support

About external objects and resource records 24-61

How VitalQIP imports external object and resource records 24-62

How VitalQIP handles external objects and resource records 24-65

Modify external objects and resource records 24-67

About external object updates to DNS 24-68

Configure VitalQIP to collect external objects and resource records 24-69

Improve DNS push functionality

What happens with improved DNS push functionality 24-73

To improve DNS push functionality 24-75

Customize user exit scripts

Stages user exits occur 24-77

Rename user exit scripts 24-79

Report failure conditions 24-81

Sample UserExit.pcy file 24-81

25 Troubleshoot DNS

Common DNS problems and questions 25-2

DNS error messages 25-10

Fixing Bad Zones 25-12

Checking named.conf files 25-32

Part VI: Advanced DHCP administration

26 Advanced DHCP configurations

Support for Windows 2003 DHCP Servers

Managing Windows 2003 DHCP Servers 26-2

Support for DHCP Multi-NIC

Support for the DHCP Failover Server

Configuring the Primary DHCP Server 26-10

Configuring the failover DHCP server 26-13

Additional policies for the Lucent DHCP server

Adding additional Lucent DHCP policies 26-17

DHCPServer and DHCPSocketAddr policies 26-19

Unique hostname resolution options

FIRST-IN/LAST-IN methodology 26-21

QIP Update Service processing of DHCP lease updates 26-23

27 Troubleshoot DHCP

Common DHCP problems and questions 27-2

DHCP error messages 27-6

GL Glossary

IN Index

List of tables

1 Administrator Reference Manual changes 1-xx
2 Typographical conventionss 1-xxvi
3 Technical support information 1-xxvii
2-1 Checklist for services running on Windows 2-4
2-2 Configure services options 2-8
2-3 DNS Controller buttons 2-12
2-4 Event Viewer tab options and buttons 2-14
2-5 Checklist for services/daemons on UNIX 2-19
3-1 Debug policy values 3-44
3-2 Debug filename format characters 3-46
3-3 Java debug features 3-48
7-1 MessageQueue values 7-9
7-2 MessageRoute values 7-11
18-1 Tomcat configuration elements 18-23
18-2 Message levels 18-24
18-3 log4J parameters 18-25
18-4 Descriptions for showIcon<icon> properties 18-43
18-5 Sample callout settings 18-70
19-1 Function values 19-12
20-1 Required troubleshooting information 20-2
20-2 Common messages in system log or <server>.log files 20-5
20-3 Login error messages 20-11
20-4 Schedule Service signal types 20-13
20-5 Login Service signal types 20-14
20-6 Message Service signal types 20-14
20-7 QIP Update Service signal types 20-15
20-8 DNS Update Service signal types 20-15
20-9 Active Lease Service signal types 20-16
20-10 Remote Service signal types 20-16

20-11 Remote Service signal types 20-17
20-12 Schedule Service message types 20-18
20-13 Login Service message types 20-18
20-14 Message Service messages types 20-19
20-15 MS DNS Update Service message types 20-20
20-16 QIP Update Service messages types 20-21
20-17 DNS Update Service message types 20-23
20-18 Active Lease Service message types 20-24
20-19 Remote Service messages types 20-26
20-20 RMI Scheduler Service messages types 20-27
21-1 Database errors 21-2
21-2 Non-database error codes 21-15
21-3 Informational and error messages 21-18
22-1 Authentication CLI command parameters and arguments 22-12
22-2 Authentication callout arguments 22-15
23-1 Legacy port names and numbers 23-4
24-1 net user command parameters 24-22
24-2 ktpass parameters *24-22*
24-3 ktutil command parameters 24-24
24-4 GSS-TSIG configuration on Lucent DNS 4.2 servers 24-25
24-5 Secure dynamic DNS environment variables 24-47
24-6 Type of Duplicate resource record validation checking 24-66
24-7 User exit stages 24-77
24-8 File Generation Service user exit stages 24-77
25-1 Location of VitalQIP services executables 25-9
25-2 DNS Error Messages 25-10
26-1 Failover parameters for the primary DHCP server 26-11
26-2 Failover parameters for the secondary DHCP server 26-13
26-3 Additional Lucent DHCP server policies 26-18
26-4 Existing hostname 26-21
26-5 FIRST-IN hostname changes 26-22
26-6 LAST-IN existing hostname 26-22
26-7 LAST-IN object receives requested name 26-22

26-8 LAST-IN existing hostname 26-23
26-9 LAST-IN original static name remains the same 26-23
27-1 DHCP error messages 27-7

List of figures

1-1 Typical VitalQIP service configuration 1-9
1-2 Alternative VitalQIP service configuration 1-10
1-3 All VitalQIP components on a single server 1-12
1-4 Enterprise server, client, and remote services on different servers 1-13
1-5 Enterprise server, client, distributed services, and remote services on different servers 1-14
1-6 Single VitalQIP enterprise server with multiple remote servers and clients 1-15
1-7 Single VitalQIP enterprise server with multiple remote servers, clients, and distributed servers 1-16
18-1 Example of a pre-defined Add-On application 18-78
18-2 Example of the Links|Customized Links application 18-78
18-3 Example of sub-menus 18-79
22-1 Validation of passwords in VitalQIP 22-3
22-2 Validation of passwords with the Authentication Callout 22-4
22-3 Example of custom message 22-7
22-4 LDAP authentication tool process 22-20
23-1 Default non-secure message flow 23-3
23-2 Non-secure message transport tunneled through Message Service 23-5
23-3 Secure message transport tunneled through SSL Tunnel and Message Service 23-6
23-4 GUI deployment without co-located Message Service 23-7
23-5 Routing of push request through secure proxy to the remote server 23-12
23-6 Routing of push request through promotion proxy 23-14
23-7 Routing of push request through demotion proxy 23-16
23-8 Routing of login request through promotion proxy 23-18
23-9 Dynamic update message flow without DNS Update Service 23-20
23-10 Dynamic update message flow for MS-DNS secure zones using the DNS Update Service 23-21
24-1 Continuous Model 24-63
24-2 Polling Model 24-64

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 Protocol (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 2003 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

The following table lists the changes to the *Administrator Reference Manual*.

Table 1 Administrator Reference Manual changes

Issue	Feature name	Description	Feature impact
5	Rename user exit scripts	Clarification provided on allowable filenames for user exit scripts. Fixes VQIP00020632.	• "Rename user exit scripts" (p. 24-79)
5	Adding Additional Lucent DHCP Policies	Instructions for performing this function with the Web client have been clarified. Fixes VQIP00020718.	• "Adding additional Lucent DHCP policies" (p. 26-17)
5	Access Control	Access control has been added to the VitalQIP GUI.	• "VitalQIP Access Control policies" (p. 16-1) • "PopulateAcSubscriberHierarchy" (p. 18-47) • "PopulateAcDeviceHierarchy" (p. 18-47)
5	DNS User Exits	Two new user exits have been added to assist in troubleshooting DNS problems.	• "Fixing Bad Zones" (p. 25-12) • "Checking named.conf files" (p. 25-32)
5	DNSSEC Support	DNSSEC Configuration instructions have been added.	• "Configure DNSSEC Support" (p. 24-56)
4	New location for named and utilities	The location for where named and associate utilities has changed to from *usr/sbin* to *$QIPHOME/usr/bin*.	• "Start DNS services manually" (p. 2-21) • "Sample qip.pcy file" (p. 3-3) • "DNSBinsDir" (p. 12-3) • "BIND 9.x support" (p. 24-3) • "To improve DNS push functionality" (p. 24-75)
4	New GUI properties	New properties have been added to the *qip.properties* file.	• "xmlrpc.client.timeout.short" (p. 18-35) • "xmlrpc.client.timeout.medium" (p. 18-36) • "xmlrpc.client.timeout.long" (p. 18-36)
3	User exit policies for the RMI QAPI Service	New user exit policies were added to the RMI QAPI Service.	• "Sample qip.pcy file" (p. 3-3)"FailOnFailedUserExit" (p. 13-4) • "UserExitErrorFile" (p. 13-4)
2	Show or hide icons	Properties were added that can show or hide icons from the toolbar.	• "PopulateAcSubscriberHierarchy" (p. 18-47) • "Sample qip.properties file" (p. 18-47)

Issue	Feature name	Description	Feature impact
2	Administrator management	A new policy was added to verify Managed Lists.	• "Sample qip.pcy file" (p. 3-3) • "backgroundOnly" (p. 17-8) • "Sample qip.properties file" (p. 18-47)
2	LDAP authentication tool changes	Updates were made to SSL encryption of LDAP authentication tool **`qip-ldapauth`** and *qip-ldapauth.pcy* file.	• "Encrypted connections with SSL/TLS" (p. 22-23) • "DeferUsers" (p. 22-34) • "DeferUsersFile" (p. 22-34) • "DnList" (p. 22-34) • "DnListFile" (p. 22-35)
2	New troubleshooting issue	A new troubleshoot issue was added for VitalQIP services.	• "Communication problems for VitalQIP service" (p. 20-33)
2	DHCP functionality in the Web client	DHCP functionality was added to the Web client.	• Chapter 26, "Advanced DHCP configurations"
2	Properties to control number of items to appear on page	New properties were added to the *qip.properties* file to control the number of items that appear on a search page for: • Address reclaims • Managed objects • Zone searches • Object Information report • Subnet Information report • Node searches • IPv4 object searches • Resource Record searches • Global searches • Quick View features	• "reclaimAddressPageSizes" (p. 18-40) • "reclaimNetworkStatusPageSizes" (p. 18-40) • "manageObjectsPageSizes" (p. 18-40) • "manageSubnetsPageSizes" (p. 18-40) • "zoneSearchPageSizes" (p. 18-40) • "PopulateAcSubscriberHierarchy" (p. 18-47)
2	Properties to set default vales fo DHCP parameters	New properties were added to the *qip.properties* file tthat set the default values to several DHCP parameters.	• "PopulateAcSubscriberHierarchy" (p. 18-47) • "dhcpServer.PingDelay" (p. 18-45) • "dhcpServer.FailoverServerType" (p. 18-45) • "dhcpServer.FailoverServerType" (p. 18-45) • "dhcpServer.ManagedRange" (p. 18-45)

Issue	Feature name	Description	Feature impact
2	Username Token	Username Token support was added.	• "Address Allocator Service policies" (p. 17-3)
2	Removed **backup_qip_dat** and **backup_qip_log** scripts	The **backup_qip_dat** and **backup_qip_log scripts** are no longer available for use.	• "VitalQIP database backup with VitalQIP scripts" was removed
2	**Add-ons** menu name change	The **Add-Ons** menu name changed to **Links**.	• "Access external applications from the web client toolbar" (p. 18-78)
1	New properties	New properties were added to the *qip.properties* file for: • Specify the version of AutoDiscovery integrated into the web client • Appliance Manager Software integration in the web client • Management of file sizes and file tranfers in the web client • Enabling VitalQIP 7.1 Web Service API	• "auto_discovery.version" (p. 18-37) • "appliance_manager.url" (p. 18-36) • "appliance_manager.version" (p. 18-36) • "folderSize" (p. 18-39) • "runtime.log.logsystem.class" (p. 18-42) • "reportFileTransferLimit" (p. 18-41) • "PopulateAcSubscriberHierarchy" (p. 18-47) • "managedFileDisplaySize" (p. 18-46) • "PopulateAcSubscriberHierarchy" (p. 18-47) and "To enable compatibility with VitalQIP 7.0/7.1 Web Service" (p. 18-69)
1	Changes to policies	New policies were added to the *qip.pcy* file for: • The Message Service to configure reconnection timeouts, UDP buffer size, and flush periods for the disk queue • The VitalQIP Schedule Service to reread the *.Lic* file • My Views to manage the number of Personal Views The **ErrorRedirectionToken** policy was removed from the [VitalQIP Remote Service] section.	• "ReConnectTimeout" (p. 7-5) • "UDPBufferSize" (p. 7-6) • "QueueFlushPeriod" (p. 7-7) • "LicenseReadInterval" (p. 9-3) • "MaxPersonalViews" (p. 17-6) • "MaxInfrastructureInstances" (p. 17-6) • "AssignedInfraTypeOnlyInAddPersonalViews" (p. 17-6) • "VitalQIP Remote Service policies" (p. 12-2)

Issue	Feature name	Description	Feature impact
1	VitalQIP Services	Revised description of VitalQIP MS DNS Update Service. Fixes VQIP00017741.	"VitalQIP MS DNS Update Service" (p. 1-7)
1	Dynamic updates to secure MS DNS zones	Revised the Note in Use DNS Update Service policy section. Fixes VQIP00017740.	"Dynamic updates to secure MS DNS zones" (p. 23-19)
1	Defining a Windows 2003 DNS server	Description of how to create a Windows DNS server in the web client UI was added.	"Configure Windows 2003 DNS server in VitalQIP" (p. 24-7)
1	Additional DHCP Server policies	Description details were added for DHCPServer and SiAddr policies.	Table 26-3, "Additional Lucent DHCP server policies" (p. 26-18)

How to use this information product

This manual is organized as follows:

Part I: "System planning"	Chapter 1, "Plan and configure your network"	This chapter provides information on planning and configuring your VitalQIP network.
Part II: "VitalQIP services administration"	Chapter 2, "Manage VitalQIP services"	This chapter describes the VitalQIP services and how to start and stop them.
	Chapter 3, "qip.pcy file management"	This chapter describes the VitalQIP policy file.
	Chapter 4, "VitalQIP QIP Update Service policies"	This chapter describes the QIP Update Service policies.
	Chapter 5, "VitalQIP DNS Update Service policies"	This chapter describes the DNS Update Service policies.
	Chapter 6, "VitalQIP MS DNS Update Service policies"	This chapter describes the MS DNS Update Service policies.
	Chapter 7, "VitalQIP Message Service policies"	This chapter describes the Message Service policies.
	Chapter 8, "VitalQIP Login Service policies"	This chapter describes the Login Service policies.
	Chapter 9, "VitalQIP Schedule Service policies"	This chapter describes the VitalQIp Schedule Service policies.

	Chapter 10, “VitalQIP Active Lease Service policies”	This chapter describes the Active Lease Service policies.
	Chapter 11, “VitalQIP MS DHCP Monitor Service policies”	This chapter describes the MS DHCP Monitor Service policies.
	Chapter 12, “VitalQIP Remote Service policies”	This chapter describes the Remote Service policies.
	Chapter 13, “VitalQIP RMI QAPI Service policies”	This chapter describes the RMI QAPI Service policies.
	Chapter 14, “VitalQIP RMI Scheduler Service policies”	This chapter describes the RMI Schedule Service policies.
	Chapter 15, “VitalQIP SSL Tunnel Service policies”	This chapter describes the SSL Tunnel Service policies.
	Chapter 17, “Web client related policies”	This chapter describes the Vital QIP Service Layer, Address Allocator Service, VitalQIP Web GUI, and MyView Service policies.
Part III: “VitalQIP system administration”	Chapter 18, “Web client configuration”	This chapter covers VitalQIP web client configuration.
	Chapter 19, “Database and server administration”	This chapter covers database administration mainly for Sybase databases. Oracle databases must be maintained by an onsite Oracle database administrator. A server section describes the order for stopping and starting services on the enterprise and remote servers.
	Chapter 20, “Troubleshoot VitalQIP”	This chapter offers advice for resolving problems with VitalQIP.
	Chapter 21, “General informational and error messages”	This chapter contains common database error messages, service error messages, and information messages.
Part IV: “Security administration”	Chapter 22, “Authenticate administrators”	This chapter describes how to use the administrator authentication callout tool.
	Chapter 23, “Secure message routes”	This chapter provides detailed information on how to set up secure message routes.
Part V: “Advanced DNS administration”	Chapter 24, “Advanced DNS configurations”	This chapter describes advanced DNS configurations, such as support of external objects, BIND, and so on.
	Chapter 25, “Troubleshoot DNS”	This chapter offers advice for resolving problems with DNS.

Part VI: “Advanced DHCP administration”	Chapter 26, “Advanced DHCP configurations”	This chapter describes advanced DHCP configuration, such as failover servers, Windows domain controller, and so on.
	Chapter 27, “Troubleshoot DHCP”	This chapter offers advice for resolving problems with DHCP.

Conventions used

The following table lists the typographical conventions used throughout this manual.

Table 2 Typographical conventionss

Convention	Meaning	Example
Trebuchet bold	Names of items on screens.	Select the **Client** check box.
	Names of buttons you should click on the screen, or names of keys on the keyboard to be pressed.	Click **OK**.
Courier	Output from commands, code listings, and log files	`# Name: Share shared-network _200_200_200_0`
Courier bold	Input that you should enter from your keyboard.	Run the following command: **`c:\setup.exe`**
	Names of commands and routines	The **`qip_getappllst`** routine returns the entire list of existing applications.
Courier bold italic	Input variable for which you must substitute another value. The angle brackets also indicate the value is a variable.	**`isql -U sa -P <sa_password>`**
Times bold	Uniform Resource Locators (URLs)	The VitalQIP product site can be found at **http://www.alcatel-lucent.com/wps/portal/products/**.
Times italics	Manual and book titles.	Refer to the *VitalQIP User's Guide*.
	Directories, paths, file names, and e-mail addresses.	A symbolic link must be created from */etc/ named.conf* that points to *named.conf*.
Times bold italic	Emphasis	***Read-only***. The name of the service element.

Related information

The following documents are referenced in this manual:

- *VitalQIP Installation Guide* (part number: 190-409-043R7.2)

  This guide describes how to install the VitalQIP product.
- *VitalQIP Command Line Interface User's Guide* (part number: 190-409-044R7.2)

  This guide discusses and describes how to use the VitalQIP Command Line Interface.

- *VitalQIP User's Guide* (part number: 190-409-068R7.2)

  This guide describes how to set up and use the VitalQIP user interface on Linux, UNIX and Windows platforms.
- *VitalQIP Web Client User's Guide* (part number: 190-409-079R7.2)

  This guide describes how to use the web client interface.

Product Training Support

Alcatel-Lucent University offers cost-effective educational programs that support the VitalQIP product. 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 contact us at 1 888 LUCENT8, option 2, option 2.

Technical support

If you need assistance with VitalQIP, you can contact the Welcome Center for your region. Contact information is provided in the following table.

Table 3 Technical support information

Region	Address	Contact information
North, Central, and South America	Alcatel-Lucent 400 Lapp Road Malvern, PA 19355 USA	Phone: 1-866-LUCENT8 (582-3688) Option 1, Option 2 Web: **https://support.lucent.com**
Europe, Middle East, Africa, and China	Alcatel-Lucent Voyager Place Shoppenhangers Road Maidenhead Berkshire SL6 2PJ UK	Phone: 00 800 00 LUCENT or +353 1 692 4579 E-mail: *emeacallcenter@alcatel-lucent.com* Web: **https://support.lucent.com**

Region	Address	Contact information
Central and South America	Alcatel-Lucent Calle 10, No. 145 San Pedro de los Pinos, 01180 Ciudad de Mexico Mexico	Mexico 01 800 123 8705 or (52) 55 5278 7235 Brazil 0800 89 19325 or (55) 193707 7900 Argentina 0800 666 1687 Venezuela 0 800 1004136 Costa Rica 0800-012-2222 or 1800 58 58877 For other local CALA numbers, consult the web site **https://support.lucent.com** or contact your local sales representative.
Asia Pacific	Alcatel-Lucent Australia 280 Botany Road Alexandria NSW 2015 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@alcatel-lucent.com*

How to order

Customers can order additional VitalQIP manuals online at **https://support.lucent.com**.

How to comment

To comment on this document, go to the Online Comment Form (**http://www.lucent-info.com/comments/**) or e-mail your comments to the Comments Hotline (**comments@alcatel-lucent.com**).

Part I: System planning

Overview

Purpose

In Part I, you will find information on planning your network, and best practices. Refer to Part I whenever you are setting up your system.

Contents

Part I contains the following chapters:

Chapter 1, “Plan and configure your network”	1-1

1 Plan and configure your network

Overview

PurposeDRAFT 2

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 and planning	1-2
VitalQIP components	1-2
VitalQIP services	1-3
VitalQIP service configurations	1-9
Plan your network	1-11
Installation configurations	1-12
Single server scenario	1-12
Single enterprise and remote server scenario	1-12
Single enterprise, remote, and distributed server scenario	1-13
Single enterprise server, multiple remote servers, and multiple clients scenario	1-14
Single enterprise server, multiple remote servers, multiple distributed servers, and clients scenario	1-16

VitalQIP components and planning

Before beginning the planning and configuration of your VitalQIP system, become familiar with the VitalQIP components. It will help in your decisions for planning your configuration.

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 **VitalQIP server** or **enterprise server** 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 **VitalQIP remote server** or **remote server** 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 **distributed server** is used for servers that run across independent systems. Refer to "VitalQIP services" (p. 1-3) for information about distributed services.

VitalQIP Web Client Interface

The browser-based **VitalQIP web client** interface or **web client** is completely new in VitalQIP 7.0 and includes network allocation features, IPv4 and IPv6 address management, as well as node management and enhanced user-defined attribute management. The web client can be installed on Windows or UNIX platforms.

VitalQIP client interface

The **VitalQIP client** or **VitalQIP graphical user interface** (GUI) refers to the system 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 **VitalQIP Command Line Interface** (CLI) provides an alternative process to using the VitalQIP GUI. Instead, VitalQIP functions are executed from a command line window. Most CLIs are documented in the *Command Line Interface User's Guide*, although the **`qip-cli`** allocation commands are located in the *VitalQIP Web Client User's Guide*.

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" (p. 3-2) for more information on policies.

For UNIX platforms, a **daemon** is the equivalent of a Windows service. Refer to "VitalQIP services on UNIX" (p. 2-17) 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, refer to 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).

> **Note:** 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, refer to 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 dynamic updates for secure zones on Active-Directory integrated MS-DNS servers. If the DNS Update service sees an update for a zone that is secure ***and*** on a Microsoft DNS server, it will forward the update to the MS DNS Update Service on the remote. The MS DNS Update service will then attempt to make the update via GSS-TSIG using the credentials stored in the sec.dat file that was generated during the last push. It will fall back to dnscmd if several GSS-TSIG updates attempts fail, though there are some exceptions to this.

Also, the VitalQIP MS DNS Update Service handles DNSDeltaNotification messages. These messages are received when a DNS Generation with "Changed Records Only" is done to an MS DNS Server (secure zones or not). The message triggers the MS DNS Update Service to read the add and delete files that are generated by the push. The service then puts the changes into DNS.

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.

> **Note:** 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 multiple threaded 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. Refer to "qapi - VitalQIP RMI QAPI Service daemon" (p. 2-31) 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.

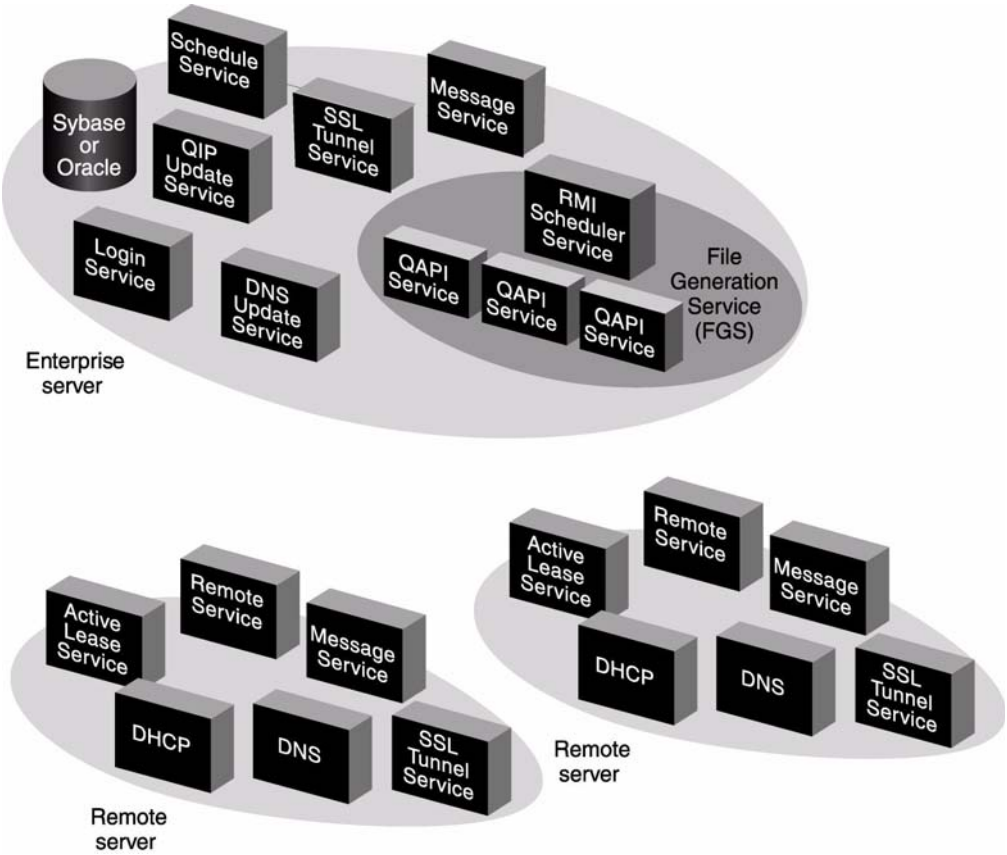
Lucent DNS Service (named)

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. The following illustration shows how services are typically configured for VitalQIP.

Figure 1-1 Typical VitalQIP service configuration

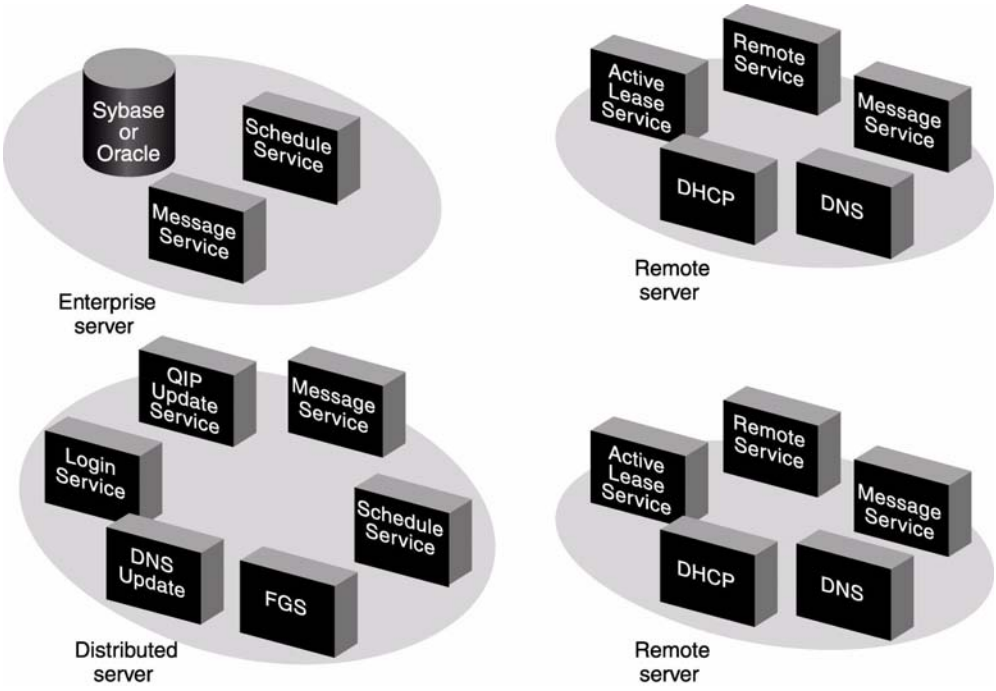


In Figure 1-1, the Schedule Service, Login Service, DNS Update Service, QIP Update Service, File Generation Service, Message Service, SSL Tunnel Service, and database reside on the VitalQIP enterprise server. The remote server has the Active Lease Service, Remote Service, DHCP, DNS, Message Service, and SSL Tunnel Service installed on it.

Alternative VitalQIP service configuration

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. The following illustration shows how services can be alternatively configured for VitalQIP using a distributed server.

Figure 1-2 Alternative VitalQIP service configuration



In Figure 1-2, 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.

Note: Figure 1-2 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. Refer to "Installation configurations" on page 12 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 (including Linux). The VitalQIP remote server can also be either Windows or UNIX servers, 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 12.

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.

> **Note:** To determine system requirements for specific platforms and components, refer to the *VitalQIP Release Notes*.

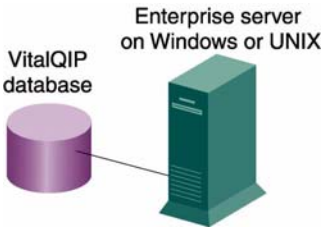
Installation configurations

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, refer to the *VitalQIP Installation Guide*.

Single server scenario

The following illustration 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 1-3 All VitalQIP components on a single server



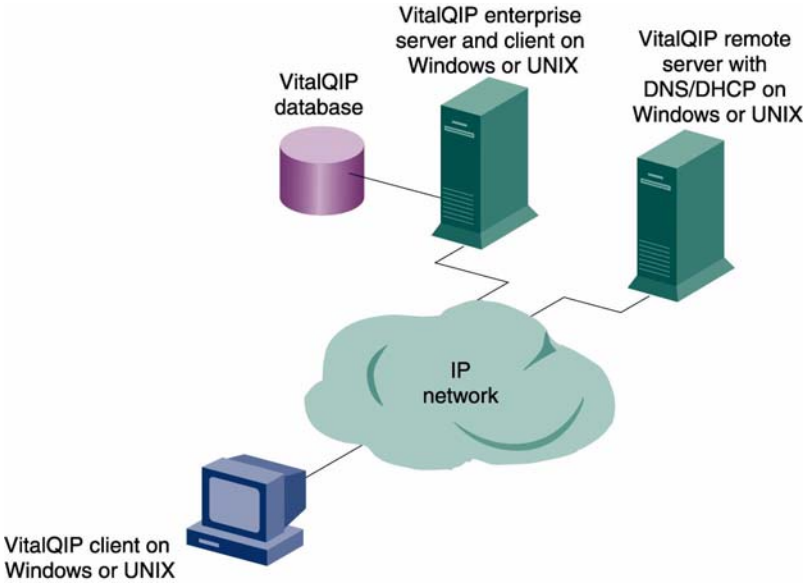
The product components required are as follows:

- 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

The following illustration 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.

Figure 1-4 Enterprise server, client, and remote services on different servers



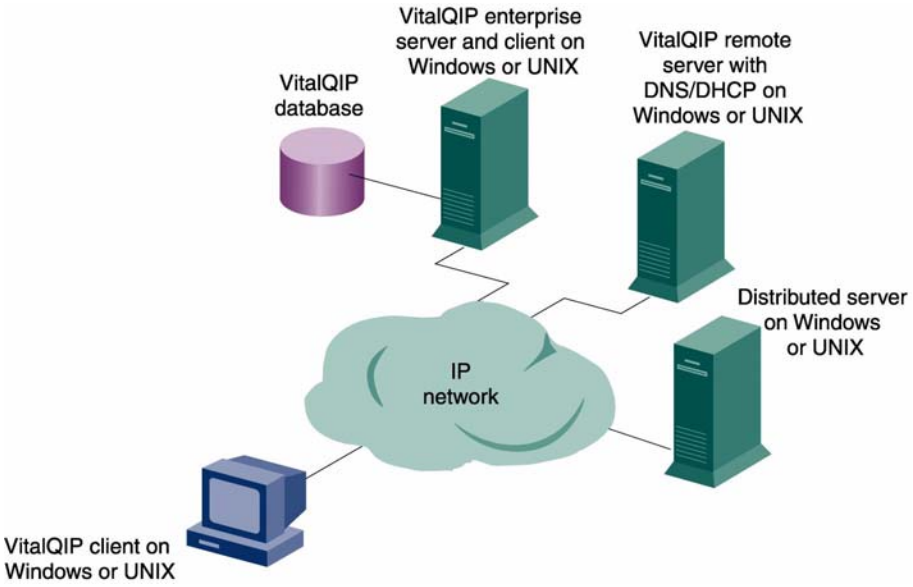
The product components required are as follows:

- 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

The following illustration 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.

Figure 1-5 Enterprise server, client, distributed services, and remote services on different servers



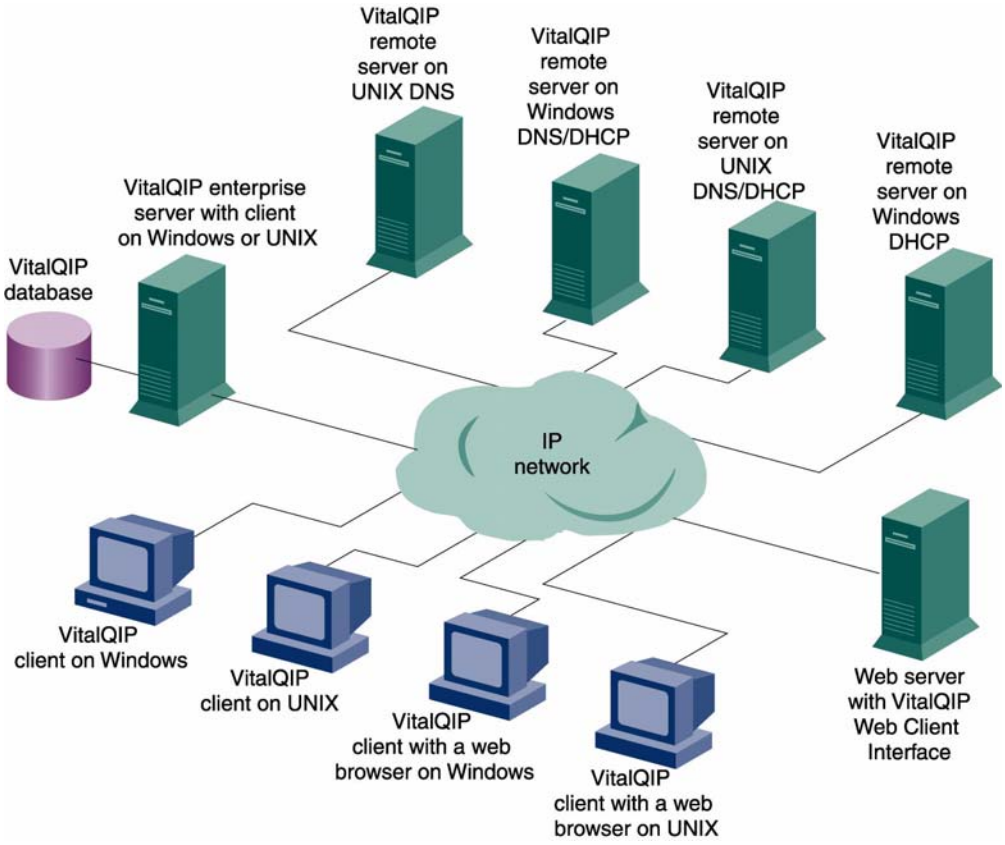
The product components required are as follows:

- 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

The following illustration 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.

Figure 1-6 Single VitalQIP enterprise server with multiple remote servers and clients



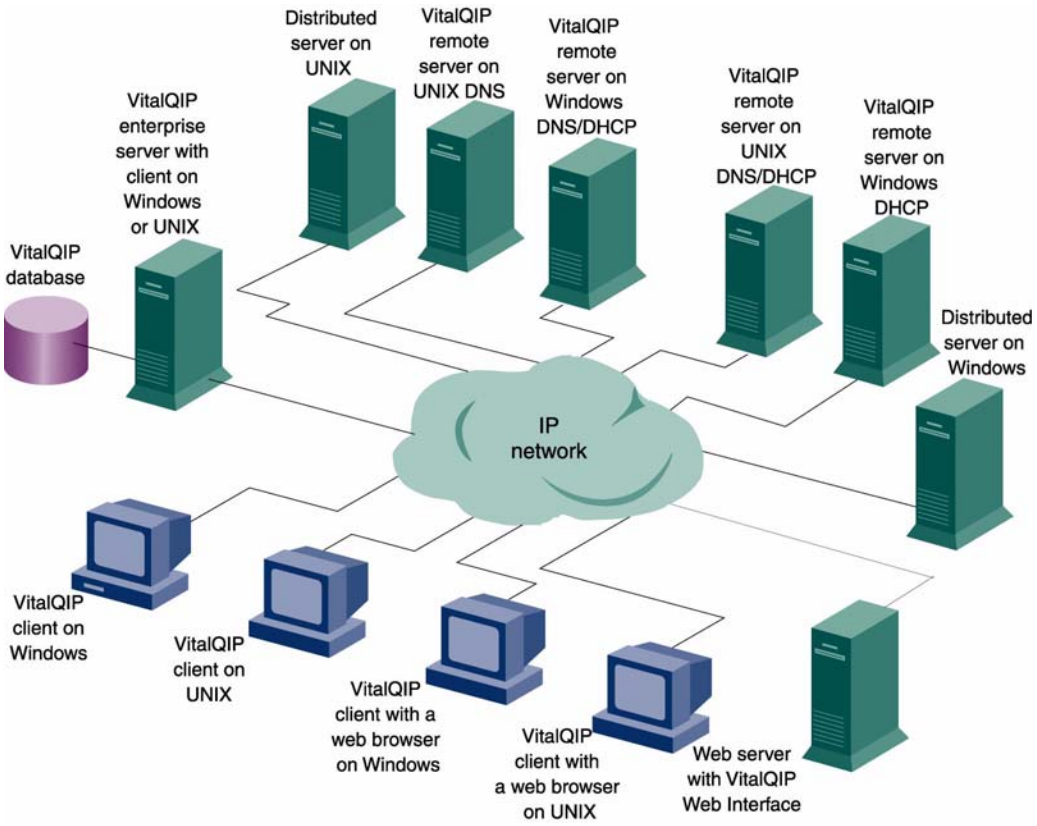
The product components required are as follows:

- 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

The following illustration 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.

Figure 1-7 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: VitalQIP services administration

Overview

Purpose

In Part II, you will find information on managing VitalQIP services. The VitalQIP services includes such things starting and stopping services and managing policies.

Contents

Part II contains the following chapters:

Chapter 2, “Manage VitalQIP services”	2-1
Chapter 3, “qip.pcy file management”	3-1
Chapter 4, “VitalQIP QIP Update Service policies”	4-1
Chapter 5, “VitalQIP DNS Update Service policies”	5-1
Chapter 6, “VitalQIP MS DNS Update Service policies”	6-1
Chapter 7, “VitalQIP Message Service policies”	7-1
Chapter 8, “VitalQIP Login Service policies”	8-1
Chapter 9, “VitalQIP Schedule Service policies”	9-1
Chapter 10, “VitalQIP Active Lease Service policies”	10-1
Chapter 11, “VitalQIP MS DHCP Monitor Service policies”	11-1
Chapter 12, “VitalQIP Remote Service policies”	12-1
Chapter 13, “VitalQIP RMI QAPI Service policies”	13-1
Chapter 14, “VitalQIP RMI Scheduler Service policies”	14-1
Chapter 15, “VitalQIP SSL Tunnel Service policies”	15-1
Chapter 17, “Web client related policies”	17-1

2 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.

VitalQIP services on Windows	2-3
Services checklist for Windows	2-3
VitalQIP Service Controller	2-6
Run services with the VitalQIP Service Controller	2-7
Configure the VitalQIP Service Controller	2-8
Lucent DNS Service options	2-11
Diagnose problems with the Event Viewer tab	2-13
Use Filtered Services	2-15
To start or stop the Tomcat server on Windows	2-16
VitalQIP services on UNIX	2-17
Services checklist for UNIX	2-17
VitalQIP startup scripts	2-20
Start DNS services manually	2-21
To start or stop the Tomcat server on UNIX	2-22

Start and stop daemons	2-22
qipd - VitalQIP Schedule Service daemon	2-23
qip-qipupdated - VitalQIP QIP Update Service daemon	2-24
qip-msgd - VitalQIP Message Service daemon	2-25
qip-logind - VitalQIP Login Service daemon	2-26
qip-dnsupdated - VitalQIP DNS Update Service daemon	2-27
qip-netd - VitalQIP Active Lease Service daemon	2-28
qip-rmtd - VitalQIP Remote Service daemon	2-29
qip-rmisched - VitalQIP RMI Scheduler Service daemon	2-30
qapi - VitalQIP RMI QAPI Service daemon	2-31
qip-ssltd - VitalQIP SSL Tunnel Service daemon	2-32
named - Lucent DNS Service daemon	2-33
dhcpd - Lucent DHCP Service daemon	2-34

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" (p. 2-3).

Note: VitalQIP services contain policies that affect the behavior of the services. For information on the policy files for VitalQIP services, refer to "Manage the VitalQIP policy file" (p. 3-2).

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
  6. Tomcat Server
- The following table displays a checklist of all services to ensure that you are starting the correct VitalQIP services on your network. The following 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 Alcatel-Lucent. The following abbreviations are used:
  - E/S - enterprise server
  - D/S - distributed server
  - R/S - remote server

  Note: 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 2-1 Checklist for services running on Windows

Type of service	Services	Where?
LUCENT DNS	Lucent DNS 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	R/S
	VitalQIP Remote Service	E/S, D/S
	VitalQIP RMI Scheduler Service	E/S, D/S
	VitalQIP QIP Update Service	E/S, D/S
LUCENT DHCP	Lucent 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 Schedule Service	R/S
	VitalQIP Remote Service	E/S, D/S
	VitalQIP RMI Schedule Service	E/S, D/S
	VitalQIP QIP Update Service	E/S, D/S
Microsoft 2000 DHCP	MS 2000 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	E/S
	VitalQIP Schedule Service	R/S
	VitalQIP Remote Service	E/S, D/S
	VitalQIP RMI Scheduler Service	E/S, D/S
	VitalQIP QIP Update Service	E/S, D/S

Type of service	Services	Where?
Microsoft 2000 DNS	MS 2000 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
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 Tomcat 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

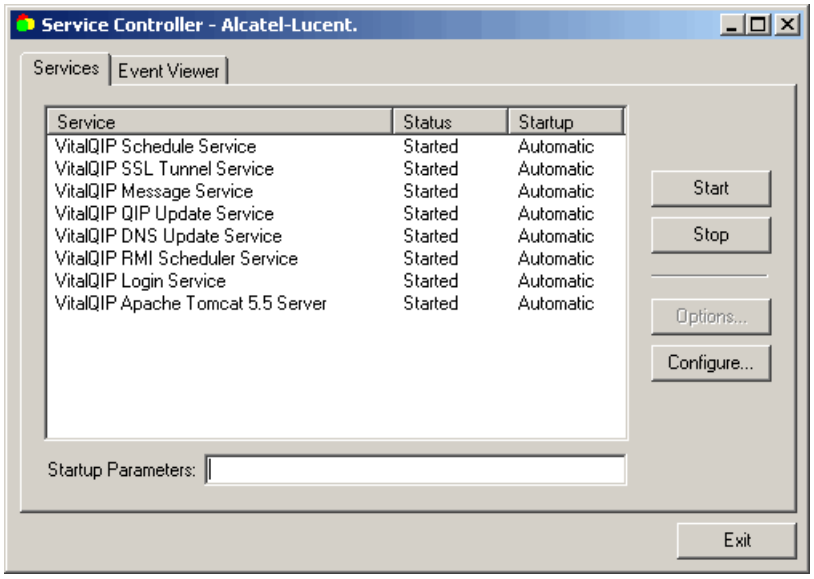
Purpose

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.



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 Parameters** 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

Purpose

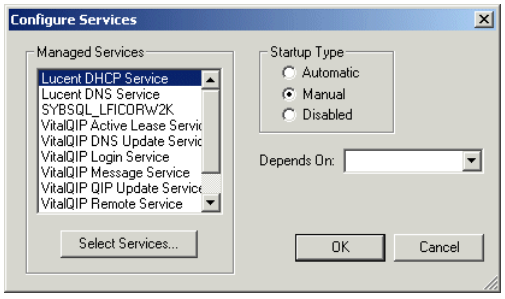
The VitalQIP Service Controller can be configured to start services automatically when a server starts or to be started manually by a user. Services can be disabled or made dependent upon other services.

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.



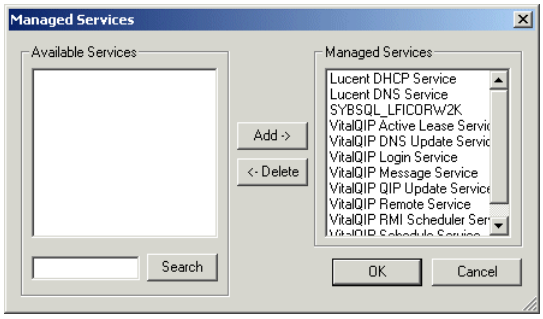
3 In the Configure Service Controller, select a service from the **Managed Services** list. The following table describes the options.

Table 2-2 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.

Option	Description
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.



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

Purpose

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

Procedure

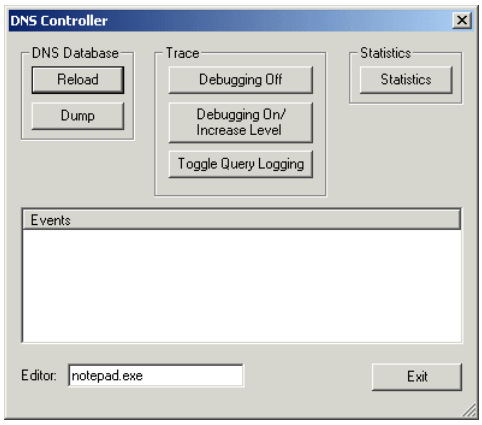
To use the Lucent DNS Service options, follow these steps:

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.



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

Table 2-3 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, refer to "The debug policy" (p. 3-44), or *DNS and BIND* 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*, 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.

END OF STEPS

Diagnose problems with the Event Viewer tab

Purpose

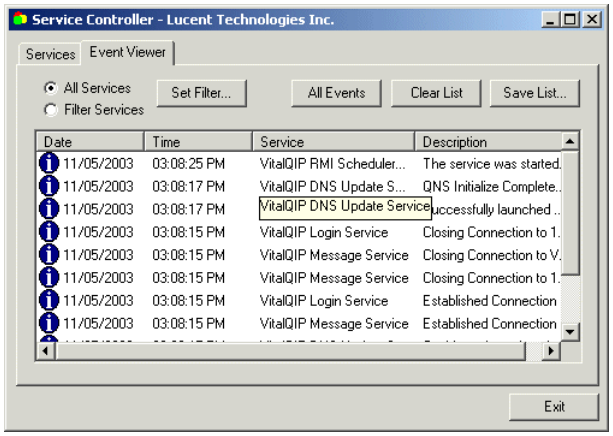
The **Event Viewer** tab is useful for diagnosing a potential problem. It is the first thing that should be reviewed to determine the problem.

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.



3. Select the function you want to perform by clicking the appropriate button. The following table describes the options and buttons.

Table 2-4 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. Refer to the next section, "Use Filtered Services" (p. 2-15).
Set Filter…	Manages the list of services that appear in the Event Viewer.
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.

END OF STEPS

Use Filtered Services

Purpose

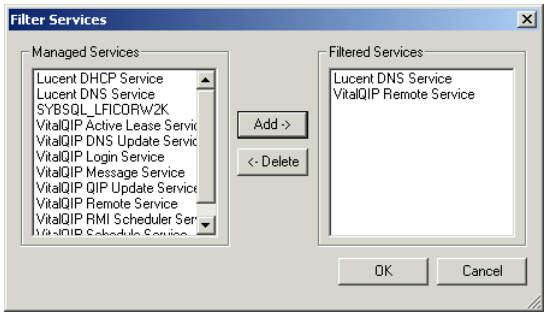
Filtered Services allows you to limit the messages displayed in the **Event Viewer** tab. You can display messages for one service or several services.

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.



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

To start or stop the Tomcat server on Windows

Purpose

Use this procedure to start and stop the Tomcat server on a Windows platform

Procedure

To start or stop the Tomcat server, follow these steps:

1. Open the Windows Service Controller.
2. To start the service, right-click on **VitalQIP Apache Tomcat 5.5 Server** and select **Start**.
3. To stop the service, right-click on **VitalQIP Apache Tomcat 5.5 Server** and select **Stop**.

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.20080510.1130* (or *cshrc.20080510.1130*) 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 prior to installing VitalQIP, so that you can modify the new environment variable file after installation.

  Note: 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):

Bourne shell

```
# . ./shrc
```

C shell

```
# source cshrc
```

  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. Refer to 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
  - Tomcat Server
- The following table 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 Alcatel-Lucent.

  Note: 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 your enterprise server.

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

Table 2-5 Checklist for services/daemons on UNIX

Type of service	Services/daemons	Location
LUCENT DNS	Lucent DNS Service (**named**) 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 Scheduler Service (**qip-rmisched**) VitalQIP QIP Update Service (**qip-qipupdated**)	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
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

Type of service	Services/daemons	Location
IBM DHCP	IBM DHCP Service (**dhcpsd**)*	R/S
	VitalQIP Active Lease Service (**qip-netd**)	R/S
	VitalQIP IBM DHCP Monitor Service (**qip-ibmdhcpd**)	R/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

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 DNS services manually

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

$QIPHOME/usr/bin

Note: A symbolic link must be created for the *named.conf* and *rndc.conf* files:

– For the *named.conf* file, it must point from the */etc/named.conf* file to the <*push_dir*>*/named.conf* file. Alternatively, the **-c** option must be used to specify the location of the startup file. Refer to "named - Lucent DNS Service daemon" (p. 2-33) for more information on the **-c** option.

– For the *rndc.conf* file, it must point from the */etc/rndc.conf* file to the <*push_dir*>*/rndc.conf* file.

To start or stop the Tomcat server on UNIX

Purpose

Use this procedure to start and stop the Tomcat server on a UNIX platform.

Procedure

To start or stop the Tomcat server, follow these steps:

1 Be sure your environment variables are set.

2 Go to $QIPHOME:

- To start the Tomcat instance:

`./startTomcat.sh`

- To stop the Tomcat instance:

`./stopTomcat.sh`

On startup, Tomcat reads *$QIPHOME/tomcat/conf/server.xml* for configuration information (port numbers, SSL, client connection and other operational parameters).

3 Check if the Tomcat server is running by entering the following command:

`ps -ef|grep apache`

A response similar to the following appears:

```
root 24707 1   0 11:22:26 pts/4 10:33 /opt1/qiprsweb/jre/bin/java
   -Djava.util.logging.manager=org.apache.juli.Class
```

END OF STEPS

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 the *qip.pcy* file, refer to Chapter 3, “qip.pcy file management”.

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**:

Note: 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**.

-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. Refer to "The debug policy" (p. 3-44) for information on VitalQIP Services debugging.
-f	If specified, a child process is not forked when the server starts. 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**, refer to 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.

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 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**:

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

-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. Refer to "The debug policy" (p. 3-44) 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.

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. Refer to "The debug policy" (p. 3-44) 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. Refer to "The debug policy" (p. 3-44) 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. Refer to "The debug policy" (p. 3-44) 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. Refer to "The debug policy" (p. 3-44) 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. Refer to "The debug policy" (p. 3-44) 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. Refer to "The debug policy" (p. 3-44) 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.

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.

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.

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. Refer to "The debug policy" (p. 3-44) 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.

Note: 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.

3 qip.pcy file management

Overview

Purpose

This chapter contains information about how to manage VitalQIP *qip.pcy* file.

Contents

This chapter presents the following topics.

Manage the VitalQIP policy file	3-2
VitalQIP services policies	3-3
Sample qip.pcy file	3-3
Common policies	3-44
The debug policy	3-44
Java debug policies	3-47
Global section	3-49

Manage the VitalQIP policy file

During the installation, a policy file called *qip.pcy* is generated. The file contains policies for all installed services and some features. The following services and features 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**)
- QIPAuthCallout
- VitalQIP Schedule Service (**qipd**)
- VitalQIP Active Lease Service (**qip-netd**)
- VitalQIP Domain Controller Logon Audit Service (**qip-dclas**)
- VitalQIP Kerberos Domain Controller Logon Audit Service (**qip-kdclas**)
- Lucent TFTP Service (**luc-tftpd**)
- VitalQIP Microsoft DHCP Monitor Service (**MSDHCPMonitorService**)
- VitalQIP IBM DHCP Monitor Service (**qip-ibmdhcpd**)
- VitalQIP Remote Service (**qip-rmtd**)
- VitalQIP RMI QAPI Service (**qip-qapi**)
- VitalQIP RMI Scheduler Service (**qip-rmisched**)
- VitalQIP SSL Tunnel Service (**qip-ssltd**)
- VitalQIP Service Layer
- Address Allocator Service
- VitalQIP Web GUI
- MyView
- Job Scheduler

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. Refer to the sample *qip.pcy* file in "Sample qip.pcy file" (p. 3-3) for examples.

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

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
```

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
;-------------------------------------------------------------------------
   ---
;
; Enable auditing for address allocation and V6 address management.
Auditing = true
User=qipman
Password=02d72a4adf44e3dca20f1e8e44be57a4454e
Server=labor641
LoginServer=10.100.25.60
;-------------------------------------------------------------------------
   ---

[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
; DenyConnectionList ACL.  To deny all access unless specifically allowed,
   use
; the AllowConnectionList ACL.  The AllowConnectionList 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 "derive", 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=derive
;
;-------------------------------------------------------------------------
   ---


[VitalQIP DNS Update Service]
DDNSGenerateSleep=3600

; 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

; This policy is only needed if scavenging is enabled on a
; SECURE zone and users do not want static object RRs to be scavenged.
; If the value is true, then any RRs in a SECURE zone that would be
; updated with the qip-strong principal will be updated using
; dnscmd instead of the API.
; The effect is that if scavenging is enabled for the zone,
; instead of being added dynamically (aka via the API) and thus
; ending up as a scavengable record, the RR will be added via
; dnscmd and thus NOT be scavenged.
;DisableScavengeStaticRRs = false
;-------------------------------------------------------------------------
   ---


[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

; This policy specifies the UDP buffer size.  If you think that messages are
; being dropped, try increasing this value.
; The default is 256K.
;UDPBufferSize = 262144
```

```
; 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

; This policy specified the number of milliseconds between flushes of the
   disk
; queue.  The on-disk queue will never be more than this number of
   milliseconds
; out-of-sync with the memory queue.
;QueueFlushPeriod=500

; 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

; When DisplaySelectMasks is set, the socket and masks specified to select
   are
; displayed in the debug file.  This can generate copious amounts of debug,
   use
; judiciously.
;DisplaySelectMasks=false

; When EnableDebugSocket is specified, sockets of accepted connections are
   put
; into debug mode by specifying the SO_DEBUG option on the socket.  Some
; operating systems allow tracing of sockets in debug mode.
;EnableDebugSocket=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.
;
; 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.B:0:DNS Update Service:VitalQIP DNS Update
   Service:10.1.2.3*20:VitalQIP DNS Update Service:10.1.2.4*20
;-------------------------------------------------------------------------
   ---


MessageRoute=DNSUpdateRR:A:0:QIP Update Service (Update RR):VitalQIP QIP
   Update Service:10.100.25.60
MessageRoute=DHCP:A:0:VitalQIP QIP Update Service - DHCP:VitalQIP QIP Update
   Service:10.100.25.60
MessageRoute=DNSUpdateRR:A:0:DNS Update Service (Update RR):VitalQIP DNS
   Update Service:10.100.25.60
MessageRoute=DNSUpdateObject:A:0:VitalQIP DNS Update Service -
   DNSUpdateObject:VitalQIP DNS Update Service:10.100.25.60
[VitalQIP Login Service]
QIP.labor641.qipadmin.Password=019e389d4a87123440a6ba897f2a7b5af5fc

; 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

; LicenseReadInterval specifies how often the .Lic file will be reread
; (and possibly rewritten) so that license changes can be applied
; This can be turned off completely by setting this value to zero.
;LicenseReadInterval = 86400

; 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]
GenDDNSConfOnPush=none
FileGenerationServer=10.100.25.60

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

; To specify a connection timeout when the remote attempts to connect to an
   FGS
; set FGSConnectTimeout to the number of seconds to wait for the connection
   to
; succeed/fail.  After this period, a connection to the backup server is
   attempted.
;FGSConnectTimeout=0

; To have communication to the FGS pass through the message service on the
   FGS, set
; CreateConduit to true.  This reduces the number of holes you will need to
   knock
; into your firewall.
;CreateConduit=false
```

```
; When CreateConduit=true, you can specify a list of Message Service proxies
   to use
; when attempting connections to the FGS message service. The list is space
   separated.
;Proxy=

; 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

; When UseFreezeThawUpdate is true, this directory will be the first searched
; when looking for rndc/journalprint commands.  {pushdir}/bins and
; {pushdir}/bin will also be searched.
;DNSBinsDir=$QIPHOME/usr/bin

; When pushing to a QIP 4.0 or a BIND 9.0 server, updates found in journal
   files
; are sent to the DNS Update Service and then sent to the DNS server of the
; pushed-to address (managed server) or the specified address.  Usefull for
; some multi-homed/NAT configurations.
;DnsServerAddress={push to IP Address}

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

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

; 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

; 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
```

```
; The maximum number of threads to start for DNS updaters.  This property
   only
; applies if UseFreezeThawUpdate is true.  Factors that should be considered
; when adjusting this are number of CPUs and probability that rndc commands
; will time-out.
;MaxDNSUpdateThreads = 5

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

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

; 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

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

; 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, 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

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

; 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

; The SSLContext configuration specifies the secure socket protocol
; implementation to use and must match the value configured for other
   services.
; Possible values include TLS and SSL.  AIX may require SSL to be used.
; You may want to place these in the global section to insure consistency.
;SSLContext = TLS
;KeyStore = JKS

; To reduce the likelihood of lost updates due to file generation time, set
; UseFreezeThawUpdate to true.  This will update zones in the DNS server
; incrementally by using the rndc freeze/thaw commands while updating the
; zone files with the latest serial numbers.
;UseFreezeThawUpdate=false

;-------------------------------------------------------------------------
   ---


[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 = qipkeystoreZ

;SSLContext = TLS
;KeyStore = JKS

; 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}

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

; After a user-exit executes, if the status was non-zero, the contents of
; the specified file are read.  If FailOnFailedUserExit is true, the contents
; are reported back to the GUI, if FailOnFailedUserExit is false, the
   contents
; are sent to the debug log.
;UserExitErrorFile=UserExit.error
;-------------------------------------------------------------------------
   ---


[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 =
;
;SSLContext = TLS
;KeyStore = JKS
;
; 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
;SSLContext = TLS
;KeyStore = JKS
;-------------------------------------------------------------------------
   ---

[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 =
;-------------------------------------------------------------------------
   ---


[VitalQIP Service Layer]
soapUrl=http://127.0.0.1:80/ws/services/VQIPWebService

; URL for Web Services.
; To disable, set the value to: bypass
;soapUrl =
;-------------------------------------------------------------------------
   ---

[Address Allocator Service]

; The following policies define the behaviour of Address Allocator Service
; These values are used for Block Allocation post-processing and for RIR
   Reporting service

; CliInputFileDir specifies the name of the diretory used for generating CLIs
   which are used
```

```
; as input for post-processing CLIs. If not specified, QIP will use the
   default temp directory
;CliInputFileDir=

; CliKeepGeneratedFile specifies whether we need to store a copy of the
   generated input files
; for CLI post-processing. A value true will store the files indefinitely.
; Should be false for production
;CliKeepGeneratedFile=false

; CliInputFileNamePrefix specifies the prefix name of the files that are
   generated for post processing
; CLI's.
;CliInputFileNamePrefix=CLI_SUBNET_

; SendEmailAlerts specifies whether we need to enable/disable email alerts
   during block allocation
; Setting it to a value "false" will disable the alerts. Setting it to any
   value other than false
; or disabling this value will enable the email alerts.
;SendEmailAlerts=true
;-------------------------------------------------------------------------
   ---

[VitalQIP Web GUI]
; The following policies define the behaviour of web GUI

; autoExpandPool specifies weather the parent pool should be automatically
   expanded in order to
; show the newly created child pool. The default value is ?true? but if the
   number of child
; pools is very large (tens of thousands of child pools) and  performance is
   impacted this value ; can be set to "false"
;autoExpandPool=true


[My View]
; This flag is used to determine if the check to determine whether a MyView
   (A) instance has  a
; Child View (B) which in turn includes the MyView instance (A). This check
   can be costly at run-time
; for peformance reasons, and it was determined to disable it in environments
   in case the check
; would slow down the system
;CircularMyViewCheck=true
```

```
; This flag is used to determine if count check is needed for infrastructure
   types when
; contents of a MyView are displayed.  The default value is true and
   therefore, by default,
; when an Admin logs in or when contents of MyView are to be displayed, a
   count is done for
; all supported infrastructure types and only infrastructure types with one
   or more items
; matching MyView criteria are displayed under MyView.  If the number of
   items in
; infrastructure types is very large and the performance of MyView Usage GUI
   is impacted,
; then this policy value can be set to false to disable empty infrastructure
   type check.
;ExcludeEmptyTypes=true

; This specifies the maximum number of Personal Views that an Administrator
   can create in the System
;
;MaxPersonalViews=5

; This specifies the maximum number of Static Instances per Infrastructure
   Type that an Administrator can
; add statically to a Personal View.

;MaxInfrastructureInstances=50

;
;In the Personal View -> Add Contents screen, if Administrators wish to see
   only those Infrastructure types
;that they have access to, then set this value to true. However, this will
   have a serious performance impact
;
;By default, this value is set to false and will  display all the
   Infrastructure types to improve performance
;
;
;AssignedInfraTypeOnlyInAddPersonalView=false

[Job Scheduler]
; Starting in the QIP 7.1 WEB UI, users have the ability to run jobs/tasks
; in the background or on a schedule.  Properties in this section
; control the behavior of the scheduler.

; How often (in milliseconds) should the scheduler check the database for new
   jobs.
; Default 120000 (2 minutes)
```

```
;checkInterval=120000

; How far in the future (beyond the checkInterval ) to get scheduled jobs.
; If the default check interval is 2 minutes, the scheduler will get jobs
; from the database which are scheduled to occur between NOW and 2 minutes
; and 30 seconds in the future. This is to ensure that no jobs are missed due
   to
; unsynchronized clocks or network latency.
; Default .25 (25% of the checkinterval time)
;jobReadAhead=.25


; Number of threads available for running scheduled and background jobs.
; If you have 5 threads and schedule 6 jobs to occur at the same time, the
   last job
; will have to wait for one of the other jobs to complete before running,
; Default =5.  This value is highly dependent on how the customer uses the
; Scheduling  feature. The upper limit of threads is dependent on available
; memory,number of processors and the type of job being run
;jobThreads=5


; Should the start time of jobs be randomized with in certain boundries.
; Default = true;
;randomizeStart=true

; Upper limit of the random offset (in milliseconds) applied to scheduled
   jobs
;(only applicable if randomizeStart=true)
;randomBoundry=15000

; Number of days to keep records of completed or failed jobs in the database.
; After the specified time limit the job records are deleted.
;retentionDays=60

; How often (in hours) should the scheduler look for jobs to purge from the
   database
;purgeCheckInterval=12

; Specify the hostname of the scheduler.  This will be displayed in the
   scheduler page
; to identify the server that ran the job.  By default this is set to
   localhost, and
; this will be replaced with the hostname as queried from the operating
   system.
;schedulerHostName=localhost
```

```
; Specify if by default the scheduler auto refresh check box on the
; Job Scheduler page should be checked.  Checked if true, Unchecked if false.
;autoRefreshDefault=false

; Specify if this scheduler should only run background jobs.  This parameter
; should be set to false for only one server.
;backgroundOnly=false
```

Common policies

The *qip.pcy* files has several policies that are used by most VitalQIP services. Common policies appear multiple times in the file under different service sections. There is also a Global section where policies are set for the entire *qip.pcy* file.

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.

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

The following table shows the values for the debug policy that can compose a string.

Table 3-1 Debug policy values

Function	Values
Debug level	The following values can be used for the 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.

Function	Values
List of application/library layers from which to display debug messages	The following values are the application/library layers: • **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	The following values are 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>`**.

> **Note:** 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 the following table.

Table 3-2 Debug filename format characters

Character	Description
%a	Abbreviated weekday name
%A	Full weekday name

Character	Description
%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
- RMI QAPI Service
- RMI Scheduler Service
- SSL Tunnel Service

DebugLevel only accepts a debug level, described as follows:

DebugLevel

Value: Default: None

Allowed: Debug | Info | Warning | Error | Critical | None

Description: 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 the following table can also be specified in the global section of the policy file.

Table 3-3 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.

Debug feature	Description
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.

The following policies are available:

Debug

Value: Default: None

Allowed: Refer to Table 3-1, "Debug policy values" (p. 3-44)

Description: This policy sets the global debug level and is passed to every service. Refer to "The debug policy" (p. 3-44) for more information.

CLIdebug

Value: Default: None

Allowed: Refer to Table 3-1, "Debug policy values" (p. 3-44)

Description: This policy sets the global debug level for CLIs and is passed to every CLI.

GUIdebug

Value: Default: None

Allowed: Refer to Table 3-1, "Debug policy values" (p. 3-44)

Description: This policy sets the global debug level for GUIs and is passed to every GUI.

LoginServer

Value: Default: 127.0.0.1

Allowed: IP address

Description: 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

Value: Default: qip-msgd=2468 | qip-dns=3119 | qip-qdhcp=2490

Allowed: Any port number

Description: 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.

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.

SecureIncoming, MessageService, Connections

Value: Default: False

Allowed: True | False

Description: 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 "Secure message routes" (p. 23-1).

Auditing

Value: Default: True

Allowed: True | False

Description: Determines if auditing is enabled or disabled for address allocation and IPv6 address management. This policy controls only address allocation and IPv6 address management. Because IPv4 subnet management is controlled by auditing rules in the VitalQIP client, IPv4 subnet reports and an Administrator Audit report of an IPv4 subnet are not disabled by this policy.

4 VitalQIP QIP Update Service policies

Overview

Purpose

This chapter contains information about the VitalQIP QIP Update Service policies.

Contents

This chapter presents the following topics.

Management of QIP Update Service policies	4-2
The qip-qipupdated.pcy file versus the qip.pcy file	4-2
VitalQIP QIP Update Service policies	4-2
Avoid dynamic name collisions	4-9

Management of QIP Update Service policies

This section describes:

- The management of the QIP Updated Service polices in a file
- A description of the policies
- Instructions on how to avoid dynamic name collisions.

The qip-qipupdated.pcy file versus the qip.pcy file

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. Refer to Chapter 3, "qip.pcy file management" for more information about the *qip.pcy* file.

VitalQIP QIP Update Service policies

The following policies are available:

Note: The QIP Update Service has a registered port number of 2490.

Debug

Value: Default: None

Allowed: Refer to Table 3-1, "Debug policy values" (p. 3-44)

Description: This policy sets the debug level. Refer to "The debug policy" (p. 3-44) for more information about the **Debug** policy.

DebugFile

Value: Default: *qip-qipupdated.log*

Allowable value: Relative or absolute filename

Description: This policy specifies the filename where the debug output is sent.

Master

Value: Default: False

Allowed: True | False | Any number

Description: ***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.

AddIDToLogFilenameOnFork

Value: Default: True

Allowed: True | False

Description: ***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 the **Master** policy is set to True and the **FeatureRotateLogs** debug policy is set.

AcceptSleepTime

Value: Default: 0

Allowed: Any number

Description: 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

Value: Default: True

Allowed: True | False

Description: 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. Refer to "Unique hostname resolution options" (p. 26-21) for more information about name resolution.

SendRenewsToDNS

Value: Default: OnChange

Allowed: Yes | OnChange | No

Description: 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" (p. 24-68) 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 2003 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

Value: Default: Solaris - 256 | Windows - 1024 | Linux - 1024

Allowed: A number between 1 and 1014

Description: 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.

Note: The service adds 10 to the configured value so that processes total 10 to 1024 available files and connections.

ConnectQueueDepth

Value: Default: 10

Allowed: Any number between 5 and 1000

Description: 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.

ConvertSpaces

Value: Default: False

Allowed: True | False

Description: 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

Value: Default: False

Allowed: True | False

Description: 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

Value: Default: False

Allowed: True | False

Description: 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

Value: Default: False

Allowed: True | False

Description: 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.

Refer to the *Audit Manager User's Guide* for more information on the Audit Update Service.

DenyConnectionList

Value: Default: None

Allowed: A comma delimited list of IP addresses and CIDR - style IP address ranges | All

Description: 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.

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

Value: Default: 127.0.0.1

Allowed: A comma delimited list of IP addresses and CIDR - style IP address ranges | All

Description: 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

Value: Default: Unpublished for security reasons

Allowed: Alphanumeric

Description: This policy specifies the VitalQIP administrator's password that the QIP Update Service uses to authenticate itself to VitalQIP.

Server

Value: Default: Valid database server name

Allowed: Alphanumeric (up to 30 characters)

Description: This policy specifies the name of the database server to use. Setting this value overrides the environment variable **`QIPDATASERVER`**.

LoginServer

Value: Default: 127.0.0.1

Allowed: IP address

Description: 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

Value: Default: Ephemeral

Allowed: Ephemeral | Any valid port number | Any service name in */etc/services*

Description: 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

Value: Default: False

Allowed: True | False

Description: This policy allows you to specify whether the service should log events about the status of external updates to the system log.

MessageServicePort

Value: Default: 2468

Allowed: Numeric

Description: This policy allows you to specify an alternate port to communicate with the Message Service.

DNSUpdatePrincipal

Value: Default: Derived

Allowed: Derived | Weak | Strong

Description: 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

Purpose

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" (p. 26-21) 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.

END OF STEPS

5 VitalQIP DNS Update Service policies

Overview

Purpose

This chapter contains information about the VitalQIP DNS Service policies. Contents

This chapter presents the following topics.

Management of DNS Update Service policies	5-2
The dnsupdated.pcy file versus the qip.pcy file	5-2
VitalQIP DNS Update Service policies	5-2

Management of DNS Update Service policies

This section describes:

- The management of the DNS Updated Service polices in a file
- A description of the policies

The dnsupdated.pcy file versus the qip.pcy file

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. Refer to Chapter 3, "qip.pcy file management" for more information about the *qip.pcy* file

VitalQIP DNS Update Service policies

The following policies are available:

Note: The DNS Update Service uses unregistered port number 3119.

Debug

Value: Default: None

Allowed: Refer to Table 3-1, "Debug policy values" (p. 3-44)

Description: This policy sets the debug level. Refer to "The debug policy" (p. 3-44) for more information about the **Debug** policy.

DebugFile

Value: Default: *qip-dnsupdated.log*

Allowed: Relative or absolute filename

Description: This policy specifies the filename where the debug output is sent.

DenyConnectionList

Value: Default: None

Allowed: A comma delimited list of IP addresses and CIDR-style IP address ranges | All

Description: 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.

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

Value: Default: 127.0.0.1

Allowed: A comma delimited list of IP addresses and CIDR-style IP address ranges | All

Description: 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

Value: Default: Ephemeral

Allowed: Ephemeral | Any valid port number | Any service name in */etc/services*

Description: 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

Value: Default: False

Allowed: True | False | any number

Description: ***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.

AddIDToLogFilenameOnFork

Value: Default: True

Allowed: True | False

Description: ***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

Value: Default: 0

Allowed: Any number

Description: 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

Value: Default: 1

Allowed: Numeric

Description: 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

Value: Default: 5

Allowed: Numeric

Description: 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

Value: Default: 0 (no value is set)

Allowed: Numeric

Description: 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

Value: Default: Solaris - 256 | Windows - 1024 | Linux - 1024
Allowed: A number between 1 and 1014

Description: 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.

Note: The service adds 10 to the configured value so that processes total 10 to 1024 available files and connections.

SendRenewsToDNS

Value: Default: No

Allowed: Yes | No

Description: 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 2003 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.

Note: 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. For more information on the **SendRenewsToDNS** policy for the QIP Update Service, refer to "SendRenewsToDNS" (p. 4-4).

ConnectQueueDepth

Value: Default: 10

Allowed: Any number between 5 and 1000

Description: 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

Value: Default: False

Allowed: True | False

Description: 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

Value: Default: False

Allowed: True | False

Description: 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.

DoSecureUpdates

Value: Default: True

Allowed: True | False

Description: 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

Value: Default: (None)

Allowed: Free form string

Description: 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. Refer to "Secure dynamic updates" (p. 24-21) for information about Kerberos.

GSSAPIImmediateRetries

Value: Default: 1

Allowed: Numeric

Description: The number of times that the DNS Update Service tries a GSSAPI /SSPI operation before reporting a failure.

GSSAPIRetryDelay

Value: Default: 900

Allowed: Numeric

Description: The number of seconds after a failure the DNS Update Service waits before attempting to send another secure update to a DNS server.

DoKinit

Value: Default: False

Allowed: True | False

Description: 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

Value: Default: None

Allowable value: */etc/krb5.keytab*

Description: 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.

DDNSGenerateSleep

Value: Default: 0 0 (the *ddns.conf* file only generated manually by running **`qip-genddnsconfs`**)

Allowed: Numeric with an option modifier

Description: 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

Value: Default: 0

Allowed: Numeric

Description: 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.

6 VitalQIP MS DNS Update Service policies

Overview

Purpose

This chapter contains information about the VitalQIP MS DNS Update Service policies.

Contents

This chapter presents the following topics.

Management of MS DNS Update Service policies	6-2
The qip-msdnsupdated.pcy file versus qip.pcy file	6-2
VitalQIP MS DNS Update Service policies	6-2

Management of MS DNS Update Service policies

This section describes:

- The management of the MS DNS Update Service polices in a file
- A description of the policies

The qip-msdnsupdated.pcy file versus qip.pcy file

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. Refer to Chapter 3, "qip.pcy file management" for more information about the *qip.pcy* file.

VitalQIP MS DNS Update Service policies

The following policies are available:

Debug

Value: Default: None

Allowed: Refer to Table 3-1, “Debug policy values” (p. 3-44)

Description: This policy sets the debug level. Refer to “The debug policy” (p. 3-44) for more information about the **Debug** policy.

DebugFile

Value: Default: *msdnsupdated.log*

Allowable value: Relative or absolute filename

Description: This policy specifies the filename where the debug output is sent.

ListenPort

Value: Default: Ephemeral

Allowed: Ephemeral | Any valid port number | Any service name in */etc/services*

Description: 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

Value: Default: None

Allowed: A comma delimited list of IP addresses and CIDR - style IP address ranges | All

Description: 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.

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

Value: Default: 127.0.0.1

Allowed: A comma delimited list of IP addresses and CIDR - style IP address ranges | All

Description: 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

Value: Default: 5

Allowed: A positive number

Description: The number of resource record updates that can fail per zone before a push failure is reported to the caller.

UpdateRetry

Value: Default: 5

Allowed: A positive number

Description: The number of times the service tries to add a resource record before considering the resource record update a failure.

UpdateDelay

Value: Default: 50

Allowed: A positive number of milliseconds

Description: 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

Value: Default: 100

Allowed: A positive number

Description: 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

Value: Default: 10,000

Allowed: A positive number

Description: 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

Value: Default: 5

Allowed: A positive number

Description: 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.

AXFRQueueLength

Value: Default: 5

Allowed: A positive number

Description: 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

Value: Default: 50

Allowed: A positive number

Description: 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.

DisableScavengeStaticRRs

Value: Default: False

Allowed: True or False

Description: If this is set to True, then any resource records in a SECURE zone that would be updated with the qip-strong principal is updated using **`dnscmd`** instead of the API.

Therefore, if scavenging is enabled for the zone, instead of being added dynamically (that is, using the API) and becoming a scavenged record, the RR is added using **`dnscmd`** and cannot be scavenged.

This policy is only needed if scavenging is enabled on a secure zone and users do not want static object RRs to be scavenged.

7 VitalQIP Message Service policies

Overview

Purpose

This chapter contains information about the VitalQIP Message Service policies.

Contents

This chapter presents the following topics.

Management of the Message Service policies	7-2
Message Service behavior	7-2
The qip-msgd.pcy versus the qip.pcy file	7-3
VitalQIP Message Service policies	7-3
MessageQueue policy	7-9
MessageRoute policy	7-10

Management of the Message Service policies

This section describes:

- Behavior of the Message Service
- The management of the Message Service polices in a file
- A description of the policies
- In depth description of the MessageRoute and MessageQueue policies

Message Service behavior

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.

> **Note:** 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.

The qip-msgd.pcy versus the qip.pcy file

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. Refer to Chapter 3, "qip.pcy file management" for more information about the *qip.pcy* file.

VitalQIP Message Service policies

The following policies are available:

Note: 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.

Debug

Value: Default: None

Allowed: Refer to Table 3-1, "Debug policy values" (p. 3-44)

Description: Sets the debug level. Refer to "The debug policy" (p. 3-44) for more information about the **Debug** policy.

DebugFile

Value: Default: *qip-msgd.log*

Allowable value: Relative or absolute filename

Description: Specifies the filename where the debug output is sent.

DenyConnectionList

Value: Default: None

Allowed: A comma delimited list of IP addresses and CIDR - style IP address ranges | All

Description: Prevents 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.

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 **DenyConnectionList**.

AllowConnectionList

Value: Default: All

Allowed: A comma delimited list of IP addresses and CIDR - style IP address ranges | All

Description: 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 **DenyConnectionList**.

AdditionalIPList

Value: Default:

Allowed: A comma delimited list of IP addresses

Description: 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

Value: Default: Solaris - 256 | Linux - 1024 | Windows - 1024

Allowed: A number between 1 and 1014

Description: 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. The service adds 10 to the configured value so that processes total 10 to 1024 available files and connections.

Note: 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

Value: Default: 180

Allowed: Seconds

Description: When a connection is lost, the Message Service waits a number of seconds before attempting to re-establish a connection. This waiting period is configured according to the number of seconds specified in the **ReConnectTimeout** policy and a random time that is up to 1/6 as large as the re-connect timeout. For example, 180 + (0...30).

DumpStatsOnExit

Value: Default: False

Allowed: True | False

Description: 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.

MessageServicePort

Value: Default: 2468

Allowed: Any valid port number

Description: Specifies on what port the service listens for UDP and TCP messages. Ports are usually less than 32,000.

ReceiveBufferSize

Value: Default: 16K

Allowed: A number between 4096 and 65536

Description: Truncates messages larger than the specified value. This policy is useful for handling larger messages. This is used for UDP messages only.

UDPBufferSize

Value: Default: 256K (262144 bytes)

Allowed: Bytes

Description: Specifies the UDP buffer size. If messages are being dropped, increase this value to reduce the number of dropped messages.

AllowUDPControlMessages

Value: Default: False

Allowed: True | False

Description: 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

Value: Default: 0

Allowed: A number between 0 and 2^{32} (that is, 4,294,967,295 seconds).

Description: 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.

QueueFlushPeriod

Value: Default: 500

Allowed: Milliseconds

Description: Determines the number of milliseconds between flushes of the disk queue. The on-disk queue can never be more out-of-sync with the memory queue than the number of milliseconds specified by this policy.

SecureOutgoingProxyConnections

Value: Default: False

Allowed: True | False

Description: 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.

DisplaySelectMasks

Value: Default: False

Allowed: True | False

Description: When set to True, the socket and masks specified to select() are displayed in the debug file.

Note: Use this policy with care since its use can generate copious amounts of debug data.

EnableDebugSocket

Value: Default: False

Allowed: True | False

Description: Facilitates the use of third-party tools when diagnosing Message Service issues. When set to True, sockets of accepted connections are put into debug mode by specifying the SO_DEBUG option on the socket. Some operating systems allow tracing of sockets in debug mode.

MessageQueue

Value: Default: Refer to Table 7-1, "MessageQueue values" (p. 7-9) for definitions of the values

Allowed: <id>: <memq>: <diskq>: <ttl>: <warn>: <message directory>

Description: 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.

MessageRoute

Value: Default: Refer to Table 7-2, "MessageRoute values" (p. 7-11) for definitions of the values

Allowed: <id>, <flags>, <ACK timeout>, <desc>, <port/servicename>, <Server IP>, <backup>

Description: 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

Value: Default: Refer to Table 7-2, "MessageRoute values" (p. 7-11) for definitions of the values

Allowed: <id>, <flags>, <ACK timeout>, <desc>, <port/servicename>, <Server IP>, <backup>

Description: 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 23, "Secure message routes".

MessageQueue policy

This is an optional policy and is specified as:

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

Note: 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 the following table.

Table 7-1 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. **Note:** 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. **Note:** 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.

MessageQueue value	Description
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).
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>:<service_name>:<Server_IP>:…<*#>
```

Note: 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. The following table describes the MessageRoute values.

Table 7-2 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. **Note:** 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. **Note:** 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, as described in "Asynchronous example" (p. 7-13). Valid values for field 2 are: • **L** - Lockstep. Used in combination with the first field as follows: **.** - 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. 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. Refer to "Load Balance example" (p. 7-14). • **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). Refer to "Primary Reconnect example" (p. 7-14).
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 port and service names are as follows: • **qip-dns** - VitalQIP DNS Update Service • **qip-qdhcp** - VitalQIP QIP Update Service • **qip-msgd** - VitalQIP Message Service 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.

  Note: 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.

8 VitalQIP Login Service policies

Overview

Purpose

This chapter contains information about the VitalQIP Login Service policies.

Contents

This chapter presents the following topics.

Management of Login Service policies	8-2
The qip-logind.pcy versus the qip.pcy file	8-2
VitalQIP Login Service policies	8-2

Management of Login Service policies

This section describes:

- The management of the Login Service polices in a file
- A description of the policies

The qip-logind.pcy versus the qip.pcy file

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.

> **Note:** 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. Refer to Chapter 3, "qip.pcy file management" for more information about the *qip.pcy* file.

VitalQIP Login Service policies

The following policies are available:

> **Note:** 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.

Debug

Value: Default: None

Allowed: Refer to Table 3-1, "Debug policy values" (p. 3-44)

Description: This policy sets the debug level. Refer to "The debug policy" (p. 3-44) for more information about the **Debug** policy.

DebugFile

Value: Default: *qip-logind.log*

Allowed: Relative or absolute filename

Description: This policy specifies the filename where the debug output is sent.

ListenPort

Value: Default: Ephemeral

Allowed: Ephemeral | Any valid port number | Any service name in */etc/services*

Description: 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.

ReConnectTimeout

Value: Default: 5

Allowed: Any positive number of seconds

Description: 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

Value: Default: False

Allowed: True | False

Description: 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

Value: Default: False

Allowed: True | False

Description: 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

Value: Default: 5

Allowed: A number between 5 and 1000.

Description: 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

Value: Default: Solaris - 256 | Windows - 1024 | Linux - 1024
Allowed: A number between 1 and 1014

Description: 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.

Note: The service adds 10 to the configured value so that processes total 10 to 1024 available files and connections.

DenyConnectionList

Value: Default: None

Allowable value: A comma delimited list of IP addresses and CIDR-style IP address ranges | All

Description: 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.

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

Value: Default: 127.0.0.1

Allowed: A comma delimited list of IP addresses and CIDR-style IP address ranges | All

Description: 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

Value: Default: 2468

Allowed: Any valid port number

Description: This policy specifies the TCP and UDP ports to which the Message Service will bind.

qip.<database_name>.<database_user>.password

Value: Default: None

Allowed: Alphanumeric (up to 30 characters)

Description: 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. Refer to "Password encryption" (p. 19-9) for more information about encrypting passwords.

audit.<database_name>.<database_user>.password

Value: Default: None

Allowed: Alphanumeric (up to 30 characters)

Description: 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. Refer to "Password encryption" (p. 19-9) for more information about encrypting passwords.

qip.<backup_database_name>.IsBackupOf = <database_name>

Value: Default: None

Allowed:

Description: 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

Value: Default: None

Allowed: Any valid path name

Description: The path name of a callout library which provides custom administrator authentication for VitalQIP or Audit Manager Administrators. Refer to "Administrator authentication tool" (p. 22-2).

9 VitalQIP Schedule Service policies

Overview

Purpose

This chapter contains information about the VitalQIP Schedule Service policies.

Contents

This chapter presents the following topics.

Management of Schedule Service policies	9-2
The qipd.pcy file versus the qip.pcy file	9-2
VitalQIP Schedule Service policies	9-2
Redundant Schedule Service	9-5

Management of Schedule Service policies

This section describes:

- The management of the Schedule Service policies in a file
- A description of the policies
- The use of redundant Schedule Services

The qipd.pcy file versus the qip.pcy file

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. Refer to Chapter 3, "qip.pcy file management" for more information about the *qip.pcy* file.

VitalQIP Schedule Service policies

The following policies are available:

Debug

Value: Default: None

Allowed: Refer to Table 3-1, "Debug policy values" (p. 3-44)

Description: This policy sets the debug level. Refer to "The debug policy" (p. 3-44) for more information about the **Debug** policy.

DebugFile

Value: Default: *qipd.log*

Allowable value: Relative or absolute filename

Description: This policy specifies the filename where the debug output is sent.

ProcessInterval

Value: Default: 60 seconds

Allowable value: Numeric

Description: 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

Value: Default: 1800 seconds (30 minutes)

Allowed: Numeric

Description: 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

Value: Default: 95

Allowed: Numeric

Description: This policy specifies the level (as a percent) at which warnings about license key limits should be produced.

LicenseReadInterval

Value: Default: 86400

Allowed: Seconds

Description: This policy specifies how often the *.Lic* file is reread and rewritten so that license changes can be applied. Set the policy to zero to disable this function.

ObjectCountInterval

Value: Default: 1800 seconds (30 minutes)

Allowed: Numeric

Description: Specifies how often to check if the maximum object count has been exceeded in any organization. Minimum value is 300.

CheckTime

Value: Default: 00:00

Allowed: HH:MM

Description: The time at which **`qip-check`** and **`qip-unlock`** are run.

DumpStatsOnExit

Value: Default: False

Allowed: True | False

Description: 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

Value: Default: Unpublished for security reasons

Allowable value: Alphanumeric

Description: The encrypted password of the Schedule Service administrator.

Server

Value: Default: Appropriate database server name

Allowable value: Alphanumeric (up to 30 characters)

Description: 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

Value: Default: None

Allowable value: IP address

Description: 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

Value: Default: 1h

Allowable value: time values with suffixes of ***`s`*** (seconds), ***`m`*** (minutes), ***`h`*** (hours), ***`d`*** (days)

Description: 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).

RemoveStaleSchedServiceInterval

Value: Default: 7

Allowable value: 1 to 2 billion

Description: 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 (described in "SSFailoverDelay" (p. 9-4)) 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.

10 VitalQIP Active Lease Service policies

Overview

Purpose

This chapter contains information about the VitalQIP Active Lease Service policies. These policies appear in the *qip.pcy* file.

Contents

This chapter presents the following topics.

Management of Active Lease Service policies	10-2
The qip-netd.pcy file versus the qip.pcy file	10-2
VitalQIP Active Lease Service policies	10-2

Management of Active Lease Service policies

This section describes:

- The management of the Active Service polices in a file
- A description of the policies

The qip-netd.pcy file versus the qip.pcy file

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. Refer to Chapter 3, "qip.pcy file management" for more information about the *qip.pcy* file.

VitalQIP Active Lease Service policies

The policies are available:

Debug

Value: Default: None

Allowed: Refer to Table 3-1, "Debug policy values" (p. 3-44)

Description: This policy sets the debug level. Refer to "The debug policy" (p. 3-44) for more information about the Debug policy.

DebugFile

Value: Default: *qip-netd.log*

Allowed: Relative or absolute filename

Description: The policy specifies the filename where the debug output is sent.

DenyConnectionList

Value: Default: (None)

Allowed: A comma delimited list of IP addresses and CIDR-style IP address ranges | All

Description: 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.

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

Value: Default: All

Allowed: A comma delimited list of IP addresses and CIDR-style IP address ranges | All

Description: 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

Value: Default: Ephemeral

Allowed: Ephemeral |
Any valid port number | Any service name in */etc/services*

Description: 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

Value: Default: 2468

Allowed: Any valid port number

Description: This policy specifies the TCP and UCP ports to which the Message Service will bind.

ConvertDomainNameCase

Value: Default: None

Allowed: None | ToLower | ToUpper

Description: Converts the case of the domain names in the active lease display to upper/lower/no conversion (default)

ConvertHostNameCase

Value: Default: None

Allowed: None| ToLower | ToUpper

Description: Converts the case of the hostnames in the active lease display to upper/lower/no conversion (default)

StripTrailingDots

Value: Default: False

Allowed: True | False

Description: If true, trailing dots in the domain name are stripped before being sent to the GUI.

ACKPeriod

Value: Default: 0

Allowed: numeric

Description: 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.

11 VitalQIP MS DHCP Monitor Service policies

Overview

Purpose

This chapter contains information about the VitalQIP MS DHCP Monitor Service policies.

Contents

This chapter presents the following topics.

Management of MS DHCP Monitor Service policies	11-2
The MSDHCPMonitorServiced.pcy file veruses the qip.pcy file	11-2
VitalQIP MS DHCP Monitor Service policies	11-2

Management of MS DHCP Monitor Service policies

This section describes:

- The management of the DHCP Monitor Service in a file
- A description of the policies

The MSDHCPMonitorServiced.pcy file veruses the qip.pcy file

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. Refer to Chapter 3, "qip.pcy file management" for more information about the *qip.pcy* file.

VitalQIP MS DHCP Monitor Service policies

The following policies are available:

Debug

Value:	Default: None
	Allowed: Refer to Table 3-1, "Debug policy values" (p. 3-44)
Description:	This policy sets the debug level. Refer to "The debug policy" (p. 3-44) for more information about the Debug policy.

DebugFile

Value:	Default: *MSDHCPMonitorService.log*
	Allowable value: Relative or absolute filename
Description:	This policy specifies the filename where the debug output is sent.

OrgID

Value: Default: None

Allowed: Numeric

Description: This policy specifies the organization of the DHCP service.

SleepTime

Value: Default: 60

Allowed: Numeric

Description: The number of seconds waiting between reading the *DhcpSrvLog* file, and thus the update rate to the VitalQIP enterprise server.

DenyConnectionList

Value: Default: None

Allowed: A comma delimited list of IP addresses and CIDR - style IP address ranges | All

Description: 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.

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

Value: Default: 127.0.0.1

Allowed: A comma delimited list of IP addresses and CIDR - style IP address ranges | All

Description: 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**.

12 VitalQIP Remote Service policies

Overview

Purpose

This chapter contains information about the VitalQIP Remote Service policies.

Contents

This chapter presents the following topics.

Management of Remote Service Policies	12-2
The qip-rmtd.pcy file versus qip.pcy file	12-2
VitalQIP Remote Service policies	12-2

Management of Remote Service Policies

This section describes:

- The management of the Remote Service polices in a file
- A description of the policies

The qip-rmtd.pcy file versus qip.pcy file

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. Refer to Chapter 3, “qip.pcy file management” for more information about the *qip.pcy* file.

VitalQIP Remote Service policies

The following policies are available.

Note: 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.

BackupFileGenerationServer

Value: Default: None

Allowed: Numeric

Description: The IP address of the backup File Generation Service that will perform the file generation if the primary File Generation Service cannot be contacted.

CreateConduit

Value: Default: False

Allowed: True

Description: The **CreateConduit** policy determines if the Remote Service routes all communication with the File Generation Service through the Message Service. This policy eliminates the need to open more holes than necessary in your firewall. When this policy is used, all the File Generation Service base port policies in the *qip.pcy* file are not required.

This policy also makes securing communication easier. All communication uses the SSL Tunnel Service if configured, which eliminates the need to configure SSL for the Remote Service and File Generation Service separately.

Debug

Value: Default: None

Allowed: Refer to Table 3-1, "Debug policy values" (p. 3-44)

Description: This policy sets the debug level. Refer to "The debug policy" (p. 3-44) for more information about the Debug policy. 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" (p. 3-47).

DNSBinsDir

Value: Default: None

Allowable value: Full path to push bin directory

Description: Allows users to specify the push bin directory if it is different from the default location. On Windows, the default location is *%QIPHOME%\named\bin*; on UNIX, the default location is *$QIPHOME/usr/bin*.

DNSServerAddress

Value: Default: <IP address of the Remote DNS Server>

Allowed: Any valid IP address

Description: This is the IP address to which the reapplied DNS updates will be sent. This prevents repeating these updates to all other DNS servers.

DebugFile

Value: Default: *qip-rmtd.log*

Allowed: Relative or absolute filename

Description: This policy specifies the filename where the Java debug output is sent.

FileBufferSize

Value: Default: 0x10000

Allowed: Number in decimal/hexadecimal/octal format

Description: 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.

FileGenerationServer

Value: Default: None

Allowable value: RMI Scheduler Service's IP address

Description: This policy specifies where this Remote Service has its files generated.

FailOnFailedUserExit

Value: Default: False

Allowed: True | False

Description: 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.

GenDDNSConfOnPush

Value: Default: One

Allowed: None | One | All

Description: 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

MaxDNSUpdateThreads

Value: Default: 5

Allowed: numeric values 0 through 50

Description: A value less than 2 will not create separate threads for processing. A value over 50 is accepted, but not recommended. If this policy is set to a value greater than 2, the remote service creates the specified number of threads and uses these to processthe zones.

MaxFileCopyThreads

Value: Default: 5

Allowed: Any number up to 100

Description: 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.

MessageServicePort

Value: Default: 2468

Allowed: Numeric

Description: Specifies an alternate port to communicate with the Message Service.

PublishOnMessageService

Value: Default: True

Allowed: True | False

Description: 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

Value: Default: VitalQIP Remote Service

Allowed: Any string

Description: Specifies what ID under which to publish the port. ***Do not change this value.***

PortNumber

Value: Default: Ephemeral

Allowed: Ephemeral | Any valid port number | Any service name in */etc/services*

Description: 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.

RegistryPort

Value: Default: 1099

Allowed: Any valid port number

Description: The port on which the RMI registry is listening.

RootReadOnlyNamedConf

Value: Default: True

Allowed: True | False

Description: 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.

RequireSSLConnection

Value: Default: False

Allowed: True | False

Description: 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.

SchedulerName

Value: Default: QAPI_Scheduler

Allowable value: Any name published in RMI Scheduler Service

Description: 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.

SpecifiedDirOnly

Value: Default: False

Allowed: True | False

Description: 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.

UseFreezeThawUpdate

Value: Default: False

Allowed: True | False

Description: This policy is only effective for Lucent DNS 4.x and BIND 9.x server types. A Lucent DNS 4.0 build of 17 or greater or an ISC BIND 9 server of version 9.3.0 or greater is required. If this policy is set to True, the Remote Service will process each zone, ensure the new serial number is greater than the current zone serial number, and re-apply any updates that might have been lost without this policy.

13 VitalQIP RMI QAPI Service policies

Overview

Purpose

This chapter contains information about the VitalQIP RMI QAPI Service policies.

Contents

This chapter presents the following topics.

Management of RMI QAPI Service policies	13-2
The qip-qapid.pcy veruses the qip.pcy file	13-2
VitalQIP RMI QAPI Service policies	13-2

Management of RMI QAPI Service policies

This section describes:

- The management of the RMI QAPI Service polices in a file
- A description of the policies

The qip-qapid.pcy veruses the qip.pcy file

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. Refer to Chapter 3, “qip.pcy file management” for more information about the *qip.pcy* file.

VitalQIP RMI QAPI Service policies

The following policies are available.

Note: The **DebugLevel** policy can be used to debug this policy. Refer to “Java debug policies” (p. 3-47).

Note: The RMI QAPI Service uses RMI to communicate with other Services. The default port for RMI is 1099.

MaxZipFileSize

Value: Default: 1M

Allowed: Numeric

Description: 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.

Note: 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.

QAPISSL_Scheduler.Secure

Value: Default: True

Allowed: True | False

Description: 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" (p. 23-8) for more information on how to set up secure file transmission with SSL.

QAPISSL_Scheduler.PassPhrase

Value: Default: None

Allowable value: Any string

Description: 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

Value: Default: *qipkeystore*

Allowable value: Any filename

Description: 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

Value: Default: *QAPI.%d.log*

Allowed: Relative or absolute filename

Description: 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

Value: Default: 127.0.0.1

Allowed: IP address

Description: This policy specifies where the RMI Scheduler Service can be found.

ServerAddressOverride

Value: Default: IP address

Allowed: Any valid IP address

Description: 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.

Note: 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.

FailOnFailedUserExit

Value: Default: False

Allowed: True or False

Description: If this policy is set to True, you can cause the push to fail by having the user exit return a non-zero value. Otherwise, the File Generation Service continues with the push even if the user-exit fails.

UserExitErrorFile

Value: Default: UserExit.error

Allowed: Error file

Description: After a user exit executes, the contents of the specified file are read if the status is non-zero. If FailOnFailedUserExit is set to True, the contents are reported back to the VitalQIP client. If FailOnFailedUserExit is set to False, the contents are sent to the debug log.

14 VitalQIP RMI Scheduler Service policies

Overview

Purpose

This chapter contains information about the VitalQIP RMI Scheduler Service policies.

Contents

This chapter presents the following topics.

Management of RMI Scheduler Service policies	14-2
The qip-rmisched.pcy versus the qip.pcy file	14-2
VitalQIP RMI Scheduler Service policies	14-2

Management of RMI Scheduler Service policies

This section describes:

- The management of the RMI Scheduler Service polices in a file
- A description of the policies

The qip-rmisched.pcy versus the qip.pcy file

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. Refer to Chapter 3, "qip.pcy file management" for more information about the *qip.pcy* file.

VitalQIP RMI Scheduler Service policies

The following policies are available:.

Debug

Value: Default: None

Allowed: Refer to Table 3-1, "Debug policy values" (p. 3-44)

Description: This policy sets the debug level. Refer to "VitalQIP RMI Scheduler Service policies" (p. 14-2) for more information about the **Debug** policy. 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" (p. 3-47).

DebugFile

Value: Default: *qip-rmisched.log*

Allowed: Relative or absolute filename

Description: This policy specifies the filename where the Java debug output is sent.

RegistryPort

Value: Default: 1099

Allowable value: Any valid port number

Description: The port on which the RMI registry is listening. The registered port is 1099.

Note: The VitalQIP RMI Scheduler Service uses RMI to communicate with other services. The default port for RMI is 1099.

BasePort

Value: Default: 0

Allowed: Any valid port number

Description: 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

Value: Default: IP address

Allowed:Any valid IP address

Description: 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

Value: Default: QAPI_Scheduler

Allowed: List of strings

Description: 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 space or comma separated list of schedulers. For example:

```
SchedulerNames = QAPISSL_Scheduler QAPI_Scheduler
SchedulerNames = QAPISSL_Scheduler,QAPI_Scheduler
```

The policy **SchedulerName** in the Remote Service must match one of the names specified here. Refer to "SchedulerName" (p. 12-6) for more information on the **SchedulerName** policy.

<SchedulerName>.BasePort

Value: Default: 0

Allowed: Any valid port number

Description: This policy specifies the base port number that executors will be bound to.

<SchedulerName>.Port

Value: Default: 0

Allowed: Any valid port number

Description: 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

Value: Default: QAPI

Allowed: Any string

Description: This policy specifies the name used when scheduling access. It is typically the same name as the class of the Executor.

<SchedulerName>.ExecutorClass

Value: Default: QAPI

Allowed: Any String

Description: 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

Value: Default: 1

Allowed: 0 - 64

Description: 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>.MaxExecutorCount

Value: Default: 5

Allowed: 1 - 128

Description: 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>.ExecutorRegistrationTimeout

Value: Default: 5000

Allowed: Number of milliseconds until timeout

Description: 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

Value: Default: False

Allowed: True | False

Description: 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" (p. 23-8) for more information on how to set up secure file transmission with SSL.

<SchedulerName>.PassPhrase

Value: Default: None

Allowable value: Any string

Description: A secret phrase used for key management. The **PassPhrase** policy is only needed if **<SchedulerName>.Secure** is set to True.

<SchedulerName>.KeysFile

Value: Default: None

Allowable value: Any filename

Description: The file where the secure socket layer keys is stored. The **KeysFile** policy is only needed if **<SchedulerName>.Secure** is set to True.

<SchedulerName>.VirtualMachineParams

Value: Default: None

Allowable value: Any parameters which the Java Virtual machine (JVM) accepts

Description: 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.

15 VitalQIP SSL Tunnel Service policies

Overview

Purpose

This chapter contains information about the VitalQIP SSL Tunnel Service policies.

Contents

This chapter presents the following topics.

Management of SSL Tunnel Service policies	15-2
The qip-ssltd.pcy file versus the qip.pcy file	15-2
VitalQIP SSL Tunnel Service policies	15-2

Management of SSL Tunnel Service policies

This section describes:

- The management of the SSL Tunnel Service polices in a file
- A description of the policies

The qip-ssltd.pcy file versus the qip.pcy file

Behavior of the SSL Tunnel Service (**qip-ssltd**) is controlled by policies in the *qip-ssltd.pcy* file or the *qip.pcy* file, located in the *QIPHOME* directory. The *qip-ssltd.pcy* file is used only by the QIP Update Service. This file is not created automatically. Since the *qip-ssltd.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-ssltd.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. Refer to Chapter 3, "qip.pcy file management" for more information about the *qip.pcy* file.

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**).

The following policies are available in the *qip.pcy* policy file.

SecureIncomingServiceConnections

Value: Default: False

Allowed: True | False

Description: 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.

Debug

Value: Default: None

Allowed: Refer to Table 3-1, "Debug policy values" (p. 3-44)

Description: This policy sets the debug level. Refer to "The debug policy" (p. 3-44) for more information about the Debug policy. 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" (p. 3-47).

DebugFile

Value: Default: *qip-ssltd.log*

Allowable value: Relative or absolute filename

Description: This policy specifies the filename where the debug output is sent.

MessageBufferSize

Value: Default: 0x2000

Allowed: Numeric

Description: 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.

PublishID

Value: Default: VitalQIP SSL Tunnel Service

Allowed: ***Do not change***

Description: This policy specifies under what ID to publish the service port. ***Do not change this policy.***

MessageServicePort

Value: Default: 2468

Allowed: Any valid port number | Any service name in */etc/services*

Description: 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

Value: Default: None

Allowed: A comma delimited list of IP addresses and CIDR - style IP address ranges | All

Description: 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.

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

Value: Default: All

Allowed: A comma delimited list of IP addresses and CIDR - style IP address ranges | All

Description: 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

Value: Default: <*qip-crypted keystore password*>

Allowed: encrypted password

Description: This policy specifies the password used to protect the keystore. This should be stored as an encrypted and encoded string.

KeysFile

Value: Default: qipkeystore

Allowed:

Description: Refer to "Configuring SSL" (p. 23-8) 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.

16 VitalQIP Access Control policies

Overview

Purpose

This chapter contains information about the VitalQIP Access Control policies.

Contents

This chapter presents the following topics.

Management of Access Control policies	16-2
VitalQIP Access Control policies	16-2

Management of Access Control policies

This section describes:

- The management of the Access Control polices in a file
- A description of the policies

VitalQIP Access Control policies

The following policies are available in the *qip.pcy* policy file.

CachServiceDataFile

Value: Default: *$QIPHOME/etc/CacheService.dat*

Allowed: Relative or absolute filename

Description: This policy specifies where to store cache data.

MaxQueryThreads

Value: Default: Number of cores

Allowed:

Description: This policy specifies how many threads will handle cache queries.

InitialCacheSize

Value: Default: 266671

Allowable value: Integer

Description: This policy specifies the initial size of the cache.

The cache is a hash-table with a load factor of .75, so an initial size of 266671 is sufficient to store 200,000 objects without a resize. The resize operation can be resource-intensive, so Alcatel-Lucent recommends setting this value to avoid changing it in the future.

A prime value also helps ensure even distribution of the table, so 266671 is used instead of 266666. Other examples of number of objects/cache size are:

- 300000 objects/400009 cache size
- 400000 objects/533317 cache size
- 500000 objects/666667 cache size

PortNumber

Value: Default: ephemeral port

Allowed: Numeric

Description: This policy specifies specifies what port the service should bind to

The following policies are available in the *qip.properties* file.

DeviceFolderSize

Value: Default:

Allowed: Integer

Description: This policy specifies the number of devices contained within the device folder in the ACL Hierarchy in the VitalQIP Web GUI.

SubscriberFolderSize

Value: Default:

Allowed: Integer

Description: This policy specifies the number of subscribers contained within the device folder in the ACL Hierarchy in the VitalQIP Web GUI.

acSubscriberPageSize

Value: Default: 50

Allowed: Integers separated by commas

Description: This property holds the values for the desired page sizes on the Access Control Subscribers page. The Drop Down for setting the page size of the table contains these values. The first value in the list is the default value for the page size. Separate the values with commas.

acDevicePageSize

Value: Default: 50

Allowed: Integers separated by commas

Description: This property holds the values for the desired page sizes on the Access Control Devices page. The Drop Down for setting the page size of the table contains these values. The first value in the list is the default value for the page size. Separate the values with commas.

17 Web client related policies

Overview

Purpose

This chapter describes policies related to the VitalQIP web client interface. The following service policies are covered:

- VitalQIP Service Layer
- Address Allocator Service
- VitalQIP Web GUI
- MyView Service
- Job Scheduler

Contents

This chapter presents the following topics.

Management of web client related policies	17-2
VitalQIP Service Layer policies	17-2
Address Allocator Service policies	17-3
VitalQIP Web GUI policies	17-4
MyView policies	17-5
Job Scheduler policies	17-6
backgroundOnly	17-8

Management of web client related policies

The *qip.pcy* file contains several services that impact the VitalQIP web client and related functionality. This section describes the service and policies that impact the VitalQIP web client.

VitalQIP Service Layer policies

The **`[VitalQIP Service Layer]`** section determines how VitalQIP connects and send messages to the Apache Tomcat web server. This feature is configured at the time of the installation. However, it can be changed if your configuration changes.

The follwing policies are available in the **`[VitalQIP Service Layer]`** section:

soapWssePassPolicy

Values: Default: text

Allowed: text | digest

Description: This policy controls the password handling in the SOAP Username Token Profile headers. Note that **digest** cannot be used in combination with an external authentication callout module, but do provide greater security.

soapUrl

Values: Default: Determined at time of installation.

Allowed:

<HTTP>://<*IP address of web server*>:<*port*>/VQIPManagerPort |

<HTTPS>://<*IP address of web server*>:<*port*>/VQIPManagerPort | **bypass**

Description: This policy determines if VitalQIP connects and sends messages to the Apache Tomcat web server as encrypted (HTTPS) or non-encrypted (HTTP) messages. By default, the URL is defined when the product is installed and occurs in this format:

- **HTTP://<IP address of web server>:<port>/VQIPManagerPort**
- **HTTPS://<IP address of web server>:<port>/VQIPManagerPort**

The **bypass** value does not specify a SOAP URL. ***You can only use `bypass` as a value if you are not running Address Allocation.***

Note: The following are the meanings of the variables in the URL values:

- **HTTP** - Determines all messages are not encrypted
- **HTTPS** - Determines all message are encrypted
- **<IP address of web server>** - Specifies the IP address where the Apache Tomcat web server resides.
- **<port>** - Specifies the port used to send encrypted and non-encrypted messages. By default, 80 is the default port for non-encrypted messages, and 443 for encrypted messages.

Address Allocator Service policies

During IPv4 block allocation post-processing, information for IPv4 subnet creation is passed in XML files. The Address Allocator Service policies determine the name of the directory and the filenames where those XML files are generated. Also if required, the

XML files can be kept indefinitely for administrative reasons.

The following policies are available in the *qip.pcy* policy file in the **`[Address Allocator Service]`** section:

CliInputFileDir

Value: Default: None

Windows: *C:\QIP\Cli_Subnet*

Unix/Linux: */opt/qip70/cli_subnet*

Description: This policy specifies the name of the directory used for generating the CLI data files that are used as input for post-processing CLIs. If not specified, VitalQIP uses the default temp directory.

CliKeepGeneratedFile

Value: Default: False

Allowed: True | False

Description: This policy specifies whether VitalQIP stores a copy of the generated input file for CLI post-processing. To store the files indefinitely, set the policy to True.

CliInputFileNamePrefix

Value: Default: None

Allowed: Alphanumeric string, such as CLI_SUBNET

Description: This policy specifies the prefix name of the files that are generated for post-processing CLIs.

SendEmailAlerts

Value: Default: True

Allowed: True | False

Description: This policy specifies whether VitalQIP needs to enable or disable e-mail alerts during block allocation for RIR reporting. If this policy is set to False, alerts are disabled. If this policy is set to True or disabled, e-mail alerts are enabled. Typically, e-mail alerts can be turned off in a test/lab environment but should be enabled in a production environment.

VitalQIP Web GUI policies

In the **`[VitalQIP Web GUI]`** section, there is policy available to determine if a parent pool is automatically expanded to show a child pools.

The **autoExpandPool** policy is used as follows.

autoExpandPool

Value: Default: True

Allowed: True | False

Description: Determines if the parent pool is automatically expanded to show newly created child pools. The default value is True but if the number of child pools is very large (for example, tens of thousands of child pools) and performance is impacted, this value can be set to False.

MyView policies

The **[MyView]** section of the *qip.pcy file* contains policies for configuring the MyView feature. The following is a description of the policy.

CircularMyViewCheck

Value: Default: True

Allowed: True | False

Description: Checks whether a MyView instance (A) has a Child View (B), which in turn includes the MyView instance (A). If this policy is set to True, it enables the MyView instance and Child View checking. This checking slows down the performance of the MyView feature. This policy can be set to False to disable the checking for a circular relationship among the MyView instances.

ExcludeEmptyTypes

Value: Default: True

Allowed: True | False

Description: Determines if a count check is needed for infrastructure types when the contents of a MyView are displayed. When this policy is set to True, a count occurs for all supported infrastructure types and only the infrastructure types with one or more items matching MyView criteria are displayed under MyView. The count occurs wht hen an administrator logs in or contents of MyView are displayed. If the number of items in infrastructure types is very large and the performance of web client is impacted, set this policy to False to disable empty infrastructure type check.

MaxPersonalViews

Value: Default: 5

Allowed: Numeric

Description: Specifies the maximum number of Personal Views that an administrator can create in the system.

MaxInfrastructureInstances

Value: Default: 50

Allowed: Numeric

Description: Specifies the maximum number of static instances per infrastructure type that an administrator can add statically to a Personal View.

AssignedInfraTypeOnlyInAddPersonalView

Value: Default: False

Allowed: True | False

Description: Determines if administrators only see those infrastructure types to which they have access in the Personal View -> Add Contents screen, or if they see all infrastructure types. The policy is set to False for performance reasons. Be aware there may be a serious performance impact if set to True.

Job Scheduler policies

The **`[Job Scheduler]`** section in the *qip.pcy* file is used to configure the Job Scheduler. Job Scheduler allows jobs to run in the background or on a schedule. The following is a description of the policies:

checkInterval

Value: Default: 120000 (2 minutes)

Allowed: milliseconds

Description: Specifies how often (in milliseconds) the job scheduler checks the database for new jobs.

jobReadAhead

Value: Default: .25 (25% of the checkinterval time)

Allowed: Percentage of **checkInterval** milliseconds

Description: Determines how far in the future beyond the time specified in checkInterval policy to retrieve scheduled jobs from the database.If the default check interval is 2 minutes, the scheduler retrieves jobs from the database that are scheduled to occur between the current time, 2 minutes, and 30 seconds in the future. This is to ensure that no jobs are missed due to unsynchronized clocks or network latency.

jobThreads

Value: Default: 5

Allowed: Number of threads

Description: Specifies the number of threads available for running scheduled and background jobs. For example, if there are 5 threads and 6 jobs are scheduled to occur at the same time, the last job has to wait for one of the other jobs to complete before running. The upper limit of threads is dependent on available memory, number of processors, and the type of job being run.

randomizeStart

Value: Default: True

Allowed: True | False

Description: Determines if the start time of jobs is randomized within certain boundaries.

randomBoundry

Value: Default: 15000

Allowed: milliseconds

Description: Only applicable if the **randomizeStart** policy is set to True. Specifies a upper limit of the random offset (in milliseconds) applied to scheduled jobs.

retentionDays

Value: Default: 60

Allowed: Number of days

Description: Specifies the number of days to keep records of completed or failed jobs in the database. After the specified time limit, job records are deleted.

purgeCheckInterval

Value: Default: 12

Allowed: Number of hours

Description: Determines how often (in hours) the job scheduler looks for jobs to delete from the database.

schedulerHostName

Value: Default: localhost

Allowed: Hostname of the scheduler

Description: Specifies the hostname of the scheduler. This name is displayed in the scheduler page to identify the server that ran the job. By default, this policy is set to localhost, and is replaced with the hostname as queried from the operating system.

autoRefreshDefault

Value: Default: False

Allowed: True | False

Description: Determines whether, by default, the **Auto Refresh** checkbox on the Job Scheduler page in the VitalQIP web client is set to refresh automatically. If this policy is set to True, the **Auto Refresh** checkbox in the Job Scheduler page is checked by default. The checkbox is unchecked by default when the policy value is set to false.

backgroundOnly

Value: Default: False

Allowed: True | False

Description: Determines if a job scheduler runs only background jobs. If this policy is set to True, the job schuduler runs only background jobs. Only one server should have this policy set to False.

Part III: VitalQIP system administration

Overview

Purpose

In Part III, you will find information on configuring the VitalQIP web client, database administration, server administration, troubleshooting VitalQIP, and general messages.

Contents

Part III contains the following chapters:

Chapter 18, “Web client configuration”	18-1
Chapter 19, “Database and server administration”	19-1
Chapter 20, “Troubleshoot VitalQIP”	20-1
Chapter 21, “General informational and error messages”	21-1

18 Web client configuration

Overview

Purpose

This chapter describes how to customize the behavior of the web client, modify log settings, customize UDA callouts, and support multiple character sets and languages.

Contents

This chapter covers these topics.

Character set and language configuration	18-3
Overview	18-3
To prepare the translation catalog	18-5
To create directories for localization	18-9
To update graphics with the new language	18-13
To make localized directories available	18-14
To customize specific labels	18-15
To migrate customized or localized labels and messages	18-17
To set linguistic sorting	18-20
Web client services configuration	18-22
Overview	18-22
Tomcat server configuration	18-22
Web Service configuration	18-23
Web client logging	18-24
log4j message levels	18-24
Sample log4j.properties file	18-26

qip.properties file	18-33
Sample qip.properties file	18-47
To configure session time-outs	18-65
To enable authentication via the Login Service	18-67
To enable compatibility with VitalQIP 7.0/7.1 Web Service	18-69
UDA callout	18-70
UDA callout configuration	18-70
Java callout classes	18-71
To create a UDA callout	18-75
User-defined object naming policy	18-73
To configure a user-defined object naming policy for the web client	18-73
Access external applications from the web client toolbar	18-78
To add an external application to the web client	18-80

Character set and language configuration

Overview

Purpose

This section describes how to configure VitalQIP to support multiple character sets/languages within the web client interface.

VitalQIP allows you to configure the web client interface to support multiple languages by allowing you to substitute a string in a different language globally throughout the web client interface screens. For example, you can change all instances of text on a certain button (such as **Submit**) through the entire web client interface.

VitalQIP also allows you to change text on just one screen within the web client interface. Each screen consists of three HTML documents, a menu, a hierarchy, and the user data/input document. You can localize text in each HTML document that composes a screen. You can use this feature to clarify text on a specific HTML document that makes up part of a particular screen within the web client interface, without affecting the same text elsewhere. VitalQIP permits this because of the context in which a word might be used.

The examples in this section show how to implement French language support. You can implement other languages as appropriate for your site.

Requirements

Supporting international characters requires:

- The user's desktop must support UTF-8, including the target character set. By default, neither UNIX nor Windows are configured with this support. The client toolset must support UTF-8; even the ubiquitous **`xterm`** has to be run with suitable options for UTF-8 to render correctly.
- Java requires the ResourceBundle (that is, catalog) properties file be encoded with ISO-8859-1, not UTF-8. Native characters are represented using Unicode \uNNNN notation. Various tools exist to convert a file containing native characters into this form, including a Java developer tool named **`native2ascii`**.
- An 8-bit clean connection is needed from the client system to the server system. Tools, such as **`telnet`** and **`rlogin`**, default to 7-bit data unless the **`-8`** option is specified.

To install localization files

Purpose

This section describes the steps you need to follow to create and install locale-specific files. Use this procedure when you want the web client interface to display prompts, labels and messages in a language other than US English.

Procedure

To create and install locale specific files, follow these steps:

1 Login as root. Enter the following command to change to the directory containing the compressed localization files:

```
cd $QIPHOME/web/templates
```

The localization files are delivered in one of the following compressed files depending on the operating system where your web client interface resides:

- For UNIX, *qipTemplates.tar.gz*
- For Windows, *qipTemplates.zip*

2 Uncompress the files:

- For UNIX platforms, type the following commands, and the expanded files are placed in a directory structure under *QIPHOME/web/templates* similar to the *QIPHOME/web* directory:

```
gunzip qipTemplates.tar.gz
tar -vxf  qipTemplates.tar
```

- For Windows platforms, unzip the *qipTemplates.zip* file. The files are expanded into a directory structure under *QIPHOME/templates*.

END OF STEPS

To prepare the translation catalog

Purpose

Use this procedure to assign translated values to elements in the VitalQIP web client interface, services, and the **`qip-cli`** command.

Before you begin

International character support requires the following:

- The user's desktop must support UTF-8, including the target character set. By default, neither UNIX nor Windows are configured with this support.
- The client toolset must support UTF-8, including text editors and terminal emulators.
- Java requires that the ResourceBundle (that is, catalog) properties file be encoded with ISO-8859-1, not UTF-8. Native characters are represented using a Unicode \uNNNN notation. Various tools exist to convert a file containing native characters into this format.
- An 8-bit clean connection is needed from the client system to the server system. Tools, such as **`telnet`** and **`rlogin`** default to 7-bit data unless the **`-8`** option is specified.

Preparation for the web client interface

Use the following steps to assign translated values to elements in the VitalQIP web client interface:

1 Change to the directory containing the *GuiCatalog_en.properties* file by entering the following command:

- For UNX, type:

```
cd $QIPHOME/web/templates/web/catalog
```

- For Windows, opens Windows Explorer and go to *QIPHOME/web/templates/web/catalog*.

2 In the same directory, copy the *GuiCatalog_en.properties* file and create a new file with the name *GuiCatalog_?.properties*, where ? is the language code for the language you are implementing. You may also include an optional country code. A full listing of the ISO 639-2 character language codes is available at:

http://www.loc.gov/standards/iso639-2/php/English_list.php

A full listing of the ISO 3166 country codes is available at:

http://www.iso.ch/iso/en/prods-services/iso3166ma/02iso-3166-code-lists/list-en1.html

For example:

If you want...	Then...
To create a catalog for French on UNIX	Use the following command: • With the language code only: **`cp GuiCatalog_en.properties GuiCatalog_fr.properties`** • With the language code and country code: **`cp GuiCatalog_en.properties GuiCatalog_fr_CA.properties`**
To create a catalog for French on Windows	Do the following: • With the language code only, copy the *GuiCatalog_en.properties* to a different directory and rename it **`GuiCatalog_fr.properties`**. • With the language code and country code, copy the *GuiCatalog_en.properties* to a different directory and rename it **`GuiCatalog_fr_CA.properties`**.

3 Open the new file you just created, and edit the file.

> **Note:** The editor you use must support the character set of the language you are implementing, and must be able to save the file as UTF-8 (Unicode).

The file contains a series of ***`tag=value`*** pairs. The tag indicates the element in the GUI to which it is assigned, and the value indicates what text to display when the GUI is opened using the language specified in this properties file.

An example of a ***`tag=value`*** pair is:

```
submit=Submit
```

4 For all ***`tag=value`*** pairs, change the ***`value`*** to the corresponding word in the new language.

> **Note:** Do not change the ***`tag`*** or `UNDEFINED` appears anywhere for this value within the web client interface.

5 Save the file. The translation of catalog is complete.

6 Copy *htmlTemplateCatalog.properties* to a new file with a country prefix. This file is used by **`TemplateMaker.sh`** (UNIX) and **`TemplateMaker.bat`** (Windows) to generate the translated files for web client interface. To do so:

If your platform is...	Then...
UNIX	1. Copy the *htmlTemplateCatalog.properties* file with the new name*:* **`cp htmlTemplateCatalog.properties frenchTemplateCatalog.properties`** 2. Edit the file so that it points to the appropriated catalog file. Change **`templateFactory.Catalogs=catalog/GuiCatalog_en.properties`** to **`templateFactory.Catalogs=catalog/GuiCatalog_fr.properties`**. 3. Save the file.
Windows	1. Copy the *htmlTemplateCatalog.properties* file to a different directory and rename it **`frenchTemplateCatalog.properties`** 2. Open the file and change **`templateFactory.Catalogs=catalog/GuiCatalog_en.properties`** to **`templateFactory.Catalogs=catalog/GuiCatalog_fr.properties`** 3. Save the file.

END OF STEPS

Preparation for services and qip-cli

Use the following steps to prepare localized files for services and **`qip-cli`**.

1 Change to the catalog directory:

- For UNIX, enter the following command:

`cd $QIPHOME/web/templates/web/catalog`

- For Windows, open Windows Explorer and go to the *QIPHOME/web/templates/web/catalog* directory.

2 Copy all the services/cli catalog files (***except*** *logMessageCatalog.properties)* to file names with a language code or with optional country code) extension. For example:

If your platform is...	Then...
UNIX	Use the following command: • With language code: **cp udaCatalog.properties udaCatalog_fr.properties** • With language code and country code: **cp udaCatalog.properties udaCatalog_fr_CA.properties**
Windows	Do the following: • With language code, copy the *udaCatalog.properties* file to a different directory and rename it **udaCatalog_fr.properties** • With language code and country code, copy the **udaCatalog.properties** file to a different directory and rename it **udaCatalog_fr_CA.properties**

3 Translate the new catalog files into a desired language using an editor that supports multiple languages (for example, **gedit** on the Solaris platform) and save them as UTF8-Unicode files.

4 Encode the translated files into an ISO-8859-1 file in the same native environment.

Note: It is recommended that you save the file to another name before you encode it, for example:

```
mv udaCatalog_fr.properties udaCatalog-fr-UTF8.properties
native2ascii -encoding UTF-8 udaCatalog-fr-UTF8.properties
udaCatalog_fr.properties
```

Note: **native2ascii** is a tool used to convert UTF-8 file into the *ISO-8859-1* file.

END OF STEPS

To create directories for localization

Purpose

Use this procedure to create a set of directories for languages you want to support in the VitalQIP web client interface, service, or the **`qip-cli`** command.

Before you begin

For instructions on how to stop and start the Tomcat server, refer to "To start or stop the Tomcat server on Windows" (p. 2-16) and "To start or stop the Tomcat server on UNIX" (p. 2-22).

Create directories for the web client interface

Use the following steps to create a set of directories for languages you want to support in the VitalQIP web client interface:

1. Change to the directory containing the **`TemplateMaker.sh`** (UNIX) or **`TemplateMaker.sh`** (Windows) file.
   - For UNIX, enter the following command:

   `cd $QIPHOME/web/templates/`
   - For Windows, open Windows Explorer and go to the *QIPHOME/web/templates* directory.

2. Edit **`Templatemaker.sh`** (UNIX) or **`Templatemaker.bat`** (Windows) the and make the following changes (the following examples use **`fr`** to indicate French. Substitute other 2 character language codes as appropriate for your site:

   Note: The following instructions generate both default templates and dynamic templates.
   - Change **`OUTDIR=./web/html_en/`** to **`OUTDIR=./web/html_fr/`**
   - Change **`CAT_NAME=htmlTemplatesCatalog.properties`** to **`CAT_NAME=frenchTemplatesCatalog.properties`**
   - Change **`OUTDIR=./web/dynamic_templates/en/`** to **`OUTDIR=./web/dynamic_templates/fr/`**

3. Save the file.

4 Make sure the file is executable. If it is not an executable:

- For UNIX, enter the following command:

```
chmod +x TemplateMaker.sh
```

- For Windows, right-click **`Template.bat`**, select **Properties**, click **Security**, and ensure **Allow** is checked for all permission settings.

5 Run **`TemplateMaker.sh`** (UNIX) or **`TemplateMaker.bat`** (Windows). The following directories are created that contain the localization files for the new language:

- *QIPHOME/web/templates/web/html_fr*
- *QIPHOME/web/templates/web/dynamic_templates/fr*

6 Copy the VitalQIP® (QIP) image files to the new directory:

If your platform is...	Then...
UNIX	1. Copy the local images files to the directory created in step 5: **`cp -R $QIPHOME/web/templates/web/html_en/locale_images $QIPHOME/web/templates/web/html_fr`** 2. Copy the images file to the new directory: **`cp -R $QIPHOME/web/templates/web/dynamic_templates/en/images $QIPHOME/web/templates/web/dynamic_templates/fr`** 3. Follow steps in "To update graphics with the new language" (p. 18-13) for translation of text in the image files.
Windows	1. Copy the local images files to the directory created in step 5: *QIPHOME/web/templates/web/html_en/locale_images* directory to *QIPHOME/web/templates/web/html_fr* directory. 2. Copy the images file from the *QIPHOME/web/templates/web/dynamic_templates/en/images* to *QIPHOME/web/templates/web/dynamic_templates/fr.* 3. Follow steps in "To update graphics with the new language" (p. 18-13) for translation of text in the image files.

7 Copy VitalQIP help files and documentation:

If your platform is...	Then...
UNIX	1. Create a help directory under *$QIPHOME/web/termplates/web/html_fr*. 2. Copy the archived files to this directory: **`cp $QIPHOME/web/templates/web/help/QIP.tar.gz $QIPHOME/web/templates/web/html_fr`** 3. Change the directory: **`cd $QIPHOME/web/templates/web/html_fr`** 4. Untar the file: **`gunzip QIP.tar.gz`** **`tar -xvf QIP.tar`** **Note:** The help files and documentation are placed in the *QIPHOME/web/templates/web/help* directory. **Note:** The help files and documentation are in English. If needed, translate all help files and documentation into your default language.
Windows	1. Open Windows Explorer and create a help directory under *%QIPHOME%\web\templates\web\html_fr*. 2. Copy the archived files to this directory. 3. Go to the *QIPHOME/web/templates/web/html_fr*. 4. Unzip the *QIP.zip* file. **Note:** The help files and documentation are placed in the *QIPHOME/web/templates/web/help* directory. **Note:** The help files and documentation are in English. If needed, translate all help files and documentation into your default language.

8 Copy the new directories to the appropriate location in QIPHOME (in this example, for French):

- For UNIX, enter the following command:

```
cp -R $QIPHOME/web/templates/web/html_fr $QIPHOME/web
cp -R $QIPHOME/web/templates/web/dynamic_templates/fr
   $QIPHOME/web/dynamic_templates
```

- For Windows, copy the *QIPHOME/web/templates/web/html_fr* directory to the *QIPHOME/web* directory and the *QIPHOME/web/templates/web/dynamic_templates/fr* directory to the *QIPHOME/web/dynamic_templates* directory.

END OF STEPS

Create directories for services and qip-cli

Use the following steps to create a set of directories for languages you want to support in the services and the **`qip-cli`** command:

1. Stop the Tomcat service.

2. Copy the translated to *cliCatalog_fr.properties* file to *$QIPHOME/classes/catalog* directory:
   - On UNIX, enter the following command:

```
cp cliCatalog_fr.properties  $QIPHOME/classes/catalog
```

   - On Windows, open Windows Explorer and copy the *cliCatalog_fr.properties* file to the *QIPHOME/classes/catalog* directory.

     Note: *logMessageCatalog.properties* should not be copied.

3. Copy all the translated services catalog files except the *cliCatalog*.properties* file, which is for **`qip-cli`**, to *QIPHOME/web/WEB-INF/classes/catalog* directory. For example:
   - On UNIX, enter the following command:

```
cp $QIPHOME/web/templates/web/catalog/*fr.properties
$QIPHOME/web/WEB-INF/classes/catalog
```

   - On Windows, copy all the files except the *cliCatalog.properties* file in the *QIPHOME/web/templates/web/catalog* directory to the *QIPHOME/web/WEB-INF/classes/catalog* directory.

     Note: *logMessageCatalog.properties* should not be copied.

4. Start the Tomcat service.

END OF STEPS

To update graphics with the new language

Purpose

Use this procedure to translate text in graphics in the VitalQIP web client interface.

Before you begin

You must have access to an image editor that allows you to place text in the image file in the language you are implementing.

Procedure

Use the following steps to translate text in graphics in the VitalQIP web client interface:

1 Change to the directory that contains local images (the following example is for French language implementation):

- For UNIX, enter the following command:

```
cd $QIPHOME/web/templates/web/html_fr/local_images
```

- For Windows, open Windows Explorer and go to the *QIPHOME/web/templates/web/html_fr/local_images directory.*

2 Use an image editor to translate labels/text into French in the images files in this directory.

3 Change to the directory that contains dynamic template images (the following example is for French language implementation):

- For UNIX, enter the following command:

```
cd $QIPHOME/web/templates/web/dynamic_templates/fr/images
```

- For Windows, go to *QIPHOME/web/templates/web/dynamic_templates/fr/images* in Windows Explorer.

4 Use an image editor to translate labels and text into French in the images files in this directory.

END OF STEPS

To make localized directories available

Purpose

Use this procedure to make the languages you want to support visible on the login screen of the VitalQIP web client interface.

Procedure

Use the following steps to make the languages you want to support visible in the VitalQIP web client interface:

1. Change to the directory containing the *locales.properties* file:
   - On UNIX, enter the following command:

     ```
     cd  $QIPHOME/web/conf
     ```
   - On Windows, open Windows Explorer and go to *QIPHOME/web/conf*.

2. Edit the file and add the following line:

   ```
   fr=French
   ```

3. ***Optional.*** To change the default language from English to French, edit the *qip.properties* file and add the following line:

   ```
   locales.default = fr
   ```

   Note: This line sets the default language only for the initial user's login on a workstation. Subsequent logins default to the language last used by the user on that workstation.

4. Save the file.

5. Stop and start the Tomcat server.

   Note: You can now select French on the Login screen and work with a localized web client interface.

END OF STEPS

To customize specific labels

Purpose

Use this procedure to modify the text of an individual label in the VitalQIP web client interface. Each VitalQIP web client interface page is broken into three sections, each containing an individual HTML web page. You can modify a tag, for example a button name, on one specific HTML document in that page and not globally through the entire web client interface.

Before you begin

If you are customizing individual labels in a foreign language, verify the following:

- Be sure the text editor you use supports the local language you are implementing, and that you can save a file as UTF-8 (Unicode).
- Set up your workstation's environment to support the language you are implementing.

Procedure

Use the following steps to assign translated values to individual labels in the VitalQIP web client interface:

1 Change to the directory containing the *GuiCatalog_en.properties* file:

- On UNIX, enter the following command:

```
cd $QIPHOME/web/templates/web/catalog
```

- On Windows, open Window Explorer and go to *QIPHOME/web/templates/web/catalog*.

2 Edit the *GuiCatalog_en.properties* file, or the equivalent catalog file for another supported language.

> **Note:** The editor you use must support the character set of the language you are implementing, and must be able to save the file as Unicode.

The file contains a series of ***tag=value*** pairs. The tag indicates the element in the GUI to which it is assigned, and the value indicates what text displays when the GUI is opened using the language specified in this properties file.

The lower portion of this file contains ***tag=value*** pairs that are commented out. Each of these ***tag=value*** pairs are specific to one element in a specific HTML page in the web client interface.

An example of a ***tag=value*** pair for a specific screen is below:

```
AddAvailableBlockToPool.registry=Registry
```

If there are three pages with the same tag “registry”, each page can be customized so that the displayed values are distinct.

3 For the ***tag=value*** pairs you want to customize, uncomment the line and change the ***value*** to the new word you want to appear in the web client interface.

Note: Do not change the ***tag*** or `UNDEFINED` appears anywhere for this value within the web client interface.

4 Save the file.

5 Stop the Tomcat server.

6 Run **`TemplateMaker.sh`** (UNIX) or **`TemplateMaker.bat`** (Windows). The following directories are created that contain the localization files for the new language:

- *QIPHOME/web/templates/web/html_fr*
- *QIPHOME/web/templates/web/dynamic_templates/fr*

Note: You can now see the customized labels in the web client interface.

7 Start the Tomcat server.

END OF STEPS

To migrate customized or localized labels and messages

Purpose

Customized or localized GUI labels and messages can be migrated from one VitalQIP release to a current VitalQIP release. The **`CatalogMaker.sh`** (UNIX) and **`CatalogMaker.bat`** (Windows) can merge existing customized or localized labels and message with a newly delivered *GuiCatalog_en.properties* file. This way, information is not lost when upgrading to a new VitalQIP release.

Before you begin

- **`CatalogMaker.sh`** or **`CatalogMaker.bat`** are located under *QIPHOME/web/templates* after *qipTemplates.zip* (or *qipTemplate.tar.gzip*) is expanded. If a user wants to migrate existing customized/localized labels/messages, these lines in this script need to be modified:
  - **`OUTFILE`** - file name for the output from **`CatalogMaker`**. This is also the file to be used as input to **`TemplateMaker.sh`** (or **`TemplateMaker.bat`**) to generate *html* files for the GUI.
  - **`PROPERTIES`** - master catalog file delivered in the new delivery, that may contain new tag-values pairs.
  - **`OLD_PROPERTIES`** - existing catalog file that contains the customized or localization tag/value pairs that user has been using.
- Double quotation marks (i.e. ") is reserved and cannot be used as part of localized messages. The exception is for those messages already implemanted in the English version and for those messages and the resulting localized messages. For example:

```
search_error_empty_attribute_value=Attribute Value is required!
  To search for empty value, enter \"\".
```

  \"\" would not be changed in the localized version.

CatalogMaker.bat content

CatalogMaker.bat contains the following:

```
SET OUTFILE=./web/catalog/NewGuiCatalog_en.properties
SET PROPERTIES=./web/catalog/GuiCatalog_en.properties
SET OLD_PROPERTIES=./web/catalog/OldGuiCatalog_en.properties
```

Input example

The following is a usage example on UNIX:

```
CatalogMaker.sh -t "./web/default_templates/.html"
-o "./catalog/Default_QIP7_Catalog.properties"
```

```
-p
   "./catalog/addralloc_english.properties;./catalog/msg_addralloc_english.p
   roperties"
-g "./catalog/previous_catalog.properties"
```

The following is a usage example on Windows:

```
CatalogMaker.bat -t "./web/default_templates/.html"
-o "./catalog/Default_QIP7_Catalog.properties"
-p
   "./catalog/addralloc_english.properties;./catalog/msg_addralloc_english.p
   roperties"
-g "./catalog/previous_catalog.properties"
```

Output file example

In the following example, two HTML files, *Screen1.html* and *Screen2.html*, have custom labels. *Screen1.html* has labels tagged “label11”and “label12”. *Screen2.html* has the labels tagged “label21” and “label22”. The master catalog file (PROPERTIES) is *./web/catalog/addralloc_english* and contains the following:

```
label11=Label 1.1
label12=Label 1.2
label21=Label 2.1
label22=Label 2.2
```

The previously generated catalog file (OLD_PROPERTIES) is *./catalog/previous_catalog.properties* and contains the following:

```
Screen2.label21=Custom Label 2.1
```

After CatalogMaker.sh/CatalogMaker.bat is successfully executed, the resulting output file looks similar to the following:

```
# tags used in file: Screen1
# Screen1.label11=Label 1.1
# Screen1.label12=Label 1.2

# tags used in file: Screen2
Screen2.label21=Custom Label 2.1
```

Output file layout

The newly generated output file contains two sections. The first section of the output file contains all entries from all catalogs specified in the **-p** parameter.

The second section of the output file contains one entry for each tag found in the template file list specified in the **-t** parameter. These entries are organized in subsections for each template file. The entries created by prepending the name of the file followed by a dot to

each tag found, and the value being the value associated to that tag in the master catalog. All the entries in the second section are commented out. You can uncomment if you desire.

The second section also contains the all entries from the previous catalog specified in the -g parameter. If any entries in the second section of the previous catalog that are not commented out, the corresponding entries in the new catalog are uncommented and the values used in the previous catalog are used. If you want to change the value associated to a tag (label), edit the generated catalog by uncommenting the entry that matches the respective screen and tag and enter the desired value.

When the **TemplateMaker** is run on the generated catalog and if it finds an entry in the **`screenName.tag`** format, it uses the format in the file. If not, it looks for the entry that has only the tag.

To set linguistic sorting

Purpose

The VitalQIP web client does not sort by order of defined language in the pool hierarchy and MyView by default, but by binary order. As a result, the pool hierarchy results returned from a search for pools, and MyView are not displayed in the sorting order for the defined language. Instead, the information can appear out of order. Use this procedure to define the linguistic sorting order.

Note: This procedure does not need to be used when English is the defined language.

Before you begin

The VitalQIP Oracle or Sybase database must be configured for multiple language support for the defined language. Refer to the *VitalQIP Installation Guide* for more details.

Procedure

To define the sorting order for the defined language, follow these steps:

1. Go to *QIPHOME/conf* directory and copy the *linguisticSort.properties* file to a different directory. This copy is a backup in case you need to roll back to the original file.

2. Open the *linguisticSort.properties* in the *QIPHOME/web/conf* directory. The file appears as follows:

```
oracle_linguisticSort_english = GENERIC_BASELETTER
oracle_linguisticSort_french = FRENCH
oracle_linguisticSort_german = GERMAN
oracle_linguisticSort_italian = ITALIAN
oracle_linguisticSort_japanese = JAPANESE_M
oracle_linguisticSort_korean = KOREAN_M
oracle_linguisticSort_chinese = SCHINESE_PINYIN_M
sybase_linguisticSort_english = dict
sybase_linguisticSort_french = dict
sybase_linguisticSort_german = dict
sybase_linguisticSort_chinese = gbpinyin
```

3. To configure the linguistic sorting order, change the value of the properties. Supported Oracle linguistic sorting and locale data can be found at:
   - For Oracle:

http://dba-services.berkeley.edu/docs/oracle/manual-9iR2/server.920/a96529/ch4.htm

http://dba-services.berkeley.edu/docs/oracle/manual-9iR2/server.920/a96529/appa.htm#958278

- For Sybase:

http://manuals.sybase.com/onlinebooks/group-as/asg1250e/refman/@Generic__BookTextView/19612;pt=5472/*

Take this example. An administrator configures VitalQIP to use Chinese in the VitalQIP web client. Assume VitalQIP is using Oracle as its database. There are five possible values that can be assigned to **`oracle_linguisticSort_chinese`** property:

- SCHINESE_RADICAL_M
- SCHINESE_STROKE_M
- SCHINESE_PINYIN_M
- TCHINESE_RADICAL_M
- TCHINESE_STROKE_M

If the administrator does not want to use the default simplified Chinese PinYin sorting order (SCHINESE_PINYIN_M), the administrator can set the value of the **`oracle_linguisticSort_chinese`** property to use one of the other four values. For example, use SCHINESE_STROKE_M which uses number of strokes as primary order and radical as secondary order.

4 Save the file after the changes have been made.

END OF STEPS

Web client services configuration

Overview

An administrator can modify the behavior of some aspects of the VitalQIP web client interface, for example, to establish different log file settings and session timeouts, as well as to set up default contact e-mail addresses and UDA callouts. These are controlled by properties in the following files:

- *QIPHOME/tomcat/conf/server.xml*
- *QIPHOME/web/WEB-INF/web.xml*
- *QIPHOME/web/conf/log4j.properties*
- *QIPHOME/web/conf/qip.properties*
- *QIPHOME/web/conf/udaApplicationContext.xml*

These files are described in the following sections.

Tomcat server configuration

The Tomcat configuration file (*QIPHOME/tomcat/conf/server.xml*) defines parameters that control the server instance. The file is in *xml* format, where the parameters are expressed as XML Elements and attributes. The elements are arranged in the following hierarchy.

Note: The syntax for commenting in XML is:

```
<!--     -->
```

where

`<!--` is the start of the comment and `-->` is the end of the comment.

```
<Server>
    <GlobalNamingResources>
    <Service>
        <Connector>
        <Connector>
        <Engine>
            <Host>
                <Context>
```

The following table describes the syntax.

Table 18-1 Tomcat configuration elements

Element	Description
Server	The <Server> element defines which port the server listens on for stop notifications. The default is 8005. When the **`stopTomcat.sh`** script is run, the script sends a message to port 8005 to instruct the running instance to shut down.
GlobalNamingResources	The <GlobalNamingResources> and <Service> elements are not user configurable.
Service	The <Service> element should contain one or more <Connector> elements.
Connectors	The <Connectors> define the ports that the server listens on for client connections (browser or soap requests). Depending upon the options selected at installation, 1 or 2 connectors are defined. • The <Connector port="80".../> defines the parameters for NON-SSL (plain text) connections. • The <Connector port="443" .../> defines the parameters for SSL (encrypted) connections.
Host	The <Host> element is not configurable by the user.
Context	The <Context> element defines where the files for the VitalQIP application reside (defaults to *$QIPHOME/web*)

Web Service configuration

The *$QIPHOME/web/WEB-INF/web.xml* file defines how the VitalQIP Web Service is configured.

Note: Unless you want the ability to redefine logging without restarting the server, or are directed to do so by Alcatel-Lucent support, you should ***not*** modify this file.

The following XML fragment can be inserted below the `<context-param> </context-param>` that defines `log4jConfigLocation`. This instructs the logging system to check for logging configuration changes every 20000 milliseconds (20 seconds)

```
<context-param>
     <param-name>log4jConfigLocation</param-name>
     <param-value>conf/log4j.properties</param-value>
</context-param>
<context-param>
     <param-name>log4jRefreshInterval</param-name>
```

```
        <param-value>20000</param-value>
</context-param>
```

Web client logging

The VitalQIP web client uses the **log4j** programming application interface (API) to log messages for Java-based services. The **log4j** API is an open source tool that can be configured to control which log statements are logged.

log4j is configurable using a configuration file called *log4j.properties*. This file is located in the *QIPHOME/web/conf* directory. **log4j** provides flexibility in the level of logging through modification of the property files. In addition, changes to the **log4j** property files take effect dynamically within 60 seconds without the need to start and stop the associated service.

Documentation for **log4j** can be obtained at:

http://jakarta.apache.org/log4j/docs/index.html

Log SOAP error messages

SOAP error messages from the Northbound Bound interface can be logged by **log4j**. To do so, set the following:

```
log4j.logger.localized.com.lucent.qip.nb.LoggingHandler=DEBUG
```

log4j message levels

Log4j has several levels of logging filters that can range from capturing all events pertaining to the service (Debug) or only specific events associated with a particular state of the service (Fatal). These logging levels can be configured globally for an entire service or locally for specific classes in the *log4j.properties* files. Global messaging is configured in the Default Settings section, and it applies to all levels. Specific Class Level messaging is configured in the Level Setting section.

If a Global log setting is set to a less detailed event filter than a Class Level log setting, the Class Level takes precedence. The following table described the message levels.

Table 18-2 Message levels

Message level	Description
FATAL	Critical errors which cause the service or application to stop.

Message level	Description
ERROR	Errors that are unexpected, but the service/application continues processing.
WARN	Unexpected conditions that does not cause any functional problems with the service/application.
INFO	Logging of standard events and processing of service/application.
DEBUG	Very verbose output. Used for diagnosing software or system problems. **Note:** Setting `LoggingHandler=DEBUG` enables logging of both SOAP requests and responses, including error messages, over North Bound Interface.

The following table explains the most commonly changed parameters.

Table 18-3 log4J parameters

Parameter	Description
`log4j.rootCategory=WARN, RollingFile`	Specifies that WARN level messages are sent to the RollingFile appender. To log to the console, you can add `Console` to the comma-delimited list.
`log4j.category.com.lucent.XXX`	Specifies what level of output is logged for various portions of the software. This is where to turn the logging levels up/down for the particular service/application.
`log4j.appender.RollingFile.File`	Specifies the name and location of the logging output files.
`log4j.appender.RollingFile.MaxFileSize`	Specifies the size that the logging output file can grow to, before being "rolled-over" to a backup file (see next property).
`log4j.threshold=WARN`	Specifies the level of logging that occurs. Log messages that are lower than the threshold setting are not displayed.
`log4j.appender.RollingFile.MaxBackupIndex`	Specifies the number of rolled-over backup logs to keep. The most recent # number of files are kept, where # is the value specified for this property.

Sample log4j.properties file

The following is a sample *log4j.properties* file. Your configuration may vary from the configuration shown in this sample file.

```
#-------------------------------------------------------------------------
   -
#  QIP 7.0 Logging configuration file
#
#  This file is split into three sections Appenders, Global logging and
#  Module logging.
#
#  APPENDERS:
#    Define where information is logged.
#    By default, there are 3 types of appenders, Console, File, SystemLog
#
#    Console Appenders write to STDOUT,
#    File Appenders write to a file on the local file system (in
   $QIPHOME/logs)
#    SystemLog Appenders write to the Syslog (Unix) or Event Log (Windows)
#
#    Note: There are other appenders that you can use
#          see http://logging.apache.org/log4j/docs/
#
#  GLOBAL LOGGING LEVELS:
#    Defines what is logged and where it is logged.
#
#  MODULE LOGGING LEVELS:
#    Defines what is logged on a MODULE by MODULE basis.

#    Define what is logged - DEBUG, INFO, WARN, ERROR, FATAL
#      +  DEBUG level  - messages are in english
#      +  INFO level (and higher)  - messages are in the locale language
#                                    (if the translations are available)
#-------------------------------------------------------------------------
   -

#+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
   +
#   APPENDERS    - define where log messages go
#+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
   +

#-------------------------------------------------------------------------
   -
#    Default Console Logging -
```

```
#    shows time since start (ms), category, message
#-------------------------------------------------------------------------
   -
### direct log messages to stdout ###
log4j.appender.Console=org.apache.log4j.ConsoleAppender
log4j.appender.Console.Target=System.out
log4j.appender.Console.layout=org.apache.log4j.PatternLayout
log4j.appender.Console.layout.ConversionPattern=%-4r %-5p %c{1}: %m%n

#-------------------------------------------------------------------------
   -
#    Terse Console Logging -
#    shows category, message
#-------------------------------------------------------------------------
   -
### direct log messages to stdout ###
#log4j.appender.Console=org.apache.log4j.ConsoleAppender
#log4j.appender.Console.Target=System.out
#log4j.appender.Console.layout=org.apache.log4j.PatternLayout
#log4j.appender.Console.layout.ConversionPattern=%-5p %c{1}: %m%n

#-------------------------------------------------------------------------
   -
#    Minimal Console Logging
#  show message
#-------------------------------------------------------------------------
   -
### Minimal log messages to stdout ###
#log4j.appender.Console=org.apache.log4j.ConsoleAppender
#log4j.appender.Console.Target=System.out
#log4j.appender.Console.layout=org.apache.log4j.PatternLayout
#log4j.appender.Console.layout.ConversionPattern=%-5p: %m%n

#-------------------------------------------------------------------------
   -
#    Default Rolling File Settings (10MB logs with 4 backups)
#    Shows: date, category, Class, Message
#-------------------------------------------------------------------------
   -
log4j.appender.File=org.apache.log4j.RollingFileAppender
#log4j.appender.File.File=${user.dir}/qip.log
log4j.appender.File.File=/opt/qip70web/log/qip7.log
log4j.appender.File.layout=org.apache.log4j.PatternLayout
log4j.appender.File.MaxBackupIndex=4
log4j.appender.File.MaxFileSize=1000KB
log4j.appender.File.layout.ConversionPattern=%d{ISO8601} %3p %c{1}: %m%n
```

```
#-------------------------------------------------------------------------
   -
#    Verbose Rolling File Settings (10MB logs with 4 backups)
#    Shows: date, category, Class, Line number Message
#    Warning: Line number reporting is EXTREMELY SLOW only use this at the
#    direction of Lucent Techincal Support.
#-------------------------------------------------------------------------
   -
#log4j.appender.File=org.apache.log4j.RollingFileAppender
##log4j.appender.File.File=${user.dir}/qip.log
#log4j.appender.File.layout=org.apache.log4j.PatternLayout
#log4j.appender.FollingFile.MaxBackupIndex=4
#log4j.appender.File.MaxFileSize=1000KB
#log4j.appender.File.layout.ConversionPattern=%d{ISO8601} %3p %c{1}:(%L):
   %m%n

#-------------------------------------------------------------------------
   -
#    Non-Appending Rolling File Settings (10MB logs with 4 backups, new logs
   at startup)
#    Shows: date, category, Class, Message
#-------------------------------------------------------------------------
   -
#log4j.appender.File=org.apache.log4j.RollingFileAppender
##log4j.appender.File.File=${user.dir}/qip.log
#log4j.appender.File.layout=org.apache.log4j.PatternLayout
#log4j.appender.File.MaxBackupIndex=4
#log4j.appender.File.MaxFileSize=1000KB
#log4j.appender.File.Append=false
#log4j.appender.File.layout.ConversionPattern=%d{ISO8601} %3p %c{1}: %m%n

#-------------------------------------------------------------------------
   -
#    NT Syslog Settings - log to a local event log
#-------------------------------------------------------------------------
   -
# Do not forget to place the file NTEventLogAppender.dll in a directory
# that is on the PATH of the Windows system. Otherwise, you will get a
# java.lang.UnsatisfiedLinkError.

# Uncomment to use the NT Event log appender.
#log4j.appender.Syslog=org.apache.log4j.nt.NTEventLogAppender
#log4j.appender.Syslog.source=VitalQIP
#log4j.appender.Syslog.layout=org.apache.log4j.PatternLayout
#log4j.appender.Syslog.layout.ConversionPattern=%d %-5p %c{1}: %m%n
```

```
#-------------------------------------------------------------------------
   -
#    Unix Syslog Settings - log to a remote syslog
#-------------------------------------------------------------------------
   -
#Uncomment to use the unix event log appender
#log4j.appender.Syslog=org.apache.log4j.net.SyslogAppender
#log4j.appender.Syslog.sysloghost=localhost
#log4j.appender.Syslog.syslogFacility=LOG_USER
#log4j.appender.Syslog.layout=org.apache.log4j.PatternLayout
#log4j.appender.Syslog.layout.ConversionPattern=%d %-5p %c{1}: %m%n

#++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
   +
#  GLOBAL LOGGING LEVELS  - define globally what is logged
#++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
   +

#
# Lowest level of messages which will be logged.  This supersedes any
# category setting defined below.  (IF set to FATAL, only FATAL messages
# will be logged)
#
# Default=WARN

#
# When submitting bugs, please change WARN to DEBUG
log4j.threshold=WARN

#
# Defines log levels for all undefined categories (Everything except what is
# defined below). Setting this DOES NOT override explicit settings below.
#
# Also defines WHERE info will be logged TO.  The "TO" is a previously
# defined Appender ( multiple appenders are allowed)
#
# - for more verbose logging change 'WARN' to 'INFO' ###
#

log4j.rootLogger=DEBUG

log4j.logger.localized=DEBUG, File

log4j.logger.org=WARN, File
```

```
log4j.logger.com=WARN, File

log4j.logger.net=WARN, File

log4j.logger.syslog=WARN

#+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
   +
#  MODULE LOGGING LEVELS  - define specifically what is logged
#+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
   +

# Startup/Initialization messages from Tomcat and Spring
# Default - INFO
log4j.logger.org.springframework.web.context.ContextLoader=INFO
log4j.logger.org.springframework.beans.factory.xml.XmlBeanDefinitionReader=I
   NFO
log4j.logger.org.apache.catalina.startup=INFO

#
#
# QIP Components
# Default - WARN
#
# When submitting bugs, please change WARN to DEBUG.
# This will be limited by the log4j.threshold above.

#
# qip.pcy File Reader
#
log4j.logger.localized.com.lucent.qip.config=WARN

#
# Persistence Layer DAOs
#
log4j.logger.localized.com.lucent.qip.dao.impl=WARN

#
# Hibernate Support Code
#
log4j.logger.localized.com.lucent.qip.hibernate=WARN

#
# AOP Interceptors
#
log4j.logger.localized.com.lucent.qip.intercept=WARN
```

```
#
# QIP Domain Model
#
log4j.logger.localized.com.lucent.qip.model=WARN

#
# Web Service API
#
log4j.logger.localized.com.lucent.qip.nb=WARN

#
# Report Generation
#
log4j.logger.localized.com.lucent.qip.reports=WARN

#
# QIP Job Scheduler
#
log4j.logger.localized.com.lucent.qip.schedule=WARN

#
# Authentication, License Verification
#
log4j.logger.localized.com.lucent.qip.server=WARN

#
# Business Layer Implementation
#
log4j.logger.localized.com.lucent.qip.service.impl=WARN

#
# GUI/CLI Interface
#
log4j.logger.localized.com.lucent.qip.ui.gui.util=WARN

#
# Common Utilities
#
log4j.logger.localized.com.lucent.qip.utils=WARN

#
# XML-RPC Interface
#
log4j.logger.localized.com.lucent.qip.xmlrpc=WARN
```

```
#
# Hibernate Cache Provider
#
log4j.logger.org.ehcache=DEBUG

#
# Hibernate 2nd-level Cache and Query Cache
#
log4j.logger.org.hibernate.cache=DEBUG


#-------------------------------------------------------------------------
   -#
# WEB interface actions
# Default - WARN
#
#
log4j.logger.localized.com.lucent.qip.ui.gui.handler=DEBUG

#
# Set to INFO to see client Response data in XML format
# Default - WARN
#
log4j.logger.localized.com.lucent.qip.ui.gui.ControllerServlet=WARN
#
# unit tests
#
log4j.logger.test.com.lucent=INFO

#
# Backend Internals
#
log4j.logger.org.springframework=INFO
log4j.logger.org.hibernate=INFO

#
# Log in the system specific locale.
#
log4j.logger.localized.com.lucent.qip=DEBUG

#
# Log to the syslog by default at level WARN
#
#log4j.logger.syslog.com.lucent.qip=WARN
```

```
#
# SQL calls
#
log4j.logger.org.hibernate.SQL=WARN

#
# JDBC bind parameters
#
log4j.logger.org.hibernate.type=INFO

#
# Schema export/update
#   (Disabled by default)
#
#log4j.logger.org.hibernate.tool.hbm2ddl=DEBUG
#
# Database connection initialization and pooling.
# If you cannot initialize the Database connection on startup
# un-comment the "mchange" logger below,
# change "log4j.threshold=WARN" to "log4j.threshold=DEBUG" at the
# top of this file, and restart Tomcat
#
# This will should show the SQL exception returned by the database driver.
#
#   (Disabled by default)
#
#log4j.logger.com.mchange=DEBUG
```

qip.properties file

Description

When the VitalQIP web client is installed, the *qip.properties* file is installed in *QIPHOME/web/conf*. You can modify this file to change the behavior of the web client interface, as needed.

If you modify the *qip.properties* file, you need to stop and start the Tomcat server to implement the changes. Refer to "To start or stop the Tomcat server on Windows" (p. 2-16) or "To start or stop the Tomcat server on UNIX" (p. 2-22).

Properties must be specified in the `property=value` format. For example:

```
enforceNoCache=false
```

Some properties are not specifically documented in this section. Properties that indicate a page sizeor folder size for a specific element of the Web GUI exist throughout the *qip.properties* file. The page size indicates the number of items that appear on a list.

The following properties are used for the web client templates:

qippath

Value: Default: None

Allowed: Full path to $QIPHOME

Description: The full path to $QIPHOME. This value is established during the installation of VitalQIP.

Note: On Windows, the back slash (\) must be used in the path name. For example: *c:\\bin\\qip*

qip_login_server

Value: Default: None

Allowed: IP address or resolvable name

Description: The IP address or resolvable name of the VitalQIP Login Service. Refer to "qip-logind - VitalQIP Login Service daemon" (p. 2-26) for more details on this service. This value is established during the installation of VitalQIP.

qip_database

Value: Default: None

Allowed: Database name

Description: The name of the database. This value is established during the installation of VitalQIP.

locales.default

Value: Default: en

Allowed: ISO 639-1 language abbreviation

Description: Indicates the default language in use by the web client. Alternative codes are located at:

http://www.loc.gov/standards/iso639-2/php/English_list.php

Country codes may also be specified, separated by an underscore, for example en_US, or en_GB. Country codes are located at:

http://www.iso.ch/iso/en/prods-services/iso3166ma/02iso-3166-code-lists/list-en1.html

locales.file

Value: Default: *conf/locales.properties*

Allowed: Relative path and filename

Description: Indicates from which file to read the available locales.

allowTimeOutRecovery

Value: Default: False

Allowed: True | False

Description: Allows transactions that were blocked as the result of a User Session timeout to be processed. The user is prompted to complete the transaction.

session.timeout

Value: Default: 0

Allowed: Seconds

Description: Establishes the keepalive interval for individual browser sessions. The value should be set to at most half of the Tomcat session timeout to allow persistent idle connections. A value of 0 indicates that the application should not perform a session check, and idle connections will be terminated after 20 minutes (the default Tomcat session timeout value). If a different idle time is required before the Tomcat session times out, refer to "To configure session time-outs" (p. 18-65) and enter appropriate values for the **`session.timeout`** and **`session.check_interval`** properties.

session.check_interval

Value: Default: 600

Allowed: Seconds

Description: Indicates the session check interval in seconds. The value should be set to at most half of the session timeout. A value of 0 indicates that the browser should not perform a session check.

xmlrpc.client.timeout.short

Value: Default: 60

Allowed: Seconds

Description: Indicates the number of seconds before a short timeout occurs.

Various functions in VitalQIP are assigned to this timeout value. If a function assigned to this value times out repeatedly, you can increase it with this property.

xmlrpc.client.timeout.medium

Value: Default: 300

Allowed: Seconds

Description: Indicates the number of seconds before a medium timeout occurs..

Various functions in VitalQIP are assigned to this timeout value. If a function assigned to this value times out repeatedly, you can increase it with this property.

xmlrpc.client.timeout.long

Value: Default: 500

Allowed: Seconds

Description: Indicates the number of seconds before a long timeout occurs..

Various functions in VitalQIP are assigned to this timeout value. If a function assigned to this value times out repeatedly, you can increase it with this property.

appliance_manager.url

Value: Default: None

Allowed: URL

Description: Indicates where the Appiance Manager Software can be accessed from the VitalQIP web client. The format is as follows:

appliance_manager.url = http://*<servername or IP address of Appliance Manager>*:*<port number>*

appliance_manager.version

Value: Default:None

Allowed: Version number

Description: The version number of Appliance Manager Software being integrated into the VitalQIP web client.

auto_discovery.url

Value: Default: None

Allowed: URL

Description: Indicates where AutoDiscovery can be accessed from the VitalQIP web client. The format is as follows:

auto_discovery.url = http://*<servername or IP address of AutoDiscovery>*:*<port number>*

Note: Since AutoDiscovery and VitalQIP must run on the same machine, the IP address entered for this property should be the same as the machine where VitalQIP is running.

auto_discovery.version

Value: Default: None

Allowed: Version number

Description: The version number of AutoDiscovery being integrated into the VitalQIP web client.

documentRoot

Value: Default: */html*

Allowed: Directory name

Description: Indicates the root of the HTML document tree for the VitalQIP web client.

poolContact

Value: Default: None

Allowed: Valid e-mail address

Description: Provides a default value for the **Contact** field when adding a seed pool and a child pool. Enter the e-mail address of the contact person who will be notified on allocation activities, for example, ***admin@mycompany.com***. Multiple contacts are allowed.

blockNetworkContact

Value: Default: None

Allowed: Valid e-mail address

Description: Provides a default value for the **Network Contact** field when allocating a block. Enter the e-mail address of the network contact person who will be notified on allocation activities, for example, ***admin@mycompany.com***.. Multiple contacts are allowed.

customerAddress

Value: Default: None

Allowed: Street address of the customer

Description: Determines the street address of the customer that appears on the SWIP report.

customerCity

Value: Default: None

Allowed: City where the customer is located

Description: Determines the customer's city that appears on the SWIP report.

customerState

Value: Default: None

Allowed: State/Province where the customer is located

Description: Determines the state where the customer is located that appears on the SWIP report.

customerPostalCode

Value: Default: None

Allowed: Postal code of the customer

Description: Determines the zip code or postal code of the customer that appears on the SWIP report.

customerCountryCode

Value: Default: None

Allowed: Code indicating the country where the customer is located

Description: Determines the country where the customer is located that appears on the SWIP report.

publicComments

Value: Default: None

Allowed: General comments about the customer

Description: Determines text comments that appears on the SWIP report.

privateCustomerName

Value: Default: Pool name

Allowed: Name of the customer

Description: Determines the name of the customer that appears on the SWIP report. Overrides the value in the Customer Name field on the SWIP Report.

privateCustomerAddress

Value: Default: None

Allowed: Additional address information for the customer

Description: Determines the street address of the customer that appears on the SWIP report.

folderSize

Value: Default: 50

Allowed: Number greater than 0

Description: Determines the number of infrastructure objects (pools, blocks, IPv4 addresses, and IPv6 addresses) that may appear in a hierarchical view before a folder is created to hold them. A folder is displayed only if the total number of objects at the same level is equal to or greater than the value entered for this property.

reclaimAddressPageSizes

Value: Default: 62,126,254

Allowed: Comma separted numeric list

Description: Folds the values for the desired page sizes on the Reclaim Details page. The drop down for setting the displayable page size of the table is set to these values. The first value in the list is the default value for the page size. The values are separated by commas.

reclaimNetworkStatusPageSizes

Value: Default: 50,100,250

Allowed: Comma separate numeric list

Description: Holds the values for the desired page sizes on the Network Reclaim Status page. The drop down for setting the displayable page size of the table will be set to these values. The first value in the list is the default value for the page size. The values should be separated by commas.

manageObjectsPageSizes

Value: Default: 62,126,254

Allowed: Comma separated numeric list

Description: Holds the values for the desired page sizes on the Manage Objects page. The drop down for setting the displayable page size of the table is set to these values. The first value in the list is the default value for the page size. The values are separated by commas.

manageSubnetsPageSizes

Value: Default: 62,126,254

Allowed: Comma separated numeric list

Description: Holds the values for the desired page sizes on the Manage Subnets page. The drop down for setting the displayable page size of the table is set to these values. The first value in the list is the default value for the page size. The values are separated by commas.

zoneSearchPageSizes

Value: Default: 50,100,250

Allowed: Comma separated list

Description: Holds the values for the desired page sizes on the Zone Search page. The drop down for setting the displayable page size of the table is set to these values. The first value in the list is the default value for the page size. The values are separated by commas.

DOMAIN_QUICK_VIEW_PAGE_SIZE

Value: Default: 350

Allowed: Numeric

Description: Determines the Domain Quick View page size in which the quci View report code retrieves data from the database.

NETWORK_QUICK_VIEW_PAGE_SIZE

Value: Default: 350

Allowed: Numeric

Description: Determines the Network Quick View page size in which the quci View report code retrieves data from the database.

SUBNET_QUICK_VIEW_PAGE_SIZE

Value: Default: 350

Allowed: Numeric

Description: Determines the Subnet Quick View page size in which the quci View report code retrieves data from the database.

reportFileTransferLimit

Value: Default: 5242880 (5 MB)

Allowed: Bytes

Description: Holds the size limit of the report file that is transferred to the client. If the generated report file size is greater than the specified value, the system does not transfer the file. Instead, it copies the file to a *temp* directory and sends the file details, which are displayed to the user.

file.resource.loader.cache

Value: Default: True

Allowed: True | False

Description: Tells the Template Engine whether to cache HTML templates. To use this property, the **`file.resource.loader.modificationCheckInterval`** property must be set. Descriptions of values are as follows:

- **`True`** - caches HTML templates. Use this value to improve the performance of the web client interface. The templates will not be loaded and compiled for every transformation.
- **`False`** - does not cache HTML templates.

file.resource.loader.modificationCheckInterval

Value: Default: 0

Allowed: Seconds

Description: If **`file.resource.loader.cache`** property is set to **`True`**, this property must be set.

Enter the interval in seconds for the Template Engine to check for revised files. If this property set to 0, the Template Engine always checks for revised files.

file.resource.loader.path

Value: Default: *dynamic_templates*

Allowed: Directory name

Description: The directory that contains the templates for the web client interface.

runtime.log

Value: Default: velocity.log

Allowed: Filename

Description: The name of the log file for template transformation messages.

runtime.log.logsystem.class

Value: Default: org.apache.velocity.runtime.log.NullLogSystem

Allowed: Java class

Description: The class used for logging transformation messages.

useFileIO

Value: Default: True

Allowed: True | False

Description: Indicates the type of input/output to use with underlying executables. If this property is set to **`true`**, temporary files are written that are used by the executables. If this property is set to **`false`**, data is written to the executable's STDIN stream.

Note: FileIO is slower than Stream IO, but makes debugging easier since the data files are available for viewing.

deleteTempFiles

Value: Default: True

Allowed: True | False

Description: Prevents temporary files from being deleted. Set this property to **`false`** if **`FileIO`** is set to **`true`** and you wish to preserve temporary files.

showIcon.<icon> properties

Value: Default: true

Allowed: true | false

Description: Determines whether or not the toolbar icons are displayed in the toolbar menu. Note that:

- Only a feature is applicable to the administrator and its corresponding preference value shown in the following table is set the true. The feature's icon is displayed in the toolbar menu.
- Settings have no impact on the visibility of main menu items.

The following table describe the showIcon properties for each icon.

Table 18-4 Descriptions for showIcon<icon> properties

Property	Description
showIcon.MY_VIEW	Shows or hides the MyView Hierarchy icon.
showIcon.MY_VIEW_PERSONALIZATION	Shows or hides the MyView Personalization icon.
showIcon.IPV4_MANAGEMENT	Shows or hides the IPv4 Management icon.
showIcon.IPV6_MANAGEMENT	Shows or hides the IPv6 Management icon.
showIcon.NODE_MANAGEMENT	Shows or hides the Node Management icon.
showIcon.ACL_TEMPLATE	Shows or hides the ACL Templates icon.
showIcon.DOMAIN	Shows or hides the Zone icon.
showIcon.DNS_SERVER	Shows or hides the DNS Server icon.
showIcon.NON_MANAGED_DNS_SERVER	Shows or hides the Non-Managed DNS Server icon.
showIcon.DNS_GENERATION	Shows or hides the DNS Generation icon.

Property	Description
showIcon.MANAGED_FILES	Shows or hides the Managed Files icon.
showIcon.NETWORK_ACCESS_CONTROL	Shows or hides the Network Access Control icon
showIcon.DHCP_SERVER	Shows or hides the DHCP server icon.
showIcon.DHCP_TEMPLATE	Shows or hides the DHCP Template icon.
showIcon.DHCP_CLIENT_CLASS	Shows or hides the DHCP Client Class icon.
showIcon.POOL_HIERARCHY	Shows or hides the Pool Hierarchy icon.
showIcon.BLOCK_HIERARCHY	Shows or hides the Block Hiearachy icon.
showIcon.RULE_HIERARCHY	Shows or hides the Rules icon.
showIcon.INTERNET_REG_HIERARCHY	Shows or hides the Internet Registries icon.
showIcon.DC_SERVER	Shows or hides the Domain Controller icon
showIcon.MY_VIEW_MANAGEMENT	Shows or hides the MyView Management icon.
showIcon.TEMPLATE_HIERARCHY	Shows or hides the Templates icon.
showIcon.SCHEDULER	Shows or hides the Job Scheduler icon.
showIcon.ATTRIBUTES	Shows or hides the Attributes icon.
showIcon.OBJECT_CLASSES	Shows or hides the Object Classes icon.
showIcon.GLOBAL_POLICIES	Shows or hides the Global Policies icon.
showIcon.CONTACT	Shows or hides the Contact icon.
showIcon.LOCATION	Shows or hides the Location icon.
showIcon.ADMINISTRATOR	Shows or hides the Administrators icon.
showIcon.ADMIN_ROLE	Shows or hides the Admin Roles icon.
showIcon.ADMIN_SECURITY	Shows or hides the Adminstrator Security icon.
showIcon.REPORTS	Shows or hides the Reports icon.
showIcon.AUTO_DISCOVERY	Shows or hides the AutoDiscovery icon.
showIcon.APPLIANCE_MANAGER	Shows or hides the Appliance Manager icon.
showIcon.CUSTOMIZED_ADD_ONS	Shows or hides the Customized Links icon.
showIcon.ONLINE_DOCUMENT	Shows or hides the Online Documents icon.
showIcon.ONLINE_HELP	Shows or hides the Online Help icon.
showIcon.ABOUT_VITALQIP	Shows or hides the About VitalQIP icon.

dhcpServer.DefaultDirectory

Value: Default: */opt/qip/dhcp*

Allowed: Path to directory

Description: Determines the default value for the **Default Directory** parameter in the DHCP Profile.

dhcpServer.PingDelay

Value: Default: 500

Allowed: Seconds

Description: Determines the default value for the **PingDelay** parameter in the DHCP Profile.

dhcpServer.DHCPTemplate

Value: Default: general

Allowed: general, microsoft_clients, or multicast

Description: Determines the default value for the **DHCP Template** parameter in the DHCP Profile.

dhcpServer.FailoverServerType

Value: Default: Standalone/Primary

Allowed: Standalone/Primary or Failover/Secondary

Description: Determines the default value for the **Failover Server Type** parameter in the DHCP Profile.

dhcpServer.ManagedRange

Value: Default: Corporation

Allowed: Corporation, Domain, Network, OSPF Area, Subnet Organization, or Subnet

Description: Determines the default value for the **Managed Range** parameter in the DHCP Profile.

disableDNSCheckboxes

Value: Default: False

Allowed: True or False

Description:

Allows you to disable DNS checkboxes in the Web Client.

disableNormalAdmUpdates

Value: Default: False

Allowed: True | False

Description: Controls whether or not a normal administrator can perform operations other than add or search for zones, networks, subnets and subnet organizations.

enable71WebServiceApi

Value: Default: False

Allowed: True | False

Description: Controls whether VitalQIP 7.1 release compatible web service based API inteface is enabled or not. The 7.1 release APIs (VQIPManagerPort) are deprecated. Use the VitalQIP 7.2 release or later APIs.

enforceNoCache

Value: Default: False

Allowed: True | False

Description: Ignores the browser cache setting on the client and disables caching. The default allows the browser client (or proxy) to cache responses.

managedFileDisplaySize

Value: Default: 1048576 (1MB)

Allowed: Bytes

Description: Defines the size limit of managed files that can be displayed in the web client.

managedFileMaxSizeSoftLimit

Value: Default: 2097152 (2MB)

Allowed: Bytes

Description: Defines the size limit of managed files that can be downloaded via the web client. VitalQIP does not enforce a hard limit on the maximum size of a managed file, but performance will degrade if the managed file is larger than 2MB.

ruleNamePrefix

Value: Default: None

Allowed: String

Description: Identifies the string appended to the CIDR block mask to derive the rule name to use for block assign operation over Northbound SOAP Interface.

PopulateAcSubscriberHierarchy

Value: Default:

Allowed: True|False

Description: This policy specifies whether the subscriber hierarchy is prepopulated instead of presenting the search screen and having the user search for entries to add to the hierarchy.

PopulateAcDeviceHierarchy

Value: Default:

Allowed: True|False

Description: This policy specifies whether the device hierarchy is prepopulated instead of presenting the search screen and having the user search for entries to add to the hierarchy.

Sample qip.properties file

The following is a sample *qip.properties* file:

```
#
# Path to $QIPHOME
#
#qippath=c:\\bin\\qip60sp1
qippath=/opt/qip

#
# IP Address or resolvable name of the QIPLoginService
#
#qip_login_server=##.###.##.###
qip_login_server=135.114.106.35

#
# The name of the database
#
#qip_database=labsun02
qip_database=orasol10

#
# Default locale
#
locales.default=en
```

```
#
# File to read the available locales from
#
locales.file=conf/locales.properties

#
# Allows transactions that were blocked as the result of a User
   Session
# timeout to be processed. The user will be prompted to complete
   the
# transaction.
#
allowTimeOutRecovery=true

#
# Session timeout in seconds. Value should be set to at most 1/2
   of the
# tomcat session timeout. A value of 0 indicates that the
   application
# should not time out the session.
#
session.timeout=0

#
# Session check interval in seconds. Value should be set to at
   most 1/2
# of the session timeout. A value of 0 indicates the browser
   should not
# perform a session check.
#
session.check_interval=600


#################################################################
   #######
#
# Add-On Properties
#
#################################################################
   #######

#
# Timeouts for browser operations.  These lines should remain
   commented out unless users are experiencing
# frequent timeouts.
#
```

```
#xmlrpc.client.timeout.short = 60
#xmlrpc.client.timeout.medium = 300
#xmlrpc.client.timeout.long = 900

#
# Set the following two properties to integrate Appliance Manager
   add-on.
#
# URL to access the Appliance Manager add-on.
# Syntax: appliance_manager.url = http://<servername or IP address
   of Appliance Manager>:<port number>
# Ex: http://localhost:8080, Default Port: 8080
#appliance_manager.url=
# Version of the Appliance Manager add-on
#appliance_manager.version = 1.4

#
# Set the following two properties to integrate Auto Discovery
   add-on.
#
# URL to access the Auto Discovery add-on.
# Syntax: auto_discovery.url = http://<servername or IP address of
   AutoDiscovery>:<port number>
# Ex: http://localhost:8082, Default Port: 8082
#auto_discovery.url=
# Version of the Auto Discovery add-on
#auto_discovery.version = 2.3

#
# The root of the HTML document tree for Network Allocator
#
documentRoot=/html


#################################################################
   #######
#
# Contact Defaults
#
#################################################################
   #######

#
# The default value for field Contact for adding Seed Pool and
   child Pool.
```

```
# Fill the email address of the contact person who will be
   notified on
# allocation activities (multiple contacts will be allowed).
# Ex. admin@mycompany.com
#
#poolContact=

#
# The default value for field Network Contact for allocating
   block.
# Fill the email address for the Network, if Network is to be
   created in QIP.
# Ex. admin@mycompany.com
#
#blockNetworkContact=


##################################################################
   #######
#
# Customer Defaults
#
##################################################################
   #######

#
# The default value for field 'Customer Address' for generating
   ARIN-Reassign-Simple Report.
# If more than one line needed, use '\n' to represent new line.
# Ex. 12345 my Ave\nSuite E-1
#
#customerAddress=

#
# The default value for field 'Customer City' for generating ARIN-
   Reassign-Simple Report.
# Ex. New York
#
#customerCity=

#
# The default value for field 'Customer State/Province' for
   generating ARIN-Reassign-Simple Report.
# Enter the two-letter abbreviation of the state or province of
   the customer receiving the reassignment.
# Ex. PA
```

```
#
#customerState=

#
# The default value for field 'Customer Postal Code' for
   generating ARIN-Reassign-Simple Report.
# Ex. 19355
#
#customerPostalCode=

#
# The default value for field 'Customer Country Code' for
   generating ARIN-Reassign-Simple Report.
# Enter the ISO-3166 two-character country code of the customer
   receiving the reassignment.
# You can find a complete listing of country codes at:
   http://www.arin.net/community/countries.html
# Ex. 'US' For the UNITED STATES. 'GB' for the UNITED KINGDOM.
#
#customerCountryCode=

#
# The default value for field 'Public Comments' for generating
   ARIN-Reassign-Simple Report.
# Ex. This is for NY branch.
#
#publicComments=

#
# The 'private' default value for field 'Customer Name' for
   generating ARIN-Reassign-Simple Report.
# This is for the blocks size of /25 and smaller and the
   'Implement Private Name' box is checked
# while allocating block.
# Ex. Private Customer - XYZ Network
#
#privateCustomerName=

#
# The 'private' default value for field 'Customer Address' for
   generating ARIN-Reassign-Simple Report.
# This is for the blocks size of /25 and smaller and the
   'Implement Private Address' box is checked
# while allocating block. If more than one line needed, use '\n'
   to represent new line.
# Ex. 12345 my Ave\nSuite E-1
#
```

```
#privateCustomerAddress=


#################################################################
   #######
#
# Foldering Properties
#
#################################################################
   #######

# Generic folder size for objects in the various UI hierarchies
   (Domains,
# Subnets, Networks, etc).
# Note: Value can be overridden by enabling and setting the
   specific one
# for each feature listed below. Default=250
#
folderSize=250

#
# Folder size for the ACL Template objects in hierarchy display.
#
#ACLTemplateFolderSize=

#
# Folder size for the AC Device objects in hierarchy display.
#
#AcDeviceFolderSize=

#
# Folder size for the AC Subscriber objects in hierarchy display.
#
#AcSubscriberFolderSize=

#
# Folder size for the Address Pool objects in hierarchy display.
# Note: size is applicable to child pools only, folders for seed
   pools
# are not supported.
#
#AddressPoolFolderSize=

#
# Folder size for the Administrative Role objects in hierarchy
   display.
```

```
#
#AdministrativeRoleFolderSize=

#
# Folder size for the Administrator objects in hierarchy display.
#
#AdministratorFolderSize=

#
# Folder size for the Application objects in hierarchy display .
#
#ApplicationFolderSize=

#
# Folder size for the DHCP Client Class objects in hierarchy
  display.
#
#DHCPClientClassFolderSize=

#
# Folder size for the DHCP Server objects in hierarchy display.
#
#DHCPServerFolderSize=

#
# Folder size for the DHCP Template objects in hierarchy display.
#
#DHCPTemplateFolderSize=

#
# Folder size for the DNS Server objects in hierarchy display.
#
#DNSServerFolderSize=

#
# Folder size for the Domain Controller objects in hierarchy
  display.
#
#DomainControllerFolderSize=

#
# Folder size for the DNS View objects in hierarchy display.
#
#DNSViewFolderSize=
```

```
#
# Folder size for the IPv4 Address Block objects in hierarchy
  display.
#
#IPv4AddressBlockFolderSize=

#
# Folder size for the IPv4 Address objects in hierarchy display.
#
#IPv4AddressFolderSize=

#
# Folder size for the IPv4 Address Range objects in hierarchy
  display.
#
#IPv4AddressRangeFolderSize=

#
# Folder size for the IPv4 Object Range objects in hierarchy
  display.
#
#IPv4ObjectRangeFolderSize=

#
# Folder size for the IPv4 Subnet objects in hierarchy display.
#
#IPv4SubnetFolderSize=

#
# Folder size for the IPv6 Address Block objects in hierarchy
  display.
#
#IPv6AddressBlockFolderSize=

#
# Folder size for the IPv6 Address objects in hierarchy display.
#
#IPv6AddressFolderSize=

#
# Folder size for the IPv6 Address Range objects in hierarchy
  display.
#
#IPv6AddressRangeFolderSize=

#
```

```
# Folder size for the IPv6 Subnet objects in hierarchy display.
#
#IPv6SubnetFolderSize=

#
# Folder size for the Managed File objects in hierarchy display.
#
#ManagedFileFolderSize=

#
# Folder size for the Available and Managed lists in the
   Administrator
# Managed List display. Default=100
#
managedListFolderSize=100

#
# Folder size for the MyView objects in hierarchy display.
#
#MyViewFolderSize=

#
# Folder size for the Network objects in hierarchy display.
#
#NetworkFolderSize=

#
# Folder size for the Node objects in hierarchy display.
#
#NodeFolderSize=

#
# Folder size for the Mon-Managed DNS Server objects in hierarchy
   display.
#
#NonManagedDNSServerFolderSize=

#
# Folder size for the OSPF Area objects in hierarchy display.
#
#OSPFAreaFolderSize=

#
# Folder size for the Subnet Organization objects in hierarchy
   display.
#
```

```
#SubnetOrgFolderSize=

#
# Folder size for the Zone objects in hierarchy display.
#
#ZoneFolderSize=


#####################################################################
   #######
#
# Pagination Properties
#
#####################################################################
   #######

# Generic page size for objects in the various UI hierarchies
   (Domains,
# Subnets, Networks, etc).
# Note: Values can be overridden by enabling and setting the
   specific one
# for each feature listed below. Default=50,100,250
#
pageSizes=50,100,250

#
# This property holds the values for the desired page sizes on the
   AC Devices page.
# The Drop Down for setting the page size of the table will be set
   to these values.
# The first value in the list is the default value for the page
   size.
# The values should be separated by commas.
#
acDevicePageSizes=50,100,250

#
# This property holds the values for the desired page sizes on the
   AC Subscribers page.
# The Drop Down for setting the page size of the table will be set
   to these values.
# The first value in the list is the default value for the page
   size.
# The values should be separated by commas.
#
acSubscriberPageSizes=50,100,250
```

```
#
# This property holds the values for the desired page sizes on the
   Manage Objects page.
# The Drop Down for setting the page size of the table will be set
   to these values.
# The first value in the list is the default value for the page
   size.
# The values should be separated by commas.
#
manageObjectsPageSizes=62,126,254

#
# This property holds the values for the desired page sizes on the
   Manage Subnets page.
# The Drop Down for setting the page size of the table will be set
   to these values.
# The first value in the list is the default value for the page
   size.
# The values should be separated by commas.
#
manageSubnetsPageSizes=62,126,254

#
# This property holds the values for the desired page sizes on the
   Reclaim Details page.
# The Drop Down for setting the page size of the table will be set
   to these values.
# The first value in the list is the default value for the page
   size.
# The values should be separated by commas.
#
reclaimAddressPageSizes=62,126,254
#
# This property holds the values for the desired page sizes on the
   Network Reclaim Status page.
# The Drop Down for setting the page size of the table will be set
   to these values.
# The first value in the list is the default value for the page
   size.
# The values should be separated by commas.
#
reclaimNetworkStatusPageSizes=50,100,250

#
# This property holds the values for the desired page sizes on the
   Zone Search page.
```

```
# The Drop Down for setting the page size of the table will be set
   to these values.
# The first value in the list is the default value for the page
   size.
# The values should be separated by commas.
#
zoneSearchPageSizes=50,100,250

#
# This property holds the values for the desired page sizes on the
   View Site Associations page.
# The Drop Down for setting the page size of the table will be set
   to these values.
# The first value in the list is the default value for the page
   size.
# The values should be separated by commas.
#
SiteAssociationPageSizes=50,100,250

##################################################################
   ###################
#
# Report Engine Properties
#
##################################################################
   ###################

# Setting of DEFAULT_FETCH_SIZE & SPECIAL_FETCH_SIZE parameters
   will not impact the
# records included in the report.
# These properties can be tuned to optimize report performance.
   They control the
# fetch size the report engine uses to retrieve data from the DB.
# Default fetch size used by most of the reports.
DEFAULT_FETCH_SIZE=1000

# Special fetch size used by huge reports like DHCP Utilization,
   Object Info, Subnet Info,
# Node Search, V4 Object Search, RR Search and Global Search.
SPECIAL_FETCH_SIZE=10000

#Reports data can be restricted to Maximum Fetch Records
# Setting MAX_FETCH_RECORDS = -1 will remove restriction (i.e.
   display all records).
MAX_FETCH_RECORDS = 100000

# Fetch size used for Quick View reports.
```

```
DOMAIN_QUICK_VIEW_PAGE_SIZE=350
NETWORK_QUICK_VIEW_PAGE_SIZE=350
SUBNET_QUICK_VIEW_PAGE_SIZE=350

#Fetch size used for subnetJoin calculated list
SUBNETJOIN_LIST_PAGESIZE=350

#
# This property holds the limit size of the report file that will
   be
# transferred to the client. If the generated report file size is
   greater
# than this configuration, system does not transfer the file,
   Instead it
# copies to temp directory and sends the file details, which is
   displayed
# to the user. Default value is set to 5242880 (5M)
#
reportFileTransferLimit=5242880


#########################################################################
   #######
#
# Template Engine Properties
#
#########################################################################
   #######

#
# Tell the Velocity Template Engine whether to cache html
   templates.
# Setting to true will increase performance, since templates will
# not need to be loaded and compiled for every transformation.
#
file.resource.loader.cache=false

#
# If caching is set to true, identify the number of seconds the
   Template
# Engine should check for revised files. 0=check every time.
#
file.resource.loader.modificationCheckInterval=0

#
```

```
# Directory which contains the templates for the Network Allocator
   Web
# Interface
#
file.resource.loader.path=dynamic_templates

#
# Name of the log file for template transformation messages
#
runtime.log=velocity.log

#
# Class used for logging transformation messages
#
runtime.log.logsystem.class=org.apache.velocity.runtime.log.NullL
   ogSystem

#
# Type of IO to use with underlying executables.
#
# If useFileIO is true, temporary files are written
# which are used by the executables.
#
# If useFileIO is set to false, data is written to the
   executable's
# STDIN stream.
#
# FileIO is slower than Stream IO, but makes debugging easier
   since the
# data files are available for viewing.
#
useFileIO=false

#
# If FileIO is being used, setting the following to false will
   prevent
# the temporary files from being deleted.
#
deleteTempFiles=true


#################################################################
   #######
#
# Toolbar Preferences
#
```

```
#################################################################
   #######
#
# This determines whether or not the toolbar icons will be
   displayed in
# the toolbar menu.
# Notes:
# 1. An icon will be displayed for a toolbar feature only if that
   feature is
#    applicable to the type of administrator logged in and its
   corresponding
#    preference value is set to true.
# 2. Settings have no impact on the visibility of main menu items.
#
showIcon.MY_VIEW=true
showIcon.MY_VIEW_PERSONALIZATION=true

showIcon.IPV4_MANAGEMENT=true
showIcon.IPV6_MANAGEMENT=false
showIcon.NODE_MANAGEMENT=false

showIcon.ACL_TEMPLATE=false
showIcon.DOMAIN=true
showIcon.DNS_SERVER=true
showIcon.NON_MANAGED_DNS_SERVER=true
showIcon.DNS_GENERATION=true
showIcon.MANAGED_FILES=false

showIcon.NETWORK_ACCESS_CONTROL=true
showIcon.DHCP_SERVER=true
showIcon.DHCP_TEMPLATE=true
showIcon.DHCP_CLIENT_CLASS=true

showIcon.POOL_HIERARCHY=false
showIcon.BLOCK_HIERARCHY=false
showIcon.RULE_HIERARCHY=false
showIcon.INTERNET_REG_HIERARCHY=false

showIcon.DC_SERVER=true
showIcon.MY_VIEW_MANAGEMENT=true
showIcon.TEMPLATE_HIERARCHY=false
showIcon.SCHEDULER=true
showIcon.ATTRIBUTES=true
showIcon.OBJECT_CLASSES=false
showIcon.GLOBAL_POLICIES=false
showIcon.CONTACT=false
```

```
showIcon.LOCATION=false
showIcon.NETWORK_ACCESS_CONTROL=false

showIcon.ADMINISTRATOR=true
showIcon.ADMIN_ROLE=true
showIcon.ADMIN_SECURITY=true

showIcon.REPORTS=true

showIcon.AUTO_DISCOVERY=true
showIcon.APPLIANCE_MANAGER=true
showIcon.CUSTOMIZED_ADD_ONS=true

showIcon.ONLINE_DOCUMENT=true
showIcon.ONLINE_HELP=true
showIcon.ABOUT_VITALQIP=true


#################################################################
   #######
#
# Web GUI Properties
#
#################################################################
   #######

#
# DHCP server parameter defaults for the web gui.
# These are used if the db parameter does not have a default.
# These are the required parameters.
#
dhcpServer.DefaultDirectory=/opt/qip/dhcp
dhcpServer.PingDelay=500
dhcpServer.DHCPTemplate=general
dhcpServer.FailoverServerType=Standalone/Primary
dhcpServer.ManagedRange=Corporation

#
# This property will disable the web interface Static and Dynamic
   DNS
# checkboxes. The checkboxes on the Object Profile page will be
   disabled.
# The values will be shown but will not be altered from the web
   interface.
#
disableDNSCheckboxes=false
```

```
#
# This property will disable non-Add( modify, delete, split, join
   )
# on the web interface for Subnets.
#
disableNormalAdmUpdates=false

#
# This property controls whether VitalQIP 7.1 release compatible
   web
# service based API interface is enabled or not. The 7.1 release
   APIs
# (VQIPManagerPort) are deprecated and VitalQIP 7.2 release or
   later
# APIs should be used.
#
enable71WebServiceApi=false

#
# Ignore the browser cache setting on the client and disable
   caching
# Default=false (allows browser client (or proxy) to cache
   responses)
#
enforceNoCache=false

#
# Size limit of managed files that can be displayed via the GUI.
# Default is 1048576 (1M)
#
managedFileDisplaySize=1048576

#
# Size limit of managed files that can be uploaded via the GUI.
# Default is 2097152 (2M)
#
managedFileMaxSizeSoftLimit=2097152

#
# Identifies the string appended to the CIDR block mask to derive
   the
# rule name to use for block assign operation over NA SOAP
   Interface.
#
#ruleNamePrefix=
```

```
#
# Determines if AC Subscriber hierarchy will be shown or
# must be searched and built
#
#populateAcSubscriberHierarchy=

#
# Determines if AC Device hierarchy will be shown or
# must be searched and built
#
#populateAcDeviceHierarchy=
```

To configure session time-outs

Purpose

You can configure the length of the inactivity period before the VitalQIP web client times out if you find that the 20-minute default is insufficient.

Procedure

To define a Tomcat session timeout, follow these steps:

1. Open *$QIPHOME/web/WEB-INF/web.xml* (on UNIX), or *%QIPHOME%\web\WEB_INF\web.xml* on Windows, in a text editor.

2. Insert a `<session-config>` element inside the `<web-app> ... </web-app>` tags. In the following example, a 40-minute time-out is established:

```
<web-app>
    <display-name>QIP7</display-name>

...

<session-config>
    <session-timeout>40</session-timeout>
  </session-config>

</web-app>
```

3. Save the *web.xml* file.

4. Open the *qip.properties* file in a text editor.

5. Change the value of the **session.timeout** and **session.check_interval** properties as needed. For information on these properties, refer to "qip.properties file" (p. 18-33).

6. Save the *qip.properties* file.

7 Stop and start the Tomcat server to implement the changes.

END OF STEPS

To enable authentication via the Login Service

Purpose

The web service authenticates against the database by default. It can be configured to use the Login Service. Use this procedure to enable the authentication callout for the Web Service.

Before you begin

- Refer to "Administrator authentication tool" (p. 22-2) for more information on the authenication callout.
- Refer to *VitalQIP Web Client User Guide* for information on starting and stopping the Apache Tomcat server.

Procedure

To enable the authentication callout for the web server:

1. Stop the Tomcat server.

2. In the application context file *ApplicationContext.xml* located in the *$QIPHOME/web/conf/ApplicationContext.xml* directory.

3. Uncomment the **`qipCallout`** bean:

```
<bean id="qipCallout" class="com.lucent.qip.auth.QipCallout">
     <property name="qipConfig"><ref bean="qipConfig"/></property>
</bean>
```

4. Uncomment the **`commonService`** bean, so it can use the **`qipCallout`** bean.

```
<bean id="commonService" ...
     ...
     <property name="callout"><ref bean="qipCallout"/></property>
     ...
</bean>
```

5. ***Optional.*** By default, the **`qipConfig`** bean checks the *qip.pcy* file to retrieve Login Service information. In some deployments (such as with SSL), it can be more appropriate

to configure the Login Service information without using the *qip.pcy* file. To do so, add the **loginServer** property to the **`qipConfig`** bean. For example:

```
<bean id="qipConfig" class="com.lucent.qip.config.QipConfigImpl"
scope="singleton">
...
   <property name="loginServer">
     <value>127.0.0.1</value>
   <property>
</bean
```

6 Save the *ApplicationContext.xml* file.

7 Start the Tomcat server.

END OF STEPS

To enable compatibility with VitalQIP 7.0/7.1 Web Service

Purpose

In VitalQIP 7.2, the VitalQIP Web Service (SOAP interface) has been modified to be Web Services Interoperability (WS-I) compliant and therefore is not compatible with SOAP clients implemented for the VitalQIP 7.0/7.1 Web Service. Alcatel-Lucent strongly recommends that SOAP clients already using the existing VitalQIP 7.0/7.1 Web Service be updated to the VitalQIP 7.2 WSDL. Alternatively, they can continue to run as is after the transitional 7.0/7.1 Web Service provided only in VitalQIP 7.2 has been enabled.

Before you begin

The VitalQIP 7.0/7.1-compatible Web Service is accessible at the following URL (the same URL as in previous releases):

`http://<WebServerIPAddress>:<WebServerPort>/services/VQIPManagerPort`

where ***`<WebServerIPAddress>`*** and ***`<WebServerPort>`*** correspond to the IP Address and port being used by VitalQIP Tomcat Web Server.

Procedure

To enable the transitional 7.0/7.1 Web Service (SOAP interface) in VitalQIP, follow these steps:

1. Open *$QIPHOME/web/conf/qip.properties* (on UNIX) or *%QIPHOME%\web\conf\qip.properties (*on Windows) in a text editor.

2. Change the value of **`enable71WebServiceApi`** property to true. For information on this property, refer to "PopulateAcSubscriberHierarchy" (p. 18-47).

3. Save the *qip.properties* file.

4. Stop and start the Tomcat server to implement the changes.

END OF STEPS

UDA callout

The VitalQIP web client allows User-defined attribute to be customized by customers. This provides information about User-defined attributes.

UDA callout configuration

Purpose

The User-defined attribute feature in the web client interface contains three callout fields where customers who have created their own Java callout classes can select them:

- **Pre-Edit Callout** - implements the **`UdaPreEditCallout`** interface. This is called by **`udaService`** before it sends UDA data to the web client GUI, and sets the UDA value to the value returned by the callout.
- **Post-Edit Callout** - implements the **`UdaPostEditCallout`** interface. This is called by **`udaService`** before it stores in the database the UDA data that was entered in the GUI, and sets the UDA value to the value returned by the callout.
- **Validation Callout** - implements the **`UdaValidationCallout`** interface. This is called by **`udaService`** before it stores in the database the UDA data that was entered in the GUI, and returns true or false. If it returns false, **`udaService`** issues an exception and aborts the action. In this case, the GUI user sees an error message, and needs to enter a UDA value that complies with the validation.

  Note: The result of a Post-Edit Callout overwrites the result of a Pre-Edit Callout if a UDA contains both a Pre-Edit Callout and a Post-Edit Callout.

Sample callouts are supplied with the VitalQIP installation and are located in the `UDA Callouts` section of *QIPHOME/web/ conf/udaApplicationContext.xml*.

Sample callout usage

In this example, the callout types shown in the following table have been established.

Table 18-5 Sample callout settings

Callout type	Key	Function description
preEditCallout	setToUserId	Sets the UDA value to login ID
postEditCallout	setToUserId	Sets the UDA value to login ID
validationCallout	noValidation	Does nothing

1. An address block is assigned a UDA named **`createdBy`**, which is associated with the preEditCallout **`setToUserId`**.
2. When a user logs in as `qipman` and creates a block, the web client interface sends a request to **`udaService`** to get the list of UDAs assigned.
3. The UDA service calls the preEditCallout **`setToUserId`** and sets the value of **`createdBy`** to `qipman`.
4. The UDA service returns **`createdBy`** with the value `qipman` to the web client UI.
5. On the block creation screen, the **Maintainer** field already contains the value `qipman`. If the UDA is read-only, the value `qipman` is greyed out so that the user cannot make any changes.

   If the UDA **`createdBy`** had a postEditCallout **`setToUserId`** and the UDA were not read-only, `qipman` would be saved in the database no matter what was entered by the user.

Java callout classes

UDA callouts

A UDA callout is a java class that implements some or all of the UDA callout interfaces:

- *UdaPreEditCallout.java*
- *UdaPostEditCallout.java*
- *UdaValidationCallout.java*

The UDA callout is added to *udaApplicationContext.xml* as a bean so that it can be loaded when VitalQIP is started. To use a pre-edit or post-edit callout for a UDA, the callout name (also known as bean name) must be specified in the UDA definition. A validation callout is for a UDA value type.

The java classes are as follows.

UdaPreEditCallout.java interface

```
package com.lucent.qip.utils;

public interface UdaPreEditCallout {
   public Object preEditCallout(String orgName, String
    adminUserName, Object value, Class type);
}
```

UdaPostEditCallout.java interface

```
package com.lucent.qip.utils;
```

```
public interface UdaPostEditCallout {
   public Object postEditCallout(String orgName, String
    adminUserName, Object value, Class type);
}
```

UdaValidationCallout.java interface

```
package com.lucent.qip.utils;

public interface UdaValidationCallout {
   public Boolean validationCallout(String orgName, String
    adminUserName, Object value, Class type);
}
```

User-defined object naming policy

The VitalQIP web client allows customers to define their own object naming policy. If the VitalQIP client has its user-defined object naming policy set up, the web client does not use it and vice versa. The VitalQIP web client requires its own user-defined object naming policy to be set up in order to take advantage of the user-defined object naming policy.

If both the VitalQIP web client and VitalQIP are to use the same user-defined object naming policy, the user-defined object naming must be configured the same in both clients. For more information on the VitalQIP client user-defined object naming policy, refer to the *VitalQIP User's Guide*.

To configure a user-defined object naming policy for the web client

Purpose

Use this procedure to set up user-defined object naming policy for the VitalQIP web client.

Procedure

To set up user-defined object name policy for the VitalQIP web client, follow these steps:

1. Create your user-defined object naming policy Java bean. You can use the *getNewObjectNameService* as a basis for your own bean. Make sure you do not save over the original file.

2. Open the *ApplicationContext.xml* file located in the *QIPHOME/web/conf* directory.

3. Find the following lines:

```
</bean>
<bean id="getNewObjectNameService"
   class="com.lucent.qip.service.impl.GetNewObjectNameService”>
<property name="v4ObjectDao">
<ref bean="v4ObjectDao"/>
</property>
</bean>
```

4 Replace **`GetNewObjectNameService`** with the name of your custom bean and set **`property name`** and **`ref bean`** to the appropriate values as needed.

5 Save the file.

END OF STEPS

To create a UDA callout

Purpose

Use this procedure to create a UDA callout.

Before you begin

The callout **.java* files should be put in your Java development environment. You can create directory structures based on the package information in the callout **.java* files.

Procedure

To create a new UDA callout, follow these steps:

1 Create a Java class that implements one or more of the UDA callout interfaces.

The Java class **`SampleCallouts`** in the following example implements **`UdaPreEditCallout`** and **`UdaPostEditCallout`** to return the organization. It also implements **`UdaValidationCallout`** to check that there are no spaces in the name.

```
package com.lucent.qip.utils.uda.callout;

import java.util.regex.Pattern;
import java.util.regex.Matcher;
import com.lucent.qip.utils.log.LocalizedLog;
import com.lucent.qip.utils.log.LogFactory;

/**
 * preEditCallout and postEditCallout return the organization
  name.
  */


public class SampleCallouts implements
   UdaPreEditCallout,UdaPostEditCallout,UdaValidationCallout{


  private Pattern whitespace = Pattern.compile("\\s+");

  public Object preEditCallout(String orgName, String
   adminUserName,Class type){
    return orgName;
  }
```

```
            public String postEditCallout(String orgName, String
            adminUserName, String value, Class type){
              return orgName;
            }

            public Boolean validationCallout(String orgName, String
            adminUserName, String value, Class type){


              if (type == null) return false;
              if (type.getSimpleName().equals("String") == false){
                return false;
              }
              if (value == null) {
                return true;
              }

                Matcher matcher = whitespace.matcher(value);
                return (!matcher.find());

              }

          }
```

2 After a UDA callout class has been created, it must be tested.

3 To enable VitalQIP to use the callout class, place the tested callout class in *QIPHOME/web/WEB-INF/classes/com/lucent/qip/utils/uda/callout* on the VitalQIP enterprise server.

4 Open *QIPHOME/web/conf/udaApplicationContext.xml* and add the callout to the `UDA Callouts` section. For example:

```
<!-- ====================== UDA Callouts ====================== -->

 <bean id="udaCalloutUtil" class="com.lucent.qip.utils.uda.UdaCalloutUtil">
   <property name="preEditCallouts">
     <map>
      <entry key="setToUserId">
         <ref local="preDefinedCallouts"/>
      </entry>
      <entry key="setToOrg">
```

```
          <ref local="sampleCallouts"/>
        </entry>
      </map>
    </property>
    <property name="postEditCallouts">
      <map>
       <entry key="setToOrg">
          <ref local="sampleCallouts"/>
        </entry>
      </map>
    </property>
    <property name="validationCallouts">
      <map>
       <entry key="noWhitespace">
          <ref local="sampleCallouts"/>
        </entry>
        <entry key="800PhoneNumber">
          <ref local="checkPhoneNumber"/>
        </entry>
      </map>
    </property>

  </bean>

  <bean id="preDefinedCallouts"
    class="com.lucent.qip.utils.uda.callout.PreDefinedCallouts">
  </bean>

  <bean id="sampleCallouts"
    class="com.lucent.qip.utils.uda.callout.SampleCallouts">
  </bean>
<bean id="checkPhoneNumber"
    class="com.lucent.qip.utils.uda.callout.CheckPhoneNumber">
  </bean>
```

5 Stop and start the Tomcat server to implement the changes.

Result: After the system is restarted, the key of the new callout is shown in the callout field to which it was assigned in the web client Attribute Information screen, and an administrator can assign it to a user-defined attribute definition.

In the above example, `SetToOrg` is added to both the **Pre-Edit Callout** and **Post-Edit Callout** fields, and `noWhitespace` is added to the **Validation Callout** field.

END OF STEPS

Access external applications from the web client toolbar

The VitalQIP web client allows you to access external applications through the VitalQIP web client. This feature can be used to add a external web-based applications that are accessed frequently by an administrator to the web client.

How it works

Pre-defined Add-On applications

The VitalQIP web client comes with set of pre-defined VitalQIP Add-On applications that can be accessed through the web client. The Add-On applications can only be accessed when a license key is enabled for the Add-On application. If the Add-On application does not have a license key enabled, the Add-On application does not appear in the **Links** menu. The following illustration is an example of a pre-defined Add-On application.

Figure 18-1 Example of a pre-defined Add-On application

External applications

External applications can be accessed from the **Links|Customized Links** menu through a JavaScript locale-specific data file. The **Links|Customized Links** menu displays a list of external applications that can be accessed via the VitalQIP web client.

Figure 18-2 Example of the Links|Customized Links application

Second level sub-menus can be added if more than one area of the external application needs to be accessed. When no external applications are added to the **Links|Customized Links** menu, the **Links|Customized Links** menu is hidden in the VitalQIP web client. The following illustration shows an external application menu with a second level sub-menu.

Figure 18-3 Example of sub-menus

Application toolbars and the VitalQIP web client

A pre-defined and external application automatically has its toolbar loaded in the VitalQIP web client when accessed via the web client. The VitalQIP web client toolbar can be accessed by clicking on the **VitalQIP** menu.

To add an external application to the web client

Purpose

Use this procedure to add a web-based application to the VitalQIP web client **Links|Customized Links** menu.

Before you begin

- The web client can only access an *HTML* file. The *HTML* file must be located in *QIPHOME/web/html_<language_catalog>* in order for the web client to call the application.
- If you want a graphic to be displayed in the **Links|Customized Links** menu, add the graphic into the *QIPHOME/web/image* directory. The graphic must be a *gif* file.

Procedure

To add an application to the **Links|Customized Links** menu, follow these steps:

1 Go to the *QIPHOME/web/html_<language catalog>*, for example *QIPHOME/web/html_en*. Copy the *Menu_Customized.html.js* to a different directory. This copy is your backup in case you need to roll back to the file.

2 Open the *Menu_Customized.html.js* file located in the *QIPHOME/web/html_<language catalog>*.

Note: Do not make any changes to the file other than those described in this procedure.

3 Search for the following line:

```
["VitalQIP","javascript:displayVitalQIPMenu()", , ,"Return to VitalQIP", , ,
   , , ], // IMPORTANT NOTE: Do not comment out, remove or modify this item
```

4 There are two options when adding an application menu. A menu can be added with or without sub-menus. Sub-menus can be advantageous if you need to access more than one

area of an application. The following table describes how to add an applications to the **Links|Customized Links** menu.

If you want to...	Then...

Add a main menu only	1. After the line shown in Step 3, insert the following code:
	`["<name of application>", "<application's HTML page>","../images/<name of gif>","../images/<name of gif>", ,"bottomFrame", , , , ],`
	2. Replace ***<name of application>*** with the name of the application to appear in **Add-Ons | Customized Add-Ons** menu.
	3. Replace ***<application's HTML page>*** with the application's HTML file.
	4. If you are using a *gif* file in the **Links|Customized Links** menu, replace ***<name of gif>*** with the name of the *gif* file to be used on the menu. If no *gif* is being used, delete ***../images/<name of gif>***. No image will appear on the **Links|Customized Links** menu.
	The following is an example of how the code looks like if:
	• The application is called My First Add-On
	• The HTML file to be called is *mfao.html*
	• The *gif* file to be used is *mfao_lgn.gif*
	`["My First Add-On", "..mfao.html","../images/mfao_lgn.gif","../images/mfad_lgn.gif", ,"bottomFrame", , , , ],`

If you want to...	Then...

Add a main menu and a sub-menu	1. After the line shown in Step 3, insert the following code:
	`["<name of application>","", , , , , , , , ],`
	`["|<name of sub-menu>", "<application's HTML page>","../images/<name of gif>","../images/<name of gif>", ,"bottomFrame", , , , ],`
	2. Replace ***<name of application>*** with the name of the application to appear in **Add-Ons | Customized Add-Ons** menu.
	3. Replace ***<name of sub-menu>*** with the name to appear in the sub-menu.
	4. Replace ***<application's HTML page>*** with the application's HTML file.
	5. If you are using a gif file in the **Links|Customized Links** menu, replace ***<name of gif>*** with the name of the *gif* file to be used on the menu. If no *gif* is being used, delete ***../images/<name of gif>***. No image will appear on the **Links|Customized Links** menu.
	6. If you want more than one sub-menu, copy all code after the first line and paste it below the last line of code. Repeat Step 2 - Step 5. The code should look like:
	`["<name of application>","", , , , , , , , ],`
	`["|<name of sub-menu 1>", "<application's HTML page>","../images/<name of gif>","../images/<name of gif>", ,"bottomFrame", , , , ],`
	`["|<name of sub-menu 2>", "<application's HTML page>","../images/<name of gif>","../images/<name of gif>", ,"bottomFrame", , , , ],`
	The following is an example of how the code looks like if:
	• The application is called My First Add-On
	• The HTML file to be called is *mfao.html* and *mfao_rpt.html*
	• The *gif* file to be used is *mfao_lgn.gif* and *mfao_rpt.html*
	• Two sub-menus are being called **Login** and **Reports**
	`["My First Add-On"],`
	`["|Login", "..mfao.html","../images/mfao_lgn.gif","../images/mfad_lgn.gif", ,"bottomFrame", , , , ],`
	`["|Reports", "..mfao_rpt.html","../images/mfao_rpt.gif","../images/mfao_rpt.gif", ,"bottomFrame"", , , , ],`

5 If there are more applications to add, repeat Step 4.

6 Save the file. When you log into the VitalQIP web client, the application menu will appear on the **Links|Customized Links** menu.

7 ***If you are using multiple character sets and languages in VitalQIP,*** you need to repeat this procedure for each *QIPHOME/web/html_*<*language catalog*> directory with the language specific *HTML* file. You do not need to rebuild the language catalog if multiple character sets and languages are not enabled.

END OF STEPS

19 Database and server administration

Overview

Purpose

This chapter discusses administration of the VitalQIP database. Most sections in this chapter are specifically for Sybase 15.0 and higher database users. However, some sections can be used by Oracle d atabase users and are indicated accordingly. For Oracle database administration beyond this chapter, see your Oracle database administrator.

> **Note:** ***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.

Database administration	19-3
Run Sybase	19-3
Find VitalQIP version numbers with vercheck	19-3
VitalQIP Sybase database backup	19-4
Transaction log backup	19-6
Server backup	19-6
Manual backup	19-6
Change the backup medium	19-8
Change the procedure cache size and total memory size	19-8
Password encryption	19-9

Database re-initialization	19-9
qip-dbinit	19-9
Database administrative tasks using qip-util	19-11
Track stored procedures and triggers	19-15
VitalQIP data space management	19-16
VitalQIP transaction log space management	19-18
Sybase database recovery	19-20
VitalQIP database recovery	19-21
Truncate audit data	19-24
Truncate audit data	19-24
Index statistics maintenance (Sybase only)	19-25
Server maintenance	19-26
Stop and start servers	19-26
To stop the VitalQIP enterprise server for maintenance	19-27
To start the VitalQIP enterprise server	19-29
To stop the VitalQIP remote server for maintenance	19-30
To start the VitalQIP remote server	19-31

Database administration

Run 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.

Start Sybase on Windows

Sybase users can start Sybase using the Microsoft Windows Services Controller.

Start Sybase on UNIX

Sybase users can start Sybase by typing the following at a command line:

```
# cd  $SYBASE/ASE-15_0/install
# RUN_<SQL_Server_Name> <DSQUERY_value>
```

Find 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.

VitalQIP Sybase database backup

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. All backups can be done via tape or hard drive (disk file).

When to back up the VitalQIP 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.

Note: It is strongly suggested that you back up the entire VitalQIP database ***daily***. If near-point-in-time recovery is required, backup the VitalQIP transaction log ***several times a day***. 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.
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.

When to back up the transaction log

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.

When to back up the Master database

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).

It is recommended that you periodically copy the <*name of Sybase server*>*.cfg*, located in the *$SYBASE/ASE-15_0*, located in the SYBASE directory, to another directory. The Master database stores Sybase configuration information, and it can make your recovery of the Master database easier.

In the event your master database goes down, call Technical Support to help reinstall the Master database. The <*name of Sybase server*>*.cfg* file can be used to restore your Sybase configuration information

Transaction log backup

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> exit
```

Server backup

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-15_0/install/RUN_QIPSYBASE_BACKUP  //assuming the
   name of the backup server is QIPSYBASE_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, refer to "Change the backup medium" (p. 19-8).

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 "VitalQIP Sybase database backup" (p. 19-4).

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 "VitalQIP Sybase database backup" (p. 19-4).

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
```

Backing up the master database

The following commands are run on a regular basis:

```
1> dump database master to master_dump
2> go
```

Change 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, "Manual backup" (p. 19-6).

Change 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
```

Password encryption

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" (p. 3-2) for more information on the *qip.pcy* file.

For additional information about **qip-crypt**, refer to the *VitalQIP Command Line Interface User's Guide.*

Database re-initialization

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.

Note: 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.

Database administrative tasks using 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**. Table 19-1, "Function values" (p. 19-12) 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.

The following table shows the function values that can be used.

Table 19-1 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

Function values	Description	Database	Product
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
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

Function values	Description	Database	Product
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
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.

Track 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 qipadman -P <qipadman_password>
1> select * from qip_version
2> go
```

The returned information provides version information for all stored procedures and triggers in the system.

VitalQIP data space management

Purpose

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 similar to the following:

```
namedb_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
```

Note: If **qip_dat** is full, you may use **qip-clear** to remove some or all audit data. For information about **qip-clear**, refer to "Truncate audit data" (p. 19-24).

END OF STEPS

VitalQIP transaction log space management

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.

Monitor 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.

Dump 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

Enlarge 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, refer to the next section, "VitalQIP database recovery" (p. 19-21) to recover the VitalQIP database. If the Sybase master device is lost, call techinical support for assistance. For Oracle users, see your Oracle database administrator for assistance.

Note: To provide up-to-the-minute recovery, the database option "trunc" must be off.

VitalQIP database recovery

Purpose

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.

Procedure

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 Step 3 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
```

Note: **vdevno**, location, size were determined in Step 3 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
```

Note: **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 Step 1, 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:

```
>isql -U sa -P <sa_password>
> sp_dboption QIP, 'single user',true
> go
> use QIP
> go
> checkpoint
> go
```

b. Using a text editor, type the following in the lines in the *dbcc* file and save the file when you are done:

```
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
```

END OF STEPS

Truncate 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**, refer to 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 “Database administrative tasks using qip-util” (p. 19-11).

Note: **`qip-import`** runs this script automatically.

Server maintenance

Stop and start servers

This section provides information on the recommended steps to stop and start VitalQIP enterprise and remote servers for maintenance.

To stop the VitalQIP enterprise server for maintenance

Purpose

This section provides information on the recommended order for stopping the VitalQIP enterprise server for maintenance. If you are doing maintenance on the VitalQIP remote server, please follow the instructions in "To stop the VitalQIP remote server for maintenance" (p. 19-30) for stopping the remote server.

Procedure

To stop the VitalQIP enterprise server, follow these steps:

1. Have all users log out of the VitalQIP clients and web interfaces.

2. Stop the web server.

3. Stop services on the VitalQIP enterprise server in this order:
   - **`qip-logind`** - VitalQIP Login Service
   - **`qipd`** - VitalQIP Schedule Service
   - **`qip-qipupdated`** - VitalQIP QIP Update Service
   - **`qip-dnsupdated`** - VitalQIP DNS Update Service
   - **`qip-rmisched`** -RMI Scheduler Service
   - **`qapi`** - VitalQIP RMI QAPI Service
   - **`qip-msgd`** - VitalQIP Message Service

   On Windows, you can use the VitalQIP Service Controller or Windows Service Controller to shutdown all services.

   On UNIX, you can kill the PIDs.

4. Shut down the database. You can shut down the database via:
   - For Oracle, see your Oracle database administrator.
   - For Sybase on Windows, you can use the VitalQIP Service Controller or the Windows Service Controller .
   - For Sybase on UNIX, it is recommended you shut down the database using **`isql/sqlplus`**.

For example:

```
 isql –U sa –P password_for_sa
>shutdown
>go
```

END OF STEPS

To start the VitalQIP enterprise server

Purpose

This section provides information on the recommended order for starting the VitalQIP enterprise server. If you are starting the VitalQIP remote server, please follow the instructions in "To start the VitalQIP remote server" (p. 19-31) for starting the remote server.

Procedure

To start the VitalQIP enterprise server, follow these steps:

1. Start the database. You can start the database via;
   - For Sybase on Windows, the VitalQIP Windows Controller or Windows Service Controller.
   - For Sybase on UNIX, you can use Sybase startup script. For example:

```
# cd $SYBASE/ASE-15_0/install
# RUN_<SQL_Server_Name> <value of the DSQUERY_environment variable>
```

   - For Oracle on Windows and UNIX, see your Oracle database administrator.

2. Start the services on the VitalQIP enterprise server in this order:
   - **qip-msgd** - VitalQIP Message Service
   - **qip-logind** - VitalQIP Login Service
   - **qipd** - VitalQIP Schedule Service
   - **qip-qipupdated** - VitalQIP QIP Update Service
   - **qip-dnsupdated** - VitalQIP DNS Update Service
   - **qip-rmisched** - MI Scheduler Service
   - **qapi** - VitalQIP RMI QAPI Service

3. Start the web server.

4. Have users log into VitalQIP via the VitalQIP clients and web interfaces.

END OF STEPS

To stop the VitalQIP remote server for maintenance

Purpose

This section provides information on the recommended order for stopping the VitalQIP remote server for maintenance.

Procedure

To stop the VitalQIP remote server, follow these steps:

1. Sever the connection to the QIP Update Service by stopping the **`qip-msgd`** - VitalQIP Message Service.

2. Stop the **`qip-rmtd`** - VitalQIP Remote Service and **`qip-netd`** - VitalQIP Active Lease Service.

3. If you are running distributed services, use the order specified in Step 3 on page 27 to stop the services.

4. Stop your DHCP and DNS Services.

END OF STEPS

To start the VitalQIP remote server

Purpose

This section provides information on the recommended order for starting the VitalQIP remote server.

Procedure

To start the VitalQIP remote server, follow these steps:

1. Establish a connection to the VitalQIP server by starting the **`qip-msgd`** - VitalQIP Message Service.

2. Start the **`qip-rmtd`** - VitalQIP Remote Service and **`qip-netd`** - VitalQIP Active Lease Service.

3. If you are running distributed services, use the order specified in Step 3 on page 27 to start the services.

4. Start your DHCP and DNS Services.

END OF STEPS

20 Troubleshoot 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.

General VitalQIP troubleshooting	20-2
General VitalQIP troubleshooting	20-2
Environment verification	20-4
System logs	20-5
Enable debugging in VitalQIP	20-8
Problem with logging into VitalQIP interface	20-10
Login error message	20-10
VitalQIP server failures	20-11
Automatic synchronization	20-12
Importing and exporting data	20-12
VitalQIP services/daemons	20-13
Troubleshooting services/daemons on UNIX	20-13
Troubleshooting starting and stopping services/daemons	20-30
Communication problems for VitalQIP service	20-33

General VitalQIP troubleshooting

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 the following table before calling technical support.

Table 20-1 Required troubleshooting information

Problem type	Required files or information
Installation or upgrade from previous VitalQIP version	Include the following items: • The log file from *$QIPHOME* file and any template files used • 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
VitalQIP web client	Include the following items: • VitalQIP version and build number • Browser and version number • Error message displayed • Error message details (if available) • Exact steps to reproduce the problem • If possible, the *qip7.log* file from the web server

Environment verification

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 2003, 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
```

Note: If any values need to be changed, update **shrc** before executing it.

For Windows 2003, 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 (**ip-manage**), select **About VitalQIP** from the **Help** menu 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 2003:

```
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 2003:

```
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`**). Refer to "Find VitalQIP version numbers with vercheck" (p. 19-3) for more information about this CLI command.

Where is my license key file located?

The license key is located in *%QIPHOME%\Lic* (Windows 2003) or *$QIPHOME/.Lic* (UNIX) on the enterprise server only.

System logs

Error conditions within VitalQIP (and other running applications) are recorded in the *syslog* file (UNIX) or the Event Viewer (Windows 2003) 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. Refer to 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.

The following table describes common messages in the system log or <*server*>.*log* file.

Table 20-2 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.

Error message	Message source	Probable cause of problem
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.
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.

Error message	Message source	Probable cause of problem
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.

Enable debugging in VitalQIP

Purpose

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 2003, run the following script:

```
ipmanage.exe -d
```

Note: The log file is located in *$QIPHOME/log/ipmanage.log*.

To turn on debugging for UNIX, run the following script:

```
ip-manage -d &
```

Note: 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, refer to “The debug policy” (p. 3-44).

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`**.

Note: 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

Problem with logging into VitalQIP interface

Purpose

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

The following table describes error messages you may encounter while logging into VitalQIP.

Table 20-3 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, */opt/app/oracle/product/10.2.0/client*. 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 Dynamic DNS 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 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 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 "Environment verification" on page 4 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.

Startup sequence

If you start Schedule Service (**`qipd`**) before starting the Sybase or Oracle database, **`qipd`** starts properly, and it will keep retrying the database connection until the database is back up. You can start **`qipd`** first, and then start the database. However, **`qipd`** cannot update the license key or do the scheduled activities while the database is down, even though it is running.

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

The following table describes signal types for the VitalQIP Schedule Service.

Table 20-4 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

The following table describes signal types for the Login Service.

Table 20-5 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

The following table describes the signal types for the Message Service.

Table 20-6 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

The following table describes the signal types for the QIP Update Service.

Table 20-7 QIP 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. Causes it to re-read its config files (not *qip.pcy*, just the files that specify what DNS servers to update).
SIGPIPE	Handled.

DNS Update Service signal types

The following table describes the signal types for the DNS Update Service.

Table 20-8 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. Causes it to re-read its config files (not *qip.pcy*, just the files that specify what DNS servers to update).
SIGPIPE	Handled.

Active Lease Service signal types

The following table describes the signal types for the Active Lease Service.

Table 20-9 Active Lease 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.

Remote Service signal types

The following table describes the signal types for the Remote Service.

Table 20-10 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

The following table describes the signal types for the RMI Scheduler Service.

Table 20-11 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 types for the MS DNS Update Service.

Syslog file

Informational, warning, and error messages are logged by all services to the*syslog* file. To review messages, the *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* file. Other messages may be logged.

Schedule Service message types

The following table shows a subset of messages generated by the Schedule Service.

Table 20-12 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.
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. Refer to "Schedule Service statistics" on page page 28 for a list of Statistics.

Login Service message types

The following table shows a subset of messages generated by the Login Service message types.

Table 20-13 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. Refer to "Login Service statistics" on page 28 for a list of Statistics.

Message Service message types

The following table describes the messages types for the Message Service.

Table 20-14 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.
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.

Message	Severity	Description
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. Refer to "Message Service statistics" on page 28 for a list of Statistics.

MS DNS Update Service message types

The following table describes the messages for the MS DNS Update Service.

Table 20-15 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.

Message	Severity	Description
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.
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

The following table describes the messages for the QIP Update Service.

Table 20-16 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.

Message	Severity	Description
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.
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.

Message	Severity	Description
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. Refer to "QIP Update Service statistics" on page 28 for a list of Statistics .

DNS Update Service message types

The following table describes the messages for the DNS Update Service.

Table 20-17 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.

Message	Severity	Description
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.
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. Refer to "DNS Update Service statistics" on page 29 for a list of Statistics.

Active Lease Service message types

The following table describes the messages for the Active Lease Service.

Table 20-18 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.

Message	Severity	Description
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.
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.

Message	Severity	Description
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.
Exiting service	INFOR	The service is exiting.

Remote Service message types

The following table describes the message types for the Remote Service.

Table 20-19 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.

Message	Severity	Description
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

The following table describes the message types for the RMI Scheduler Service.

Table 20-20 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*.
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.

Note: 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:

- Connected Message Services
- Total Messages Received

- Duplicate Messages Received
- Database Connection Failures
- Database Update Failures
- Database Authorization Failures
- Resource Record Update Failures
- QIP Updates/s over last 1/5/15 minutes
- QIP Updates overall
- Policy File
- Current Process Status
- Running in <mode>
- Hostname Spaces Converted
- Audit DHCP Updates
- Dump Stats on exit
- Connected to Database
- Update DNS
- Send Renews to DNS
- Allowed IP Addresses
- Denied IP Addresses

DNS Update Service statistics

The following statistics are generated by the DNS Update Service:

- 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

Purpose

If you encounter problems starting and stopping services/daemons, review this section.

Before you begin

For information on starting and stopping the Tomcat server, refer to "To start or stop the Tomcat server on UNIX" (p. 2-22).

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 2003, open the VitalQIP Service Controller via **Start | Programs | 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>
```

Note: 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" (p. 27-6).

Why can't I start the DNS Service (named)?

Refer to "DNS error messages" (p. 25-10).

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:

1 Update *qip.pcy* file by setting **Debug=AllFeatureBackup** in the **`[VitalQIP Message Service]`** section.

2 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

Communication problems for VitalQIP service

VitalQIP services are not able to receive or send messages

VitalQIP service communication issues can include:

- Pushes to DNS and DHCP servers fail
- DNS server are not able to send or receive messages
- Error messages being returned with regards about to not being able to reach their intended targets

A firewall may be blocking messages to or from the service. Check to make sure that the fire includes the port used for the service to listen or receive messages in the firewall's exception list.

A firewall may be blocking messages to or from the service. Check to make sure that the firewall allows the port used for the service to listen or receive messages. Refer to your firewall's documentation for more information.

21 General informational and error 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 “General VitalQIP troubleshooting” (p. 20-2) and “Environment verification” (p. 20-4) for more information.

Contents

This information presents the following topics.

Database error messages	21-1
Service errors and informational messages	21-15
Error and information messages generated by VitalQIP services	21-18

Database error messages

The following table describes database errors and their possible resolutions.

Note: The gap in sequence numbers is due to error messages that are no longer valid.

Table 21-1 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.
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.
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.

Database error	Description	Resolution
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.
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.

Database error	Description	Resolution
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.
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.

Database error	Description	Resolution
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.
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.

Database error	Description	Resolution
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.
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.

Database error	Description	Resolution
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).
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.

Database error	Description	Resolution
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.
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.

Database error	Description	Resolution
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.
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.

Database error	Description	Resolution
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.
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.

Database error	Description	Resolution
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.
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.

Database error	Description	Resolution
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.
262 Folder must be empty	Cannot delete a folder that contains domains.	Remove the domain(s) from the folder before deleting the folder.

Database error	Description	Resolution
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.

Database error	Description	Resolution
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.
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. The following table describes non-database error codes, the source of the problem, and the action to resolve the problem.

Table 21-2 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.
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.

Error code	Description	Resolution
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.
2604	License manager unable to retrieve the server host ID.	In Windows 2003, 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 2003. The following tabel describes informational and error messages logged by the above services.

Table 21-3 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.

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.

Part IV: Security administration

Overview

Purpose

In Part IV, you will find information on administration authentication and secure message routes.

Contents

Part IV contains the following chapters:

Chapter 22, “Authenticate administrators”	22-1
Chapter 23, “Secure message routes”	23-1

22 Authenticate administrators

Overview

Purpose

This chapter describes how to authenticate administrators and protect administrator passwords.

Contents

This chapter contains the following topics.

Administrator authentication tool	22-2
VitalQIP Authentication Callout overview	22-2
Invoke the Authentication Callout function	22-4
Sample administrator authentication callout	22-6
Authentication callout policies	22-7
Use the authentication callout as a logging tool	22-10
Use the authentication callout as authentication tool	22-11
Authentication CLI command	22-12
Custom administrator authentication callout	22-14
Create a custom authentication callout	22-14
LDAP authentication tool	22-20
LDAP authentication tool overview	22-20
Invoke the LDAP command as an authentication tool	22-22
Enable the qip-ldapauth authentication callout for Tomcat	22-28
The qip-ldapauth command	22-29
The qip-ldapauth.pcy file	22-31

Administrator authentication tool

Using an authentication callout or CLI command, the VitalQIP Login Service supports a callout that can authenticate administrators against a third party authentication service, such as Radius or the Windows 2003 user database, when they log into VitalQIP or Audit Manager.

Note: Although custom authentication routines are invoked by the Login Service to confirm an administrator's identity, VitalQIP clients do not maintain a persistent connection with the Login Service or the Authentication Callout. You cannot therefore use the authentication callout to expire active user sessions.

Load balancing

If a custom authentication routine takes a long time to execute, administrators can experience delays when they log into VitalQIP. To remedy this problem, 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.

VitalQIP Authentication Callout overview

The Login Service (**`qip-logind`**) has a configurable built-in mechanism that can call any user-defined Command Line Interface (CLI), known as the Authentication Callout or **`AuthenticationCLI`** throughout this chapter. The **`AuthenticationCLI`** can be a shell script, Perl script, or any program that can be executed as a CLI in the customer environment.

How passwords are authenticated

Passwords can be validated by VitalQIP or with the Authentication Callout tool. This section shows how administrators are validated in VitalQIP and with the Authentication Callout.

VitalQIP password validation

The following illustration shows how VitalQIP validates passwords without authentication callouts.

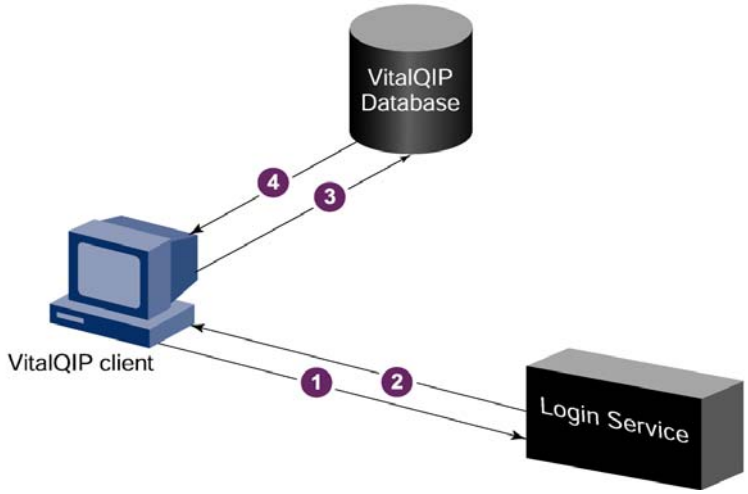


Figure 22-1 Validation of passwords in VitalQIP

Without the Authentication Callout, a password is validated as follows:

1. A VitalQIP administrator types the user name and password in the user interface. The information is sent to the Login Service.
2. The VitalQIP Login Service processes the information. Information is provided to the user interface so that the user interface can access the VitalQIP database.
3. The user interface accesses the VitalQIP database and retrieves the database record for the administrator that corresponds to the user name.
4. The user interface verifies that the typed password matches the encrypted password stored in the database. The administrator is then allowed or denied access.

Authentication Callout password validation

The following illustration shows how the default authentication process described above is modified when an authentication callout is used.

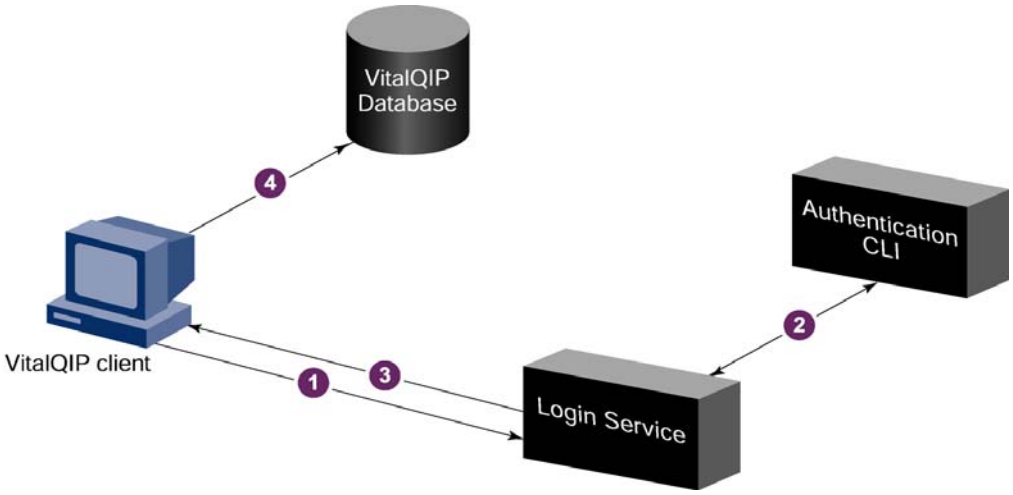


Figure 22-2 Validation of passwords with the Authentication Callout

With the Authentication Callout and **`AuthenticationCLI`** configured, a password and administrator are validated as follows:

1. A VitalQIP administrator types the user name and password in the user interface. The information is sent to the Login Service.
2. The Login Service processes the user name and password. The user name, password, and other information are sent to the user-defined callout function. The user-defined callout returns a value to the Login Service.
3. Based on the values returned by the user-defined callout function, the Login Service returns the information to the user interface. The administrator is allowed or denied access to the VitalQIP database.
4. An administrator can access the VitalQIP database only when the return code from the callout allows access.

Invoke the Authentication Callout function

A library called **`libqipauthcallout.so`** or **`qipauthcallout.dll`** is provided that enables basic authentication callout functionality. While it is possible to create your own library, it is usually not needed.

The mechanism of the callout procedure is straightforward. The Login Service calls the **AuthenticationCLI** and passes a defined set of parameters to this function. The parameters include the user name, VitalQIP product, VitalQIP database name, IP address of user interface, and password. The **AuthenticationCLI** processes these parameters and returns one of three acceptable values (return codes) to the Login Service. The returned value determines how the VitalQIP Client proceeds during the login process.

Authentication Process logic

The Authentication Process uses this logic:

1. The VitalQIP administrator enters a username and password in the user interface.
2. The user interface sends the ***<User name> <QIP|Audit> <Database> <IP Address> <Encrypted Password>*** to the VitalQIP Login Service.
3. VitalQIP Login Service calls the customer-supplied **AuthenticationCLI** specified in the *qip.pcy* file with arguments passed from the user interface.
4. The **AuthenticationCLI** then executes the instructions written by the customer. Examples include contacting an authentication server, accessing a flat file, or calling another program.
5. The **AuthenticationCLI** finishes its processing and returns a `0`, `1`, or `2` return code to the VitalQIP Login Service.
6. Based on the return code, the VitalQIP Login Service sends an appropriate response to the user interface.

Prerequisites

The following are the prerequisites for the Authentication Callout:

- Make sure the *libqipauthcallout.so* or *qipauthcallout.dll* library exists in the *$QIPHOME/usr/lib* or *%QIPHOME%\lib* directory.
- All users to be granted access must be defined in VitalQIP as administrators configured with desired privileges.
- If **AuthenticationCLI** returns a `1`, the password in VitalQIP must be known to the administrator attempting to login. If `0` or `2` are returned, the password stored in VitalQIP is not relevant, as the authentication routines override it.
- An **AuthenticationCLI** must be created and placed in the *QIPHOME/userexits* directory on the server running the Login Service.
- The *qip.pcy* file must be modified to invoke the authentication routines. The *qip.pcy* file must have the following two lines uncommented/added:
  - In the VitalQIP Login Service section:

AuthLibrary=libqipauthcallout.so (UNIX)
AuthLibrary=qipauthcallout.dll (Windows)

  - In the QIPAuthCallout section:

AuthenticationCli=*<The name of the AuthenticationCli>*

Special considerations

Provisions must be made to allow the QIP Update Service, the QIP Schedule Service, and the **qipman** administrator accounts to be able to log into the database. Both services log into VitalQIP with a default non-published password. These encrypted passwords can be changed in the *qip.pcy* file and the database if desired. (Please contact Technical Support for details.)

The **qipman** user and password is used by the VitalQIP DNS Update Service (**qip-dnsupdate**) and must be in the global section of the *qip.pcy* file on the server running DNS Update Service.

To make provisions for the QIP Update Service, the QIP Schedule Service, and the **qipman** administrator accounts to be able to log into the database:

1. The **AuthenticationCLI** can be set up to always allow (not recommended) or defer to VitalQIP. It can be further restricted by IP address or some other method for additional security.
2. The **updateservice**, **scheduleservice**, and **qipman** users can be added to the authentication server. These users can then be required to follow normal company password security policies, such as password aging.

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.

> **Note:** For the remainder of this section, *QipAuthCallout.dll* is used. You should assume *QipAuthCallout.dll* also includes the UNIX version *libqipauthcallout.so*.

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 administrator's 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:

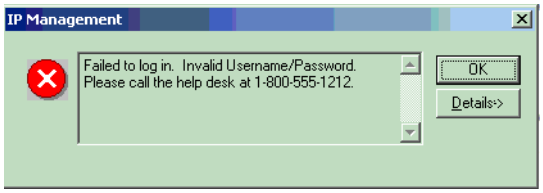


Figure 22-3 Example of custom message

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 in the **`[QIPAuthCallout]`** section of the *qip.pcy* file. This section describes the authentication callout policies.

SuccessLogLevel

Value	Default: Information Allowed: None | Information | Warning | Error
Descriptioon	The severity level at which successful authentications are written to the system log. The following values can be specified: • **None** • **Information** • **Warning** • **Error**

AttemptLogLevel

Value	Default: Information Allowed: None | Information | Warning | Error
Description	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

Value	Default: Error Allowed: None | Information | Warning | Error
Description	The severity level at which unsuccessful authentications are written to the system log. The valid values are **None, Information, Warning, and Error.**

AuthenticationCli

Value	Default: None Allowed: Any valid path name relative to *QIPHOME/userexits*

Description	The path name of a command to run to authenticate the administrator. Path names are relative to *QIPHOME/userexits*. **Note:** The path or *.exe* extension must not be used if this policy is being set up for use with the **`qip-ldapauth`** feature. For example,: `AuthenticationCli=qip-ldapauth` The CLI command should return one of the following: • **0** - allow the user to log into VitalQIP • **1**- defer authentication to VitalQIP • **2** - deny authentication When this policy is not set, the administrator authenticates against the password stored in the VitalQIP database (depending on the value of the **`AuthLibrary`** setting in the VitalQIP Login Service section of *qip.pcy*). Refer to "Authentication CLI command" (p. 22-12) for a description of input arguments and expected return values.

FailureMessage

Value	Default: None Allowed: Text up to 253 characters
Description	The text of a message to be displayed by the user interface when administrator authentication fails.

DelayOnFailure

Value	Default: 0 Allowed: 0 or any positive number
Description	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

Purpose

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

Procedure

To use the authentication callout as a logging tool, follow these steps:

1. In the **[VitalQIP Login Service]** section of the *qip.pcy* file, set the **AuthLibrary** policy to **qipauthcallout**. Refer to "AuthLibrary" (p. 8-6) 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

Purpose

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

Procedure

To use the authentication callout as an authentication tool, 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. Refer to "Authentication CLI command" (p. 22-12) 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" (p. 22-7). 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.

Sample log message

The following is a sample 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

The contents or logic of the **AuthenticationCLI** is dependent upon the user. 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 also 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>
```

The following table describes the authentication CLI command parameters.

Table 22-1 Authentication CLI command parameters and arguments

Value	Description
User name or **arg[0]**	The administrator's user name. This value is sent by the user interface.
QIP|Audit or **arg[1]**	The name of the product that the administrator is logging in to. This value is sent by the user interface. The following can be specified: • **QIP** - VitalQIP • **Audit** - Audit Manager
Database or **arg[3]**	The name of the database to which the administrator is trying to connect.
IP address or environment variable	The IP address of the host that created the login request presented in dotted decimal form. This is available to the **AuthenticationCLI** from the environment. This value is sent by the user interface to **qip-logind**, where it is set as **LUCENT_PASSWORD**. **Note:** 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.

Note: To ensure that the administrator's password is secure, the password is not passed on the command line. Instead, the Login Service sends the administrator's

password to the **AuthenticationCLI** command using the $LUCENT_PASSWORD environment variable.

Command line input example

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

Return values

The Authentication Callout uses these return codes (which are the only codes that can be returned by the **AuthenticationCLI**):

Return 0	Authenticated	User will be allowed access
Return 1	Indeterminate	Access will be deferred to VitalQIP and granted or denied via normal VitalQIP login procedures
Return 2	Denied	User will not be allowed access

Code example

The following is a code example.

```
……
customer_auth (arg0,arg1,arg2,arg3)
//define variables for Arguments
$pass=$ENV{"LUCENT_PASSWORD"};
$user=$ARGV[0];
$prod=$ARGV[1];
$db=$ARGV[2];
$ipaddr=$ARGV[3];
//Call some CLI that will authenticate the user.
$return =System(goCheckMyDatabaseCLI $user $pass);
//$return can be anything coming from the system call.
// exp. Invalid User/Password is returned
// You may choose to make this a 2
if ($return eq “Invalid User/Password”){
exit 2;}
if($return eq “Valid User/Password”){
exit 0;)
……
```

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 an individual domain administrator name to an individual VitalQIP administrator
- Maps multiple domain administrator names to a single VitalQIP administrator

In such cases, VitalQIP allows you to create a custom dynamic library that can implement auditing and authentication functions. A *QIPAuthCallout.h* header file and a sample 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`** as defined below. On Windows, **`QIPAuthCallout`** must be exported from the dynamic library. The authentication callout name **`QIPAuthCallout`** must conform to C language bindings.

> **Note:** The authentication callout implementation must not have connectivity to the VitalQIP 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
```

The following table describes the arguments that invoke the callout.

Table 22-2 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 product • **Audit** - Audit Manager product
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.

Argument	Type	Pointer	Value	Description
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.

Enable the authentication callout for the Web Service

Purpose

The VitalQIP Web Service authenticates against the database. It can also be configured to use the VitalQIP Login Service. Use this procedure to enable the authentication callout for the Apache Tomcat Server.

Procedure

To enable the authentication callout for the web server:

1. Stop the Apache Tomcat server.

2. In the application context file *ApplicationContext.xml* located in the *$QIPHOME/web/conf/ApplicationContext.xml* directory, uncomment the **`qipCallout`** bean:

```
<bean id="qipCallout" class="com.lucent.qip.auth.QipCallout">
<property name="qipConfig"><ref bean="qipConfig"/></property>
   </bean>
```

3. Uncomment the callout property of the **`commonService`** bean, so it can use the **`qipCallout`** bean.

```
<bean id="commonService" ...
...
<property name="callout"><ref bean="qipCallout"/></property>
...
</bean>
```

4. Save the *ApplicationContext.xml* file.

5. The **`qipConfig`** bean looks for and parses the *qip.pcy* file to retrieve the Login Service information. Ensure that the *qip.pcy* file is found and that it contains the appropriate configuration.

6 Last, start the Apache Tomcat server.

END OF STEPS

LDAP authentication tool

VitalQIP uses the **qip-ldapauth** command to authenticate a user against an LDAP server. The **qip-ldapauth** command is designed to be used as an external authentication callout to the VitalQIP Login Service. Since Microsoft Active Directory understands LDAP, this command can be used to authenticate users in a Microsoft Active Directory.

The **qip-ldapauth** command offers:

- Regular or secure LDAP connection using SSL
- Encryption negotiation over regular LDAP port using StartTLS (UNIX only)
- Anonymous or search after binding
- Reversible encrypted password for BIND password
- Authorization of users based on an attribute, an LDAP filter, or an LDAP group
- Logging information to a log file

LDAP authentication tool overview

The following illustration shows the LDAP authentication tool process.

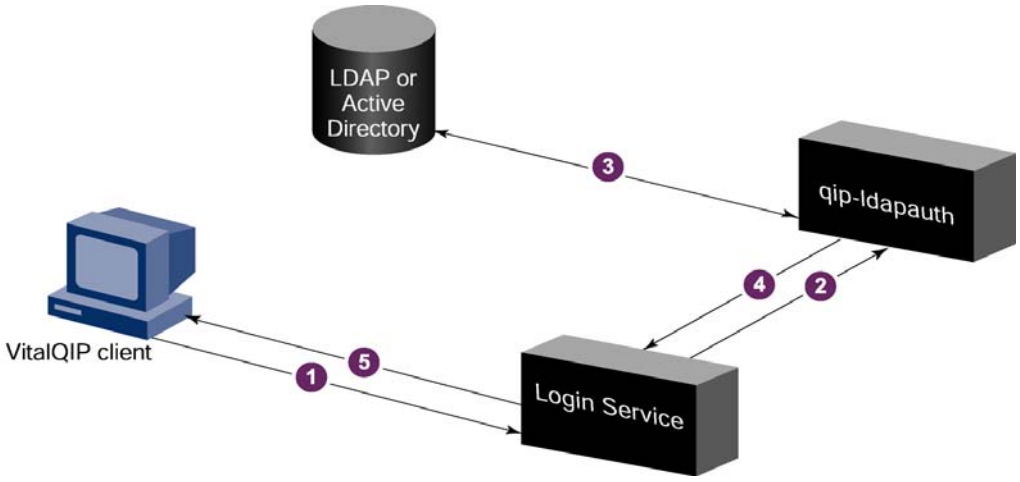


Figure 22-4 LDAP authentication tool process

In Figure 22-4, the administrator is authenticated as follows:

1. A VitalQIP administrator types the user name and password in the user interface.
2. The Login Service processes the user name and password. The user name and password are sent the **qip-ldapauth** command.
3. The **qip-ldapauth** command checks to see if the administrator is authenticated against the LDAP or Active Directory database.

4. The **`qip-ldapauth`** command generates a return value which the Login Service examines.
5. Based on that return value, the Login Service returns the information to the user interface. The administrator is either allowed or denied access to VitalQIP. The administrator can access the LDAP or Active Directory server.

Invoke the LDAP command as an authentication tool

Purpose

Use this procedure to set up the **qip-ldapauth** command as an authentication tool for an LDAP or Active Directory server.

Before you begin

- Since the **qip-ldapauth** command can be executed by external programs, the bind password and user password could be exposed if anyone lists the processes at this time. To avoid exposing these passwords, set the bind password to the **LDAP_BIND_PASS** environment variable and user password to the **LDAP_USER_PASS** environment variable. When these values have been set, do not use the **-w** parameter and do not specify the user password as the last value. The command extracts those values from the environment variables instead. For secured SSL/TLS connections, refer to "Encrypted connections with SSL/TLS" (p. 22-23).
- In Windows, **qip-ldapauth.exe** does not verify the certificate for the LDAP server when SSL is in use. In Linux/UNIX, however, **qip-ldapauth** can be configured to check if the certificate provided by the LDAP server is signed by a known Certificate Authority (CA). If the server's certificate is not signed by a known CA, it will not proceed further. Similarly, the LDAP server can be configured such that a client (**qip-ldapauth**, for example) must have a certificate signed by a known CA, or else it will not allow the client to connect. Refer to "Encrypted connections with SSL/TLS" (p. 22-23) for more information.

Set up LDAP authentication

1 Make sure that the VitalQIP Authenticaton Tool is set up. Modify the **AuthenticationCli** policy in the **QIPAuthCallout** section of *qip.pcy* to read:

AuthenticationCli=qip-ldapauth

Refer to "Administrator authentication tool" (p. 22-2) for more infomation.

2 Modify *QIPHOME/qip-ldapauth.pcy*. A sample policy file *qip-ldapauth_sample.pcy* is supplied for your reference. The following polices must be correctly specified:

LDAPServerURL
BaseDn
UidAttr

If the LDAP or Active Directory server requires binding, binding information must be specified for the following policies:

BindDn
BindPass

3 Make sure the path to the authentication callout library is specified in the **AuthLibrary** policy in *qip.pcy* file under the [VitalQIP Login Service] section. For example:

```
[VitalQIP Login Service]
AuthLibrary=/opt/qip/lib/libqipauthcallout.so
```

4 The VitalQIP Login Service sets the user password in the LUCENT_PASSWORD environment variable and runs **qip-ldapauth** as follows:

```
qip-ldapauth user_id product_name database_name client_ip
```

For example:

```
qip-ldapauth jdoe VitalQIP Sybase 10.100.30.1
```

Note: A shell script or a bat script can be specified in the **AuthenticationCli** policy if needed. Execute the **qip-ldapauth** command from the script and handle the return values as specified in "Return values" (p. 22-13).

END OF STEPS

Encrypted connections with SSL/TLS

If the **qip-ldapauth** command is running on UNIX, any clients linked with SSL/TLS enabled OpenLDAP C SDK look for an *ldaprc* file in the current working directory.

Set LDAPRC environment variable

Alcatel-Lucent recommends that the location of the *ldaprc* file be added to the **LDAPRC** environmment variable in VitalQIP's *etc/shrc* file.

The following is a sample command for setting the environment variable:

```
LDAPRC=$QIPHOME/etc/ldaprc
export LDAPRC
```

LDAP server certificate

By default, **qip-ldapauth** attempts to verify the LDAP or Active Directory server's certificate. To disable this check, set the **TLS_REQCERT** directive to "never" or "allow" in the *ldaprc* file. Otherwise, the certificate for the certificate authority needs to be specified with the appropriate directive. The certificate must be in PEM format.

An error message "`Can't contact LDAP server`" is printed if:

- A SSL/TLS connection is attempted and the *ldaprc* file cannot be found

- The **TLS_REQCERT** directive is not set appropriately
- The certificate specified in the file is not found

Sample ldaprc file

The following is a sample *ldaprc* file:

```
#################################################################
# The Unix version of qip-ldapauth is compiled with SSL enabled
# OpenLDAP SDK, hence the CLI supports ldaprc file to
# specify various directives about SSL/TLS.
#################################################################

###
# TLS_REQCERT <level>
#
# If you error message 'Can't contact LDAP server', you must specify
# an appropriate value for this directive. The default is never.
# This directive specifies what checks to perform on server certificates
# in  a TLS session, if any. The <level> can be specified as one of the
# following keywords:
#  never
#       The client will not request or  check  any  server
#       certificate. The connection will be encrypted however.
#
#  allow
#       The server certificate is requested. If no  certi-
#       ficate is provided, the session proceeds normally.
#       If a bad  certificate  is  provided,  it  will  be
#       ignored and the session proceeds normally. The connection
#       will be encrypted however.
#
#  try  The server certificate is requested. If no  certi-
#       ficate is provided, the session proceeds normally.
#       If a bad certificate is provided, the  session  is
#       immediately terminated.
#
#  demand | hard
#       These keywords are equivalent. The server certifi-
#       cate  is requested. If no certificate is provided,
#       or a bad certificate is provided, the  session  is
#       immediately  terminated.  This is the default set-
#       ting.
TLS_REQCERT never

###
```

```
#TLS_CACERT <filename>
#  Specifies the file that contains certificates  for  all
#  of  the  Certificate Authorities the client will recog-
#  nize.
#TLS_CACERT /opt/qip/qip-ldapauth/lucent.pem

###
#TLS_CACERTDIR <path>
#  Specifies the path of a directory that contains  Certi-
#  ficate  Authority  certificates  in separate individual
#  files.   The   TLS_CACERT   is   always   used   before
#  TLS_CACERTDIR.

#TLS_CACERTDIR /opt/qip/qip-ldapauth/certs

###
#TLS_CERT <filename>
#  Specifies the file that contains  the  client  certifi-
#  cate.  This is a user-only option.
#TLS_CERT /opt/qip/qip-ldapauth/qip_ldapauth.pem

###
#TLS_KEY <filename>
#  Specifies the file that contains the private  key  that
#  matches  the  certificate  stored in the TLS_CERT file.
#  Currently, the private key must not be protected with a
#  password,  so it is of critical importance that the key
#  file  is  protected  carefully.  This  is  a  user-only
#  option.
#TLS_KEY /opt/qip/qip-ldapauth/qip_ldapauth.key

###
#TLS_CIPHER_SUITE <cipher-suite-spec>
#  Specifies acceptable cipher suite and preference order.
#  <cipher-suite-spec>  should  be  a cipher specification
#  for OpenSSL, e.g., HIGH:MEDIUM:+SSLv2.
#  example: openssl ciphers -v 'RSA:!COMPLEMENTOFALL'
#TLS_CIPHER_SUITE
   ALL:!ADH:!EXPORT56:RC4+RSA:+HIGH:+MEDIUM:+LOW:+SSLv2:+EXP:+eNULL

###
#TLS_RANDFILE <filename>
#  Specifies the file to  obtain  random  bits  from  when
#  /dev/[u]random  is  not available. Generally set to the
#  name of the EGD/PRNGD socket.  The environment variable
```

```
#  RANDFILE can also be used to specify the filename.
```

Troubleshooting LDAP authentication

To troubleshoot LDAP authentication, follow these steps:

1. Make sure the VitalQIP environment is sourced (refer to "How do I "source" or set my environment variables?" (p. 20-4)).

2. Make sure the *$QIPHOME/qip-ldapauth.pcy* file is properly edited.

3. Set the user's password to the LUCENT_PASSWORD environment variable:

   ```
   LUCENT_PASSWORD="secret"
   export LUCENT_PASSWORD
   ```

4. If running on Unix and using SSL, review "Encrypted connections with SSL/TLS" (p. 22-23) and ensure that the *ldaprc* file is edited properly.

5. Run the **`qip-ldapauth`** CLI (using SSL):

   `qip-ldapauth -v -s -u "jdoe"`

6. Review the output for errors. Sample output is shown:

   ```
   30-Sep-2008 10:13:49.772 - Checking for Deferred user match
   30-Sep-2008 10:13:49.773 - User "jdoe" is not in DeferUsers list.
   continuing..
   30-Sep-2008 10:13:49.774 - Decrypting encrypted bind password
   30-Sep-2008 10:13:49.775 - Setting protocol version to: 3
   30-Sep-2008 10:13:49.781 - LDAP URL: ldaps://10.54.0.42:636
   30-Sep-2008 10:13:49.782 - Base DN: DC=labw2k3,DC=com
   30-Sep-2008 10:13:49.782 - Search attribute: sn
   30-Sep-2008 10:13:49.783 - User id: "jdoe"
   30-Sep-2008 10:13:49.784 - Initializing with LDAP URL:
   ldaps://10.54.0.42:636
   30-Sep-2008 10:13:49.784 - Checking value of LDAP_OPT_REFERREALS
   30-Sep-2008 10:13:49.785 - LDAP_OPT_REFERRALS=-1
   30-Sep-2008 10:13:49.786 - Setting LDAP_OPT_REFERRALS to:
     LDAP_OPT_OFF
   30-Sep-2008 10:13:49.787 - Checking value of LDAP_OPT_REFERREALS
   ```

```
30-Sep-2008 10:13:49.787 - LDAP_OPT_REFERRALS=0
30-Sep-2008 10:13:49.788 - Binding with DN: CN=John
   Doe,CN=Users,DC=labw2k3,DC=com
30-Sep-2008 10:13:49.789 - Bind password: (not shown)
30-Sep-2008 10:13:50.208 - Binding with supplied DN and password
   succeeded
30-Sep-2008 10:13:50.209 - Searching..
30-Sep-2008 10:13:50.209 -  base_dn: DC=labw2k3,DC=com
30-Sep-2008 10:13:50.210 -  search filter: (sn=jdoe)
30-Sep-2008 10:13:50.210 -  Scope: Subtree
30-Sep-2008 10:13:50.254 - User "jdoe" found in LDAP server
30-Sep-2008 10:13:50.254 - Obtaining user DN
30-Sep-2008 10:13:50.257 - User DN: CN=John
   Doe,CN=Users,DC=labw2k3,DC=com
30-Sep-2008 10:13:50.257 - Now binding to LDAP server with user DN
   and password
30-Sep-2008 10:13:50.263 - User "jdoe" authenticated successfully
   with supplied password
30-Sep-2008 10:13:50.264 - Not authorizing user "jdoe" by policy
30-Sep-2008 10:13:50.265 - Exiting with 0 (Authenticated)
```

Enable the qip-ldapauth authentication callout for Tomcat

Purpose

The web client authenticates against the database. It can also be configured to use the Login Service. Use this procedure to enable the **qip-adapauth** authentication callout for the Apache Tomcat Server.

Procedure

To enable the **qip-ldapauth** authentication callout for the web server, follow these steps:

1. Stop the Apache Tomcat server.

2. In the application context file *ApplicationContext.xml* located in the *$QIPHOME/web/conf/ApplicationContext.xml* directory, uncomment the **qipCallout** bean:

```
<bean id="qipCallout" class="com.lucent.qip.auth.QipCallout">
<property name="qipConfig"><ref bean="qipConfig"/></property>
   </bean>
```

3. Be sure that the existing **qipConfig** bean is commented out (the section must be preceeded by **<!--** and followed by **-->**).

4. Add the following **qipConfig** bean.

```
<bean id="qipConfig" class="com.lucent.qip.config.QipConfigImpl"
   scope="singleton">
<property name="loginServer">
<value>127.0.0.1</value>
</property>
</bean>
```

5. Uncomment the callout property of the **commonService** bean, so it can use the **qipCallout** bean.

```
<bean id="commonService" ...
...
<property name="callout"><ref bean="qipCallout"/></property>
...
```

```
</bean>
```

6 Save the *ApplicationContext.xml* file.

7 The **qipConfig** bean looks for and parses the *qip.pcy* file to retrieve the Login Service information. Ensure that the *qip.pcy* file is found and that it contains the appropriate configuration.

8 Last, start Apache Tomcat server and verify that the web client UI also uses Active Directory.

END OF STEPS

The qip-ldapauth command

Description

The **qip-ldapauth** command, located in the *QIPHOME/userexits* directory, authenticates a VitalQIP administrator from an LDAP or Microsoft Active Directory. The **qip-ldapauth** command can accept parameters from the command line, a policy file, or environment variables. Parameters are processed in this order:

1. Command line
2. Environment variables
3. Policy file

Note: When running **qip-ldapauth** command from a command line, most information for parameters is taken from the *QIPHOME/qip-ldapauth.pcy* file.

Environment variables

The following environment variables are used by the **qip-ldapauth** command:

- **LUCENT_PASSWORD** - plain text user password
- **LDAP_SERVER_URL** - LDAP URL
- **LDAP_BASE_DN** - Base Distinguished Name
- **LDAP_BIND_DN** - A Distinguised Name to bind with
- **LDAP_BIND_PASS** - encrypted Bind password generated by running:

qip-ldapauth -e "plain text"

- **LDAP_USER_ID** - User ID to authenticate, such as **jdoe**, or **mjane**.

Synopsis

qip-ldapauth -c *<path to qip-ldapauthpcy file>* -H *<URI>* -h -u *<userid>* -p *<userpass>* -b *<basedn>* -d *<binddn>* -w *<passwd>* -P *<version>*
-a *<attribute>* -s -S -v -V -Z -e *<plain_text>* -x

Parameters

The following parameters can be used:

-c *<path to qip-ldapauth.pcy file>*	The path to the *qip-ldapauth* policy file. The default is *$QIPHOME/qip-ldapauth.pcy.*
-H *<URI>*	The LDAP Uniform Resource Identifier(s).
-h	Display the syntax without processing the command.
-u *<userid>*	User ID to authenticate.
-p *<userpass>*	Plain text password for the User ID.
-b *<basedn>*	Base distiguished name for search.
-d *<binddn>*	***Required if LDAP or Active Directory server requires binding before search.*** Bind distinguished name.
-w *<passwd>*	The encryped bind password (generated with the **-e** parameter).
-P *<version>*	The version of the LDAP protocol. The default is 3.
-a *<attribute>*	Attribute to use in LDAP search.
-s	Writes message to stderr.l.
-S	Write status message to stdout.
-v	Runs in verbose mode.
-V	Shows version information.
-Z	StartTLS request
-e *<plain_text>*	Generates encrypted password and exits.
-x	Shows examples.

Input example

The following example generates an encrypted password:

```
qip-ldapauth -e plain_text
```

Return values

Refer to "Return values" (p. 22-13) for a description of return values.

The qip-ldapauth.pcy file

When VitalQIP is installed, a sample *qip-ldapauth.pcy* file is installed in the QIPHOME directory. The sample file is called *qip-ldapauth.pcy_sample.pcy*. The file can be renamed to *qip-ldapauth.pcy* to define the policies for the **`qip-ldapauth`** command.

The *qip-ldapauth.pcy* file contains the following policies:

LDAPServerURL

Value	Default: None Allowed: “ldaps://<LDAP_Server_IP>:<Port>”, “ldap://<LDAP_Server_IP>:<Port>”
Description	The URL for LDAP server. The values that can be specified for either secure LDAP or non-secure LDAP. The URL must be entered as: • For secure LDAP: **ldaps://<Server_IP_address>:<port for encrypted connection over SSL>** • For non-secure LDAP: **ldap://<Server_IP_address>:<port for encrypted connection over SSL>** This policy is the command line equivalent of the **-H** parameter.

LogFile

Value	Default: None Allowed: “<path to qip-ldapauth.log>”
Description	The path to the *qip-ldapauth.log* file. If no path is specified, log messages are written to **stderr**.

Debug

Value	Default: Yes Allowed: Yes, No
Description	Determines if messages are written to *qip-ldapauth.log* file. If the policy is set to Yes, messages are written to the *qip-dapauth.log* file. This policy is the command line equivalent of the **-v** parameter.

LDAPProtocolVersion

Value	Default: 3 Allowed: numeric value
Description	The version of LDAP Protocol. This policy is the command line equivalent of the **-P** parameter.

BaseDn

Value	Default: None Allowed: Valid distinguished name
Description	Base distiguished name for search. This policy is the command line equivalent of the **-d** parameter.

BindDn

Value	Default: None Allowed: Valid Bind distinguished name
Description	***This policy must be set if the LDAP or Active Directory server requires binding before doing a search.*** Bind distinguished name. This policy is the command line equivalent of the **-b** parameter.

BindPass

Value	Default: None Allowed: encrypted password
Description	***This policy must be set if the LDAP or Active Directory server requires binding before doing a search*** Encrypted password generated by the **qip-ldapauth** commed. The encrypted password must be generated with: `qip-ldapauth -e plain_text` The encryped password is used to browse the policy file and keep the plain text password safe from hackers. Keep the *qip-ldapauth.pcy* file readable for the user who runs it. This policy is the command line equivalent of the **-w** parameter.

UidAttr

Value	Default: “sn” Allowed: **CN** for Miscrosoft Active Directory, **userid** for Open LDAP or iPlanet server.
Description	Attribute used for creating a search filter. The Microsoft Active Directory server uses the value **CN**, and the OpenLDAP or iPlanet server uses the **userid** as the value for the attribute. This policy is the command line equivalent of the **-a** parameter.

StartTLS

Value	Default: No Allowed: Yes, No
Description	***Available when qip-ldapauth command is running on UNIX.*** Determines if encryption occurs over a regular port. If this policy is set to Yes, encryption occurs over a regular port, and the LDAPServerURL policy can be set to: **`ldap://<server_IP_address>:389`**

DeferUsers

Value	Default: **`"qipman|updateservice|scheduleservice"`** Allowed: list of users to bypass authentication
Description	A list of users allowed to bypass the **`qip-ldapauth`** authentication process and instead allow VitalQIP to authenticate user. Use the | character to separate users in a list. For example: **`DeferUsers="qipman|updateservice|scheduleservice"`** The list of users may be truncated if the length of the line exceeds 512 bytes in Windows, 1024 bytes in Solaris, and 8192 bytes in Linux. If you wish to defer a lot of users, use the **`DeferUsersFile`** policy instead.

DeferUsersFile

Value	Default: None Allowed: File containing list of users
Description	Path of the file that contains one user per line. For example: **`DeferUsersFile="/path/deferusers.txt"`** If **`DeferUsers`** and **`DeferUsersFile`** policies are both used, the users are combined.

DnList

Value	Default: None Allowed: Distinguished name list

Description	A search for the user is performed in each distinguished name to find a match. Use **`DnList`** if the LDAP or Active Directory server is configured so that there is no bind or user. Only the user can bind with her/his password. In that case we have to know the DN of the user in order to bind first as we will not be able to bind as another user in order to search. The user will be authenticated with the first successful bind. The list of DNs may be truncated if the length of the line exceeds 512 bytes in Windows, 1024 bytes in Solaris, and 8192 bytes in Linux. If you wish to list a lot of DNs, use the **`DnListFile`** policy instead.

DnListFile

Value	Default: None Allowed: File containing Distinguished name list
Description	Reads the list of DNs from the file containing one DN per line. For example: **`DnListFile="/path/dnlist.txt"`** If **`DnList`** and **`DnListFile`** policies are both used, the DNs are combined.

The following policies are used together to refine the authentication process further:

Authorize

Value	Default: No Allowed: Yes, No
Description	Determines if a user is authorized after authentication. This policy is used with the following policies.

AuthorizationMode

Value	Default: None Allowed: All, None, RequireAttrVal, RequireFilter, RequireGroup
Description	Detemines if additional policies are used to verify a user. The following values can be used: • **All** - Uses the RequireAttrVal, RequireFilter, and policies to verify a user • **None** - Does not use the RequireAttrVal, RequireFilter, and policies to verify a user • **RequireAttrVa**l - Uses only the RequireAttrVal policy to verify a user • **RequireFilter** - Uses only the RequireFilter policy to verify a user • **RequireGroup** - Uses only the RequireGroup policy to verify a user

RequireAttrVal

Value	Default: None Allowed: Any LDAP valid attribute and value for the user
Description	If the Authorize policy is set to Yes, use this policy to verify the user. The Attribute and value are separated by |. For example, a user is authorized if the attribute room number is 7990. It is specified in this policy as: **`RequireAttrVal="roomNumber|7990"`**

RequireFilter

Value	Default: None Allowed: A valid LDAP filter
Description	If the Authorize policy is set to Yes, this policy verifies the user based on the specified LDAP filter.

RequireGroup

Value	Default: None Allowed: A valid LDAP group Sub DN

Description	If the Authorize policy is set to Yes, verifies that the user belongs to a group DN.

GroupAttr

Value	Default: None Allowed: A valid LDAP group attribute
Description	If the RequireGroup policy is set, specify the attribute for the group. For example, a member, uniqueMember, and so on. Microsoft Active Directory server uses `member`; the iPlanet and OpenLDAP server use `uniqueMember`.

23 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	23-2
Port tunneling	23-3
Backward compatibility of remote servers	23-4
Secure message transport	23-5
Configuring SSL	23-8
Remote proxies	23-11
Cleartext proxy	23-11
Secure proxy	23-12
Promotion proxy	23-13
Demotion proxy	23-15
Login proxies	23-18
Dynamic updates to secure MS DNS zones	23-19
Dynamic update messaging without DNS Update Service	23-19
Dynamic update messaging using the DNS Update Service	23-21
QIP Update Service policy to support secure zone updates	23-22

Introduction to secure message routes

VitalQIP provides increased security for message routes, and supports :

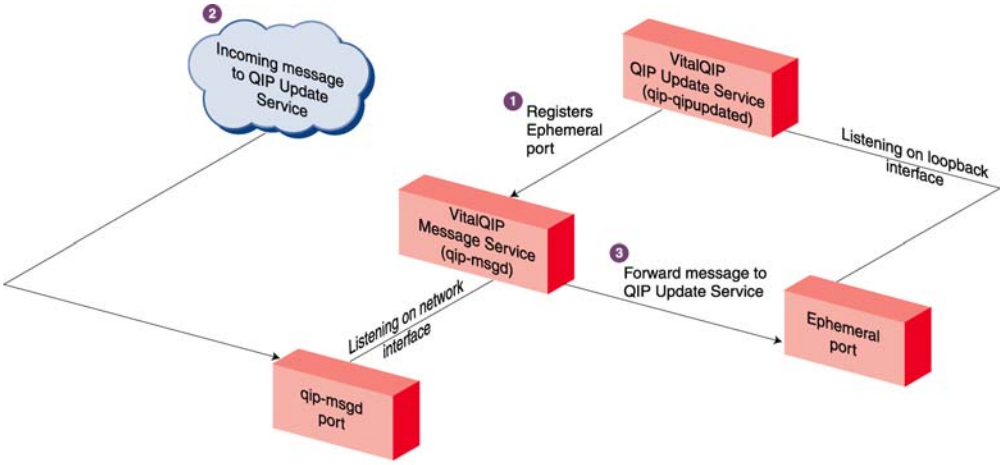
- Tunneling of all messages through the VitalQIP Message Service
- Use of Secure Sockets Layer (SSL) to provide message encryption and authentication of VitalQIP components
- Ability to make dynamic DNS updates to a secure zone hosted on a Microsoft DNS server

Note: 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 the following illustration.

Figure 23-1 Default non-secure message flow



After VitalQIP 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 number, as listed in the following table (for example, **`ListenPort=2490`**).

Note: Additionally, when an enterprise server is upgraded from pre-6.2 VitalQIP, the **AllowConnectionList** policy needs to be set to allow connections from remotes to the VitalQIP QIP Update Service and DNS Update Service by setting **`AllowConnectionList=All`** in *qip.pcy* file.

If you also have Audit Manager installed on remote servers, 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 23-1 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 or 7.x systems with new message route formats work.

After all systems are upgraded to 6.2/7.x and 6.2/7.x-style message routes, the services on the enterprise server should be set to listen on ephemeral ports.

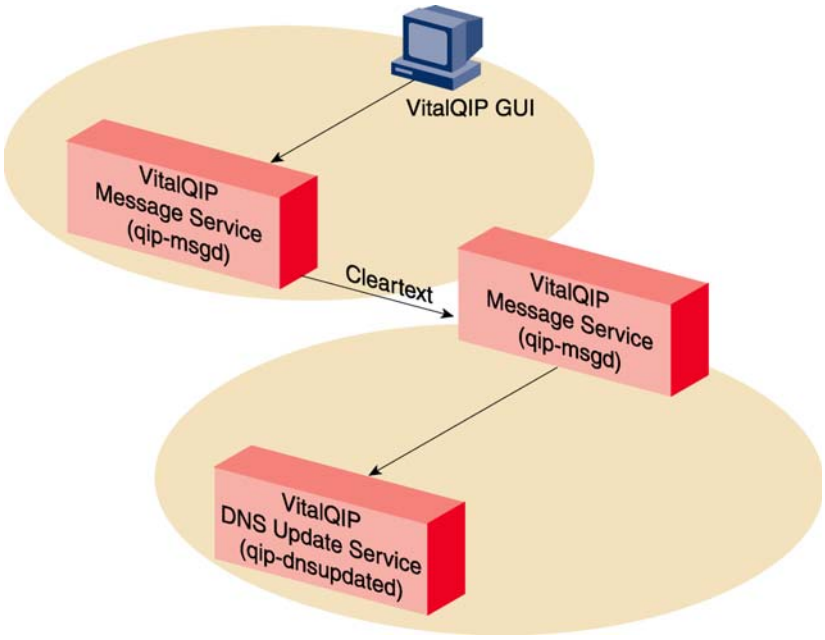
Note: SSL ***cannot*** be enabled for remote servers that are using legacy ports.

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 and 7.x, all messages are tunneled through a local Message Service on the sending machine, as shown in the following illustration.

Figure 23-2 Non-secure message transport tunneled through Message Service



In Figure 23-2, unsecured cleartext messages using the standard **MessageRoute** policy are sent from the VitalQIP GUI to the DNS Update Service:

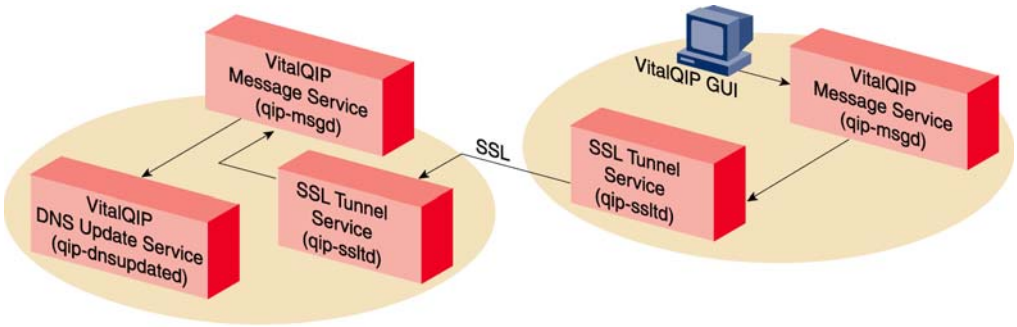
```
MessageRoute =
   <id>:<flags>:<ACK_timeout>:<desc>:<service_name>:<Server_IP>:…<*#>
```

This policy is described in detail on page 10.

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 the following illustration.

Figure 23-3 Secure message transport tunneled through SSL Tunnel and Message Service



In Figure 23-3, 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>:…<*#>
```

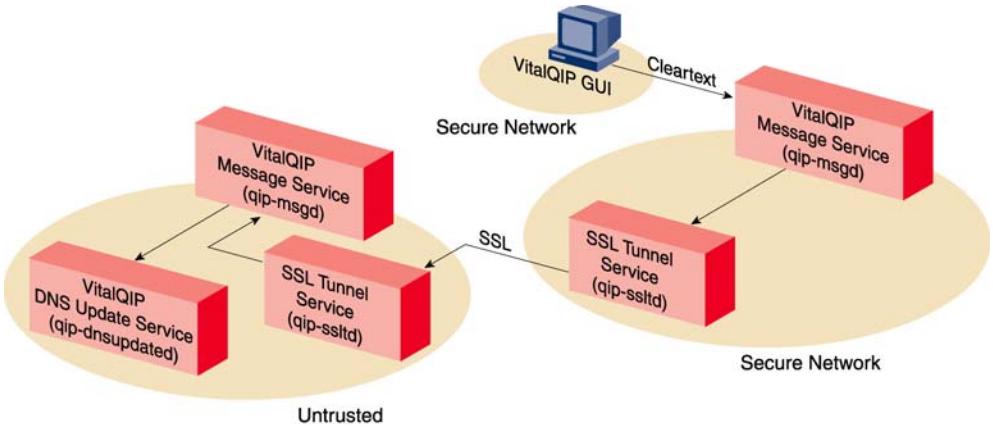
This policy uses the same settings as the MessageRoute policy.

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 2003 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 the following illustration.

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.

Figure 23-4 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” (p. 23-11).

Configuring SSL

Purpose

SSL is configured if you answered Yes in the Message Service Transport Communication Mode Selection screen during the installation of VitalQIP and entered SSL Self Signed Certificate fields. The information provided here 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

- Complete instructions on creating and managing a keys file can be found at:

  http://java.sun.com/j2se/1.4.2/docs/tooldocs/windows/keytool.html
- Alcatel-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 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 remote server installation, you are also 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.

Note: 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*, as described in "Global section" (p. 3-49).

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 for the following:

- Login Service
- Active Lease Service
- IBM DHCP Monitor Service
- MS DNS Update Service
- All push requests to the Remote Service.

Per route security

Message route traffic can also 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.

> **Note:** 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).

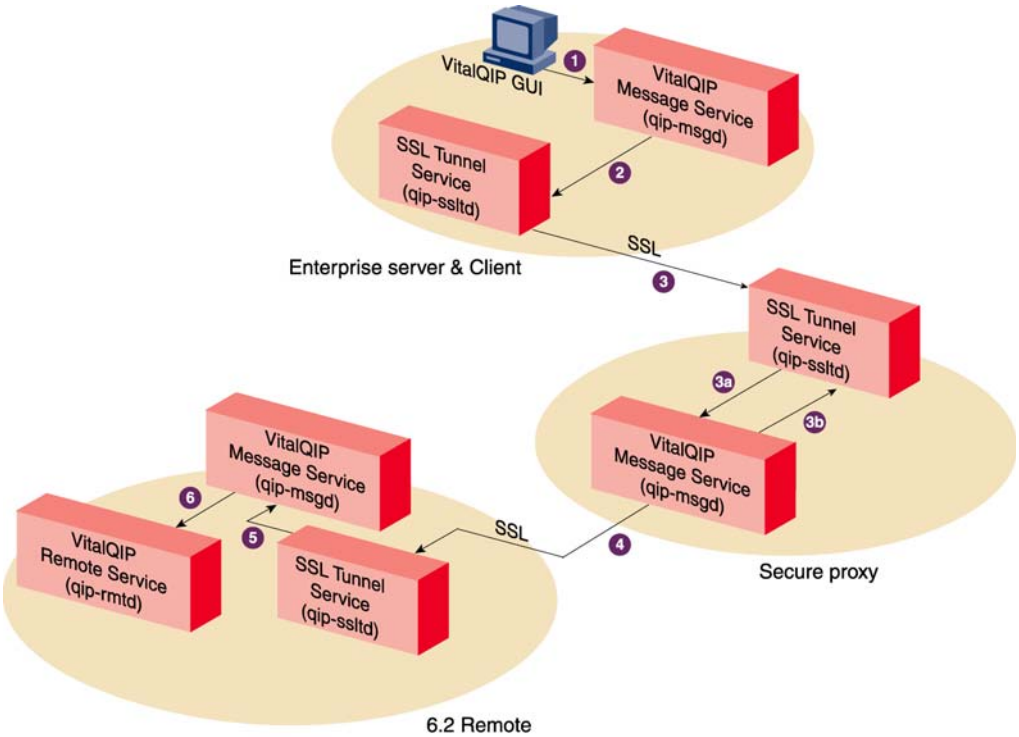
Note: Using a proxy to promote cleartext incoming connections to secure outgoing connections could be dangerous. If you need to do this, it is a good idea to lock down the **AllowConnectionsList** policy to be the minimum set necessary.

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 the following illustration.

Figure 23-5 Routing of push request through secure proxy to the remote server



The following message flow occurs:

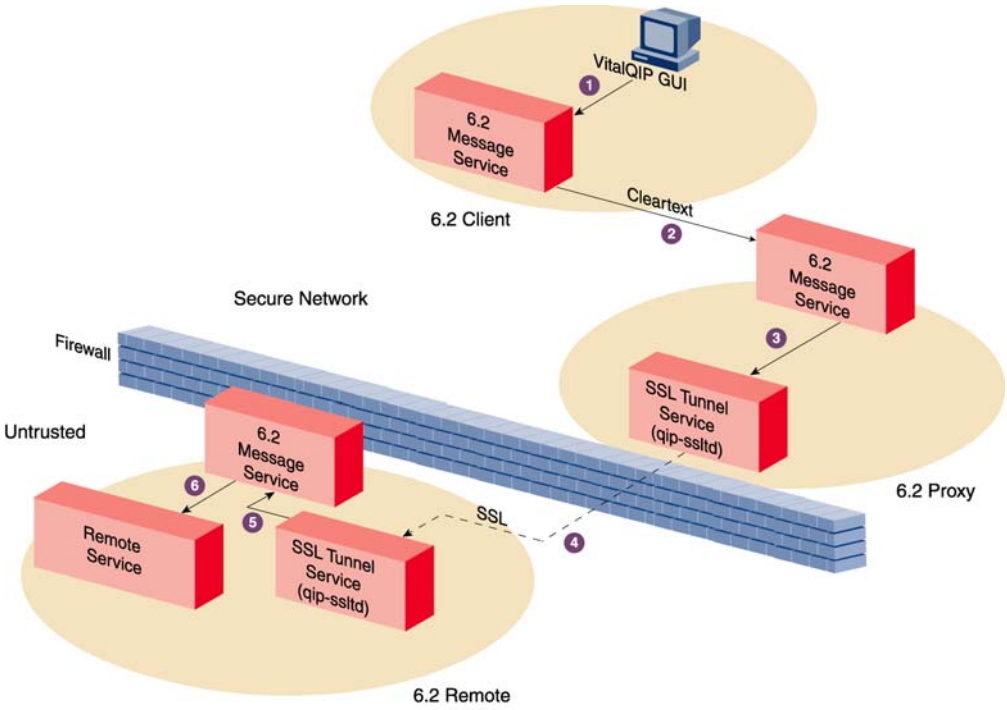
1. The GUI needs to send a push request to a remote, and sends a conduit connection request (with the IP address of the remote server, and the address of the proxy) to the local Message Service. The Message Service is bound to the Message Service port on the loopback interface from which it only accepts 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.
    a. 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.
    b. 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 23-6 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.

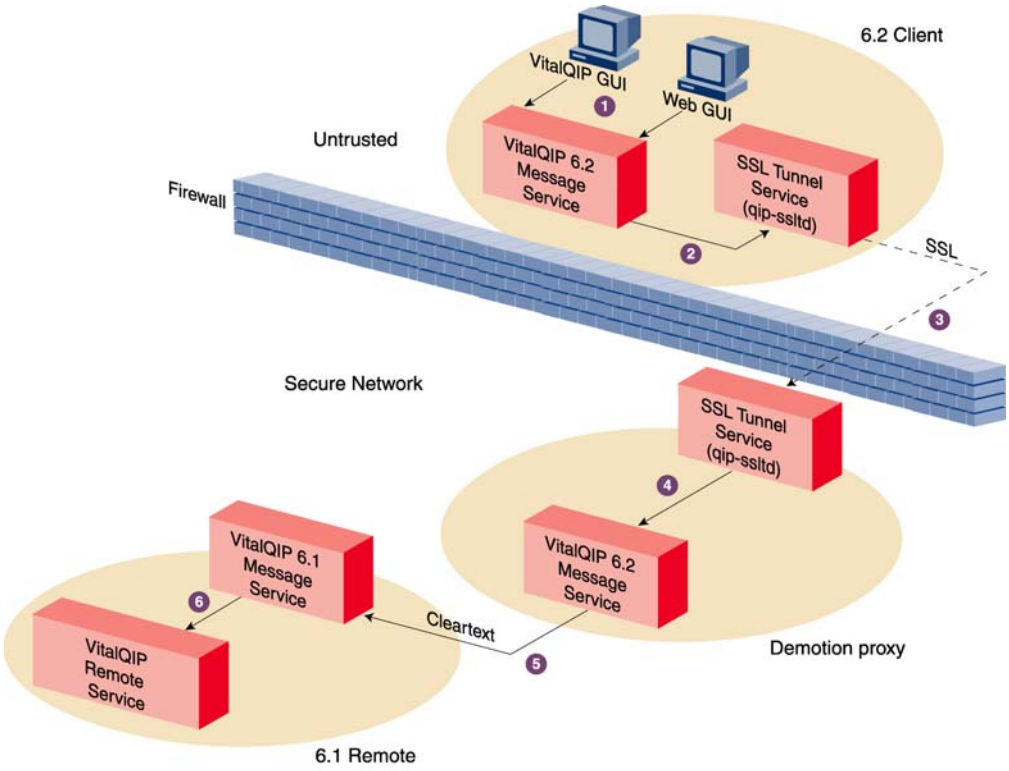
4. The proxy SSL tunnel makes a connection to the remote SSL tunnel. The remote SSL Tunnel is running on the Message Service port on the remote machine's network interface. Encrypted and authenticated TCP communication occur where both the initiator and recipient must be able to authenticate each other's 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.

Figure 23-7 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.

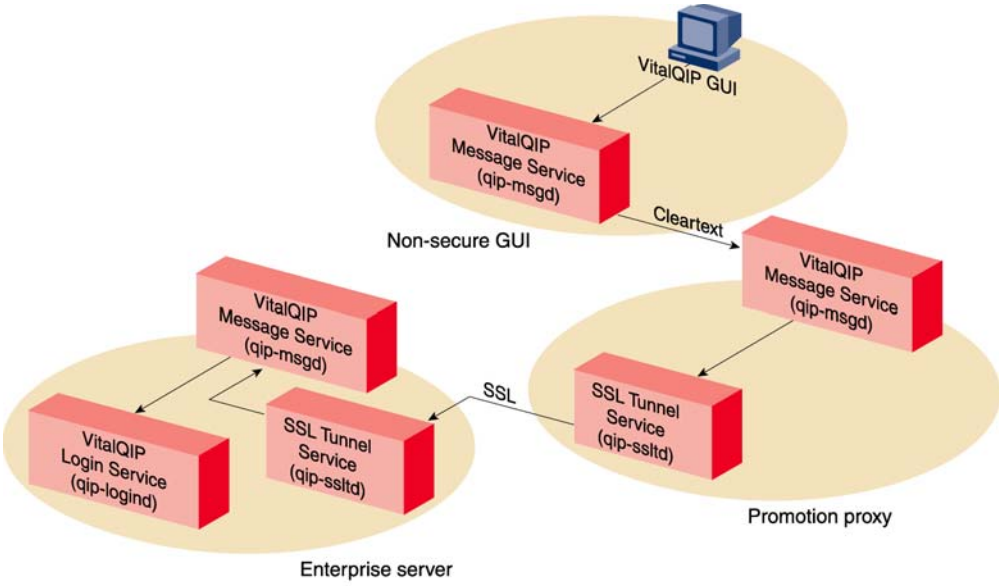
4. The proxy SSL Tunnel removes the encryption, and sends the conduit request to the proxy's Message Service. The proxy Message Service only accepts TCP cleartext communication on the loopback interface because the **SecureOutgoingProxyConnections** policy is set to False.
5. The proxy Message Service 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 takes the insecure login request from the client and promotes it to be a secure request to the Login Service, as shown in the following illustration.

Figure 23-8 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 2003 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.

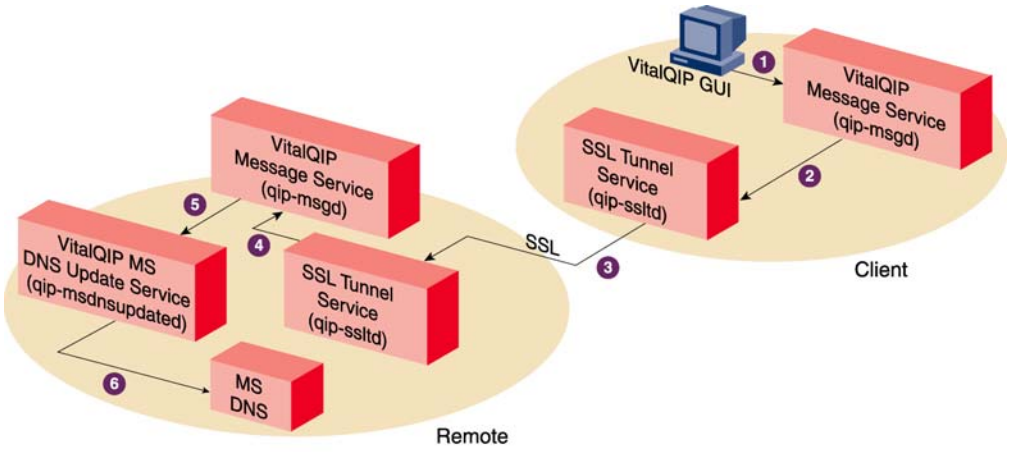
The DNS Update Service honors the proxies defined in the Server Profile and in the **Remote Server Proxy** global policy and routes messages through those proxies.

> **Note:** The update process for a Microsoft DNS server with secure zones (which are always Directory - integrated) is somewhat different from the process used for other types of DNS server or from other types of zones in MS-DNS. For other DNS server types or other zone types, the DNS Update Service makes the RFC 2136 dynamic update directly to the target DNS server.
>
> For secure zones on 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 the following illustration. The MS DNS Update Service applies the update securely to the DNS server.

Figure 23-9 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.

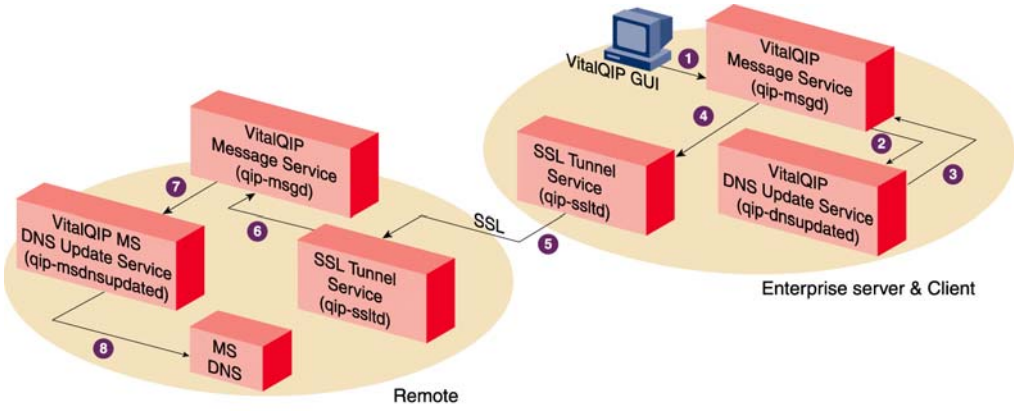
   Note: The information is passed along as a VitalQIP message until it reaches the MS DNS Update Service. This service uses the GSS-TSIG update protocol to apply the

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 to True, update information is passed through the DNS Update Service. For any other type of DNS server, or a non-AD-integrated MS-DNS server, or even a non-secure zone in a Directory-integrated MS-DNS, the DNS Update Service forwards an RFC 2135 dynamic update to DNS. But for a secure zone, the DNS Update Service forwards the information to the MS DNS Update Service to be applied to the Microsoft DNS server.

Figure 23-10 Dynamic update message flow for MS-DNS secure zones using the DNS Update Service



Note: 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 "Sample qip.pcy file" (p. 3-3).

Part V: Advanced DNS administration

Overview

Purpose

In Part V, you will find information on advanced DNS configurations and troubleshooting DNS.

Contents

Part V contains the following chapters:

Chapter 24, “Advanced DNS configurations”	24-1
Chapter 25, “Troubleshoot DNS”	25-1

24 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	24-3
Lucent DNS directives	24-3
BIND 9.x support	24-3
Manage Windows 2003 DNS servers	24-6
Configure Windows 2003 DNS server in VitalQIP	24-7
Windows 2003 DNS secure zones support	24-10
Background	24-10
Configure VitalQIP to manage Windows 2003 DNS secure zones	24-12
Set secure zone policies	24-16
Set secure zone policies at global level	24-17
Set secure zone policies at subnet level	24-18
Set secure zone policies at object level	24-19
ddns.conf with proxies for Windows 2003 secure servers	24-20
Secure dynamic updates	24-21
Lucent DNS server setup	24-25
Windows 2003 DC as KDC and Lucent DNS running on UNIX	24-26

Windows 2003 DC as KDC and Lucent DNS running on Windows	24-34
DNS client setup	24-44
VitalQIP DNS Update Service setup on UNIX	24-46
VitalQIP DNS Update Service setup on Windows	24-48
VitalQIP client setup on UNIX	24-48
VitalQIP client setup on Windows	24-50
Setup required if Lucent DNS Server is primary for the Windows Domain zone	24-51
Configure DNSSEC Support	24-56
About DNSSEC	24-56
DNSSEC userexit	24-57
Configure DNSSEC	24-58
External objects and resource records support	24-61
About external objects and resource records	24-61
How VitalQIP imports external object and resource records	24-62
How VitalQIP handles external objects and resource records	24-65
Modify external objects and resource records	24-67
About external object updates to DNS	24-68
Configure VitalQIP to collect external objects and resource records	24-69
Improve DNS push functionality	24-73
What happens with improved DNS push functionality	24-73
To improve DNS push functionality	24-75
Customize user exit scripts	24-77
Stages user exits occur	24-77
Rename user exit scripts	24-79
Sample UserExit.pcy file	24-81

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 4.x or Windows 2003, BIND 9.x.

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 4.x and BIND 9.x provide a configuration file called *named.conf*. The data files created by BIND require 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 may not function properly.

You can add extensions to *named.conf* Lucent DNS 4.x, and BIND 9.x) file through the VitalQIP client or Web Client. For more information on setting up the Corporate Extension, refer to the *VitalQIP User's Guide* or the *Web Client User's Guide*.

Note: For additional information on BIND, refer to *DNS and BIND* by Paul Albitz & Cricket Liu, published by O'Reilly & Associates.

Lucent DNS directives

In Lucent DNS 4.x and above, you can specify Lucent DNS-specific directives in a **`qddns`** block inside the **`options`** directive. 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.

In Lucent DNS 4.0, some directives are supported or supported on a limited basis. For detailed information on supported directives, refer to the *Lucent DNS Release Notes* for the specific version you have installed (Lucent DNS 4.0, Lucent DNS 4.1, or Lucent DNS 4.2).

BIND 9.x support

VitalQIP supports BIND 9.x by providing DNS server type, BIND 9.x in the Server Profile. The BIND 9.x 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 value is base 64 encoded and placed in the *named.conf* file and the *rndc.conf* file, so **rndc** can communicate with the BIND 9.x server. The **RNDC Path** parameter defines the pathname for the directory of the **rndc** executable (that is, *$QIPHOME/usr/bin* and ***not*** */usr/sbin/rndc*).

Note: When BIND 9.x configuration files are pushed on UNIX, the *named.conf* file and the *rndc.conf* file are placed in the same directory as the zone files by default. (You should make a link to each of these in */etc*.)

Note: 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
```

Note: The **Generate rndc.conf** parameter defaults to False on existing Bind 9.x 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, that value is used. If the directive is not specified, VitalQIP uses the Default TTL value from the Domain Profile as the value for the $TTL statement that is added to the beginning of the zone file.
- If a **`control`** statement is not specified, VitalQIP automatically adds the following lines to *named.conf* 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>";
};
```

Note: For BIND 9.x servers only, if the server is 9.2.x or earlier and accepts dynamic updates to pushed zones, you must have a user exit to stop named, remove *db.<zone>.jnl* files, and start named. If the server is 9.3.x or later and accepts dynamic updates to pushed zones, you can use the 9.2.x method or utilize the rndc freeze/thaw functions in a user exit. This prevents errors and failures for the pushed zones.

- 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.)

Manage Windows 2003 DNS servers

Microsoft supplies a DNS server with its Windows 2003 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 2003 DNS servers from a centralized platform. VitalQIP can also manage an Active Directory-integrated or file-based Windows 2003 DNS server (both secure and non-secure zones).

In addition to storing their zone information on disk, Windows 2003 DNS servers can store this information in Active Directory. When VitalQIP manages Active Directory-based Windows 2003 DNS servers, Windows DNS 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).

> **Note:** After you install the Windows 2003 DNS service, it is strongly recommended that you apply or reapply the latest Windows 2003 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 2003 DNS server. Refer to the instructions in the *VitalQIP Installation Guide* on how to install these components. For secure updates to Windows 2003 DNS zones, you must also install the VitalQIP MS DNS Update Service. The Remote Service is described in "qip-rmtd - VitalQIP Remote Service daemon" (p. 2-29) and Chapter 12, "VitalQIP Remote Service policies".

Zone information

The VitalQIP client allows Windows 2003 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 2003 DNS zone options, refer to 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 Lucent and BIND DNS servers. However, all server and zone options are set through **`dnscmd`**. Refer to Microsoft documentation for information on the **`dnscmd`** utility. The **`dnscmd.exe`** command from Microsoft Support Tools is required on the remote Windows 2003 DNS to support Windows 2003 DNS.

Configure Windows 2003 DNS server in VitalQIP

Purpose

This sectiondescribes how to configure a Windows 2003 DNS server in VitalQIP.

Procedure

To configure the Windows 2003 DNS server within VitalQIP, follow these steps:

1 Follow the instructions for the user interface in which you are working.

If you are using …	Then …
VitalQIP thick client UI	1. Select **Infrastructure | Server**. **Result:** The Server Profile Option window opens. 2. Click **OK** (since **Add New Server** is the default). **Result:** 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 can also include the dash (-) and underscore (_) characters, but cannot contain plus signs (+), exclamation marks (!), or underscores (_). 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**.

If you are using ...	Then ...
VitalQIP web client UI	1. Mouse over **DNS** and click **DNS Server**. **Result:** The **DNS Server Hierarchy** opens. 2. Mouse over the **Actions** menu and select **Add Server**. **Result:** The Add server page opens. 3. Select **DNS** from the **Server Class** drop-down list. 4. Select **Windows 2000 DNS** from the **Server Type** drop-down list. 5. Enter a new server host name and click **OK**. You can enter a maximum of 63 alphanumeric characters beginning with an alphanumeric. The host name can also include the dash (-) and underscore (_) characters character, but cannot contain plus signs (+), exclamation marks (!), or underscores (_) . 6. Enter a domain in the **Domain Name** field or click **...** to open the Domain Search page. a. Enter the first few letters of the domain name and/or the wildcard character (*) as needed, and click **Search**. **Result:** The results appear in the **Search Results** list. b. Highlight a domain in the **Search Results** list and click **OK**. **Result:** The domain you selected appears in the **Domain Name** field.

2 Define server parameters based on your functional requirements. Refer to “DNS servers” in Chapter 4 of the *VitalQIP User’s Guide* for more information on fields not discussed below.

- The **Default Directory** is a read-only field and is automatically set to *%systemroot%\system32\dns*.
- The **Email address for local and reverse zones** field is required. Enter a valid email address. This address appears in the SOA record of a zone file.
- **Prefix** and **Suffix Corporate Extensions** - These are free form text fields. The value is always expected in **`dnscmd`** format. Refer to the Microsoft documentation for more information on the **`dnscmd`** utility.
- The **Create “db.127.0.0”** option should always be set to “False”. This zone is automatically created by Windows 2003.
- **Boot Method** - Valid values are “File” based or Active “Directory” based. If Boot Method is set to Directory, Secure DNS updates for Windows 2003 can be enabled with the **Secure DNS Updates** parameter. Refer to “Windows 2003 DNS secure zones support” (p. 24-10) for details.

3 When you have set up the Windows 2003 DNS server parameters, follow the instructions for the user interface in which you are working.

If you are using …	Then …
VitalQIP thick client UI	1. Click **OK**. **Result:** A confirmation dialog box opens with the message `Server saved`. 2. Click **OK**. **Result:** The Server Profile window closes as the profile is saved.
VitalQIP web client UI	Click **Submit**. **Result:** The properties for the server you just defined open in the Server Properties page.

4 Define the rest of your infrastructure including domains, networks, subnets, and IP objects. Be sure to attach domains to this DNS server.

5 Use **Network Services | DNS Generation** to push data to this DNS server.

6 The DNS server must be started to generate the DNS data. If it is not started, use **Control Panel | Services** to start the Windows 2003 DNS server.

END OF STEPS

Windows 2003 DNS secure zones support

VitalQIP can manage and update secure zones on a Windows 2003 DNS server. This includes all zones, all records, changed zones only, and changed resource records.

Note: VitalQIP also supports secure zones with Windows 2000 in the same way as it does Windows 2003.

Important! Alcatel-Lucent does not recommend you manage a zone with both Windows 2003 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 for objects located in a secure Windows 2003 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, with pre-configured permissions, 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 “Secure message routes” (p. 23-1).

Background

Before you set up secure zones with Window 2003 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 2003 DNS server receives the update and uses the user’s security context to update Active Directory with the new information. The security context can be configured in VitalQIP. 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 2003 DNS secure zones

Purpose

To configure VitalQIP to use secure zones with Windows 2003 DNS, there are several steps you must complete.

Set up Window 2003 users

To use secure zones with a Windows 2003 DNS server, you need to set up two users on your Windows 2003 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 2003 DNS server documentation.

Once you have added the users, you will need the name and password of these users to set up your Windows 2003 DNS server.

Define a Windows 2003 DNS server with secure zone capabilities

To define a Windows 2003 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 2003 DNS server, select the server from the **Existing Server List** and click **OK**. If you are adding a new Windows 2003 DNS server, select **Add New Server** and click **OK**. The Server Profile windows opens.

3. Follow the procedures discussed in "Servers" in Chapter 4 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 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 2003 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 2003 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 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. Refer to the *VitalQIP User's Guide* for more information.
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 2003 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 2003 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

Purpose

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 DNS updates for DHCP clients to be made at a global level.

END OF STEPS

Set secure zone policies at subnet level

Purpose

You can set Windows 2003 DNS secure zone policies that allow DHCP clients to take ownership of dynamic object resource records for all dynamic objects in a subnet.

Procedure

To set 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.

3. Right-click on a subnet and select **Properties**. The Subnet Profile window opens.

4. Click the **Policies** tab.

5. Select **Subnet Level** under Windows 2000 Secure DNS Update Policies and Allow DHCP Client to Modify Dynamic Object Resource Records 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 assigned to Windows 2003 DNS.
   - **True** - enables the allow of DHCP clients to modify dynamic objects at the subnet level (that is, assigns from the proxy user).

   Note: The **Global Policy Level** is a read-only policy that shows the current Global Policy setting.

END OF STEPS

Set secure zone policies at object level

Purpose

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 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** under Windows 2000 Secure DNS Update Policies and Allow DHCP Client to Modify Dynamic Object Resource Records 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 2003 DNS.
- **True** - enables DHCP clients to modify dynamic objects at the subnet level (that is, assigns from the proxy user).

END OF STEPS

ddns.conf with proxies for Windows 2003 secure servers

The *ddns.conf* format has changed to support a potential list of proxies for each Windows 2003 DNS server. The existing *ddns.conf* format will continue to be generated for existing remote servers, since VitalQIP supports pushing to downlevel remote servers 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 a Windows 2003 DNS Server accepting secure updates through a proxy for the reverse zone 10.in-addr-arpa.

```
<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

Purpose

VitalQIP can be configured to send secure dynamic updates (GSS-TSIG) to the Lucent DNS 4.2 server, using Kerberos authentication. The Lucent DNS 4.2 server uses the ISC BIND implementation of GSS-TSIG for UNIX, which has been repurposed for Windows, and can provide secure dynamic DNS support in a Windows 2003 Kerberos environment, using the Windows Key Distribution Center (KDC). It can accept secure dynamic updates sent from:

- Windows 2003 and Windows 2008 native clients
- Windows 2003 Domain Controller (DC)
- VitalQIP clients
- VitalQIP DNS Update Service

Configuration for secure dynamic updates involves setup of the following components:

- Lucent DNS 4.2 server
- VitalQIP DNS Update Service
- VitalQIP client

Important! Kerberos uses timestamps during authentication. Ensure that all machines (KDC, DNS, and clients) are time-synchronized. A small time differential is tolerated, usually up to five minutes.

The instructions in this document apply to Lucent DNS 4.2 only. Instructions on using the previous implementation of secure dynamic updates that is compatible with Lucent DNS 4.0 and 4.1 are contained in the following manuals and can be downloaded from the support website.

- Lucent DNS 4.0: 190-409-042R7.0, Issue 2
  190-409-042R7.1, Issue 5
  190-409-042R7.2, Issue 4
- Lucent DNS 4.1: 190-409-042R7.1, Issue 5
  190-409-042R7.2, Issue 4

Reference information

For convenience, descriptions of parameters referenced in the following sections are included here. More comprehensive information can be found in online Kerberos and Microsoft documentation.

Table 24-1 net user command parameters

Parameter	Description
username	The name of the user account you want to add, delete, modify, or view. The name of the user account can have as many as 20 characters.
password	Assigns or changes a password for the user's account. It can contain as many as 14 characters.
/domain	Performs the operation on the primary domain controller of the current domain.
/add	Adds a user account to the user accounts database.
/comment:"text"	Provides a descriptive comment about the user's account (maximum of 48 characters). Be sure to put quotation marks around the text you use.
/expires:never	The never option sets no time limit on the account.
/fullname:"name"	Represents a user's full name (rather than a user name). Enclose the name in quotation marks.
/times:all	The all option specifies that a user can always log on, and a blank value specifies that a user can never log on.
/countrycode:0	Uses the operating system country code to implement the specified language files for a user's help and error messages. A value of 0 signifies the default country code.

ktpass utility syntax

The **ktpass** utility uses the following syntax. Refer to the following table for a description of the parameters.

```
ktpass -in <User>.keytab -out <User>.keytab
   -princ <User>/<hostname>.<domain>@<REALM> -mapuser <User> -pass <password>
   -crypto DES-CBC-MD5
   -ptype KRB5_NT_PRINCIPAL|KRB5_NT_SRV_HST|KRB5_NT_SRV_INST /?|/h|/help
```

Table 24-2 ktpass parameters

Parameter	Description
-in	Specifies the *keytab* file to read. ***<User>*** indicates the AD user account that updates DNS.
-out	Specifies the name of the Kerberos *keytab* file to generate. ***<User>*** indicates the user account that updates DNS. **Note:** This is the *keytab* file you copy to the DNS server.

Parameter	Description
-princ	Specifies the principal name, as follows: ***<User>*** User that updates DNS. ***<hostname>*** The host running the VitalQIP client. If this value is a generic principal name to be used on multiple hosts, the ***<hostname>*** is not needed. The principal name would look like ***<User>/<domain>@<REALM>***. ***<domain>*** The domain to which the ***<User>*** is assigned. ***<REALM>*** The Windows 2003 domain of the KDC in uppercase.
-mapuser	Maps the name of the Kerberos principal specified by the **princ** parameter to the specified local AD user name.
-pass	Specifies a password for the Kerberos principal user name specified in the **princ** parameter. The password must match the password that was defined for that AD user account in the Active Directory Users and Computers MMC snap-in or with **net user**. Otherwise, **ktpass** will write out a *keytab* with an incorrect encryption key. **Note:** If the argument to **-pass** is set to *****, **ktpass** prompts for the password on the console.
-crypto	Sets the encryption type to use. Alcatel-Lucent requires **DES-CBC-MD5** for use with Microsoft Windows 2003 KDC.
-ptype	Specifies the principal type. • **KRB5_NT_PRINCIPAL** General principal type. Use this type for the VitalQIP client's principal. • **KRB5_NT_SRV_HST** Host service instance. Use this type for the DNS Server's principal. • **KRB5_NT_SRV_INST** User service instance. Use this type for the DNS Update Service's principal.
/?|/h|/help	Displays command-line help.

ktutil utility syntax

ktutil uses the commands listed in the following table.

Table 24-3 ktutil command parameters

Command	Description
rkt *keytab_file*	Reads the contents of ***keytab_file*** and adds the keys to the keylist.
l | list	Displays the principals for the keys that are stored in the keylist.
wkt *keytab_file*	Writes the content of the keylist into ***keytab_file***. Rather than overwrite a *keytab* file, **wkt** appends the keylist to the existing file. To delete an entry from the *keytab* file, use **rkt** to read in the file, use **delent** to delete the entry, and then use **wkt** to write to a new file. Finally, replace the old system *krb5.keytab* file with the new file.
delent *slot_no*	Deletes an entry from the current keylist. Specify the entry by the keylist slot number.
q | quit | exit	Exits the utility.

Lucent DNS server setup

Overview

Purpose

The Lucent DNS 4.2 server supports GSS-TSIG updates in two different scenarios, as shown in the following table.

Table 24-4 GSS-TSIG configuration on Lucent DNS 4.2 servers

Scenario	KDC	qddns	Client	Configuration
1	Windows 2003 DC	UNIX	Windows	Refer to "Windows 2003 DC as KDC and Lucent DNS running on UNIX" (p. 24-26).
2	Windows 2003 DC	Windows	Windows	Refer to "Windows 2003 DC as KDC and Lucent DNS running on Windows" (p. 24-34).

Important! Windows 2008 DC is not supported as a KDC at this time.

Windows 2003 DC as KDC and Lucent DNS running on UNIX

The following steps need to be performed as an Administrator user:

1. Create a user account in the Windows Domain. Follow the steps in "Create an Active Directory (AD) user account" (p. 24-27).
2. Extract the Kerberos DNS principal to a *keytab* file in the Windows Domain. Follow the steps in "Extract the Kerberos principal" (p. 24-27).
3. Transfer the *keytab* file to the UNIX machine where Lucent DNS 4.2 will be running and insert the DNS principal in */etc/krb5.keytab*. Follow the steps in "Transfer keytab file and create/update /etc/krb5.keytab" (p. 24-27).
4. Set up the Kerberos configuration file (*krb5.conf*) on the DNS server machine. Follow the steps in "Create or edit the Kerberos configuration file" (p. 24-28).
5. Configure Lucent DNS 4.2 for GSS-TSIG from the VitalQIP GUI. Follow the steps in "Configure the Lucent DNS 4.2 server" (p. 24-30) and in "Modify the hosts file" (p. 24-33).

Prerequisites

Ensure that you have met the following requirements.

- Windows 2003 Domain Controller with Microsoft 2003 Server Resource Kit installed. The tool **`ktpass`** is needed to extract the Kerberos principal for the Lucent DNS 4.2 server.

  Note: Secure update with Windows 2008 Domain Controller is not currently supported.
- Lucent DNS 4.2 installed.
- Domain name (or IP address) of the Windows Domain Controller.
- FQDN of the UNIX machine where Lucent DNS 4.2 will be running.
- Kerberos tools must be installed on the UNIX machine where Lucent DNS 4.2 will be running. The tool **`ktutil`** is needed to update/create the */etc/krb5.keytab* file.
- The user account must be a Windows 2003 domain account and not just a local machine account.

Create an Active Directory (AD) user account

A unique user is required. This name can be any meaningful name with a strong password. User names and principals must be unique within AD.

> **Important!** Alcatel-Lucent recommends that the password for a Kerberos user account be set to ***never*** expire, otherwise you will need to recreate the *keytab* file, as well as update all servers that are using the expired account with the updated key information each time a password expires.

On the Windows KDC machine, add the new user account with the Active Directory Users and Computers MMC snap-in, or use the **`net user`** command, described in Table 24-1, "net user command parameters" (p. 24-22). In the following example (entered without a line break), `dnsTest` is created with a password of `test123`:

```
C:\> net user "dnsTest" "test123" /domain /add /comment:"NA" /expires:never
   /fullname:"dnsTest" /times:all /countrycode:0
```

Extract the Kerberos principal

On the Windows KDC, extract the Kerberos principal for the DNS server and map it to the user account. Use the **`ktpass`** utility, described in Table 24-2, "ktpass parameters" (p. 24-22), to generate the *keytab* file. The following example uses `dns1.example.com` as the FQDN in the EXAMPLE.COM Windows Domain and maps the user `dnsTest`, as follows.

```
C:\>ktpass -out dnsTest.keytab -princ DNS/dns1.example.com@EXAMPLE.COM
   -mapuser dnsTest -pass test123 -crypto DES-CBC-MD5 -ptype KRB5_NT_SRV_HST
```

You should see messages similar to the following:

```
Targeting domain controller: windc.example.com
Successfully mapped DNS/dns1.example.com@EXAMPLE.COM to dnsTest.
Building salt with principalname DNS/dns1.example.com and domain EXAMPLE.COM...
Hashing password with salt "EXAMPLE.COMDNSdns1.example.com".
Key created.
Output keytab to dnsTest.keytab:
Keytab version: 0x502
keysize 61 DNS/dns1.example.com@EXAMPLE.COM ptype 2 (KRB5_NT_SRV_HST) vno 3
etype 0x3 (DES-CBC-MD5) keylength 8 (0xe923e50167078a64)
Account dnsTest has been set for DES-only encryption.
```

Transfer keytab file and create/update /etc/krb5.keytab

First, securely transfer the *keytab* file with the Kerberos principal to the DNS server (`dnsTest.keytab` in the example). If you use **`ftp`**, ensure that you use binary mode.

Second, as root user on the DNS server, merge the principal to the system *krb5.keytab* file. Follow these steps.

1 Run the Kerberos keytab file maintenance utility **ktutil**, described in Table 24-3, "ktutil command parameters" (p. 24-24). The following example uses `dnsTest.keytab`.

```
# ktutil
ktutil:  rkt dnsTest.keytab
ktutil:  l
slot KVNO Principal
---- ---- ---------------------------------
1    3    DNS/dns1.example.com@EXAMPLE.COM
```

Result: You should see the DNS principal that you exported.

2 Add the principal to */etc/krb5.keytab*. with the **wkt** command. For example:

```
ktutil:  wkt /etc/krb5.keytab
ktutil:  quit
```

3 Ensure the principal is added correctly to the system *krb5.keytab* file with the **rkt** command. For example:

```
# ktutil
ktutil:  rkt /etc/krb5.keytab
ktutil:  l
slot KVNO Principal
---- ---- -------------------------------------------------
1    3    DNS/dns1.example.com@EXAMPLE.COM
ktutil:  quit
```

Note: This output should include the output from step 1, with possibly other principal data.

END OF STEPS

Create or edit the Kerberos configuration file

The Kerberos configuration file needs to be verified to ensure that the Windows 2003 domain and appropriate KDC are listed.

1 Open a text editor and create (or modify) the Kerberos configuration file (*/etc/krb5.conf*).

2 Add (or modify) the following lines:

a. Ensure there is a realm defined for the Windows 2003 Domain under the **[realms]** tag. The format is as follows:

```
[realms]
<REALM> =
{
   kdc = <IP_address_or_name_of_Windows_KDC>:88
   admin_server = <IP_address_or_name_of_Windows_KDC>:749
   default_domain =
   <DNS_zone_of_local_machine_or_other_zone_if_desired>
```

Note: Using the IP address instead of the FQDN for the kdc and admin_server eliminates some potential resolution issues that might cause GSS-TSIG not to work on the server.

[realms] example

The example below uses EXAMPLE.COM Windows Domain for the realm, the Domain Controller's FQDN (windc.example.com) for the kdc and admin server, and the DNS server's domain (example.com) for default_domain:

```
EXAMPLE.COM =
{
    kdc =  windc.example.com
    admin_server = windc.example.com
    default_domain = example.com
}
```

b. Ensure the default realm and encryption types are set under the **[libdefaults]** tag. The format is as follows:

```
[libdefaults]
default_realm = <REALM>
default_tkt_enctypes = des-cbc-md5
default_tgs_enctypes = des-cbc-md5
```

Note: Encryption types must be des-cbc-md5 when using a Windows KDC.

[libdefaults] example

The example below uses EXAMPLE.COM Windows Domain for the realm:

```
default_realm = EXAMPLE.COM
default_tkt_enctypes = des-cbc-md5
default_tgs_enctypes = des-cbc-md5
```

c. Ensure the domain-to-realm mappings include the DNS server's local domain under the **[domain_realm]** tag. The format is as follows:

```
[domain_realm]
.<DNS_zone> = <REALM>
<DNS_zone> = <REALM>
```

Note: The **`[domain_realm]`** heading lists DNS zones and the realm to be associated with that zone. Typing a period (.) before the DNS zone name means to use the realm for all subdomains of that domain. A DNS zone with no period means to use the realm for this domain.

[domain_realms] example

The example below uses example.com:

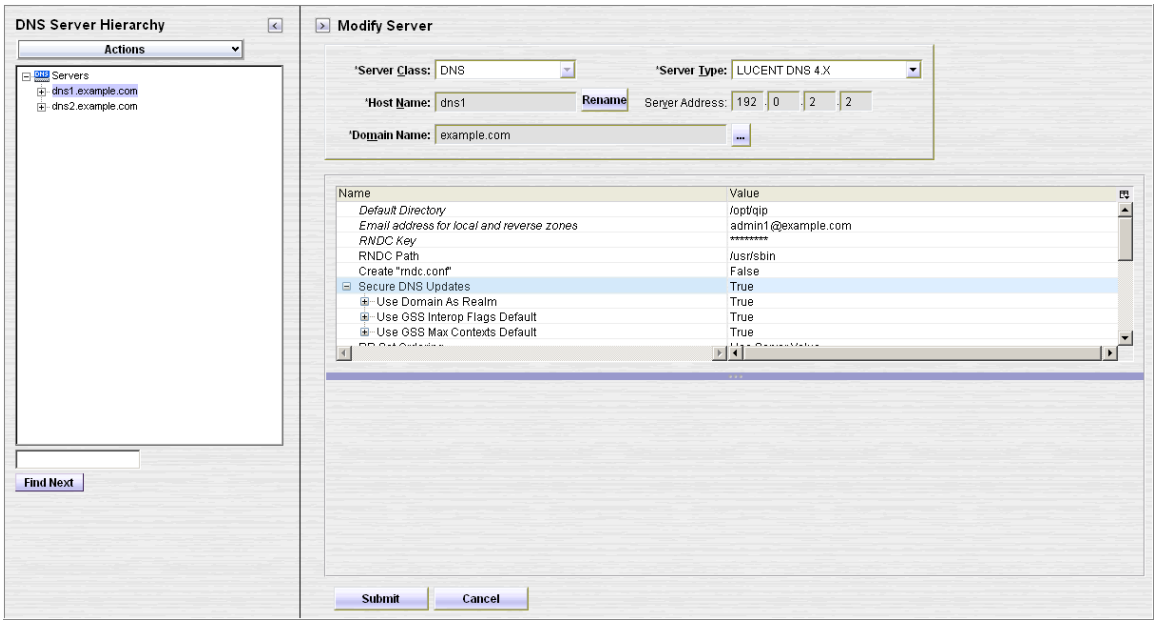
```
example.com = EXAMPLE.COM
```

3 Save the file as **`/etc/krb5.conf`**.

END OF STEPS

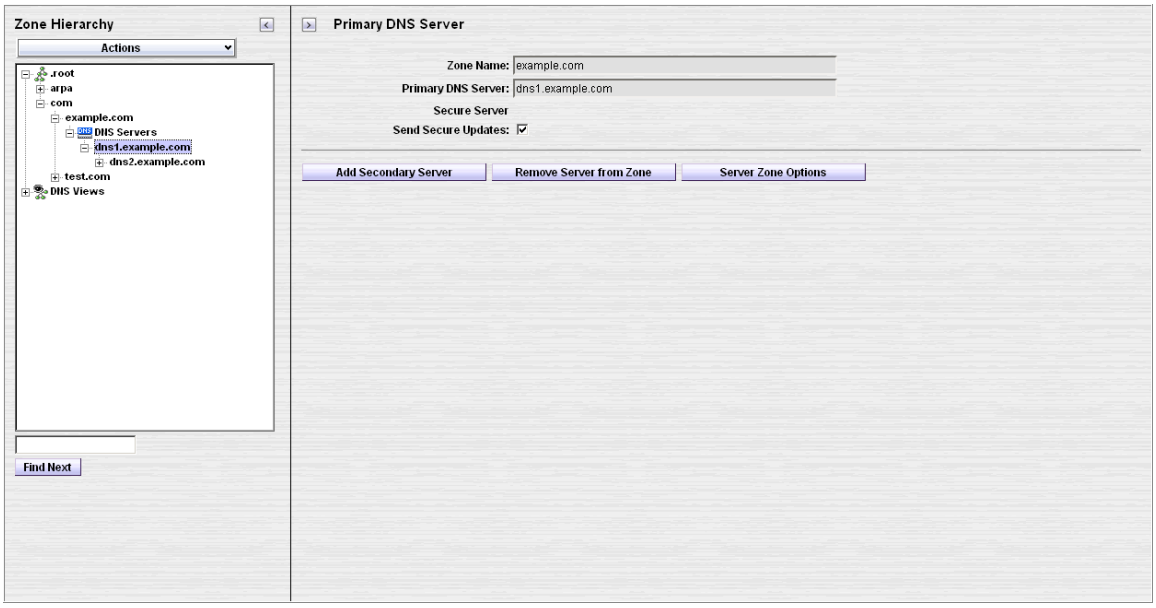
Configure the Lucent DNS 4.2 server

1 In the VitalQIP web client, locate the Lucent DNS 4.2 server in the DNS Server Hierarchy and set the **Secure DNS Updates** parameter to **True**.



2 If the realm name is not the same as the domain name, set the **Use Domain as Realm** parameter to False and specify a different realm name with the **Override Realm** parameter. The **GSS Interop Flags** sub-parameter has no affect on Lucent DNS 4.2 and should not be changed.

3 In the VitalQIP web client, locate the same server under ***each*** zone in the Zone Hierarchy. In the Primary DNS Server page, ensure that the **Send Secure Updates** check box is checked. Repeat this step in each zone where the server is the primary.



Important! Any Secondary DNS servers (such as dns2.example.com in the illustration above) that you wish to accept secure updates should also have the **Send Secure Updates** check box checked in the desired zones.

4 To check that the configuration files generated by VitalQIP are set up correctly, generate the DNS files (using DNS Generation in the VitalQIP web client), and check the **`gss-principal`** setting in the **`options`** block and the **`secure-updates`** setting in the **`zone`** block of each secure zone. A sample *named.conf* using earlier examples is shown below:

```
##########################################
# Lucent 4.x name server configuration file
##########################################

controls {
        inet 127.0.0.1 allow { localhost; } keys { "rndc-key"; };
};

key "rndc-key" {
        algorithm "hmac-md5";
        secret "dGVzdA==";
```

```
};
options {
    directory "/opt/dns_test/gss_tsig";
    pid-file "named.pid";
    tkey-domain ".";
    qddns
    {
       gss-principal "DNS@dns1.example.com";
    };
};

zone "example.com" in {
        type master;
        file "db.example.com";
        allow-update { any; };
        allow-query { any; };
        allow-transfer { any; };
        notify no;
    qddns
    {
        secure-updates yes;
    };
};

zone "0.0.127.in-addr.arpa" in {
        type master;
        file "db.127.0.0";
};
```

Note that the DNS principal is specified with the directive **`gss-principal`** in the **`qddns`** block in the **`options`** block. The zone `example.com` is configured as a secure zone by the directive **`secure-updates yes`** in the **`qddns`** block of the **`zone`** block. Also note that the Kerberos realm EXAMPLE.COM is not appended at the end of the principal; the default realm from *krb5.conf* is used.

5 Also, check the zone file and verify that the MNAME and a matching NS record exist in the zone. The NS glue record must resolve to an address record for native Windows clients.Otherwise, secure updates from native Windows clients will not work. In the sample *db.example.com* shown below, the MNAME (`dns1.example.com`) is highlighted and matches an NS record with the NS glue record (`dns1.example.com. IN A 10.100.30.50`):

```
$ORIGIN .
$TTL 10800  ; 3 hours
example.com     IN SOA  dns1.example.com. root.example.com. (
```

```
                    129         ; serial
                    10800       ; refresh (3 hours)
                    3600        ; retry (1 hour)
                    604800      ; expire (1 week)
                    3600        ; minimum (1 hour)
                    )
                NS  dns1.example.com.
dns1.example.com.    IN  A   10.100.30.50
```

Note: An additional configuration step is necessary for a Lucent DNS server that is primary for the Windows Domain zone. Refer to "Setup required if Lucent DNS Server is primary for the Windows Domain zone" (p. 24-51).

END OF STEPS

Modify the hosts file

Ensure that a reverse lookup for the hostname of the DNS server's IP address either does not resolve or resolves to a host or host/domain that maps to the Kerberos realm defined in */etc/krb5.conf*. Typically, the */etc/hosts* file needs to be edited to include the DNS server's IP address. Note that the first value after the IP address in the hosts file must be the FQDN of the DNS Server, and must also match the DNS server Kerberos principal's FQDN.

1 If your DNS principal name is fully qualified when it is entered into the *keytab* file, the first host entry in */etc/hosts* 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's domain name, as specified in the *keytab* file).

2 Ensure that the name of the KDC does not resolve in */etc/hosts* or resolves to the proper IP address, or that the IP address of the KDC does not resolve in */etc/hosts* or resolves to the proper domain.

A sample *hosts* file is shown below:

```
# hosts file
# dns1.example.com is the FQDN for the DNS server
# which will accept secure dynamic update
127.0.0.1       localhost
10.100.30.50    dns1.example.com dns1
```

END OF STEPS

Windows 2003 DC as KDC and Lucent DNS running on Windows

The following steps need to be performed:

1. Create a user account in the Domain. Follow the steps in "Create an Active Directory (AD) user account" (p. 24-27).
2. Extract the Kerberos DNS principal to a *keytab* file. Follow the steps in "Extract the Kerberos principal" (p. 24-27).
3. Transfer the *keytab* file to the Windows machine where Lucent DNS 4.2 will be running and copy the DNS principal to *%QIPHOME%\named\etc\krb5.keytab*. Follow the steps in "Transfer keytab file and create %QIPHOME%\named\etc\krb5.keytab" (p. 24-35).
4. Set the KRB5_CONFIG system environment variable on the DNS server machine. Follow the steps in "Set the KRB5_CONFIG environment variable" (p. 24-35).
5. Set up the Kerberos configuration file (*krb5.conf*) on the DNS server machine. Follow the steps in "Create or edit the Kerberos configuration file" (p. 24-37).
6. Configure Lucent DNS 4.2 for GSS-TSIG from the VitalQIP GUI. Follow the steps in "Configure the Lucent DNS 4.2 server" (p. 24-30) and "Modify the hosts file" (p. 24-33).

Prerequisites

Ensure that you have met the following requirements.

- Windows 2003 Domain Controller with Microsoft 2003 Server Resource Kit installed. The tool **`ktpass`** is needed to extract the Kerberos principal for the Lucent DNS 4.2 server.

  Note: Secure update with Windows 2008 Domain Controller is not currently supported.
- Lucent DNS 4.2 installed.
- Domain name of the Windows Domain Controller.
- IP address of the Windows Domain Controller.
- The file *c:\WINDOWS\system32\config\netlogon.dns* from the Windows Domain Controller (only if server is hosting the primary DC zone).
- FQDN of the Windows machine where Lucent DNS 4.2 will be running.
- The user account must be a Windows 2003 domain account and not just a local machine account.

Create an Active Directory (AD) user account

A unique user is required. This name can be any meaningful name with a strong password. User names and principals must be unique within AD.

> **Important!** Alcatel-Lucent recommends that the password for a Kerberos user account be set to ***never*** expire, otherwise you will need to recreate the *keytab* file, as well as update all servers that are Gusing the expired account with the updated key information each time a password expires.

Add the new user account with the Active Directory Users and Computers MMC snap-in, or use the **`net user`** command, described in Table 24-1, “net user command parameters” (p. 24-22). In the following example (entered without a line break), `dnsTest` is created with a password of `test123`:

```
C:\> net user "dnsTest" "test123" /domain /add /comment:"NA" /expires:never
   /fullname:"dnsTest" /times:all /countrycode:0
```

Extract the Kerberos principal

Extract the Kerberos principal for the DNS server and map it to the Windows user account. Use the **`ktpass`** utility, described in Table 24-2, “ktpass parameters” (p. 24-22), to generate the *keytab* file. The following example uses `dns1.example.com` as the FQDN and maps the user `dnsTest`, as follows.

```
C:\>ktpass -out dnsTest.keytab -princ DNS/dns1.example.com@EXAMPLE.COM
   -mapuser dnsTest -pass test123 -crypto DES-CBC-MD5 -ptype KRB5_NT_SRV_HST
```

You should see messages similar to the following:

```
Targeting domain controller: windc.example.com
Successfully mapped DNS/dns1.example.com@EXAMPLE.COM to dnsTest.
Building salt with principalname DNS/dns1.example.com and domain EXAMPLE.COM...
Hashing password with salt "EXAMPLE.COMDNSdns1.example.com".
Key created.
Output keytab to dnsTest.keytab:
Keytab version: 0x502
keysize 61 DNS/dns1.example.com@EXAMPLE.COM ptype 2 (KRB5_NT_SRV_HST) vno 3
etype 0x3 (DES-CBC-MD5) keylength 8 (0xe923e50167078a64)
Account dnsTest has been set for DES-only encryption.
```

Transfer keytab file and create %QIPHOME%\named\etc\krb5.keytab

First, securely transfer the *keytab* file with the Kerberos principal to the DNS server (`dnsTest.keytab` in the example). If you use **`ftp`**, ensure that you use binary mode.

Second, on the DNS server, copy the keytab file to *%QIPHOME%\named\etc\krb5.keytab*.

Set the KRB5_CONFIG environment variable

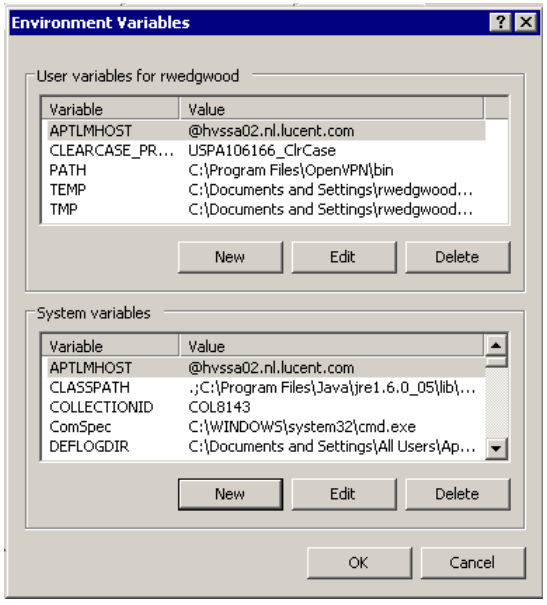
Set the path of the Kerberos configuration file in the **`KRB5_CONFIG`** system environment variable. Follow these steps.

1 Open **Control Panel | System** and click the **Advanced** tab.

Result: The System Properties window opens.

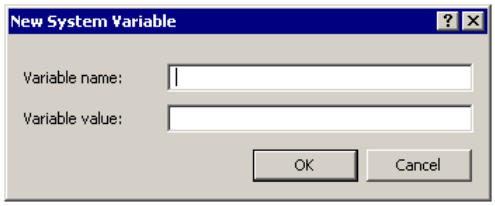
2 Click **Environment Variables**.

Result: The Environment Variables window opens.



3 Click **New** in the **System variables** section.

Result: The New System Variable window opens.



4 Enter **KRB5_CONFIG** in the **Variable name** field.

5 Enter the path of *krb5.conf* in the **Variable value** field, such as *c:\qip\named\etc\krb5.conf*.

6 Click the **OK** button to close the three open windows.

END OF STEPS

Create or edit the Kerberos configuration file

The Kerberos configuration file needs to be verified to ensure that the Windows 2003 domain and appropriate KDC are listed, as well as the path of the keytab file.

1 Open a text editor and create (or modify) a Kerberos configuration file.

2 Add (or modify) the following lines:

a. Ensure there is a realm defined for the Windows 2003 Domain under the **[realms]** tag. The format is as follows:

```
[realms]
<REALM> =
{
   kdc = <IP_address_or_name_of_Windows_KDC>:88
   admin_server = <IP_address_or_name_of_Windows_KDC>:749
   default_domain =
   <DNS_zone_of_local_machine_or_other_zone_if_desired>
```

Note: Using the IP address instead of the FQDN for the kdc and admin_server eliminates some potential resolution issues that might cause GSS-TSIG not to work on the server.

[realms] example

The example below uses EXAMPLE.COM Windows Domain for the realm, the Domain Controller's FQDN (windc.example.com) for the kdc and admin server, and the DNS server's domain (example.com) for default_domain:

```
EXAMPLE.COM =
{
    kdc =  windc.example.com
    admin_server = windc.example.com
    default_domain = example.com
}
```

b. Ensure the default realm and encryption types are set under the **[libdefaults]** tag. The format is as follows:

```
[libdefaults]
default_realm = <REALM>
default_keytab_name=<path_of_keytab_file>
default_tkt_enctypes = des-cbc-md5
default_tgs_enctypes = des-cbc-md5
```

Note: Encryption types must be des-cbc-md5 when using a Windows KDC.

[libdefaults] example

The example below uses EXAMPLE.COM Windows Domain for the realm:

```
default_realm = EXAMPLE.COM
default_keytab_name="FILE:c:/qip/named/etc/krb5.keytab"
default_tkt_enctypes = des-cbc-md5
default_tgs_enctypes = des-cbc-md5
```

c. Ensure the domain-to-realm mappings include the DNS server's local domain under the **[domain_realm]** tag. The format is as follows:

```
[domain_realm]
.<DNS_zone> = <REALM>
<DNS_zone> = <REALM>
```

Note: The **[domain_realm]** heading lists DNS zones and the realm to be associated with that zone. Typing a period (.) before the DNS zone name means to use the realm for all subdomains of that domain. A DNS zone with no period means to use the realm for this domain.

[domain_realms] example

The example below uses example.com:

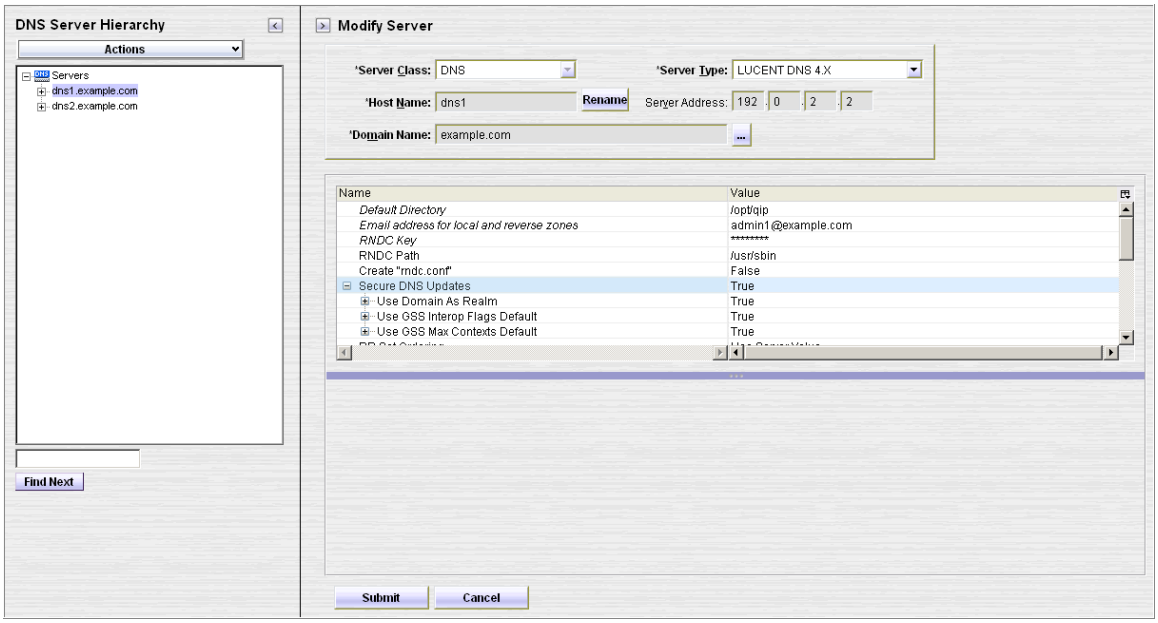
```
example.com = EXAMPLE.COM
```

3 Save the file as **krb5.conf** in the same directory you specified earlier when you set up the environment variable for KRB5_CONFIG (*c:\qip\named\etc\krb5.conf* for example).

END OF STEPS

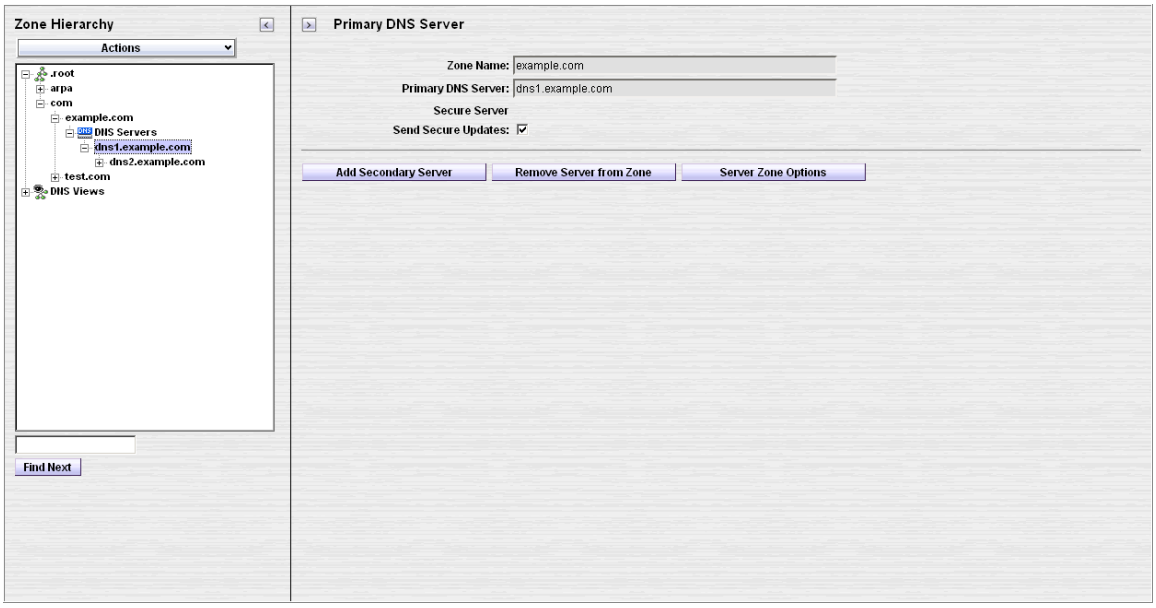
Configure the Lucent DNS 4.2 server

1 In the VitalQIP web client, modify locate the Lucent DNS 4.2 server in the DNS Server Hierarchy and set the **Secure DNS Updates** parameter to **True**.



2 If the realm name is not the same as the domain name, set the **Use Domain as Realm** parameter to False and specify a different realm name with the **Override Realm** parameter. The **GSS Interop Flags** sub-parameter has no affect on Lucent DNS 4.2 and should not be changed.

3 In the VitalQIP web client, locate the same server under ***each*** zone in the Zone Hierarchy. In the Primary DNS Server page, ensure that the **Send Secure Updates** check box is checked. Repeat this step in each zone where the server is the primary.



Important! Any Secondary DNS servers (such as dns2.example.com in the illustration above) that are associated with the Primary DNS servers you wish to accept secure updates should also have the **Send Secure Updates** check box checked.

4 To check that the configuration files generated by VitalQIP are set up correctly, generate the DNS files (using DNS Generation in the VitalQIP web client), and check the **`gss-principal`** setting in the **`options`** block and the **`secure-updates`** setting in the **`zone`** block of each secure zone. A sample *named.conf* using earlier examples is shown below:

```
##########################################
# Lucent 4.x name server configuration file
##########################################

controls {
        inet 127.0.0.1 allow { localhost; } keys { "rndc-key"; };
};

key "rndc-key" {
        algorithm "hmac-md5";
        secret "dGVzdA==";
```

```
};
options {
    directory "C:\qip\named";
    pid-file "named.pid";
    tkey-domain ".";
    qddns
    {
       gss-principal "DNS@dns1.example.com";
    };
};

zone "example.com" in {
        type master;
        file "db.example.com";
        allow-update { any; };
        allow-query { any; };
        allow-transfer { any; };
        notify no;
    qddns
    {
        secure-updates yes;
    };
};

zone "0.0.127.in-addr.arpa" in {
        type master;
        file "db.127.0.0";
};
```

Note that the DNS principal is specified with the directive **`gss-principal`** in the **`qddns`** block. The zone `example.com` is configured as a secure zone by the directive **`secure-updates yes`**. Also note that the Kerberos realm EXAMPLE.COM is not appended at the end of the principal.

Note that the DNS principal is specified with the directive **`gss-principal`** in the **`qddns`** block in the **`options`** block. The zone `example.com` is configured as a secure zone by the directive **`secure-updates yes`** in the **`qddns`** block of the **`zone`** block. Also note that the Kerberos realm EXAMPLE.COM is not appended at the end of the principal; the default realm from *krb5.conf* is used.

5 Also, check the zone file and verify that the MNAME and a matching NS record exist in the zone. The NS glue record must resolve to an address record for native Windows clients.Otherwise, secure updates from native Windows clients will not work. In the sample *db.example.com* shown below, the MNAME (`dns1.example.com`) is highlighted

and matches an NS record with the NS glue record (`dns1.example.com. IN A 10.100.30.50`):

```
$ORIGIN .
$TTL 10800  ; 3 hours
example.com     IN SOA  dns1.example.com. root.example.com. (
                129         ; serial
                10800       ; refresh (3 hours)
                3600        ; retry (1 hour)
                604800      ; expire (1 week)
                3600        ; minimum (1 hour)
                )
            NS  dns1.example.com.
dns1.example.com.   IN  A   10.100.30.50
```

END OF STEPS

Note: An additional configuration step is necessary for a Lucent DNS server that is primary for the Windows Domain zone. Refer to "Setup required if Lucent DNS Server is primary for the Windows Domain zone" (p. 24-51).

Modify the hosts file

Ensure that a reverse lookup for the hostname of the DNS server's IP address either does not resolve or resolves to a host or host/domain that maps to the Kerberos realm defined in *%QIPHOME%\named\etc\krb5.conf*. Typically, the *hosts* file needs to be edited to include the DNS server's IP address. Note that the first value after the IP address in the hosts file must be the FQDN of the DNS Server, and must also match the DNS server Kerberos principal's FQDN.

1. If your DNS principal name is fully qualified when it is entered into the *keytab* file, the first host entry in *C:\WINDOWS\system32\drivers\etc\hosts* 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's domain name, as specified in the *keytab* file).

2. Ensure that the name of the KDC does not resolve in *C:\WINDOWS\system32\drivers\etc\hosts* or resolves to the proper IP address, or that the IP address of the KDC does not resolve in *C:\WINDOWS\system32\drivers\etc\hosts* or resolves to the proper domain.

   A sample *hosts* file is shown below:

   ```
   # hosts file
   ```

```
# dns1.example.com is the FQDN for the DNS server
# which will accept secure dynamic update
127.0.0.1       localhost
10.100.30.50    dns1.example.com dns1
```

END OF STEPS

DNS client setup

Overview

Purpose

This section describes how to set up the VitalQIP DNS Update Service and VitalQIP clients so they send secure dynamic updates to the DNS server.

The DNS client supports GSS-TSIG updates in four different scenarios, as shown in the following table.

GSS-TSIG configuration on Lucent DNS clients

Scenario	Component	Client	Configuration
1	DNS Update Service	UNIX	Refer to "VitalQIP DNS Update Service setup on UNIX" (p. 24-46).
2	DNS Update Service	Windows	Refer to "VitalQIP DNS Update Service setup on Windows" (p. 24-48).
3	VitalQIP	UNIX	Refer to "VitalQIP client setup on UNIX" (p. 24-48).
4	VitalQIP	Windows	Refer to "VitalQIP client setup on Windows" (p. 24-50).

Secure dynamic updates via DNS Update Service

The easiest configuration to manage is to set up ***all*** VitalQIP clients so that they send their updates through the VitalQIP DNS Update Service. This means setting either the **`QIP_USE_DNS_UPDATE_SVC`** environment variable or the **Use DNS Update Service** global policy on all clients to True. The global policy is set in either the VitalQIP thick client UI, the web client, or the **`qip-setpolicy`** CLI.

The DNSUpdateObject and DNSUpdateRR MessageRoutes policies also need to be set up on each client to route dynamic update packets to the DNS Update Service. In this configuration, GSS-TSIG setup is only required for the VitalQIP DNS Update Service that sends the secure dynamic update to DNS. For information on MessageRoute policy configuration, refer to "MessageRoute policy", in ##-##

COMMENT: Cross-reference to be added when merged with Admin Ref Manual.

Secure dynamic DNS updates via GUI or CLI

For VitalQIP clients that are running on UNIX, direct secure dynamic updates of DNS are feasible but each client needs its own keytab file with its corresponding user account created on the Windows KDC. The **`QIP_USE_DNS_UPDATE_SVC`** environment variable needs to be set on all clients, or the **Use DNS Update Service** global policy needs to be set to False. Administratively, this configuration can be burdensome.

If VitalQIP clients are running on Windows, secure dynamic updates of DNS are feasible but the VitalQIP client must be a member of the same Windows 2003 as the Windows 2003 KDC. The **`QIP_USE_DNS_UPDATE_SVC`** environment variable needs to be set on all clients, or the **Use DNS Update Service** global policy needs to be set to False. Each VitalQIP client also needs to be authorized to contact a DNS server directly.

VitalQIP DNS Update Service setup on UNIX

Prerequisites

Ensure that you have met the following requirements.

- Windows 2003 Domain Controller with Microsoft 2003 Server Resource Kit installed. The tool **`ktpass`** is needed to extract the Kerberos principal for the DNS Update Service.

  Note: Secure update with Windows 2008 Domain Controller is not currently supported.
- Domain name of the Windows Domain Controller.
- IP address of the Windows Domain Controller.
- FQDN of the UNIX machine where the DNS Update Service will be running.
- Kerberos tools must be installed on the UNIX machine where the DNS Update Service will be running. The tool **`ktutil`** is needed to update/create the */etc/krb5.keytab* file.

Procedure

The following steps need be performed:

1. Create a user account in the Domain Controller for the DNS Update Service principal. Follow the steps in "Create an Active Directory (AD) user account" (p. 24-27).
2. Extract the DNS Update Service principal to a *keytab* file in the Domain Controller. Follow the steps in "Extract the Kerberos principal" (p. 24-27).
3. Transfer the *keytab* file to the UNIX machine where the DNS Update Service will be running and insert the DNS Update Service principal in */etc/krb5.keytab*. Follow the steps in "Transfer keytab file and create/update /etc/krb5.keytab" (p. 24-27).
4. Set up the Kerberos configuration file (*krb5.conf*) on the UNIX machine. Follow the steps in "Create or edit the Kerberos configuration file" (p. 24-28).
5. Configure the DNS Update Service for GSS-TSIG on the UNIX machine, as described in "Configure the Lucent DNS Update Service" (p. 24-47).
6. *Optional*. Modify the */etc/hosts* file and add the KDC IP address and name as a precaution should the DNS Update Service be unable to resolve the KDC through DNS. Follow the steps in "Modify the hosts file" (p. 24-33).

Configure the Lucent DNS Update Service

1 Open *qip.pcy* in a text editor and locate the `[VitalQIP DNS Update Service]` section, or open *qip-dnsupdated.pcy* if you use that in preference to *qip.pcy.*

2 Enable the following policies. For the **KerberosPrincipal** policy, enter the DNS Update Service principal you recently transferred to the UNIX machine where the DNS Update Service is running.

```
DoSecureUpdates = true
KerberosPrincipal = DNSUpdate/<host>.<domain>@<REALM>
GSSAPIImmediateRetries = 1
GSSAPIRetryDelay = 900
DoKinit = true
KeytabPath = /etc/krb5.keytab
```

Note that the following environment variables can be used to specify secure update settings, and have precedence over policy file settings.

Table 24-5 Secure dynamic DNS environment variables

Policy	Environment variable
KerberosPrincipal	**QIP_PRINCIPAL_NAME**
DoKinit	**QIP_DO_KINIT**
KeytabPath	**QIP_KEYTAB_PATH**

3 Restart the DNS Update Service.

END OF STEPS

VitalQIP DNS Update Service setup on Windows

Prerequisites

Ensure that you have met the following requirements.

- The Windows-based VitalQIP DNS Update Service must be a member of the same Windows 2003 domain as the Windows 2003 KDC. This allows the DNS server to resolve the SRV records in the `_domains` for the KDC realm.
- The Windows-based DNS Update Service must run under the "localsystem" account since the Kerberos principal is derived based on the localsystem account.

Configure the Lucent DNS Update Service

1. Open *qip.pcy* in a text editor and locate the `[VitalQIP DNS Update Service]` section, or open *qip-dnsupdated.pcy* if you use that in preference to *qip.pcy.*

2. Enable the following policy.

```
DoSecureUpdates = true
```

VitalQIP client setup on UNIX

Prerequisites

Ensure that you have met the following requirements.

- Windows 2003 Domain Controller with Microsoft 2003 Server Resource Kit installed. The tool **`ktpass`** is needed to extract the Kerberos principal for the VitalQIP client.

  Note: Secure update with Windows 2008 Domain Controller is not currently supported.
- Domain name of the Windows Domain Controller.
- IP address of the Windows Domain Controller.
- FQDN of the UNIX machine where the VitalQIP client will be running.
- Kerberos tools must be installed on the UNIX machine where the VitalQIP client will be running. The tool **`ktutil`** is needed to update/create the */etc/krb5.keytab* file.

Procedure

The following steps need be performed:

1. Create a user account in the Domain Controller for the VitalQIP client principal. Follow the steps in "Create an Active Directory (AD) user account" (p. 24-27).
2. Extract the VitalQIP client principal to a *keytab* file in the Domain Controller. Follow the steps in "Extract the Kerberos principal" (p. 24-27).
3. Transfer the *keytab* file to ***all*** the UNIX machines where the VitalQIP client will be running and insert the VitalQIP client principal in */etc/krb5.keytab*. Follow the steps in "Transfer keytab file and create/update /etc/krb5.keytab" (p. 24-27).
4. Set up the Kerberos configuration file (*krb5.conf*) on all the UNIX machines. Follow the steps in "Create or edit the Kerberos configuration file" (p. 24-28).
5. Configure the VitalQIP client for GSS-TSIG on each UNIX machine, as described in "Set up global policies and message routes" (p. 24-49).

COMMENT: David: please reset the above link with the ParaText xref format (<HyperColor><$paratext>).

6. *Optional*. Modify each */etc/hosts* file and add the KDC IP address and name as a precaution should the DNS Update Service be unable to resolve the KDC through DNS. Follow the steps documented in "Modify the hosts file" (p. 24-33).
7. Restart the GUI client if already running.

Set up global policies and message routes

3 Choose one of the following actions.

If the client...	Then...
Updates DNS through the DNS Update Service	1. Set the **Use DNS Update Service** Dynamic DNS global policy to **True**. 2. Open the *qip.pcy* file on the VitalQIP DNS Update Service machine in a text editor and locate the **`[VitalQIP DNS Update Service]`** section. 3. Edit the following secure update policies. • Set the **`KerberosPrincipal`** policy to ***`<DNS_client_principal>/<domain>@<REALM>`***. This can also be set on the VitalQIP client with the **`QIP_PRINCIPAL_NAME`** environment variable. • Set the **`DoKinit`** policy to **`True`**. This can also be set on the VitalQIP client with the **`QIP_DO_KINIT`** environment variable. • Set the **`KeyTabPath`** policy to the directory where *krb5.keytab* file is stored (normally */etc/krb5.keytab*). This can also be set on the VitalQIP client with the **`QIP_KEYTAB_PATH`** environment variable. 4. Save the policy file. **Note:** The **`QIPMESSAGESERVICE`** environment variable must also be configured if the Message Service is not installed on the same machine as the VitalQIP client.

If the client...	Then...
Updates DNS via GUI or CLI	Set the following Dynamic DNS global policies in either the VitalQIP thick client, the VitalQIP web client, or the **`qip-setpolicy`** CLI: 1. Set **Use DNS Update Service** to **False**. 2. Under **Secure DNS Updates**, set: • **Principal Name (UNIX only)** to ***`<DNS_client_principal>/<domain>@<REALM>`***. This can also be set with the **`QIP_PRINCIPAL_NAME`** environment variable. • **Perform Kinit (UNIX only)** to **True**. This can also be set on the VitalQIP client with the **`QIP_DO_KINIT`** environment variable. • **Kerberos Keytab Path (UNIX only)** to the directory where *krb5.keytab* is stored (usually */etc/krb5.keytab*). This can also be set with the **`QIP_KEYTAB_PATH`** environment variable. **Note:** The settings for Secure DNS Updates are global, so if you wish to use different keys for each machine, use the environment variables to establish these settings.

VitalQIP client setup on Windows

Prerequisites

Ensure that you have met the following requirements.

- The Windows-based VitalQIP client must be a member of the same Windows 2003 domain as the Windows 2003 KDC. This allows the DNS server to resolve the SRV records in the `_domains` for the KDC realm.
- The Windows-based VitalQIP client must run under the "localsystem" account since the Kerberos principal is derived based on the localsystem account.

Procedure

Configure the VitalQIP client for GSS-TSIG on each Windows machine.

If the client...	Then...
Updates DNS through the DNS Update Service	1. Set the **Use DNS Update Service** Dynamic DNS global policy to **True**. **Note:** The **`QIPMESSAGESERVICE`** environment variable must also be configured if the Message Service is not installed on the same machine as the VitalQIP client.
Updates DNS via GUI or CLI	Set the following Dynamic DNS global policies in either the VitalQIP thick client, the VitalQIP web client, or the **`qip-setpolicy`** CLI: 1. Set **Use DNS Update Service** to **False**.

Setup required if Lucent DNS Server is primary for the Windows Domain zone

Purpose

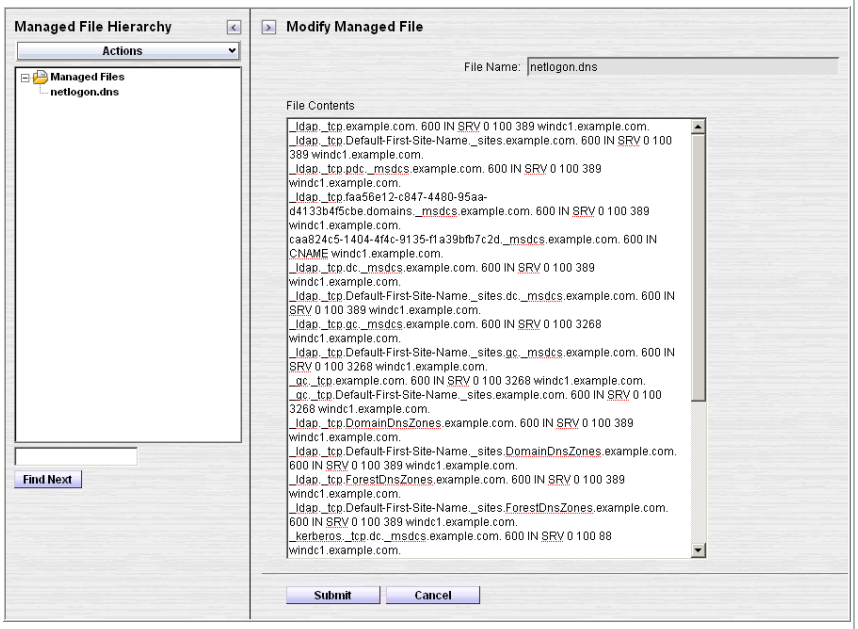
Service (SRV) records are required for VitalQIP components running on Windows and/or Windows clients to update zones securely. This procedure is necessary for secure updates if the Lucent DNS server is primary for the Windows Domain zone (for example, if the Windows domain is EXAMPLE.COM, and the DNS server will be primary for the example.com zone). The Windows clients (VitalQIP and native) need to find the Kerberos (and other) services, based on the "_*" records.

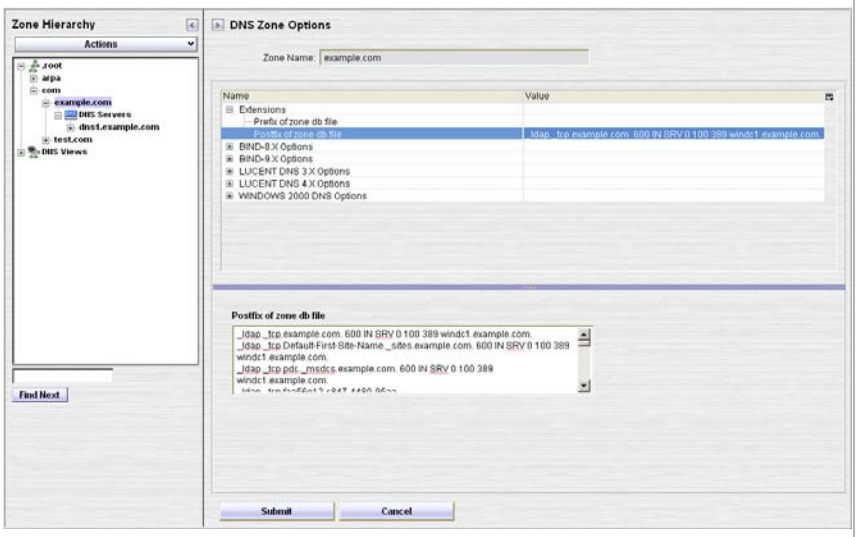
Procedure

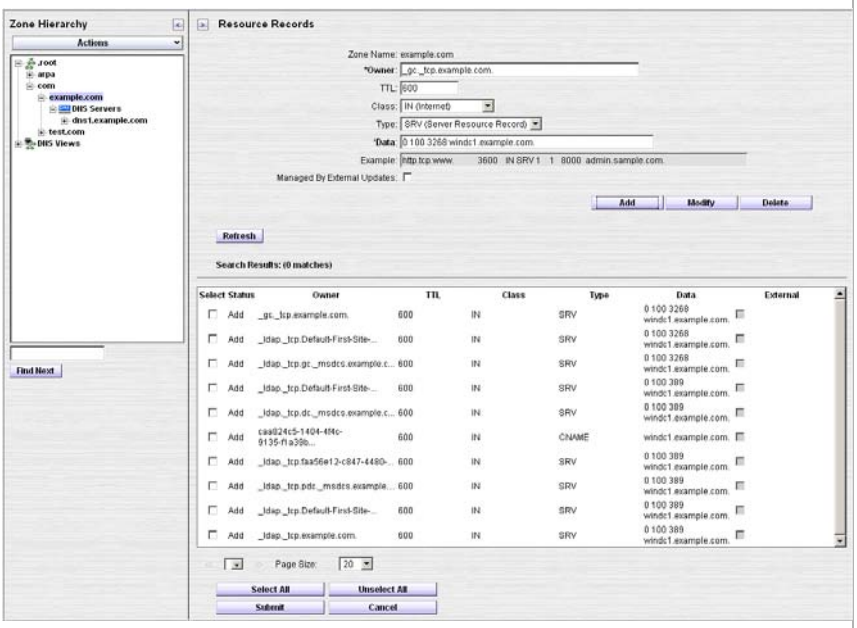
To complete the configuration of secure updates (for Windows clients and VitalQIP Services running on Windows) if the Lucent DNS server is primary for the Windows Domain zone, follow these steps.

1. Transfer the file *c:\WINDOWS\system32\config\netlogon.dns* on the Windows Domain Controller machine to your local machine.

2 Choose one of the following.

If you want to …	Then …
Add *netlogon.dns* to your DNS managed files.	1. Select **Managed Files** from the **DNS** menu in the web client. 2. Select **Add Managed File** from the **Actions** menu. 3. Upload the *netlogon.dns* file you transferred from the Windows DC.  4. Click **Submit**.

If you want to …	Then …
Copy *netlogon.dns* content into the zone extensions **Postfix of zone db file** field.	1. Open *netlogon.dns* in a text editor and copy all its contents. 2. Select **Zones** from the **DNS** menu in the web client. 3. Expand the Zone Hierarchy and select the zone. 4. Click **Zone Options** in the Zone Properties page. 5. Expand **Extensions** and select **Postfix of zone db file**. 6. Paste the content of *netlogon.dns*.  7. Click **Submit**.

If you want to ...	Then ...
Manage the *netlogon.dns* content as individual zone resource records. **Note:** This option allows you to search resource records in the VitalQIP database, as well as manage them externally.	1. Open *netlogon.dns* in a text editor. 2. Select **Zones** from the **DNS** menu in the web client. 3. Expand the Zone Hierarchy and select the zone. 4. Click **Resource Records** in the Zone Properties page. 5. Copy the Owner string from the first line of *netlogon.dns* and paste it into the **Owner** field in the Resource Records page. 6. Enter **`600`** as the **TTL**, select **`IN`** as the **Class** and a **Type** (SRV, CNAME, or A). 7. Copy the record data (RDATA) to the **Data** field. 8. Click **Add**. 9. Repeat steps 5 to 8 for each entry in *netlogon.dns*.  10. Click **Select All**. 11. Click **Submit**.

3 Select **DNS Generation** from the **DNS** menu and push the configuration files to the DNS server. A sample preview file of the zone `example.com` is shown below.

```
$TTL 86400
;============================================================================
; Addresses and other host information for zone: example.com
;============================================================================
@       IN      SOA     dns1.example.com. admin1.example.com. (
                        0       ; Serial No.
                        21600   ; Refresh
                        3600    ; Retry
                        604800  ; Expire
                        600 )   ; Negative Cache TTL
;
                        IN      NS      dns1.example.com.
                        IN      NS      dns2.example.com.
;*************
; A records
;*************
dns2            IN              A       192.0.2.6
rtp000002rts            IN              A       192.0.2.5
dns1            IN              A       192.0.2.2
rtp000001rts            IN              A       192.0.2.1
;**************************************
; Other resource records
;**************************************
DomainDnsZones.example.com.             600     IN      A       10.54.0.42
ForestDnsZones.example.com.             600     IN      A       10.54.0.42
gc._msdcs.example.com.          600     IN      A       10.54.0.42
example.com.            600     IN      A       10.54.0.42
_kpasswd._udp.example.com.              600     IN      SRV     0 100 464 windcl.example.com.
_kpasswd._tcp.example.com.              600     IN      SRV     0 100 464 windcl.example.com.
_kerberos._udp.example.com.             600     IN      SRV     0 100 88 windcl.example.com.
_kerberos._tcp.Default-First-Site-Name._sites.example.com.              600     IN      SRV     0 100 88 windcl.example.com.
_kerberos._tcp.example.com.             600     IN      SRV     0 100 88 windcl.example.com.
_kerberos._tcp.Default-First-Site-Name._sites.dc._msdcs.example.com.            600     IN      SRV     0 100 88 windcl.example.com.
_kerberos._tcp.dc._msdcs.example.com.           600     IN      SRV     0 100 88 windcl.example.com.
_ldap._tcp.Default-First-Site-Name._sites.ForestDnsZones.example.com.           600     IN      SRV     0 100 389 windcl.example.com.
_ldap._tcp.ForestDnsZones.example.com.          600     IN      SRV     0 100 389 windcl.example.com.
_ldap._tcp.Default-First-Site-Name._sites.DomainDnsZones.example.com.           600     IN      SRV     0 100 389 windcl.example.com.
_ldap._tcp.DomainDnsZones.example.com.          600     IN      SRV     0 100 389 windcl.example.com.
_gc._tcp.Default-First-Site-Name._sites.example.com.            600     IN      SRV     0 100 3268 windcl.example.com.
_gc._tcp.example.com.           600     IN      SRV     0 100 3268 windcl.example.com.
_ldap._tcp.Default-First-Site-Name._sites.gc._msdcs.example.com.                600     IN      SRV     0 100 3268 windcl.example.com.
_ldap._tcp.gc._msdcs.example.com.               600     IN      SRV     0 100 3268 windcl.example.com.
_ldap._tcp.Default-First-Site-Name._sites.dc._msdcs.example.com.                600     IN      SRV     0 100 389 windcl.example.com.
_ldap._tcp.dc._msdcs.example.com.               600     IN      SRV     0 100 389 windcl.example.com.
caa824c5-1404-4f4c-9135-f1a39bfb7c2d._msdcs.example.com.                600     IN      CNAME   windcl.example.com.
_ldap._tcp.faa56e12-c847-4480-95aa-d4l33b4f5cbe.domains._msdcs.example.com.             600     IN      SRV     0 100 389 windcl.example.com.
_ldap._tcp.pdc._msdcs.example.com.              600     IN      SRV     0 100 389 windcl.example.com.
_ldap._tcp.Default-First-Site-Name._sites.example.com.          600     IN      SRV     0 100 389 windcl.example.com.
_ldap._tcp.example.com.         600     IN      SRV     0 100 389 windcl.example.com.
_gc._tcp.example.com.           600     IN      SRV     0 100 3268 windcl.example.com.
_ldap._tcp.Default-First-Site-Name._sites.gc._msdcs.example.com.                600     IN      SRV     0 100 3268 windcl.example.com.
_ldap._tcp.gc._msdcs.example.com.               600     IN      SRV     0 100 3268 windcl.example.com.
_ldap._tcp.Default-First-Site-Name._sites.dc._msdcs.example.com.                600     IN      SRV     0 100 389 windcl.example.com.
_ldap._tcp.dc._msdcs.example.com.               600     IN      SRV     0 100 389 windcl.example.com.
caa824c5-1404-4f4c-9135-f1a39bfb7c2d._msdcs.example.com. 600 IN CNAME windcl.example.com.               600     IN      CNAME   windcl.example.com.
_ldap._tcp.faa56e12-c847-4480-95aa-d4l33b4f5cbe.domains._msdcs.example.com.             600     IN      SRV     0 100 389 windcl.example.com.
_ldap._tcp.pdc._msdcs.example.com.              600     IN      SRV     0 100 389 windcl.example.com.
_ldap._tcp.Default-First-Site-Name._sites.example.com.          600     IN      SRV     0 100 389 windcl.example.com.
_ldap._tcp.example.com.         600     IN      SRV     0 100 389 windcl.example.com.
;*************************
```

END OF STEPS

Configure DNSSEC Support

About DNSSEC

DNSSEC provides data origin authentication, data integrity, and authenticated denial of existence for DNS data. It does this by securing data with public and private keys and establishing chain of trust.

This section describes DNSSEC configuration. DNSSEC also impacts VitalQIP CLIs and the User Interface.

Important! DNSSEC and DNS Views have the following interoperability restrictions:

- A DNSSEC-enabled DNS Server cannot also have DNS Views enabled.
- A DNS Views-enabled DNS Server cannot also have DNSSEC enabled.
- A zone that is DNSSEC-enabled cannot also be part of a DNS View.
- A child zone of a DNSSEC-enabled zone cannot also be part of a DNS View.

DNSSEC background

DNSSEC (DNS Security Extensions) adds security to the Domain Name System (DNS) used on IP networks. It is a set of extensions to DNS, which provide:

- Origin authentication of DNS data
- Data integrity
- Authenticated denial of existence

DNSSEC was designed to protect the Internet from certain attacks, such as DNS cache poisoning. All answers in DNSSEC are digitally signed. By checking the signature, a DNS resolver is able to check if the information is identical (correct and complete) to the information on the authoritative DNS server.

VitalQIP support of DNSSEC

The DNSSEC feature in Vital QIP provides:

- Database support for storing and retrieving DNSSEC key data
- Userexit script to secure zones with DNSSEC keys (by signing of zones with keys) and establish chain of trust for VitalQIP managed domains
- SOAP interface enhancement to add, delete or retrieve DNSSEC keys for zones. See the *VitalQIP Web Service Interface Specification* for details.
- CLI commands. See the *Vital QIP Command Line Interface User's Guide* for details.

DNSSEC userexit

Description

VitalQIP has a userexit called *dnssec-userexit.pl*, located in the *QIPHOME/userexits* directory, that supports DNSSEC. The userexit is invoked during a DNS push. It retrieves keys for each zone in the push and signs the zone. The DS keys generated when a zone is signed are used to sign the parent zone, and establishes a chain of trust.

Note: DNS push uses the user ID setup in the *qip.pcy* file. If the user specified in the *qip.pcy* file is not a Master or an Organization Administrator, the **dnssec-userexit** fails.

Synopsis

QIPHOME/userexits/dnssec-userexit.pl [*server_name*] [*server_IP*] [*current_push_directory*] [*remote_push_directory*] [SERVER|LOCAL] [*organization name*]

Parameters

The following parameters are passed to the userexits:

Parameter	Description
server_name	The name of the server where the user exit is being pushed.
server_IP	The IP address of the server to which the user exit is being pushed.
current_push_directory	The directory where files are currently stored. This will only be different from the ***remote_push_directory*** if this user exit is being run on the File Generation Service. **Note:** The directory may be set to NONE if the directory is not available.
remote_push_directory	The directory on the Remote Service where files will be placed. **Note:** The directory may be set to NONE if the directory is not available.
Server or Local	Specifies the type of push.
organization_name	The name of the organization.

Configure DNSSEC

Configure DNSSEC by performing the following steps.

1. Set up database parameters by running*$QIPHOME/scripts/dnssec_data_71.sql*. This adds the additional parameters
   - DNSSEC enabled zone
   - DNSSEC enabled server

2. Generate the xml needed for enabling DNSSEC from the web client by executing the **`qip-getparamterlist`** CLI for parameters #20, 35 and 57

```
qip-getparamlist -n 20 -f qiptype_20.xml -s <DB_SERVER_NAME>
   -u <QIP_USER> -p <QIP_USER_PASSWORD>
qip-getparamlist -n 35 -f qiptype_35.xml -s <DB_SERVER_NAME>
   -u <QIP_USER> -p <QIP_USER_PASSWORD>
qip-getparamlist -n 57 -f qiptype_57.xml -s <DB_SERVER_NAME>
   -u <QIP_USER> -p <QIP_USER_PASSWORD>
```

   The three parameters are:
   - 20 - Lucent DNS 4.X
   - 35 - Bind 9.x
   - 57 - Zone

3. Copy the three xml parameter files to *$QIPHOME/web/xml*.

4. Enable the user exit by editing the *$QIPHOME/userexits/UserExit.pcy* file,and placing the *userexit* file name in stage 3 of the FGS push sections. This user exit must be called after the zone files are generated successfully. So it must be executed at Stage 3 of the user exit policy file.

```
; DNS Update Push (Remote)
;DNSUP0=qipdnserror
options
{
  dnssec-enable yes;
};
DNSUP1=qipprednsuserexit
DNSUP2=qipdnsuserexit2
DNSUP3=qipdnsuserexit
```

```
DNSUP4=qipdnsuserexit4
; DNS Update Push (FGS)
DNSUP1FGS=qipprednsuserexitfgs
DNSUP3FGS=dnssec-userexit.pl
;
; DNS Config Push (Remote)
DNSCFG1=qipprednscnfuserexit
DNSCFG2=qipdnscnfuserexit2
DNSCFG3=qipdnscnfuserexit
DNSCFG4=qipdnscnfuserexit4
; DNS Config Push (FGS)
DNSCFG1FGS=qipprednscnfuserexitfgs
DNSCFG3FGS=dnssec-userexit.pl
```

5 Enable DNSSEC for zones. Read the following table to determine the appropriate method to enable DNSSEC for zones.

If you are enabling…	Then
DNSSEC for zones via the **enterdomain** command	1. Create an input file with the **DNSSEC enabled zone** field set to **True** for each zone that is to be DNSSEC enabled. See New enterdomain input file fieldî (p. 1-8) for a sample input file. 2. From a command line prompt, go to the *$QIPHOME/usr/bin* directory. 3. Run the enterdomain command. For example: **enterdomain -i inputfile -df c -w** **Note:** The *VitalQIP Command Line Interface Userís Guide* contains more information on using the **enterdomain** command and input file formats.
DNSSEC for zone via the VitalQIP client	1. From the **Infrastructure** menu, select **Domain**. 2. In the Domain Profile Option screen, select a domain and click **OK**. 3. Click the **Zone Options** tab. 4. Expand the **BIND-9.X Options**, or **Lucent DNS 4.x Options** node. 5. Set **DNSSEC enabled** to **True** for the zones needed to be signed with the keys generated. 6. Click **OK**.

6 Enable DNSSEC for a DNS server. Read the following table to determine the appropriate method to enable DNSSEC for a DNS server.

If you are enabling…	Then
DNSSEC for DNS servers via the **enterserver** command	1. Create an input file with the **DNSSEC enabled server** field set to **True** for each zone that is to be DNSSEC enabled. SeeìNew enterserver input file fieldî (p. 1-9) for a sample input file. 2. From a command line prompt, go to the *$QIPHOME/usr/bin* directory. 3. Run the **enterserver** command. For example: **enterserver -i inputfile** **Note:** The *VitalQIP Command Line Interface Userís Guide* contains more information on using the **enterserver** command and input file formats.
DNSSEC for DNS servers via the VitalQIP client	1. From the **Infrastructure** menu, select **Server**. 2. In the Server Profile Option screen, select the server you wish to modify and click **OK**. 3. In the Server Profile screen, set **DNSSEC enabled server** to **True** for the DNS Server containing the zones that need to use DNSSEC. 4. Click **OK**.

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 2003, 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 can be configured to send 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 2003 administrators can modify a partially managed object. Windows 2003 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.

> **Note:** See your sale representative for more information on purchasing Audit Manager. Refer to the *Audit Manager User's Guide* for information on 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. Refer to 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.

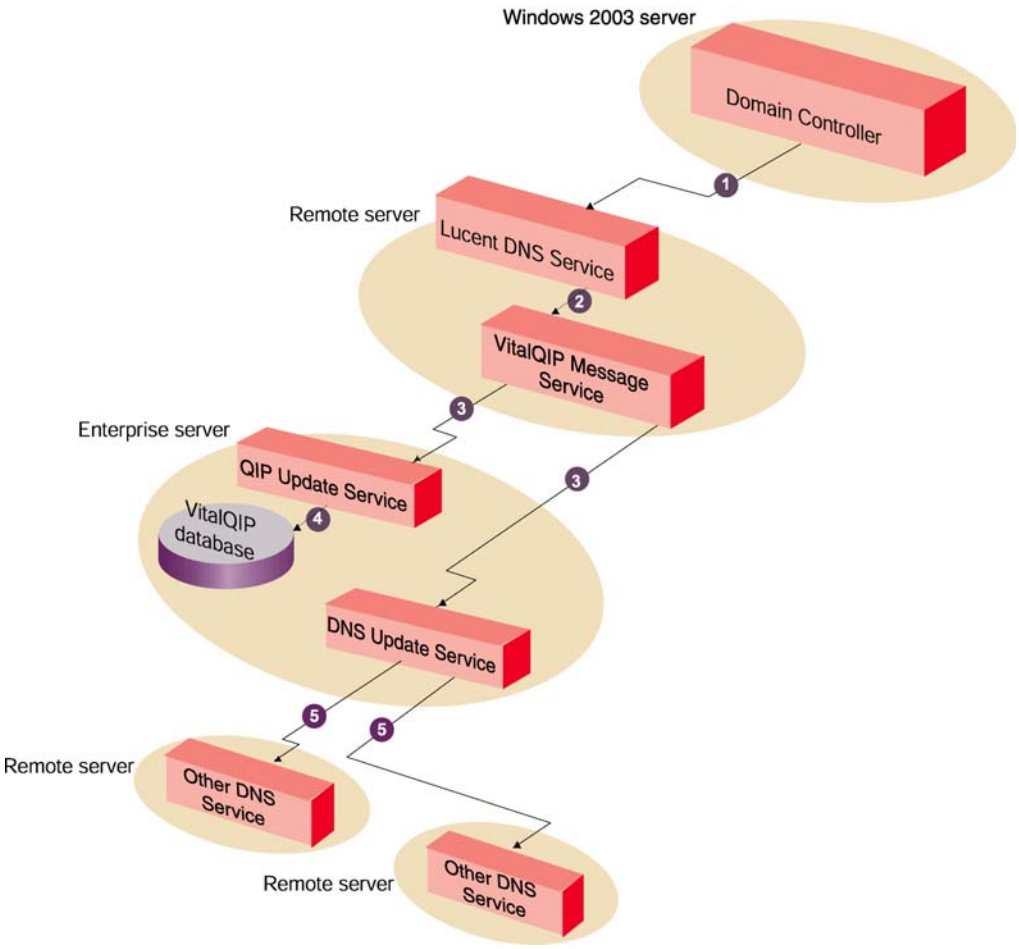
Continuous model

In the continuous model, VitalQIP is continuously being updated with external objects and resource records in “real-time”. 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 for adds and single resource records or resource record sets for deletes.

Note: Currently, only the Lucent DNS Service (version 3.1 or above) supports the continuous model.

The following illustration shows how the continuous model works.

Figure 24-1 Continuous Model



The following steps occur in Continuous Model:

1. A DNS client, such as a domain controller on a Windows 2003 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.

   Note: 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 this originating IP address to ensure it does not forward the update back to the originating DNS Service.

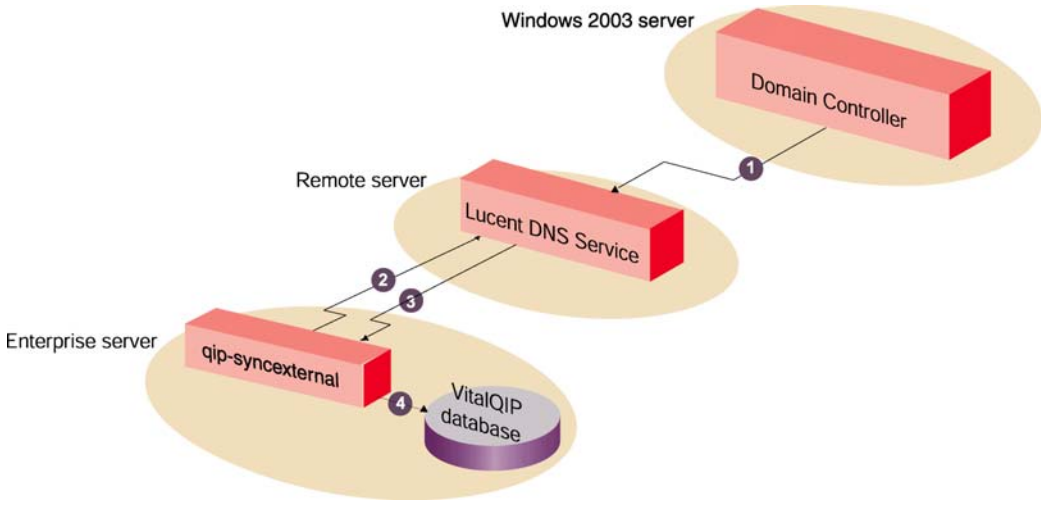
Polling Model

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.

Note: 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.

The following illustration shows how the polling model works.

Figure 24-2 Polling Model



The following steps occur in the Polling Model:

1. A DNS client, such as a domain controller on a Windows 2003 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 2003 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.

Note: 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 (on the *Resource Record* tab).
- When processing a TXT, AAAA, CNAME, or SRV resource record, VitalQIP adds an external resource record to the appropriate domain or reverse zone (on the **Resource Record** tab).
- When processing a dynamic add for a "tombstoned" external object or resource record, VitalQIP "un-tombstones" the object or resource record.

Duplicate resource record validation checking

VitalQIP checks for duplicate external resource records in the same organization. If a duplicate is found, an error message is displayed, and the external resource record cannot be added or modified. Validation checking is enabled by default via the **Validate CNAME Records** policy in the VitalQIP client. This feature can be disabled by setting the policy to False. Refer to the *VitalQIP User's Guide* for more information.

The type of external resource record validation checking is determined by the type of record being entered or modified and where it is being entered from. The following table shows what resource records are checked against.

Table 24-6 Type of Duplicate resource record validation checking

Record type	Enter from	Checked against
All resource records excluding CNAME records	• **Resource Record** tab of the Object Profile • **Resource Record** tab of the Domain Profile • **Resource Record** tab of the Reverse Zone Profile	VitalQIP checks for duplicate CNAME resource record owner data on: • Objects • Domains • Reverse zones • Fully qualified alias names
Object record	Object Profile	VitalQIP checks for duplicate CNAME resource record owner data on: • Objects • Domains • Reverse zones • Fully qualified alias names
CNAME	• **Resource Record** tab of the Object Profile • **Resource Record** tab of the Domain Profile • **Resource Record** tab of the Reverse Zone Profile	VitalQIP checks for duplicate: • Fully qualified object names • Domain names • Object level resource records • Reverse zone resource records • Fully qualified object alias records • Fully qualified MX records • ENUM Manager records • IPv6 name records

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, and arrive out of order.

For example, consider a Windows 2003 server that reboots frequently. As Windows 2003 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 2003 server when it should or will represent the Windows 2003 server when it should not.

Tombstoned objects or records contain a time stamp, which resolves conflicts such as the one described above. The Lucent DNS Service puts the current time in the DNSUpdateRR messages in the continuous model. 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 (on the **Resource Record** tab). 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.

Refer to 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, refer to 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, refer to 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 than VitalQIP, 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

Purpose

Three classes of VitalQIP infrastructure affect the creation of external objects, Lucent DNS Services, forward zones, and reverse zones.

Before you begin

The following is recommended 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 contains 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, refer to the *VitalQIP Installation Guide*.
- In Windows 2003 environments, only domain controllers and VitalQIP sources should have update permission (through ACLs) for the Lucent DNS server. It is not recommended to allow end users' Windows 2003 clients to update DNS (which could update VitalQIP when using External Update Propagation). Updates from end users' Windows 2003 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 on the ***remote server*** to send DNS updates to the QIP Update Service:

```
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" (p. 24-68) 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.

END OF STEPS

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, refer to 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.

END OF STEPS

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

Improve DNS push functionality

Improve the functionality of VitalQIP DNS Update push by enabling Remote Service policies that implement a freeze-thaw technique for updating zones on a Lucent DNS 4.0/BIND 9.x server. This functionality requires Lucent DNS 4.0 build 17 or later, or ISC BIND 9.3.0 or later. The improvement prevents the loss of updates received by the DNS Server during the push process.

What happens with improved DNS push functionality

Once VitalQIP is configured to perform DNS pushes more efficiently, the following steps occur:

1. If the DNS Server is not running, the old push logic will occur.
2. A backup copy of the *named.conf* is made and the new *named.conf* is copied from the temporary directory into the DNS working directory (push directory).
3. The *db.cache* and *db.0.0.127.in-addr.arpa* zone files are copied into the DNS working directory.
4. Any zones in the temporary push directory (pushed files) that are not in the current DNS working directory are considered new, and copied into the working directory.
5. The current *rndc.conf* file is used to issue the **`rndc reconfig`** command to the DNS Server. This causes the server to:
   a. Read the new *named.conf* file.
   b. Reload the *db.cache* file.
   c. Remove any deleted zones (zones in old *named.conf* file, but not in the new *named.conf* file).
   d. Load any new zones (zones in new *named.conf*, but not in the old *named.conf*).
6. Each of the zones for which a zone file remains in the temporary directory will have the following actions performed:
   a. Check for a journal file and process the updates within the journal file. Processing consists of queuing updates for any serial number greater than the serial number in the new zone file minus 2.
   b. Prevent dynamic updates to the zone by issuing the **`rndc freeze <zone_name>`** command to the DNS Server.
   c. Check that the new zone file serial number is greater than the current zone serial number in the current DNS zone file. If the serial number is less than or equal to the current one, replace the zone serial number in the new zone file with the current serial number plus 1.
   d. Copy the new zone file from the temporary directory into the DNS working directory.

e. Load the new zone and enable dynamic updates to the new zone by issuing the **`rndc thaw <zone_name>`** command to the DNS Server.

7. Send the queued updates (from Step 6a) to the DNS Update Service.

To improve DNS push functionality

Purpose

Use this procedure to improve DNS push functionality.

Before you begin

Ensure the following is set up on the remote DNS server:

- Both **`rndc`** and **`journalprint`** binaries must be executable and found in *<install_directory>\bin* on Windows or *$QIPHOME/usr/bin* on UNIX/Linux.
- A **`MessageRoute`** to the DNS Update Service with the message type of DNSUpdateRR must exist in the *qip.pcy* file on the Remote DNS Server. For example:

```
MessageRoute=DNSUpdateRR:A:0:DNS Update Service (Update
RR):VitalQIP DNS Update Service:10.20.30.1
```

- For each dynamic zone, allow dynamic updates from the necessary DNS Update Service's IP address. This allows any potentially lost updates to be added back into the zone.
- Leave the **max-journal-size** options unset (4MB default as of Lucent DNS 4.0 B17), or set the value to ensure access to the updates that have occurred between the time the push began and the time the zone is processed.

Procedure

To improve push functionality, follow this procedure:

1. Open the *qip.pcy* file (or *qip-rmtd.pcy* file if applicable) and locate the `[VitalQIP Remote Service]` section.

2. Set the **UseFreezeThawUpdate** policy to True. For information on this policy, refer to "UseFreezeThawUpdate" (p. 12-7).

3. Modify the **MaxDNSUpdateThread** policy if necessary. The default value of 5 should be adequate. For information on this policy, refer to "MaxDNSUpdateThreads" (p. 12-4).

4. If the remote DNS server does not listen for updates on the IP used for the push, enter the appropriate DNS server listening IP address in the **DNSServerAddress** policy. For information on this policy, refer to "DNSServerAddress" (p. 12-3).

5 If your configuration is different from the usual default location (<*push_directory*>*/bin*) for rndc and journalprint binaries, enter the path for these binaries in the **DNSBinsDir** policy. For information on this policy, refer to "DNSBinsDir" (p. 12-3).

6 Save the *.pcy* file.

7 Perform a push to changed zones only.

VitalQIP client

Perform a **DNS Generation to Changed Zones Only** from the **Network Service | DNS Generation** menu.

VitalQIP web client

Select **DNS Generation** from the **DNS** menu (or click  in the toolbar), select the server to push to in the hierarchy, followed by the **Changed Zones Only** target zone option.

END OF STEPS

Customize user exit scripts

VitalQIP allows user exits to be renamed in a file called *UserExits.pcy*. The *UserExits.pcy* file contains a set of default user exits that are processed when files are generated. You can also configure user exits to return failure conditions if a push fails.

Stages user exits occur

Process

On the remote server, VitalQIP runs user exits at separate stages. These stages correspond to the distinct stages of a push operation on a remote server. The corresponding user exit is run at the end of each stage. The stages are as follows:

Table 24-7 User exit stages

Stage	Description
0	An error occurred. The push fails.
1	The destination directory is created if necessary, but no communication with the File Generation Service occurs. By default, only DNS-type pushes run in this stage on a remote server.
2	Files are copied from the File Generation Service to the *temp* directory.
3	Files are copied to the destination directory. If the **UseFreezeThawUpdate** policy is set to True in the *qip.pcy* file, Freeze/Thaw DNS updates occur rather than regular DNS updates. Refer to "UseFreezeThawUpdate" (p. 12-7) for more information on this policy.
4	The server is restarted.

The File Generation Service runs user exits after each of these stages. The stages are as follows:

Table 24-8 File Generation Service user exit stages

Stage	Description
0	An error occurred during file generation, push fails.
1	File generation commences. Files are not created yet.
2	Not defined.
3	Occurs before the dynamic DNS *conf* file is generated.
4	File generation completes. The locked database is not yet released.

Policy names

The syntax of a user exit policy is: {prefix}{stage}[FGS]

Prefix — The following prefixes can be used in a policy:

- DNSUP - used for DNS update pushes
- DNSCFG - used for DNS data and configuration pushes
- DHCP - use for DHCP pushes
- BOOTP - used for BOOTP pushes
- NIS - used for NIS pushes
- LOCAL - used for local pushes
- REMOTE - used for remote server pushes

Stage — A numerical value between 0 - 4 that represents the stage at which the user exit is run, as described in Table 24-7.

FGS — Used at the end of a policy if it is a user exit used by the File Generation Service.

Sample user exit policy name for a DNS Update Push on a remote:

```
DNSUP1
```

Sample user exit policy name for a DNS Update Push using FGS:

```
DNSUP1FGS
```

Rename user exit scripts

Purpose

Use this procedure to rename user exit scripts.

Before you begin

Windows only. If a user exit name does not contain a file extension, the extensions *.exe*, *.cmd* and *.bat* are tried in that order. If an extension is specified, only that script is attempted.

If you want to run a user exit that ends in *.pl*, you must call it from a *.bat* file.

UNIX only. The exact filename is run as specified.

Procedure

To rename a user exit, do the following:

1. Back up and rename the user exit script:
   a. Go to the *QIPHOME/userexits* directory.
   b. Back up the user exit script that you want to rename. This script serves as a backup in case you need to use the default script.
   c. In the *QIPHOME/userexits* directory, rename the user exit script.
2. Copy the *UserExit.pcy* file from the *QIPHOME/examples* directory to the *QIPHOME/conf* directory. If this file is found in the *QIPHOME/con*f, default user exits are not used.
3. Open the *UserExit.pcy* file.
4. To prevent a particular user exit from being executed, comment the user exit definition definition. For example, the line **`DNSUP1=qipdnserror`** would like:

   ```
   ; DNSUP1=qipdnserror
   ```
5. To change the name of the user exit script, replace the name of the user exit script for the appropriate policy. For example, the policy **`DNSUP1=qipdnserror`** would change to:

   `DNSUP1=dnserror`

6 Save the file.

END OF STEPS

Report failure conditions

Purpose

User exits can be can be configured to report failure conditions.

Procedure

To configure user exits to report failure conditions, follow these steps:

1. Open the *qip.pcy* file in the *QIPHOME* directory.

2. In the **`[VitalQIP Remote Service]`** section, set the **FailOnFailedUserExit** policy to True. If a push fails, a non-zero value is used as a the return value. Refer to "FailOnFailedUserExit" (p. 12-4) for more information on this policy.

3. Save the *qip.pcy* file.

4. Modify the user exit script and add the desired error message to **`$remotePushDir/UserExit.error`** on the remote server or **`$currentPushDir/UserExit.error`** on the File Generation Server.

   The following is a sample error message code:

```
if [ ${Status} -ne 0 ] ; then
    case ${Status} in
    1) msg="Joe forgot to clear the logs." ;;
    2) msg="of NFS issues." ;;
    3) msg="the server is on fire." ;;
    *) msg="of an unknown error." ;;
    esac
    echo "The push failed because ${msg}" >
  ${remotePushDir}/UserExit.error
fi
```

END OF STEPS

Sample UserExit.pcy file

The following is a sample *UserExit.pcy* file that contains default VitalQIP user exits.

```
;-------------------------------------------------------------------------
; User-exit names are now configurable.  Defaults are displayed in this file.
; On the Remote Server, User-Exits are run at 5 separate times.  These times
; correspond to 5 distinct stages of a push operation on the Remote Server.
; stage 0: An error occurred. Push will fail.
; stage 1: the destination directory has been created (if necessary), but no
;          communication with the FGS has occurred.  By default, on the
;          remote, only DNS type pushes run stage 1.
; stage 2: files have been copied from the FGS to the temp directory
; stage 3: files have been copied to the destination directory.  Freeze/Thaw
;          DNS updates have occurred.
; stage 4: the server has been restarted
; The stages on the FGS:
; stage 0: An error occurred during file generation, push will fail.
; stage 1: File generation commencing; no files have yet been created
; stage 2: not defined
; stage 3: Before DDNS conf file generation.
; stage 4: File generation complete; DB lock not yet released.
;-------------------------------------------------------------------------
; Policy names are composed of {prefix}{stage}[FGS]
; prefixes: DNSUP/DNSCFG/DHCP/BOOTP/NIS/LOCAL/REMOTE
; stages: 0-4
; FGS is used for FGS based user-exits
;-------------------------------------------------------------------------
; This file must be copied to $QIPHOME/conf to be used, otherwise
; defaults will be used.  If this file exists in $QIPHOME/conf, defaults are
; NOT USED.  To prevent a particular user-exit from being executed, comment
; the user-exit definition.
;-------------------------------------------------------------------------
; Windows Users: If a user-exit name does not contain a file extension; the
;                extensions .exe, .cmd, .bat, .pl are tried in that order. If
;                an extension is specified, only that command is attempted.
;                Any extension can be specified.
;-------------------------------------------------------------------------
;
; DNS Update Push (Remote)
;DNSUP0=qipdnserror
DNSUP1=qipprednsuserexit
DNSUP2=qipdnsuserexit2
DNSUP3=qipdnsuserexit
DNSUP4=qipdnsuserexit4
; DNS Update Push (FGS)
DNSUP1FGS=qipprednsuserexitfgs
DNSUP3FGS=qipdnsuserexitfgs
;
```

```
; DNS Config Push (Remote)
DNSCFG1=qipprednscnfuserexit
DNSCFG2=qipdnscnfuserexit2
DNSCFG3=qipdnscnfuserexit
DNSCFG4=qipdnscnfuserexit4
; DNS Config Push (FGS)
DNSCFG1FGS=qipprednscnfuserexitfgs
DNSCFG3FGS=qipdnscnfuserexitfgs
;
; DHCP push (Remote)
;DHCP1=qipdhcpuserexit
DHCP2=qipdhcpuserexit2
DHCP3=qipdhcpuserexit
DHCP4=qipdhcpuserexit4
; DHCP push (FGS)
DHCP3FGS=qipdhcpuserexitfgs
DHCP4FGS=qipdhcpuserexitfgs
;
; BOOTP push (Remote)
;BOOTP1=qipbootpuserexit
BOOTP2=qipbootpuserexit2
BOOTP3=qipbootpuserexit
BOOTP4=qipbootpuserexit4
; BOOTP push (FGS)
BOOTP4FGS=qipbootpuserexitfgs
;
; NIS push (Remote)
;NIS1=qipnisuserexit
NIS2=qipnisuserexit2
NIS3=qipnisuserexit
NIS4=qipnisuserexit4
; NIS push (FGS)
NIS3FGS=qipnisuserexitfgs
;
; Local hosts files push (do not confuse with a LOCAL push)
;LOCAL=qiplocaluserexit
LOCAL2=qiplocaluserexit2
LOCAL3=qiplocaluserexit
LOCAL4=qiplocaluserexit4
; Local hosts push (FGS)
LOCAL3FGS=qiplocaluserexitfgs
;
; Remote push
;REMOTE1=qipremoteuserexit
REMOTE2=qipremoteuserexit2
```

```
REMOTE3=qipremoteuserexit
REMOTE4=qipremoteuserexit4
```

25 Troubleshoot 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 "General VitalQIP troubleshooting" (p. 20-2) and "Environment verification" (p. 20-4) for more information.

Contents

This chapter presents the following topics.

Common DNS problems and questions	25-2
DNS error messages	25-10
Fixing Bad Zones	25-12
Checking named.conf files	25-32

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* (Lucent DNS 3.1/4.0 and BIND 8.x/9.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 example, run the following to establish a link in UNIX:

```
ln -s /etc/named.conf /<QIPHOME>/<DNS_directory>.named.conf
```

- For Windows 2003, the *named.conf* resides in *%systemroot%\System32\drivers\etc*, and for BIND 9.x and Lucent DNS 4.x resides in *<install_dir>\etc*.
- Verify that the correct version of *named.exe* file is installed on the remote as is defined in VitalQIP. Use named **-v** command to determine the version that is installed.
- Enable debugging to determine the problem's cause. See the next question, "How do I enable debugging for DNS (named)?".
- You can also rerun the BIND 8 or BIND 9 installation (not provided by Alcatel-Lucent).

  Note: In Windows, certain registry keys must be populated before **named** is started. To do so, install **named** (Lucent DNS 3.x and 4.x only) by running:

  ```
  named -install
  ```

How do I enable debugging for DNS (named)?

To enable DNS (**named**) debugging of Lucent DNS 3.x or 4.x on Windows 2003, use the following method:

1. Open the VitalQIP Service Controller by selecting **Start | Programs | 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:

- Start **named** in debug by running:

  `named -d <debug_level>`

  For example, `named -d 10`. Level 11 is the highest debug level.

  Note: The *named.run* file is located in the configured DNS directory.

- You can change the debug level for a running DNS server in real-time. For BIND 8.x/LDNS 3.x, run **ndc**. For BIND 9.x/LDNS 4..x, run **rndc**.

  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.

- For Lucent DNS 3.x or BIND 8.x only, 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. Refer to "Signal handling" (p. 20-13) for more information about signal handling.

  Note: The **kill -USR1** signal terminates the Lucent DNS 4.x or BIND 9.x DNS service.

For additional information on troubleshooting DNS, refer to *DNS and BIND* 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 3.x or BIND 8.x, 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 on the Server Profile in the **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 2003, make sure the *%systemroot%/resolv.ini* file and the TCP/IP stack is configured properly.
- If you are running in a Windows 2003 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, refer to *DNS and BIND* by Paul Albitz and Cricket Liu.

To configure a forwarder for BIND/9.x and Lucent DNS servers, add a forwarder statement to the *named.conf* (this can be added to the Server Profile **Corporate Extension** field) 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 **`forward only;`** statement to the forwarders' directives similar to the following. (The forward only prevents the server from unnecessarily attempting internet resolution if this is not possible due to network/firewall restrictions.)

```
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 to a secondary.
- The secondary server is not configured as a slave for the expected zone.
- There are syntax or other errors preventing the zone from loading correctly on the primary or secondary 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 to 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 that the zone is properly defined within the *named.conf* on the secondary server.

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 2003) to determine if any DNS error messages were received and that the zone was loaded properly. Refer to "Service errors and informational messages" (p. 21-15) 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, refer to the *VitalQIP User's Guide*.

When I use a DNS client (like `nslookup` or `dig`) to query for an object in VitalQIP (a resource record like A, CNAME or PTR), I am seeing a non-authoritative response. Why?

The DNS server that was used (based on resolver settings) to answer the query is not authoritative for the zone in which the requested record resides. During the resolution process, an authoritative server for the requested record was found by the recursive server which responded with the answer that it learned from the authoritative server. In this case, the queried server is acting as a caching server, which performs the recursion process on behalf of the DNS client (**`nslookup`**, **`dig`**). If you verify that you are requesting the resource record directly from a DNS server that ***should*** be authoritative for the requested resource record, the zone in which the record resides is not loaded on the queried DNS server due to misconfiguration, or is rejected due to bad zone data.

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 (ACL).
- The Message Routes for the VitalQIP enterprise and the VitalQIP remote servers are not configured properly.
- The DNS Update Service is not configured with the proper DNS servers. It may need to be [re]generated using the **`qip-genddnsconfs`** CLI. For further information on **`qip-genddnsconfs`**, refer to the *Command Line Interface User's Guide*.

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 QIP Update Service (default method to avoid dynamic name collisions with static objects) the QIP Update Service message route statement is required. In addition, the *qip.pcy* file must have UpdateDNS = True (set in the **`[VitalQIP QIP Update Service]`** section) on the machine where the QIP Update Service is running. Refer to "Sample qip.pcy file" (p. 3-3) for more information on configuring the DNS updates properly.

   ```
   MessageRoute=DHCP:A:0:VitalQIP QIP Update Service:VitalQIP QIP
      Update
   ```
7. 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** (the default for new installations).
8. Use the *dhcpd.log*, *qip-msgd.log*, or the *qip-qipupdated.log* to see if the dynamic updates are being sent.

END OF STEPS

Why does Network Services|DNS Generation fail on Windows 2003?

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 get my Windows 2003 SRV records in VitalQIP?

VitalQIP can be set up to import Window 2003 SRV records. For more information about importing SRV records, refer to "External objects and resource records support" (p. 24-61).

How do I reinstall a service on Windows?

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.

The following table shows where the executables for VitalQIP services are stored.

Table 25-1 Location of VitalQIP services executables

Service	Location
VitalQIP Schedule Service	%QIPHOME%\ScheduleService.exe
VitalQIP QIP Update Service	%QIPHOME%\QIPUpdateService.exe
VitalQIP Remote Service	%QIPHOME%\Remote\RemoteService.exe
Lucent DNS Service	Lucent 4.x Service: <Install_Dir>\bin\named.exe Lucent 3.x Service: %SYSTEMROOT%\System32\named.exe
Lucent DHCP Service	%QIPHOME%\DHCP\DHCPService.exe
VitalQIP Active Lease Service	%QIPHOME%\DHCP\ActiveService.exe
VitalQIP Message Service	%QIPHOME%\MessageService.exe
MS DHCP Monitor Service	%QIPHOME%\DHCP\MSDHCPMonitorService.exe
RMI Scheduler Service	%QIPHOME%\RMIScheedulerService.exe
Login Service	QIPHOME%\LoginService.exe
SSLT Service	%QIPHOME%\SSLTService.exe
MS DNS Update Service	%QIPHOME%\MSDNSUpdateService.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, or **vercheck** ***<service_name>*** from the appropriate directory.

END OF STEPS

DNS error messages

The following table describes error messages for DNS.

Table 25-2 DNS Error Messages

Error Message	Description	Resolution
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.	Lucent DNS 3.x only: 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** **{** **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	Description	Resolution
Error Message: 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 2003). This only occurs on Lucent DNS 3.x and BIND 8.x servers.	Lucent DNS 3.x only: 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 **Note:** This option is only for a primary server serving an outbound AXFR.by using **Network Services|DNS**. If the slave does not have the axfr-serial-diff-tolerance policy set, the transfer will continue from the master, but will be rejected by the slave.
(2010)QSI Common Error 2010: Neither the Remote Service nor the Interim Remote Service has been started on the specified server.	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, refer to the “VitalQIP services/daemons” (p. 20-13) for information on enabling debugging.

Fixing Bad Zones

Overview

fixzone is a program that fixes bad DNS forward zone files. If it detects an error that will cause the zone to be rejected by a BIND 9 based name server, it comments out that specific line in the zone file. This program can be called from VitalQIP user exit script in Remote or the File Generation Service (FGS).

Alcatel-Lucent recommends that you run **fixzone** from *qipdnsuserexit2* on a Remote Server, and from *qipdnsuserexitfgs* on an FGS.

Limitations

The following are limitations to the fixzone user exit:

- **fixzone** cannot reliably fix reverse zones.
- **fixzone** cannot fix a zone if the errors are in the SOA of a zone or if *named_checkzone* does not report the line number of the zone file (e.g. bad ttl error).
- **fixzone** cannot fix the zone files generated from VitalQIP 7.2 which have DNS views.
- **fixzone** cannot fix zones with $INCLUDE statements.
- Depending on the types of severity and the location of errors in the zone file, it is possible that **fixzone** will miss some errors at the end of zone files. For example, if the zone has CNAME and other data errors before records with bad TTL, *named-checkzone* does not report the bad ttl error and hence **fixzone** does not find the error until the next push. That means **fixzone** knows about the bad TTL and the push succeeds, although the zone will be rejected by the name server.

Prerequisite

Perl must be installed on the server where you are running **fixzone**.

Synopsis

```
fixzone.pl [options]
```

Where the options include:

```
--zone-file <file>      Fix this specific forward zone file
--zone-dir  <dir>       Fix all the forward zone files in this directory
--save-dir  <dir>       Save original zone to this directory
--named-conf-dir <dir>  The directory where named.conf file is located
--pcy-file  <file>      The policy file for fixzone
--server-name <name>    Prepend this name before log and report file
--logging <yes/no>      Log messages or not. Default is yes
```

```
--gen-report <yes/no>   Create report file or not. Default is yes
--checkzone-prog <prog> Path of named-checkzone program
--debug                 Print debug messages to stderr
--quiet                 Do not print errors to stderr even if
                        the program has a interactive shell
--test                  Check the zones but do not fix anything
```

Note: If no options are specified, fixzone uses the values from the policy file *fixzone_checkconf.pcy.*

If **`--named-conf-dir`** option is used, then **`fixzone`** looks for the *named.conf* file in that directory and fixes the zone files specified in *named.conf*, otherwise it scans the directory specified with the option **`--zone-dir.`**

The command line options override the values in the policy file. Not all command line options have equivalent policy parameters and vice-versa. See the Policy File section for details.

Policy File

By default, **`fixzone`** looks for its policy file *fixzone_checkconf.pcy* in the same directory where it is located. The policy directives start under the section **`[FixZone]`**. The same configuration file is also used by the **`checkconf`** program. The policy directives of **`checkconf`** program start under the section **`[CheckConf]`**. The location of the policy file can be changed by the command line option **`-pcy-file`** if needed. Any policies before the first section are considered global to all the sections. The same policy value in a section overrides the one in the global section. The syntax of this file is **`key="value"`** or **`key='value'.`** The key and sections are not case sensitive but the values are. Environment variables can be used in the value but the environment variable must exist if used. Any line that starts with **`;`** or **`#`** is a comment and is ignored. The policy file is similar to VitalQIP's *.pcy* file with two exceptions: envrionment variables can be used and a comment line can start with **`#`**. If a command line option exists for a policy, the command line option takes precedence.

At a minimum, the following directives must be configured properly before **`fixzone`** can run correctly from VitalQIP userexit scripts.

- CheckZoneProgram
- LogDir
- ActionFile

Descriptions of all policy directives are shown below:

CheckZoneProgram

Value: Default: *$QIPHOME/usr/bin/named-checkzone*

Allowed: Valid path and filename.

Description: The path of the **`named-checkzone`** program from qddns 4.2+. **`fixzone`** uses **`named-checkzone`** to detect the errors in a zone file. You do not need to add *.exe* at the end of the filename. For Windows. **`fixzone`** detects the operating system and appends *.exe* if needed. If **`fixzone`** cannot find the program specified with this policy, it will look for it in *$QIPHOME/usr/bin* in Unix and *$QIPHOME/named/bin* in Windows. Therefore, you may not have to change it if you are in QIP environment. The command line equivalent of this policy is **`--checkzone-prog`**.

CheckZoneProgramArg

Value: Default: -i none

Allowed: String.

Description: Specify any extra arguments for **`named-checkzone`**. The option **`-i none`** prevents **`named-checkzone`** to do various integrity checks on the zone data. There is no command line equivalent for this policy.

ZoneDir

Value: Default: *$QIPHOME/named*

Allowed: Valid path

Description: The Full path of the directory where **`named.conf`** and the zone files are located. It can be overridden with command line option: **`--zone-dir`**. When **`fixzone`** is run from QIP userexit, use the **`--zone-dir`** option because fixzone should be run in stage 2 of the push to use the *tmp* directory where the zone files are located. The command line equivalent of this policy is **`--zone-dir`**.

SaveBadZones

Value: Default: Yes

Allowed: Yes|No

Description: If yes, the bad zone file is saved to the *SaveDir* directory. Any value other than “Yes” or “True” is considered false. There is no command line equivalent for this policy.

SaveDir

Value: Default: *$QIPHOME/named/safe*

Allowed: Valid path

Description: When an error is found in a zone, a copy of the bad zone file can be saved in the directory specified with this directive. If the directory does not exist, **`fixzone`** tries to create it. If it fails to create it, **`fixzone`** exits with a return code of `1`. The command line equivalent of this policy is **`--save-dir`**.

TimeStampBadZoneFile

Value: Default: Yes

Allowed: Yes|No

Description: If yes, the number of seconds since Unix epoch are appended at the end of the saved zone file. Any value other than "Yes" or "True" is considered false. There is no command line equivalent for this policy.

GenReport

Value: Default: Yes

Allowed: Yes|No

Description: **`fixzone`** can generate a report about problems found/fixed in the zone files. The default name of the report file is *fixzone_report.txt*. It is created in *LogDir*. If the program is called with **`--server-name`** option, then the name of the report file is *server_name_fixzone_report_timestamp.txt*. If **`TimeStampReportFile`** is set to `yes`, the number of seconds since Unix epoch are appended at the end of the filename, for example, *fixzone_report_1240765528.txt* The command line equivalent of this policy is **`--gen-report`**. Any value other than "yes" or "true" is considered false.

TimeStampReportFile

Value: Default: Yes

Allowed: Yes|No

Description: If Yes, seconds the number of seconds since Unix epoch is appended at the end of the report file. Any value other than "Yes" or "True" is considered false. There is no command line equivalent for this policy.

PostRunActionProg

Value: Default: null

Allowed: Valid path

Description: Full path of a program to run if any errors in a zone are fixed. The program accepts one argument, which is the path of the report file. This program must be provided by the user, it is not delivered with VitalQIP. One use of this option could be to trigger a program used to email the report. There is no command line equivalent for this policy.

Logging

Value: Default: Yes

Allowed: Yes|No

Description: If logging is not desired, specify no for the logging parameter. The command line equivalent of this policy is **`--logging`**. Any values other than "Yes" or "True" are considered false.

LogDir

Value: Default: *$QIPHOME/log*

Allowed: Valid path

Description: The full path of the log directory. It must exist if logging is desired. **`fixzone`** tries to create this directory if it does not exist. If the directory cannot be created, **`fixzone`** exits with a return code of `1`. There is no command line equivalent for this policy.

LogFile

Value: Default: *fixzone.log*

Allowed: Path/Filename

Description: Name or path of the log file. If a path is not specified, the file is created in *LogDir*. If **`fixzone`** is called with command line option **`--server-name`**, then the server name is appended to the log file. There is no command line equivalent for this policy.

Verbose

Value: Default: Yes

Allowed: Yes|No

Description: Print verbose information to log. If the value is yes, debugging is turned on unless it is run with the option **`--quiet`** Any values other than "Yes" or "True" are considered false. The command line equivalent of this policy is **`--debug`**.

IgnoreFiles

Value: Default: **.log,*.jnl,*.bak,*.log,*~,*.bat,*_include,*.cache,*.arpa*

Allowed: Comma-separated files

Description: By default **`fixzone`** verifies all the forward zone file names in the zone directory. However if there can be files in the directory which should not be scanned, use comma separated wildcards to specify the filename patterns to ignore. There is no command line equivalent for this policy.

ZoneWhiteListFile

Value: Default: null

Allowed: Path

Description: Specifies the path of a file which can contain the name of the zone files to process. The file contains one zone file name per line. If this file exists and if it contains the zone file names, only these zone files are processed. Any line that starts with **;** or **#** is considered a comment. If a line contains the word 'none' by itself **fixzone** exits immediately with a return code of 0. It is a simple way to disable fixzone. There is no command line equivalent for this policy.

ActionFile

Value: Default: *fixzone_action.cfg*

Allowed: Path/Filename

Description: Specifies the path or name of the file that contains the lines like: **BIND Error message="action"** and other key and value pairs. **fixzone** uses this information to decide what to do when an error is detected in a zone file. If the file is not a path, **fixzone** looks for it in the directory where it is running. There is no command line equivalent for this policy.

Valid actions specified in the action file are: "**fix**", "**ignore**" "**warning**" or "**abort**".

Within the file, the message and action must be inside double quotes. The messages are not case-sensitive but they must be the exact BIND 9 error messages. The action values are also not case sensitive.

> **Note:** The messages are taken from BIND 9's *lib/dns/result.c*. But **named-checkzone** has many hard coded messages. If such messages are not in this file, you must add them and add the appropriate action verb. Also many messages in this file are not encountered while checking a zone.

If the action is "**fix**", the offending line in the zone file will be commented out. The comment in the zone will look like: ;;=

If the action is "**ignore**" the error will be ignored.

If the action is "**warning**" a warning message will be logged

If the action is "**abort**", fixzone will exit immediately with exit code 1.

Use the action "abort" to force a push failure if the **fixzone** is run from *qipdnsuserexit*.

Exit Codes

fixzone exits with 0 on success and 1 on failure. **fixzone** exits with 1 in two situations:

- A problem with its own configuration (could not create log file or could not find *named_checkzone program* etc.)
- It is specifically configured to fail if an error message is detected whose action was configured as "**abort**". Use this feature to force a push failure for some error types in a zone that cannot be fixed by **fixzone** but causes the name server to reject the zone.

Calling fixzone from a Remote Server

After setting up your configuration file, *fixzone_checkconf_pcy*, and your action file, *fixzone_action.cfg*, ensure you must meet the following prerequisites:

- Make sure Perl is installed.
- Make sure network, domain are created in VitalQIP.
- Make sure DNS server is created and assigned to zone(s).

Then use the following steps to run **fixzone** from a remote server.

1. Perform a DNS Generation from the VitalQIP GUI. The first push will fail as no DNS server is running yet.
2. Start the DNS Server with the pushed zone(s). This push will fail.
3. Perform another server push. This time the push should succeed.
4. Ensure the DNS server has loaded the zone(s). Verify that by doing a zone transfer for example.

   ```
   $ dig @server zone axfr
   ```

5. Look at *$QIPHOME/userexits/UserExits.pcy* and ensure **qipdnsuserexit2** is not commented out.
6. Modify *$QIPHOME/qip.pcy*. Under the section **[VitalQIP Remote Service]**, set **FailOnFailedUserExit=true**.
7. Restart the remote service.

8 In Unix, create a file *$QIPHOME/userexits/qipdnsuserexit2* and put the following lines in it. Send the `stdout` and `stderr` to a log file. If **`fixzone`** exits before parsing the config file or could not create log file, it writes error messages to *stderr.*

9 In QIP GUI, go to the domain profile and create two resource records that will cause CNAME and Other data error.

10 Perform a DNS push. Push should succeed. Look at the zone file and ensure that CNAME and Other data error is fixed.

11 Look at the log and report files to verify.

END OF STEPS

Sample files

```
#!/bin/sh
#########
# This is just a sample script to run fixzone at push stage 2
# in the remote server.
# The zones will be fixed in the tmp directory where the zones
# are pulled from FGS. Therefore, --zone-dir must be specified.
# Please read QIP documentation and sample user exit scripts for details
# on how the user exits work.
##########

stderr_log="$QIPHOME/log/fixzone_stderr.log"

programName=`basename $0`
serverName=$1
serverIP=$2
currentPushDir=$3
remotePushDir=$4
pushType=$5
stage=$6
org=$7

# If you want to override, Change >> to > if you want

echo "=====================================================" >>$stderr_log
```

```
echo "Running ${programName} at `date`" >>$stderr_log
echo "Server Name           : ${serverName}" >>$stderr_log
echo "Server IP             : ${serverIP}" >>$stderr_log
echo "Current Push Directory: ${currentPushDir}" >>$stderr_log
echo "Remote Push Directory : ${remotePushDir}" >>$stderr_log
echo "Push Type             : ${pushType}" >>$stderr_log
echo "Stage                 : ${stage}" >>$stderr_log
echo "Organization          : ${org}" >>$stderr_log
echo "QIPHOME               : ${QIPHOME}" >>$stderr_log
echo "Current Directory     : `pwd`" >>$stderr_log

perl $QIPHOME/userexits/fixzone.pl --zone-dir $currentPushDir >> $stderr_log
   2>&1
rc=$?
exit $rc
```

In Windows, put something like the following in the file *QIPHOME%\userexits\qipdnsuserexit2.bat file*:

```
@echo off
REM
   ============================================================================
   =
REM This is just an example. Please look at QIP documentation and sample
REM exampleuserexit.bat file for details.
REM
   ============================================================================
   =

set programName=%0
set serverName=%1
set serverIP=%2
set currentPushDir=%3
set remotePushDir=%4
set pushType=%5
set stage=%6
set org=%7

set stderr_log=%QIPHOME%\log\fixzone_stderr.log

REM As we are running the script in push stage2, zone files will
REM in the temporary push directory. Therefore, the directory
REM must be specified with --zone-dir

perl fixzone.pl.pl --zone-dir %currentPushDir% > %stderr_log% 2>&1
set exit_code=%errorlevel%
REM exit with the error code
```

```
exit %exit_code%
```

Calling fixzone from FGS

When **fixzone** is running on FGS, the zone directory is specified at command line with **--zone-dir**, because, on the FGS, the zone directory is different each time. Specify the server name or IP Address if logging or **gen_report** is set to true to prevent corruption of the log and to report if multiple simultaneous generations occur. As FGS generates zone files for many servers, you may fix files for certain servers expressly. In this case the program only runs for certain servers. For example, to fix zones for server dns.domain.com:

The option **--server-name** is useful in FGS if logging and reporting is enabled. For example, if it runs with **-server-name dns.domain.com**, the name of the log file is *dns.domain.com_fixzone.log* and the name of the report file is **dns.domain.com_fixzone_report.txt**. Without this option, the name of the log and report file are *fixzone.log* and *fixzone_report.txt*.

In Unix, put the follwing lines at the end of *$QIPHOME/userexits/qipdnsuserexitfgs*:

```
#!/bin/sh
#########
# This is just a sample script to run fixzone at at FGS
# The zones will be fixed in the tmp directory where the
# FGS create for a specific server. There for --zone-dir must be specified
# with fixzone.
# Please read QIP documentation and sample user exit scripts for details
# on how the user exits work
##########

programName=`basename $0`
serverName=$1
serverIP=$2
currentPushDir=$3
remotePushDir=$4
pushType=$5
stage=$6
org=$7
stderr_log="$QIPHOME/log/fixzone_$serverName_stderr.log"

# If you want to override, Change >> to > if you want

echo "=====================================================" >>$stderr_log
echo "Running ${programName} at `date`" >>$stderr_log
echo "Server Name           : ${serverName}" >>$stderr_log
echo "Server IP             : ${serverIP}" >>$stderr_log
echo "Current Push Directory: ${currentPushDir}" >>$stderr_log
```

```
echo "Remote Push Directory : ${remotePushDir}" >>$stderr_log
echo "Push Type             : ${pushType}" >>$stderr_log
echo "Stage                 : ${stage}" >>$stderr_log
echo "Organization          : ${org}" >>$stderr_log
echo "QIPHOME               : ${QIPHOME}" >>$stderr_log
echo "Current Directory     : `pwd`" >>$stderr_log


perl $QIPHOME/userexits/fixzone.pl --server-name $serverName --zone-dir
   $currentPushDir > $stderr_log 2>&1
rc=$?
exit $rc
```

In Windows, put the following lines at the end of *qipdnsuserexitfgs.bat*:

```
@echo off
REM
   ============================================================================
   =
REM This is just an example. Please look at QIP documentation and sample
REM exampleuserexit.bat file for details.
REM
   ============================================================================
   =

set programName=%0
set serverName=%1
set serverIP=%2
set currentPushDir=%3
set remotePushDir=%4
set pushType=%5
set stage=%6
set org=%7

set stderr_log=%QIPHOME%\log\fixzone_%serverName%_stderr.log

REM As we are running the script in push stage2, zone files will
REM in the temporary push directory. Therefore, the directory
REM must be specified with --zone-dir

perl fixzone.pl.pl --server-name %serverName% --zone-dir %currentPushDir% >
   %stderr_log% 2>&1
set exit_code=%errorlevel%
REM exit with the error code
exit %exit_code%
```

Sample fixzone_checkconf.pcy file

```
;                    Policy file for fixzone and checkconf
; fixzone v1.5, checkconf v1.1
;
; fixzone is a program to fix DNS forward zone files.
; checkconf is a program to check syntax of named.conf file.
;
; The policies for fixzone program starts under the section [FixZone]
; The policies for checkconf program starts under the section [CheckConf]
;
; Any policies before the first section are considered global to all the
; sections. The same policy value in a section will override the one in the
; global section.
;
; The syntax of this file is key="value" or key='value'
; The key and sections are not case sensitive but the values are.
; Environment variables can be used in the value but the environment variable
; must exist if used.
;
; Any line that starts with ; or # is considered a comment will be
; ignored.
;
; Copyright 2009 Alcatel-Lucent
;-------------------------------------------------------------------------
   ---

; NOTE: When specifying a path, always use forward slash / instead of \ in
   all
; platforms.

;=====================================================
; Global section
;=====================================================

;
;-------------------------------------------------------------------------
   ---

;=====================================================
; The configuration information of checkconf follows
;=====================================================
[CheckConf]

;;
; NamedConfFile: Filename/Path
; The Path or name of named.conf file. If it is not a path, the name.conf file
```

```
; will be looked in the directory specified with --named-dir command line
; option.
NamedConfFile="named.conf"

;;
; CheckConfProgram: Path
; The Path of the named-checkconf program from qddns 4.2+.
; checkconf uses it for checking the syntax of named.conf file.
; There is no no need to add .exe at the end for Windows. checkconf will
; detect the OS and append .exe if needed.
CheckConfProgram="$QIPHOME/usr/bin/named-checkconf"

;;
; Logging: Yes/No
; If logging is not desired, specify no for the logging parameter.
; The command line equivalent of this policy is --logging.
; Any values other than "Yes" or "True" is considered false.
Logging="Yes"

;;
; LogDir: Path
; The directory where the log file will be created. checkconf will try to
; create the directory if it does not exist. If the directory could not be
; created, checkconf will exit immediately with 1.
; There is no command line equivalent for this policy.
LogDir="$QIPHOME/log"

;;
; LogFile: Filename/Path
; The name of the log file. If file is not a path, it will be created in
; LogDir.
; There is no command line equivalent for this policy.
LogFile="checkconf.log"

;;
; Verbose: Yes/No
; Print verbose information to log.  If the value is yes, debugging will be
; turned on unless ran with --quiet
; Any values other than "Yes" or "True" is considered false.
; The command line equivalent of this policy is --debug.
Verbose="Yes"

;;
; GenReport: Yes/No
; checkconf can generate report about problems found in named.conf file.
```

```
; The default name of the report file is checkconf_report.txt.  It will be
; created in LogDir. If the program is called with --server-name option,
; then the name of the report file will be
; server_name_fixzone_report_timestamp.txt. If TimeStampReportFile is set to
; yes seconds since Unix epoch will be appended at the end of the file. For
; example: checkconf_report_1240765528.txt
; The command line equivalent of this policy is --gen-report.
; Any values other than "Yes" or "True" is considered false.
GenReport="Yes"

;;
; TimeStampReportFile: Yes/No
; If Yes, seconds since Unix epoch will be appended at the end of the report
; file
; There is no command line equivalent for this policy.
TimeStampReportFile="Yes"

;;
; PostRunActionProg: Path
; Full path of a program to run if any errors are detected in the named.conf
; file.  The program accepts one argument, which is the path of the report
   file.
; This program does not exist, it has to be written. The program can be used
   to
; email the report for example if desired.
; There is no command line equivalent for this policy.
#PostRunActionProg="/usr/local/bin/post_checkconf.sh"

;-------------------------------------------------------------------------
   ---

;=====================================================
; The configuration information for fixzone follows.
;=====================================================
[FixZone]

;;
; CheckZoneProgram: Path
; The Path of the named-checkzone program from qddns 4.2+.
; fixzone uses named-checkzone to detect the errors in a zone file.
; There is no need to add .exe at the end for Windows. fixzone will detect
; the OS and append .exe if needed.
; The command line equivalent of this policy is --checkzone-prog.
CheckZoneProgram="$QIPHOME/usr/bin/named-checkzone"

;;
```

```
; ZoneDir: Path
; The Full path of the directory where named.conf and the zone files
; are located.  It can be overridden with command line option: --zone-dir.
; When fixzone is run from QIP userexit, --zone-dir option should be used
; because fixzone should be run in stage 2 of the push in order to use the tmp
; directory where the zone files will be located.
; The command line equivalent of this policy is --zone-dir.
ZoneDir="$QIPHOME/named"

;;
; SaveBadZones: Yes/No
; If yes, the bad zone file will be saved to SaveDir directory.
; Any value other than "Yes" or "True" is considered false.
; There is no command line equivalent for this policy.
SaveBadZones="Yes"

;;
; SaveDir: Path
; When an error is found in a zone, a copy of the bad zone file can be saved
; in the directory specified with this directive. If the directory does not
; exist, fixzone will try to create it. If it fails to create it, fixzone
   will
; exit with 1.
; The command line equivalent of this policy is --save-dir.
SaveDir="$QIPHOME/named/safe"

;;
; TimeStampBadZoneFile: Yes/No
; If yes, seconds since Unix epoch will be appended at the end of the saved
; zone file.
; Any value other than "Yes" or "True" is considered false.
; There is no command line equivalent for this policy.
TimeStampBadZoneFile="Yes"

;;
; GenReport: Yes/No
; fixzone can generate report about problems found/fixed in the zone files.
; The default name of the report file is fixzone_report.txt.  It will be
; created in LogDir. If the program is called with --server-name option,
; then the name of the report file will be
; server_name_fixzone_report_timestamp.txt. If TimeStampReportFile is set to
; yes seconds since Unix epoch will be appended at the end of the file. For
; example: fixzone_report_1240765528.txt
; The command line equivalent of this policy is --gen-report.
; Any value other than "yes" or "true" is considered false.
```

```
GenReport="Yes"

;;
; TimeStampReportFile: Yes/No
; If Yes, seconds since Unix epoch will be appended at the end of the report
; file. Any value other than "Yes" or "True" is considered false.
; There is no command line equivalent for this policy.
TimeStampReportFile="Yes"

;;
; PostRunActionProg: Path
; Full path of a program to run if any errors in a zone are fixed.
; The program accepts one argument, which is the path of the report file.
; This program does not exist, it has to be written. The program be used to
; email the report for example if desired.
; There is no command line equivalent for this policy.
#PostRunActionProg="/usr/local/bin/post_fixzone.sh"

;;
; Logging: Yes/No
; If logging is not desired, specify no for the logging parameter.
; The command line equivalent of this policy is --logging.
; Any values other than "Yes" or "True" is considered false.
Logging="Yes"

;;
; LogDir: Path
; The full path of the log directory. It must exist if logging is desired.
; fixzone will try to create the directory if it does not exist. If the
; directory could not be created, fixzone will exit with 1.
; There is no command line equivalent for this policy.
LogDir="$QIPHOME/log"

;;
; LogFile: Filename/Path
; Name or path of the log file. If path is not specified, the file will be
; created in LogDir. If the fixzone is called with command line option
; --server-name, then the server name will be appended to the log file.
; There is no command line equivalent for this policy.
LogFile="fixzone.log"

;;
; Verbose: Yes/No
; Print verbose information to log.  If the value is yes, debugging will be
; turned on unless ran with --quiet
```

```
; Any values other than "Yes" or "True" is considered false.
; The command line equivalent of this policy is --debug.
Verbose="Yes"

;;
; IgnoreFiles: Comma separated wildcard patterns
; By default fixzone will verify all the forward zone file names in the zone
; directory. However if there can be files in the directory which should not
; be scanned, use comma separated wildcards to specify the filename patterns
; to ignore.
; There is no command line equivalent for this policy.
IgnoreFiles="*.log,*.jnl,*.bak,*.log,*~,*.bat,*_include,*.cache,*.arpa"

;;
; ZoneWhiteListFile: Path
; Specifies the path of a file which can contain the name of the zone
; files to process.
; The file contains one zone file name per line. If this file exists and
; if it contains the zone file names, -only- these zone files will be
; processed. Any line that start with ; or # is will be considered a comment.
; If a line contains the word 'none' by itself fixzone will exit immediately
; with 0. It is a simple way to disable fixzone.
; There is no command line equivalent for this policy.
#ZoneWhiteListFile="$QIPHOME/userexits/zones_whitelist.txt"

;;
; ActionFile: Filename/Path
; Specifies the path or name of the file which contains the lines like:
; BIND Error message="action". fixzone uses this information to decide what
   to
; do when an error is detected in a zone file. If the file is not a path,
; fixzone will look for it in the directory where it is running from.
; Please look at the comments at the top of the file for details.
; There is no command line equivalent for this policy.
ActionFile="fixzone_action.cfg"
```

Sample fixzone_action.cfg file

```
; For fixzone v1.5.
; The file contains the lines like:
; "BIND 9.5.1 Error message"="action"
;
; Any line that starts with ; or ; are considered comments and will be
; ignored.
;
; Valid actions are: "fix", "ignore" "warning" or "abort".
```

```
;
; The message and action must be inside double quotes.
;
; The messages are not case sensitive but they must be exact
; BIND 9 error messages. The action values are not case sensitive either.
;
; Note: the messages are taken from BIND 9's lib/dns/result.c.
; But named-checkzone has many hard coded messages, if such
; messages are not in this file, please add them and put appropriate
; action. Also lot of messages in this file are not encountered while
; checking a zone.
;
; If the action is "fix", the offending line in the zone file will be
; commented out. The comment in the zone will look like: ;;=
;
; If the action is "ignore" the error will be ignored.
;
; If the action is "warning" a warning message will be logged
;
; If the action is "abort", fixzone will exit immediately with exit code 1.
;
; The action "abort" should be used to force a push failure if the fixzone
; is run from qipdnsuserexit
;
; Do not change any error message if you're not sure.
; Remove error messages if you are sure that they are useless for zone
; validation.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
"CNAME and other data"="fix"
"multiple RRs of singleton type"="fix"
"unexpected end of line"="fix"
"unexpected end of input"="fix"
"label too long"="fix"
"unknown class/type"="fix"
"unknown RR type"="fix"
"not a valid number"="fix"
"bad IPv6 address"="fix"
;"has 0 SOA records"="ignore"
;"has no NS records"="ignore"
; named_checkzone does not report the line; of bad TTL, so we cannot fix bad
; ttl errors. Exit with 1
"bad ttl"="abort"
"MX is an address"="warning"
; we cannot fix out of range error. Exit with 1
"out of range"="abort"
```

```
"bad escape"="ignore"
"bad bitstring"="ignore"
"bitstring too long"="ignore"
"empty label"="ignore"
"bad dotted quad"="ignore"
"invalid NS owner name (wildcard)"="ignore"
"bad label type"="ignore"
"bad compression pointer"="ignore"
"too many hops"="ignore"
"disallowed (by application policy)"="ignore"
"extra input text"="ignore"
"extra input data"="ignore"
"text too long"="ignore"
"not at top of zone"="ignore"
"syntax error"="ignore"
"bad checksum"="ignore"
"no owner"="ignore"
"no ttl"="ignore"
"bad class"="ignore"
"name too long"="ignore"
"partial match"="ignore"
"new origin"="ignore"
"more data needed/to be rendered"="ignore"
"continue"="ignore"
"bad database"="ignore"
"bad zone"="ignore"
"tsig verify failure"="ignore"
"tsig indicates error"="ignore"
"RRSIG failed to verify"="ignore"
"RRSIG has expired"="ignore"
"RRSIG validity period has not begun"="ignore"
"key is unauthorized to sign data"="ignore"
"invalid time"="ignore"
"expected a TSIG or SIG(0)"="ignore"
"did not expect a TSIG or SIG(0)"="ignore"
"TKEY is unacceptable"="ignore"
"hint"="ignore"
"drop"="ignore"
"zone not loaded"="ignore"
"ncache nxdomain"="ignore"
"ncache nxrrset"="ignore"
"wait"="ignore"
"not verified yet"="ignore"
"no identity"="ignore"
"no journal"="ignore"
```

```
"alias"="ignore"
"use TCP"="ignore"
"no valid RRSIG"="ignore"
"no valid NSEC"="ignore"
"not insecure"="ignore"
"unknown service"="ignore"
"recoverable error occurred"="ignore"
"unknown opt attribute record"="ignore"
"unexpected message id"="ignore"
"seen include file"="ignore"
"not exact"="ignore"
"address blackholed"="ignore"
"bad algorithm"="ignore"
"invalid use of a meta type"="ignore"
"hint nxrrset"="ignore"
"no master file configured"="ignore"
"unknown protocol"="ignore"
"clocks are unsynchronized"="ignore"
"IXFR failed"="ignore"
"not authoritative"="ignore"
"no valid KEY"="ignore"
"obsolete"="ignore"
"already frozen"="ignore"
"unknown flag"="ignore"
"expected a response"="ignore"
"no valid DS"="ignore"
"NS is an address"="ignore"
"received FORMERR"="ignore"
"truncated TCP response"="ignore"
"lame server detected"="ignore"
"unexpected RCODE"="ignore"
"unexpected OPCODE"="ignore"
"chase DS servers"="ignore"
"empty name"="ignore"
"empty wild"="ignore"
"bad bitmap"="ignore"
"from wildcard"="ignore"
"check-names"="ignore"
"dynamic zone"="ignore"
"must-be-secure"="ignore"
"covering NSEC record returned"="ignore"
"duplicate query"="ignore"
"out-of-zone data"="ignore"
"bad dotted quad"="fix"
```

Checking named.conf files

Overview

checkconf is a program to check syntax of a named configuration file. **checkconf** uses the **named-checkconf** program from Lucent DNS (*qddns*) 4.2+ to check the syntax of the *named.conf* file. It is intended to run from a VitalQIP userexit script during a DNS push before the *named.conf* file is placed in the DNS server's directory. The program must be run before the **fixzone** program in *qipdnsuserexit2* and if an error in *named.conf* is detected, the DNS push must be forced to fail immediately.

VitalQIP users can add additional configuration parameters in server's profile using the corporate extension mechanism, and this can cause mistakes to appear in the *named.conf* file after the DNS push. Depending on the type of mistakes, *named* rejects the change, rejects part of the configuration file or fails to to start. This command checks the configuration file before it becomes available to *named* and if it finds a mistake, it forces the DNS push to fail. Alcatel-Lucent recommends that the VitalQIP Remote or FGS (depending where **checkconf** will be run) should be configured to **FailOnFailedUserExit=true**.

Limitations

checkconf cannot fix the *named* configuration file, it can only identify problems.

Prerequisite

The following are prerequisites or checkconf:

- Perl must be installed.
- Must have **named-checkconf** program from Lucent DNS (*qddns*) 4.2+
- Must be used with Lucent DNS (*qddns*) 4.2+.

Synopsis

```
checkconf.pl [options]
    --conf-file <file>       Check this named configuration file
    --named-dir <dir>        The base directoy of named
    --named-conf-file file   Just the name of the named.conf file, not path
    --pcy-file <file>        The policy file of checkconf
    --server-name <name>     Prepend this name before log and report file
    --logging <yes/no>       Log messages or not. Default is yes
    --gen-report <yes/no>    Create report file or not. Default is yes
    --checkconf-prog <prog>  Path of named-checkconf program
    --debug                  Print debug messages to stderr
    --quiet                  Do not print errors to stderr even if
                             the program has a interactive shell
```

Note: If no options are specified, **fixzone** uses the values from the configuration file *fixzone_checkconf.pcy* under section **[checkconf]**

The command line options override the values in policy file. Not all command line options have equivalent policy parameters and vice versa. See the Policy File section for details.

Policy File

checkconf uses the same policy file that **fixzone** uses. The policy parameters for **checkconf** starts under the section [CheckConf]. See "Policy File" (p. 25-13) for the sample *fixzone_checkconf.pcy*_file for the example and description of the supported policies of checkconf.

By default, **checkconf** looks for its policy file **fixzone_checkconf.pcy** in the same directory where it is running. The policy directives start under the section [CheckConf]. The same configuration file also used by the **fixzone** program. You can change the location of the policy file by using the command line option **-pcy-file** if needed. Any policies before the first section are considered global to all the sections. The same policy value in a section overrides the one in the global section. The syntax of this file is **key="value"** or **key='value'**. The key and sections are not case sensitive but the values are. Environment variables can be used in the value but the environment variable must exist if used. Any line that starts with **;** or **#** is considered a comment and is ignored. The policy file is similar to VitalQIP's pcy file with two exceptions: envrionment variables can be used and a comment line can start with **#**. If there is a command line option for a policy, the command line option takes precedence when invoked.

At minimum, the following directives must be configured properly before **checkconf** can run correctly from QIP userexit scripts.

- NamedCheckConfProgram
- LogDir

NamedConfFile

Value: Default: *named.conf*

Allowed: Filename/Path

Description: The Path or name of the *named.conf* file. If it is not a path, the *named.conf* file is looked for in the directory specified with **--named-dir** command line option. The command line equivalent of this policy is **--conf-file**.

CheckConfProgram

Value: Default: *$QIPHOME/usr/bin/named-checkconf*

Allowed: Path

Description: The path of the **named-checkconf** program from qddns 4.2+. **checkconf** uses *named-checkconf* for checking the syntax of the *named.conf* file. There is no need to add *.exe* at the end for Windows. **checkconf** detects the operating system and appends *.exe* if needed. If **checkconf** cannot find the program specified with this policy, it looks for it in *$QIPHOME/usr/bin* in Unix and *$QIPHOME/named/bin* in Windows. Therefore, you may not have to change it if you are in a VitalQIP environment. The command line equivalent of this policy is **--checkzone-prog**.

CheckConfProgramArg

Value: Default: N/A

Allowed: String

Description: Specify any extra argument for *named-checkconf*.

Logging

Value: Default: Yes

Allowed: Yes|No

Description: If logging is not desired, specify **no** for the logging parameter. The command line equivalent of this policy is **--logging**. Any values other than "**Yes**" or "**True**" are considered the same as "**no**".

LogDir

Value: Default: *$QIPHOME/log*

Allowed: Path

Description: The directory where the log file will be created. **checkconf** tries to create the directory if it does not exist. If the directory cannot be created, **checkconf** exits immediately with an exit code of `1`. There is no command line equivalent for this policy.

LogFile

Value: Default: *checkconf.log*

Allowed: Path/Filename

Description: The directory where the log file will be created. **checkconf** tries to create the directory if it does not exist. If the directory cannot be created, **checkconf** exits immediately with a result of `1`. There is no command line equivalent for this policy.

Verbose

Value: Default: Yes

Allowed: Yes|No

Description: Print verbose information to log. If the value is yes, debugging is turned on unless **`--quiet`** is specified. Any values other than "Yes" or "True" are considered false. The command line equivalent of this policy is **`--debug`**.

GenReport

Value: Default: Yes

Allowed: Yes|No

Description: **`checkconf`** can generate report about problems found in the *named.conf* file. The default name of the report file is *checkconf_report.txt*. It is created in *LogDir*. If the program is called with **`--server-name`** option, then the name of the report file is *server_name_fixzone_report_timestamp.txt*. If **`TimeStampReportFile`** is set to yes the seconds since Unix epoch are appended at the end of the file, for example: *checkconf_report_1240765528.txt*. The command line equivalent of this policy is **`--gen-report`**. Any values other than "**`Yes`**" or "**`True`**" are treated as **`no`**.

TimeStampReportFile

Value: Default: Yes

Allowed: Yes|No

Description: If Yes, the number of seconds since Unix epoch are appended at the end of the report file There is no command line equivalent for this policy

PostRunActionProg

Value: Default: N/A

Allowed: Path

Description: Full path of a program to run if any errors are detected in the *named.conf* file. The program accepts one argument, which is the path of the report file. This program must be provided by the user, it is not delivered with VitalQIP. An example of this program would be one used to email the report. There is no command line equivalent for this policy.

Exit Codes

`checkconf` exits with `0` on success and `1` on failure.

Calling checkconf from a Remote Server

After setting up your configuration file, fixzone_checkconf_pcy, and your action file, fixzone_action.cfg, ensure you must meet the following prerequisites:

- Make sure Perl is installed.
- Make sure network, domain are created in VitalQIP.
- Make sure DNS server is created and assigned to zone(s).

Then use the following steps to run fixzone from a remote server.

1. Perform a DNS Generation from the VitalQIP GUI. The first push will fail as no DNS server is running yet.

2. Start the DNS Server with the pushed zone(s).This push will fail.

3. Perform another server push. This time the push should succeed.

4. Ensure the DNS server has loaded the zone(s). Verify that by doing a zone transfer for example.

   ```
   $ dig @server zone axfr
   ```

5. Look at *$QIPHOME/userexits/UserExits.pcy* and ensure **`qipdnsuserexit2`** is not commented out.

6. Modify *$QIPHOME/qip.pcy*. Under the section **`[VitalQIP Remote Service]`**, set **`FailOnFailedUserExit=true`**.

7. Restart the remote service.

8. In Unix, create a file *$QIPHOME/userexits/qipdnsuserexit2* and put the following lines in it. Send the `stdout` and `stderr` to a log file. If **`fixzone`** exits before parsing the config file or could not create log file, it writes error messages to *stderr*.

9 In the VitalQIP GUI, go to the domain profile and create two resource records that cause CNAME and Other data error .

10 Perform a DNS push. Push should succeed. Look at the zone file and ensure that CNAME and Other data error is fixed.

11 Look at the log and report files to verify.

END OF STEPS

Calling checkconf from FGS

checkconf does not work in FGS withthe **named-checkconf** program from qddns 4.2 B10. You must use the **named-checkconf** program from qddns 4.2 B11. The command must also be changed in the script to run **named-checkconf** with **-x push_dir**.

In Unix, put the follwing lines at the end of *$QIPHOME/userexits/qipdnsuserexitfgs*:

```
#!/bin/sh
#########
# This is just a sample script to run fixzone at at FGS
# The zones will be fixed in the tmp directory where the
# FGS create for a specific server. There for --zone-dir must be specified
# with fixzone.
# Please read QIP documentation and sample user exit scripts for details
# on how the user exits work
##########

programName=`basename $0`
serverName=$1
serverIP=$2
currentPushDir=$3
remotePushDir=$4
pushType=$5
stage=$6
org=$7
stderr_log="$QIPHOME/log/checkconf_$serverName_stderr.log"

# If you want to override, Change >> to > if you want

echo "=====================================================" >>$stderr_log
echo "Running ${programName} at `date`" >>$stderr_log
```

```
echo "Server Name           : ${serverName}" >>$stderr_log
echo "Server IP             : ${serverIP}" >>$stderr_log
echo "Current Push Directory: ${currentPushDir}" >>$stderr_log
echo "Remote Push Directory : ${remotePushDir}" >>$stderr_log
echo "Push Type             : ${pushType}" >>$stderr_log
echo "Stage                 : ${stage}" >>$stderr_log
echo "Organization          : ${org}" >>$stderr_log
echo "QIPHOME               : ${QIPHOME}" >>$stderr_log
echo "Current Directory     : `pwd`" >>$stderr_log


perl $QIPHOME/userexits/fixzone.pl --server-name $serverName --named-dir
   $currentPushDir > $stderr_log 2>&1
rc=$?
exit $rc
```

In Windows, put the following lines at the end of qipdnsuserexitfgs.bat:

```
@echo off
REM
   ============================================================================
   =
REM This is just an example. Please look at QIP documentation and sample
REM exampleuserexit.bat file for details.
REM
   ============================================================================
   =

set programName=%0
set serverName=%1
set serverIP=%2
set currentPushDir=%3
set remotePushDir=%4
set pushType=%5
set stage=%6
set org=%7

set stderr_log=%QIPHOME%\log\checkconf_%serverName%_stderr.log

REM FGS creates a unique tmp directory for each server where it
REM creates the named.conf and zone file. Therefore, the directory
REM must be specified with --named-dir.

perl fixzone.pl.pl --server-name %serverName% --named-dir %currentPushDir% >
   %stderr_log% 2>&1
set exit_code=%errorlevel%
```

```
REM exit with the error code
exit %exit_code%
```

Logging and Reporting

If the policy Verbose is "**yes**", all the errors are logged. The same is true for the report file. Logging can be turned off by specifying **Logging="No"** in the policy file or by specifying the command line option **--logging=no**.

Part VI: Advanced DHCP administration

Overview

Purpose

In Part VI, you will find information on advanced DHCP configurations and troubleshooting DHCP.

Contents

Part VI contains the following chapters:

Chapter 26, “Advanced DHCP configurations”	26-1
Chapter 27, “Troubleshoot DHCP”	27-1

26 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 Windows 2003 DHCP Servers	26-2
Managing Windows 2003 DHCP Servers	26-2
Support for DHCP Multi-NIC	26-8
Support for the DHCP Failover Server	26-9
Configuring the Primary DHCP Server	26-10
Configuring the failover DHCP server	26-13
Additional policies for the Lucent DHCP server	26-16
Adding additional Lucent DHCP policies	26-17
DHCPServer and DHCPSocketAddr policies	26-19
Unique hostname resolution options	26-21
FIRST-IN/LAST-IN methodology	26-21
QIP Update Service processing of DHCP lease updates	26-23

Support for Windows 2003 DHCP Servers

Windows 2003 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 2003 DHCP server
- The ability to manage multicast addresses in the VitalQIP GUI and push those out to the Windows 2003 DHCP server

Note: VitalQIP also supports Windows 2000 DHCP servers in the same way as it does Windows 2003 DHCP servers.

Managing Windows 2003 DHCP Servers

VitalQIP supports management of Microsoft's Windows 2003 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 2003 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 2003 DHCP is accomplished through a DHCP server type in VitalQIP called "Windows 2003 DHCP".

The Windows 2003 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 2003 DHCP server can be specified in the **Additional Policies** field. Anything placed in the **Additional Policies** field must be in **`netsh dhcp`** format. For more information on this format, access your Microsoft 2003 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 2003:

- 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 the *VitalQIP User's Guide* or the *VitalQIP Web Client User's Guide.*

VitalQIP Interface with Windows 2003 Limitations

If you are managing Microsoft DHCP 2003 servers, you must be aware that there are limitations in the VitalQIP interface to the Microsoft 2003 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 2003 server to configure the Microsoft 2003 DHCP server, even if the VitalQIP client is running on the Microsoft 2003 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 2003 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 2003 DHCP treats them as either Bootp or DHCP as needed.
- Microsoft 2003 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 2003 configuration.
- When specifying additional policies in the Microsoft Windows 2003 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 2003 DHCP server can be imported into the VitalQIP system to facilitate the management of Microsoft 2003 DHCP servers, or to assist in the conversion of DHCP servers (for example, Microsoft 2003 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 2003 DHCP information into VitalQIP:

1. Install the VitalQIP remote server, including support for the Microsoft 2003 DHCP server, on the system that is currently running the Microsoft 2003 DHCP server.

2. Install the VitalQIP Command Line Interface on the system that is currently running the Microsoft 2003 DHCP server.

3. Create a temporary directory to store temporary import information (for example, *\temp)*.

   Note: 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 2003 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`**, refer to 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 2003 Dynamic DHCP dynamic clients (D-DHCP)
   - *ms_setmanobjects.txt* - contains all Microsoft 2003 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 2003 DHCP server within VitalQIP. Note the server name is case sensitive and must match exactly.

9 Define all networks that Microsoft 2003 DHCP scopes reference within VitalQIP.

10 Define all subnets that Microsoft 2003 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
```

Note: Check the reject files (*<filename>.rej*) for any records that are not accepted.

END OF STEPS

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" (p. 3-3) for more information.

Updating VitalQIP with Information from the Microsoft 2003 DHCP Server

VitalQIP provides new services that allow information written to the Microsoft 2003 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 2003 DHCP server gives out new leases, renews leases, and deletes leases.

> **Note:** The **VitalQIP MSDHCPMonitorService** starts the MS DHCP server automatically if it is not already running.

The Microsoft 2003 DHCP server must be configured with logging turned "on". Optionally, logging can be turned "on" via the registry; however, the VitalQIP Microsoft 2003 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 2003 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 pushing to the server via **Network Services | DHCP Generation** in the VitalQIP client or via **Infrastructure | DHCP Server | Perform DHCP Generation** in Web client 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.

> **Note:** 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. Refer to 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 2003 DHCP server, you must have Active Lease Service running on the Microsoft 2003 DHCP server.

You cannot delete Active Leases on the Microsoft 2003 DHCP server.

CLI Commands and Utilities

The **`enterdhcpsvr`** and **`exportdhcpsvr`** CLI commands support Window 2003 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, refer to 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 2003 and UNIX platforms.

On Solaris 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. Refer to "Additional policies for the Lucent DHCP server" (p. 26-16) 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** in the VitalQIP client or **Infrastructure | General Policies | General | Allow Dotted Hostname** in the Web client, 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 the *VitalQIP User's Guide* or *VitalQIP Web Client 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. (Refer to "Configuring the Primary DHCP Server" (p. 26-10) 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.

– After the installation of the Remote Server, including the DHCP Server, that is intended to be configured as a Primary DHCP server in a failover configuration, for which there is an actively running secondary/failover DHCP server, the primary DHCP server/daemon should not be started until after a DHCP generation has been performed to push the correct configuration and policy files to the server. The primary DHCP server must then be started following the push, so that the lease synchronization properly occurs between the secondary DHCP server and the newly installed primary DHCP server.

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.

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).

Note: 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 the followng table.

Table 26-1 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.
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`**.

Parameter name	Values type	Description
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 the *VitalQIP User's Guide* or *VitalQIP Web Client User's Guide*) except the failover policies that are specific to the primary server.

> **Note:** 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 the following table.

Table 26-2 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`**.

Parameter name	Value type	Description
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.
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 the *VitalQIP User's Guide* for more information.

Note: If you are editing the *dhcpd.pcy* file, do not change policies that are not documented. They are intended for internal use by Alcatel-Lucent only.

Note: 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

Purpose

This section describes how to add additional Lucent DHCP policies.

Procedure

To add additional policies through the VitalQIP client, follow these steps:

1 Choose one of the following.

If you are in..	Then..,
The VitalQIP client	1. Access **Infrastructure | Server**. **Result:** The Server Profile Option window opens. 2. Select your Lucent DHCP server from the **Existing Server** list. 3. Select **Modify Server**. 4. Click **OK**. **Result:** The Server Profile window opens. 5. Select **Additional Policies** from the **Parameters/Values** list. **Result:** The **Value** field appears.
The Web client	1. Access **DHCP | DHCP Server**. **Result:** The DHCP Server page opens. 2. Expand the hierarchy until you come to the desired Lucent DHCP server and select your Lucent DHCP server from hierarchy. **Result:** The DHCP Properties page opens. 3. Select **Modify**. **Result:** The Modify Server page opens. 4. Select **Additional Policies**. **Result:** A **Additional Policies** field appears.

2 Type the policy in Parameter=Value format. The following table describes the additional policies that are available.

Table 26-3 Additional Lucent DHCP server policies

Policy	Values	Descriptions
DHCPServer	IP address Default: None	Defines the "local" subnet for the server in a multi-NIC configuration. Any LOCAL broadcast DHCP client discover, regardless of which NIC it is received on, will result in an address being offered from the subnet containing this address. The server still honors the giaddr criterion for non-local broadcast discovers. The DHCPServer policy, when specified, is also used to populate the server ID option (option 54) in outgoing OFFER and ACK messages, as well as the server ID field in the outgoing update messages (to VitalQIP, and so on). When the policy is not specified, these fields are set to the NIC address on which the client request was received. Refer to "DHCPServer and DHCPSocketAddr policies" (p. 26-19) for more information about this policy.
DHCPSocketAddr	IP address Default: None	Refer to "DHCPServer and DHCPSocketAddr policies" (p. 26-19) for more information about this policy.
SiAddr	IP address|Hostname Default: Dotted decimal IP address	Defines the IP address that is placed in the siaddr field of outgoing packets. If there is no value specified, SiAddr defaults to the DHCPServer policy value, if it is specified. A special value of "sname" can be specified for the SiAddr policy. When this is the case, the server will use the address specified by the TFTP Server option (option 66), that is configured for this client address, in the siaddr field of outgoing packets.

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.

3 Click **Apply**. The policy and value appear in the **Value** column next to **Additional Policies**.

4 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.

Note: 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 **Policies | Global Policies | General | FirstIn LastIn** in the VitalQIP client or via **Infrastructure | Global Policies | General | FirstIn LastIn** 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** - **defaultname** or **hostname** - **lastusedname** (see Example 1).

Example 1

1. The existing hostname is shown in the following table.

Table 26-4 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 in at IP address 192.200.138.1.
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 the following table) stays the same.

Table 26-5 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** - **defaultname** or **hostname** - **lastusedname** (see Example 2).

Example 2:

1. The existing hostname is shown in the following table.

Table 26-6 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 at IP address 192.200.138.1.
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 the following table shows.

Table 26-7 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

Note: 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 the following table.

Table 26-8 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 at IP address 192.200.138.1.
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 the following table. The original static name stays the same.

Table 26-9 LAST-IN original static name remains the same

IP Address	Hostname	Status	Type
192.200.138.1 192.200.138.2	myserver myserver-pc0002pc	used used	static 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 the *VitalQIP User's Guide* and *VitalQIP Web Client User's Guide*), and then the business logic is outlined.

Note: The term "incoming" in this section describes the Lucent 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.

Validate CNAME Records

This global policy determines whether or not the DHCP lease update process validates DHCP fully qualified hostnames against CNAME owner resource record data on objects, domains and reverse zones. When the policy is set to the default of true, validation proceeds. If a collision is detected, the CNAME resource record retains its hostname

information, and the DHCP hostname is renamed. If the policy is set to false, the validation of the DHCP fully qualified hostname against CNAME resource record owner data will not occur.

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 only takes 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 an existing CNAME resource record defined on an object, domain or reverse zone. If the DHCP fully qualified domain name collides with an

existing CNAME resource record, the CNAME resource record owner data retains its fully qualified hostname data, and the incoming DHCP hostname is modified to a unique generated hostname.

The third 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, the static object keeps its hostname, and the incoming DHCP lease hostname is modified to a unique generated hostname.
- If the static object is a tombstoned object, the tombstoned object hostname is modified to a unique generated hostname, and the DHCP lease update object is allowed to keep its requested FQDN.
- If a MDHCP/Bootp object has the same FQDN, and the Protect MDHCP/Bootp object policy is set to true, the MDCHP/Bootp object is treated like a static object, and is allowed to keep its fully qualified domain name. The incoming DHCP lease update hostname has 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, it is assumed 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 is requesting, and the existing dynamic object in the database with the same FQDN will be reverted back to its 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 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.

27 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 "General VitalQIP troubleshooting" (p. 20-2) and "Environment verification" (p. 20-4) for more information.

Contents

This chapter presents the following topics.

Common DHCP problems and questions	27-2
DHCP error messages	27-6

Common DHCP problems and questions

Purpose

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 "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, as well as in **Network Services|DHCP Generation**. For information on the Server Profile, refer to the*VitalQIP User's Guide*.

After debugging is enabled, use **Network Services | DHCP Generation** to specify a DHCP server and the debug level. As the DHCP server 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, refer to 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.

  Note: "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 the *VitalQIP User's Guide*.

Why does Network Services|DHCP Generation fail on Windows 2003?

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" (p. 20-13) 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 "Avoid dynamic name collisions" (p. 4-9) for more information on configuring the DNS updates properly.

DHCP error messages

The following table describes error messages for DHCP.

Table 27-1 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" (p. 20-13) 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 6 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" (p. 20-13) for information on enabling debugging if the service does not start.

Glossary

A

AD
Active Directory

AXFR
Asynchronous Full Transfer Zone

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.

CLI
Command Line Interface

G

GSS-TSIG
Generic Security Service - Transaction Signature

J

J2EE
Java 2 Platform, Enterprise Edition

K

key pair
A key pair consists of a matching public and private key.

N

NAT
Network Address Translation

R

RFC
Request for Comments

RMI
Remote Method Invocation

S

SDK
Software Development Kit

SSL
Secure Sockets Layer

T

TCP
Transmission Control Protocol

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.

U

UDA
User Defined Attribute

W

WSDL
Web Service Definition Language

WS-I
Web Services Interoperability Organization

Index

A

AcceptSleepTime policy, 4-3, 5-4

Access Control, 16-1

ACKPeriod policy, 10-4

Active Lease Service
- daemon parameters, 2-28
- description of, 1-6
- message types, 20-24
- policies, 10-2
- signal types, 20-15
- starting, 20-32
- statistics, 20-29

Active Lease Service Message Types, 20-24

AdditionalIPList policy, 7-5

Address Allocator Service policies, 17-3

Admin Service policy, 17-8

administration
- database, 19-11

AllowConnectionList policy, 4-6, 5-3, 6-4, 7-5, 8-5, 10-3, 11-3, 15-5

allowTimeOutRecovery property, 18-39

AllowUDPControlMessages policy, 7-6

appliance_manager.url property, 18-36

appliance_manager.version property, 18-36

AssignedInfraTypeOnlyInAddPersonalView policy, 17-6

Asynchronous message flag, 7-12

AttemptLogLevel callout policy, 22-8

audit data
- truncate, 19-24

audit.database_name.database_user.password policy, 8-6

Auditing policy, 3-50

AuditUpdates policy, 4-6

Authentication Callout
- arguments, 22-15
- enable for Tomcat Server, 22-18
- enable qip-ldapauth for Tomcat, 22-28
- LDAP server, 22-20
- library, 22-4
- overview, 22-2
- prerequisites, 22-5

AuthenticationCLI, 22-2

AuthenticationCli callout policy, 22-8, 22-11

AuthenticationCli policy, 22-12, 22-22

AuthLibrary policy, 8-6, 22-11

AuthorizationMode policy, 22-36

Authorize policy, 22-36

auto_discovery.url property, 18-36

auto_discovery.version property, 18-37

autoExpandPool policy, 17-5

autoRefreshDefault policy, 17-8

AXFRQueueLength policy, 6-5

B

backup
- change medium, 19-8

backup server, 19-6

backup_qip_log, 19-5

BaseDn policy, 22-33

BasePort policy, 14-3

BIND
- Lucent DNS directives, 24-3
- more support of, 24-3
- named.conf, 24-3
- support of BIND 8.x, 24-3
- version 9.x directives, 24-3

BindDn policy, 22-33

BindPass policy, 22-33

blockNetworkContact property, 18-37

C

cacerts keystore, 23-9

change backup medium, 19-8

checkInterval policy, 17-6

CheckTime policy, 9-3

CircularMyViewCheck policy, 17-5

client setup
- secure DDNS updates, 24-44

CliInputFileDir policy, 17-4

CliInputFileNamePrefix policy, 17-4

CliKeepGeneratedFile policy, 17-4

commonService bean
- callout property, 22-18, 22-28

configuration

Soap service, 18-23
Tomcat server, 18-22
UDA callout, 18-70
Web Service, 18-23
configurations, installation, 1-12
configure session time-outs, 18-65
ConnectQueueDepth policy, 4-5, 5-6, 8-4
Continuous Model, 24-62
ConvertDomainNameCase policy, 10-4
ConvertHostNameCase policy, 10-4
ConvertSpaces policy, 4-5, 5-6
CtlReqRetryMax policy, 26-11

D

daemons
descriptions of, 2-22
parameters, 2-22
statistics, 20-28
stopping, 2-22, 20-30
syslogd, 20-17
troubleshooting, 20-13, 20-30
data space
check, 19-16
database
administration, 19-1
administrative tasks
qip-util, 19-11
administrative tasks with qip-util, 19-11
back up of Sybase, 19-4
backing up the VitalQIP Database & Transaction Log, 19-7
backup server, 19-6
change procedure cache size, 19-8
change total memory size, 19-8
changing the Backup Medium, 19-8
create database dump device, 19-7
dump devices, 19-6
encrypting password, 19-9
index statistics maintenance, 19-25
manage transaction log space, 19-18
managing data space, 19-16
manual backup, 19-6
recovery, 19-20
re-initialize, 19-9
running Sybase, 19-3
stored procedures/triggers, 19-15
Sybase transaction log, 19-6
transaction log dump devices, 19-7
truncating audit data, 19-24
version, 19-4
version of, 19-3
database failover, 19-20
DDNS_Attempts policy, 5-4
DDNS_Timeout policy, 5-5
DDNS_TTL policy, 5-5
Debug debug, 6-3, 14-3
debug filenames
date strings, 3-46
Debug policy, 3-49, 4-2, 5-2, 7-4, 8-3, 9-2, 10-2, 11-2, 12-3, 15-3, 16-2, 16-4, 22-32
debug policy
description, 3-44
filter program, 3-46
Java, 3-47
DebugFile policy, 4-2, 5-2, 6-3, 7-4, 8-3, 9-2, 10-3, 11-2, 12-3, 13-4, 14-3, 15-3, 16-3
debugging
DHCP, 27-2
DNS, 25-2
enabling, 20-8
DebugRotateFileSize debug policy, 3-46
DebugRotateInterval debug policy, 3-46
Default
None, 8-6, 11-3
True, 12-6
DeferUsers policy, 22-34
DeferUsersFile policy, 22-34
DelayOnFailure callout policy, 22-9
deleteTempFiles property, 18-43
DenyConnectionList policy, 4-6, 5-3, 6-4, 7-4, 8-4, 10-3, 11-3, 15-4
DHCP
daemon parameters, 2-34
Lucent DHCP, 26-16
support of failover server, 26-9
support of multi-NIC, 26-8
troubleshooting, 27-1, 27-2
Windows 2000, 26-2
dhcpd
parameters, 2-34
starting, 20-32
DHCPServer policy, 26-8, 26-18, 26-19
dhcpServer.DefaultDirectory, 18-47
dhcpServer.DHCPTemplate, 18-45
dhcpServer.FailoverServerType, 18-45

dhcpServer.ManagedRange, 18-45
dhcpServer.PingDelay, 18-45
DHCPSocketAddr policy, 26-8, 26-18, 26-19
directives
  Lucent DNS, 24-3
disableDNSCheckboxes, 18-45
DisplaySelectMasks policy, 7-7
distributed server
  definition of, 1-2
  installation configurations, 1-12
distributed services
  configuratons, 1-9
  definition, 1-3
  network considerations, 1-11
DnList policy, 22-34
DnListFile policy, 22-35
DNS
  daemon parameters, 2-33
  debugging, 25-2
  error messages, 25-10
  forwarder, 25-4
  Microsoft, 24-6
  roundrobin, 25-4
  starting services manually, 2-21
  troubleshooting, 25-1
  Windows 2000, 24-6
  zone transfers, 25-5
DNS server setup
  secure DDNS updates, 24-25
DNS Update Service
  daemon parameters, 2-27
  description of, 1-6
  message types, 20-23
  policies, 5-2
  signal types, 20-15
  statistics, 20-29
DNS Views and DNSSEC, 24-56
DNSBinsDir policy, 12-3
dnscmd, 24-6
dnscmd command, 23-20
DNSGenerateSleep policy, 5-8
DNSSEC, 24-56
  DNS Views, 24-56
DNSServerAddress policy, 12-3
DNSUpdatePrincipal policy, 4-8
documentRoot property, 18-39
DoKinit policy, 5-7
DOMAIN_QUICK_VIEW_PAGE_SIZE, 18-47
DoSecureUpdates policy, 5-7
DropRequests policy, 4-5, 8-3
DumpStatsOnExit policy, 4-6, 5-6, 7-6, 8-4, 9-4

E

edit
  web.xml file, 18-65
enable71WebServiceApi property, 18-40, 18-69
EnableDebugSocket policy, 7-7
encrypt password, 19-9
encrypted connection
  LDAP authentication, 22-23
enforceNoCache property, 18-46
enterprise server
  Automatic Synchronization, 20-12
  installation configurations, 1-12
  order for starting services, 19-29
  order for stopping services, 19-27
  synchronizing servers, 20-11
environment variable
  $LUCENT_PASSWORD, 22-13
  LDAP_BIND_PASS, 22-22
  LDAP_USER_PASS, 22-22
  LDAPRC, 22-23
  LUCENT_PASSWORD, 22-26
environment variables, 20-4
error messages
  DNS, 25-10
  login, 20-11
  services, 21-18
ExcludeEmptyTypes policy, 17-5
External Objects
  configuring VitalQIP, 24-69
  Continuous Model, 24-62
  description of, 24-61
  how VitalQIP handles, 24-65
  importing into VitalQIP, 24-62
  modifying, 24-67
  Polling Model, 24-64
  support of, 24-61
External Resource Records
  configuring VitalQIP, 24-69
  Continuous Model, 24-62
  description of, 24-61
  how VitalQIP handles, 24-65
  importing into VitalQIP, 24-62
  modifying, 24-67
  Polling Model, 24-64
  support of, 24-61
ExternalStatusToSystemLog policy, 4-8

F

FailOnFailedUserExit policy, 13-4
Failover server
  configuring dhcpd.pcy on the failover server, 26-13

configuring dhcpd.pcy on the primary DHCP server, 26-10
policies, 26-11, 26-13
support of, 26-9
FailureLogLevel callout policy, 22-8, 22-11
FailureMessage callout policy, 22-9
FailureThreshold policy, 6-4
File Generation Service, 1-7
description of, 1-7
File Generation Service, VitalQIP, 1-7
file.resource.loader.cache property, 18-42
file.resource.loader.modification CheckInterval property, 18-42
FileBufferSize policy, 12-3
FileGenerationServer policy, 12-4
Finding the Program's Version Numbers with overcook, 19-3
FIRST-IN/LAST-IN, 24-68
FIRST-IN/LAST-IN Methodology, 26-21
foldersize property, 18-39

G
Generation Service, VitalQIP File, 1-7
GroupAttr policy, 22-37
GSSAPIImmediateRetries policy, 5-7
GSSAPIRetryDelay policy, 5-7
gss-principal directive, 24-32, 24-41
GSS-TSIG
DNS client setup, 24-44
DNS server setup, 24-25

H
hosts file
modfy, 24-42
modify, 24-33

I
IBM Services, 2-21
index statistics maintenance, 19-25
informational messages, 21-15
installation configurations, 1-12
IsBackupOf policy, 8-6

J
java class UDA callout, 18-71
Java debug policies, 3-47
Job Scheduling Service policies, 17-6
jobReadAhead policy, 17-6
jobThreads policy, 17-7

K
KerberosPrincipal policy, 5-7
KeysFile policy, 23-8
keystore file name, 23-8
KeytabPath policy, 5-8
keytool utility, 23-8
kill command, 20-17, 20-30, 20-31
krb5.conf file, 24-30, 24-38
KRB5_CONFIG environment variable, 24-35
ktpass utility, 24-22
ktutil utility, 24-24

L
LDAP authentication
encrypted connection, 22-23
overview, 22-20
PEM format, 22-23
qip-ldapauth command, 22-20
setup, 22-22
troubleshoot, 22-26
LDAP server
disable certificate, 22-23
LDAPProtocolVersion policy, 22-32
LDAPRC environment variable, 22-23
ldaprc file
sample, 22-23
LDAPServerURL policy, 22-32
libqipauthcallout.so, 22-6
license key, 20-5
LicenseInterval policy, 9-3
LicenseReadInterval policy, 9-3
ListenOnLoopback policy, 26-19
ListenPort policy, 4-7, 5-3, 6-3, 8-3, 10-4
Load Balance message flag, 7-12
locales.default property, 18-35
log file rotation, 3-46
log4J
message parameters, 18-25
log4j, 18-24
message levels, 18-24
log4j.properties, 18-24
sample file, 18-26
LogFile policy, 22-32
Login Error Message, 20-10
Login Service
daemon parameters, 2-26
description of, 1-6
message types, 20-18
moving, 8-2
policies, 8-2
signal types, 20-14
statistics, 20-28
LoginServer policy, 4-7, 9-4
Lucent DHCP

additional policies, 26-16
Lucent DHCP Service
daemon parameters, 2-34
description of, 1-9
starting, 20-32
Lucent DNS
configuration via VitalQIP Service Controller, 2-11
directives, 24-3
WINS, 24-6
Lucent DNS Service
daemon parameters, 2-33
description of, 1-9

M
managedFileDisplaySize property, 18-46
manageObjectsPageSizes, 18-40
manageSubnetsPageSizes, 18-40
managing
Active Lease Service policies, 10-2
Master policy, 4-3, 5-4
MaxConnections policy, 4-5, 5-5, 7-5, 8-4
MaxDNSUpdateThreads policy, 12-4
MaxFileCopyThreads policy, 12-5
MaxInfrastructureInstances policy, 17-6
MaxPersonalViews policy, 17-6
MaxZipFileSize policy, 13-2
message routes
SSL-enabled, 23-5
Message Service
daemon parameters, 2-25
description of, 1-4
message types, 20-19
policies, 7-3
signal types, 20-14
starting, 20-32
statistics, 20-28
message transport
secure, 23-5
message types
Active Lease Service, 20-24
DNS Update Service, 20-23
Login Service, 20-18
Message Service, 20-19
MS DNS Update Service, 20-20
QIP Update Service, 20-21
Remote Service, 20-26
RMI QAPI Service, 20-27
RMI Scheduler Service, 20-27
MessageBufferSize policy, 15-3
MessageQueue policy, 7-8, 7-9
MessageRoute policy, 7-8, 7-10
MessageServicePort Numeric policy, 12-7
MessageServicePort policy, 4-8, 7-6, 8-5, 10-4, 15-4
Microsoft DHCP Monitor Service
description of, 1-6
policies, 11-2
Windows 2000, 26-5
Microsoft DNS
configuring in VitalQIP, 24-6
support of, 24-6
MS DNS Update Service
message types, 20-20
signal types, 20-17
MyView Service policies, 17-5

N
named
description of, 1-9
parameters, 2-33
named.conf, 24-3, 24-4, 24-5
net user commands, 24-22
network
configuration considerations, 1-11
planning, 1-11
NIC, 26-8
NumberofThreads policy, 26-19

O
OBJECT_INFO_PAGE_SIZE, 18-47
ObjectCountInterval policy, 9-3
Oracle
administration, 19-1
administrative tasks with qip-util, 19-11
encrypting password, 19-9
managing data space, 19-16
managing transaction log space, 19-18
re-initializing, 19-9
OrgID policy, 11-3

P
partially managed object, 24-61
PassPhrase policy, 23-8
password, 19-9
changing qipman's password, 20-10
encrypt, 19-9
PauseBetweenChildren policy, 5-8
PEM format
LDAP authentication, 22-23
PersistenceServiceHost, 18-34, 18-35, 18-36, 18-37, 18-39, 18-42, 18-43, 18-46, 25-14, 25-15, 25-16, 25-17, 25-33, 25-34, 25-35
policies
Debug, 3-44
definition of, 1-3

qip.pcy file, 3-3
Policies for VitalQIP Services, 3-3
policy file
global section, 3-49
Poll Delay policy, 26-11
Polling Model, 24-64
PollRetryMax policy, 26-13
poolContact property, 18-37
PortNumber policy, 12-5
Post-Edit Callout, 18-70
Pre-Edit Callout, 18-70
Preparing to Back Up the Transaction Log, 19-6
primary DNS server
Send Secure Updates, 24-31, 24-40
Primary Reconnect message flag, 7-12
ProcessFilesQueueLength policy, 6-5
ProcessInterval policy, 9-3
PublishID policy, 12-5, 15-4, 18-45
PublishOnMessageService policy, 12-5
purgeCheckInterval policy, 17-8

Q

qapi, 13-2
description of, 1-8
message types, 20-27
parameters, 2-31
policies, 13-2
statistics, 20-29
qdhcp
description of, 1-9
QIP Update Service
daemon parameters, 2-24
description of, 1-4
message types, 20-21
policies, 4-2
signal types, 20-14
starting, 20-33
statistics, 20-28
qip.database_name.database_user.password policy, 8-6
qip.pcy, 3-2, 3-3
Access Control, 16-2
DNSSEC, 24-57
qip.properties
Access Contro, 18-47
qip.properties file, 18-33
qip_dat
full, 19-17
qip_database property, 18-34
QIP_DO_KINIT environment variable, 24-47
QIP_KEYTAB_PATH environment variable, 24-47
qip_login_server property, 18-34
QIP_PRINCIPAL_NAME environment variable, 24-47
QIP_USE_DNS_UPDATE_SVC environment variable, 24-44
QIPAuthCallout
qip.pcy section, 22-12, 22-22
QipAuthCallout.dll, 22-6
qipCallout bean, 22-18, 22-28
qip-clear
remove audit data, 19-17
truncate audit data, 19-24
qipConfig bean, 22-18, 22-28
qip-crypt, 19-9
qip-cs-startup, 2-21
qipd, 9-2
description of, 1-4
parameters, 2-23
policies, 9-2
starting, 20-31
statistics, 20-28
qip-dbinit, 19-9
qip-dnsupdated, 5-2
description of, 1-6
message types, 20-23
parameters, 2-27
statistics, 20-29
qip-ds-startup, 2-20
qip-es-startup, 2-20
qip-ldapauth
parameters, 22-30
qip-ldapauth authentication callout
enable for Tomcat, 22-28
qip-ldapauth command
location, 22-29
policy file, 22-31
SSL/TLS, 22-23
qip-ldapauth.pcy file, 22-31
qip-logind
authentication callout, 22-2
description of, 1-6
moving, 8-2
parameters, 2-26
policies, 8-2
statistics, 20-28
qip-msdnsupdated, 6-2
qip-msgd, 7-3
description of, 1-4
parameters, 2-25
policies, 7-3
starting, 20-32
statistics, 20-28
qip-netd, 10-2
daemon parameters, 2-28
description of, 1-6
policies, 10-2
starting, 20-32
statistics, 20-29
qippath property, 18-34

qip-qipupdated
description of, 1-4
parameters, 2-24
starting, 20-33
statistics, 20-28
qip-rmisched, 14-2
description of, 1-8
message types, 20-27
parameters, 2-30
policies, 14-2
statistics, 20-29
qip-rmtd, 12-2
description of, 1-7
parameters, 2-29, 2-32
starting, 20-32
qip-ssltd, 15-2
qip-syncexternal, 24-10, 24-64
qip-updated
policies, 4-2
qip-util, 19-11
function values, 19-12
qiprs-startup, 2-20
QueueFlushPeriod policy, 7-7

R

randomBoundry policy, 17-7
randomizeStart policy, 17-7
ReceiveBufferSize policy, 7-6
reclaimAddressPageSizes, 18-40
reclaimNetworkStatusPageSizes, 18-40
ReConnectTimeout policy, 7-5, 8-3
recover database, 19-20
RecsPerPacket policy, 6-5
RegistryPort policy, 12-6, 14-3
remote server
definition of, 1-2
installation configurations, 1-12
order for starting services, 19-31
order for stopping services, 19-30
Remote Service
daemon parameters, 2-29, 2-32
description of, 1-7
message types, 20-26
signal types, 20-16
starting, 20-32
reportFileTransferLimit property, 18-41
RequireAttrVal policy, 22-36
RequireFilter policy, 22-36
RequireGroup policy, 22-36
RequireSSLConnection policy, 12-7
retentionDays policy, 17-7
RMI QAPI Service
daemon parameters, 2-31
message types, 20-27
policies, 13-2
signal types, 20-17
statistics, 20-29
RMI Scheduler Service
daemon parameters, 2-30
description of, 1-8
executor, 1-8
message types, 20-27
policies, 14-2
signal types, 20-16
statistics, 20-29
ruleNamePrefix property, 18-37
runtime.log property, 18-36
runtime.log.logsystem.class property, 18-42

S

Schedule Service
daemon parameters, 2-23
description of, 1-4
message types, 20-17
policies, 9-2
signal types, 20-13
starting, 20-31
statistics, 20-28
Schedule Service Signal Types, 20-13
SchedulePassword policy, 9-4
schedulerHostName policy, 17-6
SchedulerName policy, 12-6
SchedulerName.ExecutorClass policy, 14-5
SchedulerName.ExecutorCount policy, 14-5
SchedulerName.ExecutorName policy, 14-5
SchedulerName.ExecutorRegistrationTimeout policy, 14-5
SchedulerName.KeysFile policy, 14-6
SchedulerName.MaxExecutorCount policy, 14-5
SchedulerName.PassPhrase policy, 14-6
SchedulerName.Port policy, 14-4
SchedulerName.Secure policy, 14-6
SchedulerName.VirtualMachineParams policy, 14-6
SchedulerNames policy, 14-4
SchedulerServerIP policy, 13-4
secondary DNS server
Send Secure Updates, 24-31, 24-40
Secure DNS Updates parameter, 24-30, 24-39
Secure Dynamic Updates
support of, 24-20
Secure Socket Layer

configure, 23-8

configure proxy server, 23-12

Secure Sockets Layer
introduction, 23-2

Secure Zones
background of, 24-10

configuring VitalQIP to manage secure zones, 24-12

generating, 24-20

setting Global Policies, 24-17

setting object level policies, 24-19

setting subnet level policies, 24-18

support for Windows 2000 DNS, 24-10

SecureIncomingMessageService Connections policy, 7-2, 15-3

SecureMessageRoute policy, 7-8

SecureOutgoingProxyConnections policy, 7-7

secure-updates directive, 24-32, 24-41

self-signed certificate, 23-8

SendEmailAlerts policy, 17-4

SendRenewsToDNS policy, 4-4, 5-5

Server
Automatic Synchronization, 20-12

failover, 26-9

synchronizing servers, 20-11

Windows 2000 DNS, 24-6

VitalQIP failures, 20-11

Server policy, 4-7, 9-4

ServerAddressOverride policy, 13-4, 14-3

Service
VitalQIP File Generation, 1-7

service messages, 21-15

services

configurations, 1-9

daemons, 2-22

debugging, 20-8

descriptions of, 1-3

distributed services, 1-3

location executables, 25-9

messages, 21-18

policy file, 3-2

policy files, 3-3

reinstalling, 25-9

running on Windows, 2-3

running services on UNIX, 2-17

signal handling, 20-13

startup scripts on UNIX, 2-20

statistics, 20-28

troubleshooting, 20-30

troubleshooting , 20-13

VitalQIP Service Controller, 2-6

session.check_interval property, 18-35

session.timeout property, 18-35

session-config element, 18-65

showIcon. properties, 18-47

showIcon.ABOUT_VITALQIP property, 18-44

showIcon.ACL_TEMPLATE property, 18-43

showIcon.ADMIN_ROLE property, 18-44

showIcon.ADMIN_SECURITY property, 18-44

showIcon.ADMINISTRATOR property, 18-44

showIcon.APPLIANCE_MANAGER property, 18-44

showIcon.ATTRIBUTES property, 18-44

showIcon.AUTO_DISCOVERY property, 18-44

showIcon.BLOCK_HIERARCHY property, 18-44

showIcon.CUSTOMIZED_ADD_ONS property, 18-44

showIcon.DHCP_CLIENT_CLASS property, 18-44

showIcon.DHCP_SERVER property, 18-44

showIcon.DHCP_TEMPLATE property, 18-44

showIcon.DNS_GENERATION property, 18-43

showIcon.DNS_SERVER property, 18-43

showIcon.DOMAIN property, 18-43

showIcon.GLOBAL_POLICIES property, 18-44

showIcon.INTERNET_REG_HIERARCHY property, 18-44

showIcon.IPV4_MANAGEMENT property, 18-43

showIcon.IPV6_MANAGEMENT property, 18-43

showIcon.MANAGED_FILES property, 18-44

showIcon.MY_VIEW property, 18-43

showIcon.MY_VIEW_MANAGEMENT property, 18-44

showIcon.MY_VIEW_PERSONALIZATION property, 18-43

showIcon.NODE_MANAGEMENT property, 18-43

showIcon.NON_MANAGED_DNS_SERVER property, 18-43

showIcon.OBJECT_CLASSES property, 18-44

showIcon.ONLINE_DOCUMENT property, 18-44

showIcon.ONLINE_HELP property, 18-44
showIcon.ORGANIZATION property, 18-44
showIcon.POOL_HIERARCHY property, 18-44
showIcon.REPORTS property, 18-44
showIcon.RULE_HIERARCHY property, 18-44
showIcon.SCHEDULER property, 18-44
showIcon.SWITCH_ORGANIZATION property, 18-44
showIcon.TEMPLATE_HIERARCHY property, 18-44
SiAddr policy, 26-18
signal handling, 20-13
signal types
  Active Lease Service, 20-15
  DNS Update Service, 20-15
  Login Service, 20-14
  Message Service, 20-14
  MS DNS Update Service, 20-17
  QIP Update Service, 20-14
  Remote Service, 20-16
  RMI QAPI Service, 20-17
  RMI Scheduler Service, 20-16
  Schedule Service, 20-13
SleepTime policy, 11-3
Soap service
  configuration, 18-23
soapUrl policy, 17-3, 17-5
soapWssePassPolicy, 17-3
SSL Tunnel Service, 23-8
start services
  enterprise server, 19-29
  remote server, 19-31
StartTLS policy, 22-34
statistics
  DNS Update Service, 20-29
  Login Service, 20-28
  Message Service, 20-28
  QIP Update Service, 20-28
  Schedule Service, 20-28
StatsLogPeriod policy, 7-7
stop services
  enterprise server, 19-27
  remote server, 19-30
strftime, 3-46
StripTrailingDots policy, 10-4
subnetting, 27-3
SuccessLogLevel callout policy, 22-8, 22-11
Sybase
  administration, 19-1
  administrative tasks with qip-util, 19-11
  back up of, 19-4
  backing up the VitalQIP Database & Transaction Log, 19-7
  backup server, 19-6
  changing the Backup Medium, 19-8
  changing the procedure cache size & total memory size, 19-8
  create master database dump device, 19-8
  creating Database Dump Device for hard disk, 19-7
  creating Transaction Log Dump Device for tape, 19-7
  database recovery, 19-20
  dump devices, 19-6
  encrypting password, 19-9
  index statistics maintenance, 19-25
  managing data space, 19-16
  managing transaction log space, 19-18
  manual backup, 19-6
  re-initializing, 19-9
  starting on Windows via Sybase Central, 19-3
  starting Sybase on UNIX, 19-3
  stored procedures and triggers, 19-15
  transaction log, 19-6
  transaction log dump devices, 19-7
  truncate audit data, 19-24
  version of, 19-3
SyncBindingBufSize policy, 26-12, 26-13
SyncBindings policy, 26-13
SyncBindRetryMax policy, 26-11, 26-13
SyncFailCritical policy, 26-14
SyncReqRetryMax policy, 26-14
syslog
  file description, 20-17
syslog file
  statistics, 20-28
System Logs, 20-5

T

technical support, 20-2
ThreadPoolCount policy, 6-6
time-out
  configure session length, 18-65
TLS_REQCERT directive, 22-23
Tombstoned Objects, 24-65
Tombstoned Resource Records, 24-65
Tomcat server
  configuration, 18-22

start, 2-16, 2-22
stop, 2-16, 2-22
transaction log, 19-6
dump, 19-18
transaction log space
enlarge, 19-18
troubleshooting
debugging, 20-8
DHCP, 27-1
DHCP problems, 27-1
DNS, 25-1
environment variables, 20-4
exportation of data, 20-12
importation of data, 20-12
logging in, 20-10
reinstalling services, 25-9
server failures, 20-11
services/daemons, 20-13
statistics, 20-28
syslog, 20-17
system logs, 20-5
technical support, 20-2
VitalQIP problems, 20-1
trunc database option, 19-20

U
UDA callout
create, 18-75
UDA callouts, 18-71
udaApplicationContext.xml, 18-71
UdaPostEditCallout.java interface, 18-71
UdaPreEditCallout.java interface, 18-71
UdaValidationCallout.java interface, 18-72
UDPBufferSize policy, 7-6
UidAttr policy, 22-33
Unique Hostname Resolution Options, 26-21
UpdateDelay policy, 6-5
UpdateDNS policy, 4-4
UpdatePassword policy, 4-7
UpdateQueueLength policy, 6-5
UpdateRetry policy, 6-4
Use DNS Update Service global policy, 24-44
Use Failover Server policy, 26-12
useFileIO property, 18-42
UseFreezeThawUpdate policy, 12-7
userexit
DNSSEC, 24-57
UserExitErrorFile policy, 13-4
Using the Backup Server, 19-6

V
Validation Callout, 18-70
vercheck, 19-3, 20-4
VitalQIP
Automatic Synchronization, 20-12
components, 1-2
daemons, 2-22
installation configurations, 1-12
policy file, 3-2
running services on UNIX, 2-17
running services on Windows, 2-3
server definition, 1-2
service configurations, 1-9
service messages, 21-15
support of BIND 9.x, 24-3
support of external objects & resource records, 24-61
support of failover server, 26-9
support of secure dynamic updates, 24-20
support of Windows 2000 DHCP, 26-2
support of Windows 2000 DNS, 24-6
VitalQIP Active Lease Service daemon parameters, 2-28
description of, 1-6
policies, 10-2
signal types, 20-15
starting, 20-32
statistics, 20-29
VitalQIP client
debugging, 20-8
definition of, 1-2
installation configurations, 1-12
version, 20-4
VitalQIP DNS Update Service daemon parameters, 2-27
description of, 1-6
message types, 20-23
policies, 5-2
statistics, 20-29
VitalQIP enterprise server
definition of, 1-2
installation configurations, 1-12
synchronizing servers, 20-11
VitalQIP File Generation Service, 1-7
VitalQIP Login Service daemon parameters, 2-26
description of, 1-6
moving, 8-2
policies, 8-2
statistics, 20-28

VitalQIP Message Service
daemon parameters, 2-25
description of, 1-4
MessageQueue, 1-5
MessageRoute, 1-5
policies, 7-3
starting, 20-32
statistics, 20-28
VitalQIP Microsoft DHCP Monitor Service
description of, 1-6
policies, 11-2
Windows 2000, 26-5
VitalQIP MS DNS Update
policies, 3-3
VitalQIP QIP Update Service
daemon parameters, 2-24
description of, 1-4
policies, 4-2
starting, 20-33
statistics, 20-28
VitalQIP remote server
definition of, 1-2
installation configurations, 1-12
VitalQIP Remote Service
daemon parameters, 2-29, 2-32
description of, 1-7
messages types, 20-26
signal types, 20-16
starting, 20-32
VitalQIP RMI QAPI Service
daemon parameters, 2-31
description of, 1-8
message types, 20-27
policies, 13-2
signal types, 20-17
statistics, 20-29
VitalQIP RMI Scheduler Service
daemon parameters, 2-30
description of, 1-8
message types, 20-27
policies, 14-2
statistics, 20-29
VitalQIP Schedule Service
daemon parameters, 2-23
description of, 1-4
policies, 9-2
starting, 20-31
statistics, 20-28
VitalQIP Service Controller, 2-6
configuring, 2-8
Event Viewer Tab, 2-13
Filtered Services, 2-15
Lucent DNS configuration option, 2-11
running services, 2-6
VitalQIP Service Layer policies, 17-2
VitalQIP services
descriptions, 1-3
distributed services, 1-3
messages, 21-18
policy files, 3-3
signal handling, 20-13
statistics, 20-28
troubleshooting, 20-30
VitalQIP SSL Tunnel Service, 15-2
VitalQIP web client interface
definition of, 1-2
installation configurations, 1-12
VitalQIP Web GUI policies, 17-4

W

WaitCtlRetSecs policy, 26-12
WaitPollRplSecs policy, 26-14
WaitSyncBindAckSecs policy, 26-14
WaitSyncBindAckSecspolicy, 26-12
WaitSyncBindUpdateSecspolicy, 26-12, 26-14
WaitSyncStartSecs policy, 26-14
web client
definition of, 1-2
installation configurations, 1-12
logging, 18-24
Web Service
7.0/7.1 version compatibility, 18-69
configuration, 18-23
log refresh interval, 18-23
web.xml, 18-65
Windows 2000 DHCP
active lease information, 26-6
CLIs, 26-7
configuring, 26-2
importing data, 26-3
limitations of, 26-3
multicast addressing, 26-7
support of, 26-2
updating VitalQIP, 26-6
Windows 2000 DNS
configuring sercure zones, 24-12
secure zones, 24-10
VitalQIP support of, 24-6
Windows 2000 DNS Server, 24-6
configuring in VitalQIP, 24-7
server type, 24-10
zone information, 24-6
Windows 2003 DNS
SRV records, 25-9
Windows DNS 2000
setting secure zone policies, 24-16

WINS
configuring Lucent DNS, 24-6

X xml
DNSSEC, 24-57, 24-58

Z zones
DNSSEC, 24-59
zoneSearchPageSizes, 18-40, 18-41