RoamServer 6.1.0 Unix Admin Guide

VERSION 1.0, NOVEMBER 2015

Corporate Headquarters
iPass Inc.
3800 Bridge Parkway
Redwood Shores, CA 94065 USA

www.ipass.com
+1 650-232-4100
+1 650-232-0227 fx

TABLE OF CONTENTS

TABLE OF CONTENTS

TABLE OF CONTENTS

Introduction

The RoamServer 6.1.0 for Unix Administrator Guide provides instructions for installing RoamServer 6.1.0 for UNIX. It also includes instructions on how to configure RoamServer to use UNIX, SITE, RADIUS, LDAP, or TACACS as an authentication protocol.

These instructions sometimes refer to the directory called **<RS_Home>**. This is the directory in which RoamServer is installed. The default for RoamServer 6.1.0 is **/usr/ipass/roamserver/6.1.0**.

System Requirements

Platforms

- Red Hat Enterprise Linux (RHEL 6.5), 64-bit (Kernel Version 2.6.32-431.el6.x86_64)
- Red Hat Enterprise Linux (RHEL 6.1), 64-Bit (Kernel Version 2.6.32-131.0.15.el6.x86_64)
- Solaris Sparc 5.10 Generic_147147-26 sun4vsparc SUNW, SPARC-Enterprise-T5220
- Ubuntu 14.04.1, 64-bit (Kernel Version- 3.13.0-39-generic)

> *If RoamServer 6.1.0 32-Bit installer needs to be installed on 64-bit Ubuntu machine then following libraries should be pre-installed: libc6-i386, lib32gcc1, lib32z1, lib32stdc++6, and ia32-libs. You can install these libraries by running: apt-get install libc6-i386 lib32gcc1 lib32z1 lib32stdc++6 ia32-libs.*

Server Requirements

- 512 MB to 1 GB RAM (the RoamServer process requires 256 MB of RAM) 250 MB temporary disk space
- 250 MB permanent disk space
- Root access is required (for installation and uninstallation)
- The server must have a static IP address and hostname (no DHCP)
- If installed behind a firewall, an accessible NAT IP address is required
- Installer must have administrative permissions on the host

Additional Requirements

- **Redundancy:** RoamServer must be installed on at least two separate host machines, to insure the iPass redundancy model is enabled. No iPass service guarantees apply without having failover configured between at least two RoamServer hosts.
- **Connectivity:** Connectivity to an authentication database is required.
- **iPass Transaction Centers**: iPass Transaction Centers must be able to communicate with RoamServer on port 577 (or the port you configured). Please refer to the list of iPass Transaction Centers here: http://help.ipass.com/doku.php?id=required_configurations_for_open_mobile_access#roamserver1

Default Port

The default RoamServer port is 577 (unless RoamServer is run by a user without root access in which case the port must be higher than 1024).

Internet Protocol version 4 (IPv4)

RoamServer supports IP addresses in the IPv4 format.

Installation

Prerequisites

Before installing RoamServer, you will need the following:

- Administrator rights on the RoamServer host
- Your iPass Customer ID
- Your host's private and public IP addresses
- The port number on which RoamServer will listen (defaults to 577)
- The host's operating system, including kernel and version number

General Process

To install RoamServer:

1. Download the installation file.
2. Install the software.
3. Set initial configuration and certify RoamServer.
4. Configure RoamServer to communicate with your authentication servers, and if desired, accounting servers.
5. Set any advanced options, such as:
   - Policy File
   - Secondary Servers for Failover
   - Log Files
6. Set additional properties in the ipassRS.properties file, if necessary.
7. Test the installation.
8. Repeat steps 2-7, as needed to install RoamServer on additional servers, and configure failover.

Installing Behind a Firewall

iPass recommends that you install RoamServer behind a firewall. If you choose to do so, you will need to allow TCP traffic to the external IP of RoamServer on port 577 (unless RoamServer is run by a user without root access in which case the port must be higher than 1024) through to RoamServer. The internet-facing IP must be registered with iPass. You may restrict traffic on that port to incoming packets only from the IP addresses of the iPass Transaction Centers. Please refer to the list of iPass Transaction Centers here:
http://help.ipass.com/doku.php?id=required_configurations_for_open_mobile_access#roamserver1
If your firewall is performing Network Address Translation (NAT), you will need to provide the IP address of your firewall to your iPass Installation Engineer.

Downloading Installer

Before installing, you will need to download the installation file from the iPass FTP site, ftp.ipass.com.

To download the installation file using FTP:

1. FTP to **ftp.ipass.com**.

2. Enter the following:
   - **username:** roamserver
   - **password:** pass2roAm
3. To change to binary mode, type: **bin**.
4. To obtain a complete listing of directory contents, type: **ls**.
5. To change to the directory containing the software for your platform and region, type: **CD**. Remember that directory names and filenames are case-sensitive.
6. After locating the file appropriate to your platform and region, type: **get <filename>**
7. To exit the ftp application, type: **bye**
8. Once the file is downloaded, copy it to a tmp folder. To do so, type: **cp <build server>/<filename> <tmp>**

Example

roamserver_6.1.0_linux-x86.tar.gz is the **<filename>** of an installer for a Linux 32-bit machine.

Installing RoamServer

To install RoamServer:

1. Log in to the machine as a root user.
2. Type: **cd /tmp/<roamserver_6.1.0_linux-x86.tar.gz>**
3. Type: **chmod +x /tmp/<roamserver_6.1.0_linux-x86.tar.gz>**
4. Type: **tar zxvf <roamserver_6.1.0_linux-x86.tar.gz>**

   > *Please use gtar command for extracting the build in solaris sparc machine from the path (/opt/csw/bin/gtar zxvf roamserver_6.1.0_solaris-sparcv9.tar.gz. If gtar package is not installed then install the gtar package.)*

5. A directory called **roamserver_installer** will be created in **/tmp**
6. Type: **cd roamserver_installer**
7. Execute: **./install.sh**
8. Review and approve the **End User License Agreement**.
9. Enter an absolute path, or press **<Enter>** to accept the default **[/usr/ipass]**
   This will create a hierarchy in **/usr/ipass/roamserver/6.1.0** with all the necessary directories and files. In order for RoamServer to run correctly, you must keep the file hierarchy as it is installed. However, RoamServer can be installed in any location.
10. Continue on to configuration and certification (see the Setup section.)

Non-Root User

To run RoamServer as a non-root user:

1. Set the **curr_user=<user>** in the file **roamserverd (/usr/ipass/roamserver/6.1.0/bin/)** with the <user> who will run RoamServer. By default, the user is: **root**. (Make sure you save the file and that the configured user exists.)
2. Change ownership to the user set in the previous step.
3. Set **ipass_usr=<user>** in the **chg_owner.sh script (/usr/ipass/roamserver/6.1.0/.scripts/)** and execute it.

Uninstalling RoamServer

To uninstall RoamServer:

1. Go to **<RS_Home>/UninstallerData**
2. Execute uninstaller.sh as **./uninstaller.sh** and follow the prompts to uninstall.
   You may also need to manually delete any leftover files in the **<RS_Home>** folder that were not created by the installer.

Updating to RoamServer 6.1.0

Migration Tool

If you are upgrading from RoamServer 5.x to RoamServer 6.1.0, you must run the Migration Tool manually. The Migration Tool will migrate the RoamServer 5.x configuration files, the RoamServer 5.x certificate, and the key into the keystore.

Migrating from RoamServer 5.x/6.x to 6.1.0:

To run the migration tool:

1. Go to **<RS_Home>/bin** directory and execute **rs_migration_tool.csh script** without any arguments.
2. When prompted for the path to migrate the files from, enter: **RS 5.x path /usr/ipass/roamserver/5.x**
3. Follow the instructions that appear.

The following items will migrate:

- **KeyStore** is added to ipassRS.properties.
- **AppSharedKey** is added to ipassRS.properties (This is an optional attribute. More information in the Property Glossary.)
- All .**pem files** are migrated to rs.keystore inside <RS_Home>/certs directory.
- It will migrate the **policy.txt file** and **ipassLDAP.properties** (if available in RoamServer 5.x).

ACA Support

RoamServer by default supports ACA with Active Directory. For LDAP, the LdapUacAttr property must be configured in the LDAP properties file.

Configuring LdapUacAttr

LdapUacAttr=<UserAccountControl attribute name>=<active users UserAccountControl Attribute values>
Example:

LdapUacAttr=ipassStatus=active

For details about LdapUacAttr attribute, refer to section ipassLDAP.properties.

AuthServer as LDAP

If your organization is using LDAP for authentication, the following configuration needs to be changed in in ipassRS.properties after migration:

- **LdapConfigFile** path must change to **6.1.0** path explicitly in the AuthServer attribute

RADIUS Attributes

When upgrading to RoamServer 6.1.0 and using RADIUS authentication, check your RADIUS logs to verify your RFC attributes. If an attribute is not shown in the tables in Appendix II on page 50, then you need to re-configure your RADIUS to eliminate the attribute.

.

Setup

Before running RoamServer for the first time, you need to perform certain initial setup tasks and receive and install a digital certificate from iPass.

Setting Values in ipassRS.properties

By setting properties in the file `ipassRS.properties`, you can enable or disable RoamServer functions. Some properties are turned on by default, and it is necessary to change the value of the property in order to disable a feature. (Enabling some features may involve setting more than one attribute).

You can edit the file and add, change or delete properties in two ways:

- You can run **ipassconfig.csh** in your **<RS_Home>/bin** directory. This is the recommended method and is explained in detail in the next section.
- You can also use a text editor to make changes. To set a new property value using a text editor, open the file and type in the name and value of a new attribute. (However, we strongly recommend use of the **ipassconfig.csh** script whenever possible, to ensure correct naming and formatting of property names and values.)
  - Properties are set in the following format: **<property name>=<value>**
  - Property names are case-sensitive, while property values are not. Valid values for Boolean properties are: true, false, yes, no, y, n.

See the Property Glossary for a complete list of configuration options in ipassRS.properties.

Running ipassconfig.csh

Configuration tasks can be performed quickly and easily by running a script called **ipassconfig.csh**, located in the **<RS_Home>/bin** directory, which can be used to set properties in the ipassRS.properties file.

To run ipassconfig.csh:

1. In your **<RS_Home>/bin** directory, type **ipassconfig.csh -conf**
2. The values in square brackets [] are default values. To enter a default value, press **ENTER**.

Multiple instances of **ipassconfig.csh** are not recommended. You should only run a single instance of the script at any one time, as simultaneous instances can overwrite each other's results.

Adding, Editing or Deleting Properties

You can rerun the script after initial configuration to add, edit, or delete properties, as needed. If you rerun it, the script will read values from the existing ipassRS.properties, so you will not have to re-enter those values.

For instance, two months after you install RoamServer, you decide to add a secondary authentication server. Run the **ipassconfig.csh -conf**, skip all the questions not having to do with authentication servers by pressing <Enter>, and only enter the configuration information for the new authentication server when the script requests this information.

Initial RoamServer Configuration

Initial configuration is done by running the **ipassconfig.csh** script, which sets many of the properties in your ipassRS.properties file. After setting the properties, you must then request a certificate from iPass, and install it on your server host. Finally, you must configure the server for auto-restart.

Basic Server Settings

To configure your basic server settings:

1. In the **<RS_Home>/bin** directory, run **ipassconfig.csh -conf**. Supply the requested information as outlined here. For each script entry, the value shown in square brackets [] is the default. Where applicable, you can press **Enter** to use default values for the information.
2. **Time and Date Verification:** (Default Value=YES.) The date/time stamp must be correct and correspond with the information in the iPass database in order to validate the certificate.
3. **Customer ID:** (Default Value=0) Enter your customer ID, supplied by iPass.
4. **Policy File:** (Default Value=No) If you want to use a Policy File to allow or deny users access, enter Yes.
5. **Debug Level:** (Default Value=0): Debug level determines how debugging and error messages are logged to a trace file. Debug level can be any value from 0 to 5, with 0 generating only critical error messages and 5 generating the most detailed and extensive amount of information. Production servers should normally be run with a debug level set to 0.
6. **Port:** (Default Value=577) Enter the RoamServer listening port as 577. If you want to allow users to run RoamServer without root access, you have to change this port to a number higher than 1024
7. **Authentication Servers:** (Default Value=no). If you wish to configure your authentication server(s), enter yes. You will need to enter each server's authentication protocol, IP address and other relevant configuration parameters. See Authentication Servers for more information.
8. **Accounting Servers:** (Default Value=no). If you wish to configure your accounting server(s), enter yes. You will need to enter each server's IP address and other relevant configuration parameters. Note that this is not mandatory, since. not all authentication protocols support Accounting records. (e.g. LDAP). See the Accounting Servers section for more information.
9. **KeyStore:** (Default Value=KeyStorePath=$ipass.server.home/certs/rs.keystore). You will need to mandatorily configure KeyStore settings for SSL communications. The tool will prompt with following information required to generate the keystore. Verify the similar console output as given below:

```
The next step is to configure your server KeyStore with an KeyStorePath,
KeyAlias, Salt, KeyPassword, and KeyStorePassword.

KeyStore=KeyStorePath=$ipass.server.home/certs/rs.keystore
 Please enter the
KeyStorePath:[/usr/ipass/roamserver/current_version/certs/rs.keystore]
 Please enter the KeyAlias:[rs]
 Please enter the CertAlias:[ipassca]
 Please enter the Salt:[iPassRS]
 Please enter the KeyPassword:[changeme]
 Please enter the KeyStorePassword:[changeme]
New KeyStore  settings:
KeyStorePath=/usr/ipass/roamserver/current_version/certs/rs.keystore,KeyAlias=r
s,CertAlias=ipassca,Salt=iPassRS,KeyPassword=ZyQ0s/ObjBliu64IJGbCNw==,KeyStoreP
assword=ZyQ0s/ObjBliu64IJGbCNw==
```

10. **Private key and Certificate Signing Request for KeyStore**

11. : Enter the information needed to generate your SSL certificate, including:
    - Enter key size (multiples of 512) (range between 1024 and 16384):[2048]
    - 2-character Country Code: (Default Value=US)
    - State or Province Name: (Default Value=Some-State)
    - City or Town Name: (Default Value=Some-City)
    - Company or Organization Name: (Default Value=Some-Organization)
    - Public IP Address of the RoamServer Host: (Default Value=<Local IP>). This must be the public or external IP address, and may differ from the IP address you entered previously. The IP address will not be stored by RoamServer but will be used to generate your public key certificate. If you are using NAT (Network Address Translation), please supply this external address to your iPass installation engineer as well.
    - Fully Qualified Domain Name of the RoamServer Host: (Default Value=N/A).
    - The domain name will not be stored by RoamServer but will be used to generate your public key certificate.
    - Your E-mail Address: (Default Value=N/A).

Certificate Request

After entering your basic server information, you must submit a request for a signed certificate. The x509 certificate will allow SSL 128-bit encrypted communication between the iPass transaction server and the RoamServer.

To submit your certification request:

1. Log in to the iPass Portal and open a Support Ticket requesting a signed RoamServer certificate.
2. Attach the **??? mail_cert_req.data** file to the ticket.

To finish the certification process:

1. iPass will ftp the signed certificate and then send you download instructions.
2. 

3. Save the downloaded certificate as **isp_cert.pem under <RS_Home>/certs** folder.
4. Import the certificate to **KeyStore<RS_Home>/certs/rs.keystore** using this command:
    - Run the script: **load_RS_keystore.csh in <RS_Home>/bin**

List RoamServer Keystore

To list your keystore:

1. In <RS_Home>/bin, run the script: **list_RS_keystore.csh**

Usage:

- list_RS_keystore.csh
- list_RS_keystore.csh <KeyStoreFilePath>
- list_RS_keystore.csh <KeyStoreFilePath> <Password>

For example:

- list_RS_keystore.csh ../certs/rs.keystore changeme

This will list the keystore. Keystore alias entries in keystore should be valid/non-expired in order to start RoamServer.

Automatic Restarts

Finally, you must configure RoamServer to restart automatically, in case the RoamServer host cycles through power failure or reboots. Run the below scripts once the RoamServer is started successfully.

To set up automatic restarts:

1. Log in as a user with root access.
2. Edit the **roamserverd** file located under **<RS_Home>/bin**.
3. Set the user that is running RoamServer (the default is root) in the **curr_usr** variable.
4. Make sure that this user has execution permission and ownership of the RoamServer folder:
   - Type: **ls -al <RS_Home>**
   - Make sure that the permission is 744 and ownership is to the user.
5. Add the RoamServer RC scripts into the run level by running the script setup_init.sh:
   - Type: **cd /<RS_Home>/bin**
   - Type: **./setup_init.sh**

To test automatic restarts:

1. Type the command: **reboot**
2. After you have waited a little while, log in again and grep the RoamServer process:
   - Type: **ps -ef | grep roamserver**
3. Make sure that RoamServer is started with the user set in **curr_user** variable (the default is root).

To remove automatic restarts (if necessary):

1. If you find it necessary to remove the automatic restart scripts:
   - Type: **cd <RS_Home>/bin**
   - Type: **./setup_remove.sh**

To set up ipasskeepalive monitoring:

If the RoamServer process is not running the **ipasskeepalive.sh** script will start the server process automatically based on the time interval added in the crontab entry (In step1step2)

1. Log in as a user with root access.
2. Add the RoamServer **ipasskeepalive** monitoring scripts by running the script setup_cron.sh:
   - Type: **cd /<RS_Home>/bin**
   - Type: **./setup_cron.sh**

To remove ipasskeepalive monitoring:

1. If you find it necessary to remove the automatic restart scripts:
   - Type: **cd <RS_Home>/bin**
   - Type: **./remove_cron.sh**

To test ipasskeepalive monitoring restarts:

1. Type the command: **crontab –l**

0,5,10,15,20,25,30,35,40,45,50,55 * * * * '/usr/ipass/roamserver/current_version'/bin/ipasskeepalive.sh

2. Grep the RoamServer process:
   - Type: **ps -ef | grep roamserver**

> *The RoamServer process can also be started by executing **ipasskeepalive.sh** script from the <RS_Home> /bin.*

Testing

There are two tests that should be performed following every installation and configuration of RoamServer to ensure proper functionality:

- checkipass Tool
- RoamServer Test Tool

When testing RoamServer, it is recommended that you perform all of these tests in the order that they are presented here.

Checkipass Tool

The checkipass test is a simulated request from RoamServer. to the AAA server, which stays local to your network. To test RoamServer using the checkipass test, you will need to run the **checkipass.csh** test tool as an administrator. This test verifies that RoamServer can authenticate a local user by communicating with the AAA server. This procedure only tests RoamServer. No realm should be prefixed to the user name unless it is required by your AAA, or Route-by-Realm is configured. The authentication request goes from the **checkipass.csh** test tool to RoamServer, then to the AAA server for authentication, and finally back to RoamServer and **checkipass.csh** tool.
checkipass.csh is found in the test subdirectory of your **<RS_Home>** directory. You will need to use a valid user name and password for the host on which RoamServer is installed.

To run checkipass:

1. Run: **./checkipass.csh -u <username>.** To Test ACA Request, add "-attr aca_request=true" argument to the command.
2. Enter the password when prompted.
3. If accounting start and stop status=ack, then RoamServer is properly installed, configured and working, and you may proceed to the next test.
   - Possible causes for a Reject here include:
     - **Invalid user name or password:** The user in this test must have local login privileges to that system or should be authorized in the AAA server.
     - **Invalid certificate:** If the certificate is corrupt, then it will need to be replaced.
     - **Improper configuration:** Verify that you have correctly entered all the information in the setup program.
     - **Invalid shared secret** (for RADIUS users): Verify that your shared secret is entered properly. A shared secret cannot contain the comma (,) or equals sign (=) characters.

RoamServer Test Tool

The RoamServer Test Tool extends the verification performed in the checkipass test by sending a authentication request across the iPass network. In this test:

- An authentication request is generated by the tool and sent directly to an iPass Transaction Server.
- The Transaction Server resolves the domain to your account from the iPass database and forwards this authentication request to chosen RoamServer(s) on port 577 (unless RoamServer is run by a user without root access in which case the port will be higher than 1024).
- RoamServer receives the request, and forwards the request to your AAA server.

Upon successful authentication, the request is relayed using SSL encryption back to the RoamServer Test Tool. This test tool is a Web-based tool, available from openmobile.ipass.com.

To run the Test tool:

1. Log in to openmobile.ipass.com.
2. Under the **Tools** tab, select **RoamServer Test Tool**.
3. Choose the type of test you want to perform.
4. **Enter username:** enter a username that you know works and include the domain (i.e. @domain.com). (This is a required step).
5. **Enter password:** enter the password for the username you entered. (This is a required step).
6. **Choose RoamServer:** select DEFAULT to test your primary RoamServer or select another RoamServer (by IP address) from the drop-down list (if available).
7. **Class of Service:** select the class of service from the drop-down list.
8. Click **Test Authentication**.
9. In addition to performing this test with a valid user name and password, you should also run the test with invalid credentials to ensure that the authentication attempt will be rejected.

Authentication Servers

This section discusses configuring RoamServer to communicate with your authentication servers. These instructions assume that you are installing RoamServer behind your firewall or on the same host as your AAA server. If you are installing RoamServer outside your firewall or on the firewall server, you may need to modify some of these settings. Consult with your iPass RoamServer Installation Engineer for assistance.

UNIX and SITE Authentication

If you would like RoamServer to authenticate using your UNIX system's password file, the type of authentication protocol you choose will be based on the type of passwords used.

> *If your system does not use shadow passwords, UNIX authentication should be used.*
> *If your system uses shadow passwords, SITE authentication should be used instead.*

To enable UNIX authentication:

1. Run: **ipassconfig.csh -conf**
2. When the script requests authentication server information, enter **UNIX**.

To enable SITE authentication:

1. Run: **ipassconfig.csh -conf**
2. When the script requests authentication server information, enter **Site**.
3. For Site File, enter the name of the password file (typically, this is **/etc/shadow**).

RADIUS Authentication

RoamServer will format the request as a standard RADIUS request and forward it to the RADIUS daemon at the address and port number specified during the installation. Additionally, you must make the RADIUS encryption key (shared secret) available to RoamServer. RoamServer uses this shared secret to encrypt the RADIUS packet contents before sending them to the RADIUS server. The RADIUS server then uses the shared secret to decrypt the packet contents. (A shared secret cannot contain the comma (,) or equals sign (=) characters.)

> *Your system must have a static, routable IP address, and cannot be blocked by a firewall.*

To configure RoamServer for RADIUS authentication:

1. Run **ipassconfig.csh (with option '- conf')**. Enter radius as an authentication protocol and enter:
   - the server's IP address [127.0.0.1]
   - port number [1812]
   - RADIUS shared secret [mysecret]
   - attempts [3]
   - idle timeout in milliseconds[5000]
   - if the prefix should be included [N]
   - if the domain should be included [N]

2. Verify that RoamServer is entered as a client of your RADIUS. You will need to edit the appropriate configuration file on your RADIUS server by adding the IP address of the RoamServer, and the corresponding shared secret, that you entered above.
3. If you make any changes to your RADIUS, you will have to restart it to make sure the changes take effect.
4. Restart RoamServer. RoamServer will now be able to authenticate against your RADIUS Server.

RoamServer can contain the IP address of more than one RADIUS authentication or accounting Server for failover purposes. For more information, see the Failover section.

LDAP Authentication

RoamServer can forward authentication requests to an LDAP server running on the network. RoamServer will format the request as a standard LDAP request and forward it to the LDAP daemon at the address and port number that is specified during the installation. Additionally, you must configure/customize how RoamServer will perform authentication at the LDAP server. LDAP-specific configuration is set in a file called ipassLDAP.properties. For more information, refer to ipassLDAP.properties on page 32, and the ipassLDAP.properties.example file included in the RoamServer package.

To configure RoamServer for LDAP authentication:

1. Open the **<RS_Home>/ipassLDAP.properties** file.
2. Run **ipassconfig.csh (with option '- conf').** Enter LDAP as an authentication protocol and enter:
   - The server's IP address [127.0.0.1]
   - LDAP configuration file name [/usr/ipass/roamserver/6.1.0/ipassLDAP.properties]
   - Port number [389]
   - Idle timeout in milliseconds [10000]
   - Enable SSL? [N] Enter Yes to support LDAP over SSL connections. (See Secure LDAP on page 15.)
3. Customize the contents of the ipassLDAP.properties file as needed.
4. Save and exit the file.
5. Restart RoamServer. RoamServer will now authenticate against your LDAP server.

RoamServer can contain the IP address of more than one LDAP Authentication Server for failover purposes. For more information, see the **Failover** section.

Secure LDAP

RoamServer can support LDAP over SSL connections. Server-side authentication is performed in the SSL handshake. This is done at the OS level. If enabled, will only require a list of certification authority (CA) certificates for validating the LDAP server. SSL is commonly done over port 636.

By default, most secure LDAP servers allow client authentication in the SSL handshake but do not require it. To perform only server authentication, RoamServer must have the CA certificate loaded into its JRE default keystore using the **import_CA_certificate** script.

- **To list all certificates**, run list_CA_certificates.
- **To import additional CA certificates**, run import_CA_certificate <cert-alias-name> <cert-file-name>.
- **To delete a certificate**, run delete_CA_certificate <cert-alias-name>.

TACACS Authentication

RoamServer can forward authentication requests to a TACACS server running on the network. RoamServer will format the request as a standard TACACS request and forward it to the TACACS daemon at the address and port number that is configured during the installation.. Additionally, you must make the TACACS shared secret available to RoamServer. The shared secret is configured in the TACACS configuration file called **tac_plus.conf**. RoamServer uses this shared secret to encrypt the TACACS packet contents before sending them to the TACACS server. The TACACS server then uses the shared secret to decrypt the packet contents. Refer to your TACACS documentation for more information on the **tac_plus.conf** file and shared secret. The TACACS server can be located anywhere with a routable, static IP address, including on the same host as the RoamServer.

If the TACACS server is running on an alternative host on your network (that is, not on the server running RoamServer), you will need to install a copy of the **tac_plus.conf** file on that server or on a network-addressable drive available to that server. You will also need to configure this file location in the RoamServer setup.

To configure RoamServer for TACACS authentication:

1. Run **ipassconfig.csh -conf**. Enter **tacacs** as an authentication protocol and enter:
   - the server's IP address [127.0.0.1]
   - port number [49]
   - TACACS Shared Secret [mysecret]
   - idle timeout [10000]
2. Verify the settings in the configuration file for your TACACS server. You may need to edit the appropriate configuration file within your TACACS software by adding the IP address of the RoamServer.
3. If you make any changes to your TACACS, you will have to restart it to make sure the changes take effect.
4. Restart RoamServer. RoamServer will now be able to authenticate against your TACACS server.

RoamServer can contain the IP address of more than one TACACS authentication or accounting Server for failover purposes. For more information, see the Failover section.

Accounting Servers

Accounting Log File Configuration

RoamServer can be configured to write accounting information to a log file. The log file rotation and backup process can be customized to suit your networking environment and business needs. Depending on the type of AAA used, RoamServer can use either local accounting logging or remote accounting logging.

Local Accounting

For authentication protocols that do not have a built-in remote accounting server (UNIX, SITE, and LDAP), RoamServer can be configured to keep detailed local accounting records (AcctFile) at a location specified by the user. For authentication protocols which have a remote server capable of handling accounting transactions (RADIUS, TACACS), RoamServer can forward the accounting record to the remote server for logging.

To configure RoamServer to log in to a local accounting file:

1. Run: **ipassconfig.csh -conf**.
2. Enter **Yes** when the following prompted appears **Do you wish to add a new AcctServer?**
3. If you wish to log accounting records to a local file, for Protocol, enter **AcctFile.**
4. Enter the path and name of your accounting file, or press **Enter** to use the default path.
5. After running the script, restart RoamServer.

Remote Accounting (RADIUS and TACACS users)

Customers who have a remote server capable of handling accounting transactions (RADIUS or TACACS) can forward the records to the remote server for logging,

To configure RoamServer to forward accounting records to your remote AAA server:

1. Run: **ipassconfig.csh –conf**
2. Enter **Yes** when the following prompted appears **Do you wish to add a new AcctServer?**
3. For Protocol, enter **RADIUS** or **TACACS** as appropriate.
4. Enter the details of the AAA server, as requested.
5. After running the script, restart RoamServer.

If a remote accounting server (RADIUS or TACACS) is unreachable for some reason, accounting data that was supposed to be forwarded to it can be stored in a local file until the remote server is reachable again. The data is stored in binary format in a file called **<RS_Home>/logs/failedAcct**.

If the files are not needed, they can be deleted and remote accounting can be turned off.

Resend Data

To resend the data, run the script **resendacct.csh** from **<RS_Home>/bin** folder. This forwards the **failedAcct** file to the AAA server and then deletes the file.

This task can be automated by adding it to the crontab:

1. Use **crontab -e** to edit the crontab file and add the line:

```
0 3 * * * cd /usr/ipass/roamserver/current_version/bin: ./resendacct.csh.
```

2. View the **crontab** file using **crontab –l**.

Running RoamServer

Runtime Commands

The RoamServer process is named **roamserverd**.

Starting RoamServer

In the **<RS_Home>/bin** directory, run: **roamserverd start**

Some systems will shut down all processes started by a user when that user logs off. If this is the case, run: **nohup roamserverd start**

Shutting Down

In the **<RS_Home>/bin** directory, run: **roamserverd stop**

Restarting RoamServer

Whenever the configuration is modified, RoamServer has to be restarted.

To restart Roamserver, in the **<RS_Home>/bin** directory, run: **roamserverd** restart.

rs_command

You can also perform many runtime functions by using the tool **rs_command.csh, in the <RS_Home>/bin** directory.

Usage: rs_command.csh <command options>.

Command Options

`-host <IP address>`	Specifies the IP address of the machine to send the command to.
`-port <port number>`	Specifies the server port number to send the command to. Default is the local server's listener port (577 unless RoamServer is run by a user without root access in which case the port must be higher than 1024).
`-shutdown`	The server will shut down.
`-restart`	The server will restart.
`-reload_config`	Causes the server to reload many (but not all) of the properties from the ipassRS.properties file. These are: AAA Servers (AuthServer and AcctServer properties) Policy Rules, if feature is enabled. Log Rotation parameters. DebugLevel of server. For a complete reload, you should use the -restart switch.
`dump_queue`	The server will dump the queue elements to a file.
`get <filename> -host <IP ddress> -port <port umber>`	Get a file from a remote RoamServer. Use filename ipassRS.properties to get the RoamServerproperties file. Use filename RS.trace to get the RoamServer trace file. Optionally, use any valid filename relative to the RoamServer home directory.

post<Name=value;Name1=value1> -host <IP address> -port <port number>	To post configuration changes on a remote host. Where Name=Value pairs are the properties settings separated by a semicolon. (;) <IP address> is the IP address of the remote host and <port number> is the port number of the remote host.
post_file <file> -host <IP address> -port <port number>	To post configuration changes on a remote host, where <file> contains the configuration changes to be uploaded to RoamServer, <IP address> is the IP address of the remote host, <port number> is the port number of the remote host.
version	Prints the RoamServer release version.

Scripts Usage Available Under <RS_Home>/bin

rs_get_version.csh	Retrieve roamserver version, build number and JRE version.
config_help.csh	-help <Attribute Name>: Print help/usage for a specific attribute. -list: List the attributes in the server's properties file. -listall : List all of the server's internal attributes.
list_RS_keystore.csh	To list your keystore list_RS_keystore.csh list_RS_keystore.csh <KeyStoreFilePath> list_RS_keystore.csh <KeyStoreFilePath> <Password>
load_RS_keystore.csh	To import the primary signed certificate into rs.keystore
Ipassconfig.csh <options>	<options> -import_cert Use this to import ipass CA certificate and signed primary certificate to a Java keystore. -regen_keystore Use this to Regenerate the keystore and certificate signing request.

Scripts Usage Available Under RS_Home/.scripts

create_link.sh	To create the symbolic link to the RoamServer 6.1.0.
chg_owner.sh	To change the ownership of the roamserver to ipass user.

ipassRS.properties

The ipassRS.properties file allows customization of RoamServer features. By setting properties in the file, you can enable important RoamServer functions. Enabling some features may involve setting more than one property.
Property names are case-sensitive, but property values are not. Valid values for Boolean properties are: true, false, yes, no, y, n.

Property Help

You can obtain help on any property, including those listed here, by using a tool called **config_help.csh**, found in your **<RS_Home>/bin directory**.

- To list all server properties: **config_help.csh -listall**
- To describe usage of a property: **config_help.csh -help <property name>**

Property Glossary

This glossary defines all properties found in ipassRS.properties, including configurable parameters for each property.

Property	Description
`AcctLogBackupType`	`AcctLogBackupType=<backupType>` where <backupType> is either MultipleWithTimestamp or SingleBackup. The default is MultipleWithTimestamp. `AcctLogBackupType` sets the accounting log's backup file name when rotation is to be performed on local accounting files.
`AcctLogRotationDays`	`AcctLogRotationDays=<days>` Valid range is: 1 to 30 days. The default is 7 days. `AcctLogRotationDays` control how often the local accounting file is rotated.
`AcctLogRotationMaxSize`	`AcctLogRotationMaxSize=<max size>` Minimum value is 100 kbytes. Maximum value is 20000 kbytes. The default is 10000 kbytes. `AcctLogRotationMaxSize` limits how large (in kbytes) the local accounting file can get before it is rotated..

`AcctLogRotationType`	`AcctLogRotationType=<rotationType>`
	Where <rotationType> is either FileSize or NumberOfDays.The default is FileSize.
	`AcctLogRotationType` sets the type of rotation to be performed on the local `accounting files.`

`AcctServer`	Provides accounting server information, for example
	`AcctServer1=name11=value11,name12=value12,name13=value13.....`
	`AcctServer2=name21=value21,name22=value22,name23=value23.....`
	AcctServer parameters:
	■ **Protocol**: The server's protocol. Values can be: NT/Radius/LDAP/TACACS\
	■ **EnableSsl**: Flag used to enable/disable SSL connections to the LDAP servers. It is ignored when used for other Acct servers.
	■ **IpAddress**: The server's IP address.
	■ Port: The server's port number.
	■ **LocalIpAddress**: The Local IP address to bind the socket to. (Optional and Only for RADIUS)

	■ Attempts: The number of attempts made to communicate with a server. ■ **IdleTimeout**: Timeout (in milliseconds) to wait for a response from a server for a given communication attempt. ■ **SharedSecret**: The shared secret used by a RADIUS/TACACS+ server. ■ **IncludeDomain**: Include the user's domain in the request sent to the server. ■ **IncludeDomainAsWinPrefix**: Include the user's domain, as Windows style prefix,in the request sent to the server. For example, user@ntdomain would become ntdomain\user ■ **IncludePrefix**: Include the user's routing prefix in the request sent to the server. ■ **IncludeNasPortType**: Include the NAS-Port-Type in the request sent to the RADIUS AAA server. ■ **StripRealm**: Specifies a realm suffix to strip away from the user's domain. For example, with StripRealm=example.com and IncludeDomainAsWinPrefix enabled, the login of user@ntdomain.example.com would become user@ntdomain ■ **NTDomain**: The NT domain used to authenticate window users. ■ **NTRasMode:** The NT RAS mode to use. 1=WINNT RAS mode, 0=WINNT. ■ **SiteFile:** The file used in Site (Unix Shadow file) authentication ■ **LdapConfigFile:** The file used to load LDAP specific properties for an LDAP server. ■ **ValidateAuthenticator**: Specifies in the RADIUS Authenticator should be validated. Values are YES or NO. Default is YES. ■ **ProtocolVersion**: Used by the TACACS+ server to specify the Minor Version.Values are 1 or 0. Default is 1. ■ **EnableLocalAcct:** Used by an AcctFile server to enable/disable local accounting. Values are YES or NO. Default is NO. ■ **RetryDelay**: The time delay, in minutes, before retrying a server that recently failed a connection request. When a connection fails to a server, it is reordered to the end of the list. Once the RetryDelay expires, that server is brought back to the top of the list. The default value is 15 minutes. Valid range is: >=0.
`AppSharedKey`	`AppSharedKey= <Secret key> for encryption and decryption` This entry determines the salt used for encrypting and decrypting the LDAP bind password. The passwords in LDAP property file will not be encrypted if it is not set.
`AuthCacheEnabled`	`AuthCacheEnabled=yes/no.` Determines if the caching of successful authentication requests is enabled. Default is set to YES.

`AscendDataFilter`	`AscendDataFilter1=<valid string for ascend-data-filter>`
	This is used as an anti-Spam feature for some providers and will block the email port (25) at the provider. If the AAA server does not send it to us, we will use the AscendDataFilter(s) specified to send back in the auth accept packet.
	An example entry is:
	`AscendDataFilter1=ip in forward tcp est`
	`AscendDataFilter2=ip in forward dstip xxx.xxx.xxx.xxx/yy`
	`AscendDataFilter3=ip in drop tcp dstport=25`

	`AscendDataFilter4=ip in forward` The string `"ip in drop tcp dstport=25"` is a mandatory AscendDataFilter attribute. When no AscendDataFilter is configured, this feature is disabled. See page 33 for more information.

`AuthServer`	Provides authorization server information, for example:
	`AuthServer1=name11=value11,name12=value12,name13=value13.....`
	`AuthServer2=name21=value21,name22=value22,name23=value23.....`
	AuthServer parameters:
	■ **Protocol**: The server's protocol. Values can be: NT/Radius/LDAP/TACACS
	■ **EnableSsl**: Flag used to enable/disable SSL connections to the LDAP servers. It is ignored when used for other Auth servers.
	■ **IpAddress**: The server's IP address.
	■ **Port**: The server's port number.
	■ **LocalIpAddress**: The Local IP address to bind the socket to. (Optional and Only for RADIUS)
	■ **Attempts**: The number of attempts made to communicate with a server.
	■ **IdleTimeout**: Timeout (in milliseconds) to wait for a response from a server for a given communication attempt
	■ **SharedSecret**: The shared secret used by a RADIUS/TACACS+ server.
	■ **IncludeDomain**: Include the user's domain in the request sent to the server.
	■ **IncludeDomainAsWinPrefix**: Include the user's domain, as Windows style prefix,in the request sent to the server. For example, user@ntdomain would become ntdomain\user
	■ **IncludePrefix**: Include the user's routing prefix in the request sent to the server.
	■ **IncludeNasPortType**: Include the NAS-Port-Type in the request sent to the RADIUS AAA server.
	■ **StripRealm**: Specifies a realm suffix to strip away from the user's domain. For example, with **StripRealm=example.com** and **IncludeDomainAsWinPrefix** enabled, the login of user@ntdomain.example.com would become **user@ntdomain**
	■ **NTDomain:** The NT domain used to authenticate window users.
	■ **NTRasMode:** The NT RAS mode to use. 1=WINNT RAS mode, 0=WINNT.
	■ **SiteFile:** The file used in Site (Unix Shadow file) authentication
	■ **LdapConfigFile**: The file used to load LDAP specific properties for an LDAP Server.
	■ **ValidateAuthenticator**: Specifies in the RADIUS Authenticator should be validated. Values are YES or NO. Default is YES.
	■ **ProtocolVersion**: Used by the TACACS+ server to specify the Minor Version.Values are 1 or 0. Default is 1.
	■ **EnableLocalAcct:** Used by an AcctFile server to enable/disable local accounting. Values are YES or NO. Default is NO.
	■ **RetryDelay:** The time delay, in minutes, before retrying a server that recently failed a connection request. When a connection fails to a server, it is reordered to the end of the list. Once the RetryDelay expires, that server is brought back to the top of the list.

	The default value is 15 minutes. Valid range is: >=0.
CustomerId	CustomerId=<iPass Code>. This is the same number as your iPass portal customer ID. If you do not yet have such code, or are unsure what this code is, contact your iPass representative.
DebugLevel	DebugLevel=<level>. Debug level determines if debug and error messages are logged to the trace file. The following levels are supported. Debug Level 0 - Only severe messages are logged. Debug Level 1 - Error messages are logged. Debug Level 2 - Error and Debug messages are logged. Debug Level 3 - Error, Debug, and Packet parsing information is logged. Debug Level 4 - Error, Debug, Packet parsing, and Packet dumping is logged. Debug Level 5 - Detailed Packet and debug information is logged. The default value for this property is 0 Note: Production servers should normally run with debug level 0.
FailedAcctLogDir	FailedAcctLogDir=<Failed Accounting Directory> If an accounting record cannot be sent to the AAA server due to a communication error, the RoamServer writes the record to this file. The RoamServer writes one file per failed record. The file name of these files would have the timestamp as the suffix. Use the AcctLog tool to retransmit these records to the RoamServer, which will then resend it to the Accounting Server. The failed account directory should specify either the full path to the directory or the path relative to the iPass server home via the $ipass.server.home macro. Default value for this property is set to $ipass.server.home/logs/failedAcct/
FilterRequest	FilterRequest=<filter time in minutes> This property determines the amount of time to keep users in the local authentication cache. This cache is used to filter duplicate request and authenticate cached users. Valid range is 0 to 10 minutes. A value of 0 will turn off local authentication cache. The FilterRequest default is 0 minutes.
HeartBeatInterval	HeartBeatInterval=<number of minutes> This entry determines the time interval between heartbeat messages. This is an advanced setting. The server may not function properly if this value is set incorrectly. Default value for this property is set to 15 minutes.
HeartBeatMessage	HeartBeatMessage=yes/no. This entry determines if the heartbeat is turned on or off. This is an advanced setting. The server may not function properly if this value is set incorrectly. Default value for this property is set to no (heartbeat messages are turned off).
IMonServer	Provides IMonServer information. The IMonServers are central iPass servers used to receive HeartBeat Messages from this server. Sample format of the entries: IMonServer1=name11=value11,name12=value12,... IMonServer2=name21=value21,name22=value22,... IMonServer attributes: IpAddress: The IMonServer's IP address. Port: The IMonServer's port number. Do not change the default values set internally, unless instructed by iPass. Refer to iPass NetServer Documentation for more details.
Listener	List of the Listeners for this server. Expected format: Listener1=Type=<protocol>,Port=<port number>,IpAddress=<local IP address> Listener2=Type=<protocol>,Port=<port number>,IpAddress=<local IP

	address> Default Listeners are: Listener1=Port=577 NumOfThreads: You can improve connectivity to a RoamServer by increasing the number of threads accepting requests on port 577. This can be helpful for if your RoamServer is under heavier stress, such as 10 or more requests per second. For example: Listener1=Port=577,NumOfThreads=10 This is an advanced setting. The server may not function properly if this value is set incorrectly.
KeyStore	Provides KeyStore information. Sample format of this entry: KeyStore=name11=value11,name12=value12,... Below are the list of various KeyStore attributes: (These are advanced settings. The server will not start properly if these values are set incorrectly.) KeyStorePath: This entry determines the java keystore path. Default value for this property is set to C:\ipass\roamserver 6.1.0\certs\[rs.keystore] KeyPassword: This entry determines the password required to get keys from java keystore. Default value for this property is set to changeme KeyAlias: This entry determines the java keystore private key Alias. Default value for this property is set to rs CertAlias: This entry determines the java keystore trusted certificate alias. Default value for this property is set to ipassca KeyStorePassword: This entry determines the password required to open java keytore. Default value for this property is set to changeme Salt: This entry determines the salt used for encrypting KeyPassword and KeyStorePassword. Default value for this property is set to iPassRS
LogDirFileDeletionAge	LogDirFileDeletionAge=<age in days> Valid range is: 0 to 180 days. The default is 90 days. A value of 0 means deletion is DISABLED. LogDirFileDeletionAge determines how old files in the directory <iPass ServerHome>/logs must be before they are deleted. The check for file age is done only when the log file rotation happens. See page 34 for more information.
PolicyFile	PolicyFile=<Policy file Name> This entry, when present enables policy management (access control). The policy file contains a list of access control rules. Each rule can identify a country, class of service, a username, and whether roaming access is allowed or denied. This file can be created using the Policy Tool.
ReplyClass	ReplyClass=yes/no Configuration to enable passing Class attribute coming from the AAA server. When enabled, Roamserver will pass the Class attribute coming from AAA server. Default value is no (disabled). When disabled, Roamserver will block the Class attribute coming from AAA server. However, Roamserver may add its own Class attribute values even if ReplyClass is disabled.
RouteByRealm	RouteByRealm=yes/no Configuration to enable routing based on user realms (domains). When enabled, the RoutingRealm1, RoutingRealmX... are used to specify the servers to route to for a given realm. Default value is no. Routing by realm allows routing requests to specific AAA servers, based on the user's realm or domain. Routing can also be done by routing prefix. This allows you to use different types of authentication server, if necessary. For example, you could use both a

	RADIUS server and an LDAP server simultaneously. Requests from one domain, or with one prefix, can be directed to one server while requests from another domain or with another prefix can be directed to a second server. If routing by realm is enabled on your RoamServer, you will also need to set other properties to specify your other AAA servers, including `RoutingRealm, Realm, AuthServer, AcctServer`
	Example
	`RouteByRealm=YES`
	`RoutingRealm1=Realm=example.com,AuthServer1=AuthServer1, AcctServer1=AcctServer1`
	`RoutingRealm2=Realm=XY, AuthServer1=AuthServer2, AcctServer1=AcctServer2`
	`RoutingRealm3=Realm=DEFAULT,AuthServer1=AuthServer1,AcctServer1=AcctServer 1`

`RouteByRealmScheme`	`RouteByRealmScheme=<scheme>` Where <scheme> is either `EndsWith` or `StartsWith`. The default is `EndsWith`. `RouteByRealmScheme` indicates how the `RoutingRealm` properties are matched up with the domain (or realm) of the incoming user request. See page 34 for more information on routing by realm.
`RoutingRealm`	`RoutingRealm=<valid domain or routing prefix>` See also `RouteByRealm` for examples of proper use and formatting.
`ServerInfold`	This feature is not currently in use.
`StartUpMessage`	`StartUpMessage=yes/no.` This entry determines if a message is generated by the server on startup. This is an advanced setting. The server may not function properly if this value is set incorrectly. Default value for this property is set to no (startup messages are turned off)
`StoreFailedAcct`	`StoreFailedAcct=yes/no` or `true/false.` Determines if the RoamServer will store accounting to a local file if it fails to communicate with any and all of the AAA accounting servers. The `resendacct` tool can then be used to resend each of those accounting records to the RoamServer once the AAA is back up. Default setting is: `false`
`StripDeviceInfo`	StripDeviceInfo=yes/no or true/false. When StripDeviceInfo is set to true and the request is ACA accounting request then RoamServer will strip the Device info from the userid. Default value for this property is set to true
`TraceLogBackupType`	`TraceLogBackupType=<backupType>` Where <backupType> is either MultipleWithTimestamp or SingleBackup. The default is SingleBackup. `TraceLogBackupType` sets the trace log's backup file name when rotation is to be performed on the local trace files.
`TraceLogRotationHours`	`TraceLogRotationHours=<hours>` Valid range is: 1 to 720 hours. The default is 168 hours (1 week). `TraceLogRotationHours` controls how often the local trace file is rotated.
`TraceLogRotationMaxSize`	`TraceLogRotationMaxSize=<max size>` Minimum value is 100 kB. Maximum value is 20000 kB. The default is 10000 kB. `TraceLogRotationMaxSize` limits how large (in kilobytes) the local trace file can get before it is rotated.
`UpdateInterval`	`UpdateInterval=<DayOfWeek Hour:Minute>` Where `DayOfWeek` ranges from Sunday to Saturday and `Hour` is between 0-23. Default value for this property is set to Monday 2:00. This entry determines when RoamServer contacts the update server. Note: The `UpdateInterval` mechanism synchronizes with the system clock every sixty minutes. See also `AutoUpdate`.

`UpdateServer`	Provides iPass software Update Server information. Sample format of the entries:
	`UpdateServer1=name11=value11,name12=value12,...`
	`UpdateServer2=name21=value21,name22=value22,...`

	`UpdateServer` attributes: ■ `IpAddress:` The URL of the iPass software update server ■ `RetryDelay:` The time delay, in minutes, before retrying a server that recently failed a connection request. When a connection fails to a server, it is reordered to the end of the list. Once the `RetryDelay` expires, that server is brought back to the top of the list. The default value is 15 minutes. Valid range is: >=0. ■ `FailureThreshold:` Once the failure count exceeds the `FailureThreshold`, the server is reordered to the end of the list. The default value is 0. Refer to iPass NetServer Documentation for more details.

`UploadServer`	Provides iPass software Upload Server information. Sample format of the entries:
	`UploadServer1=name11=value11,name12=value12,...`
	`UploadServer2=name21=value21,name22=value22,...`
	
	`UploadServer` attributes:
	■ `IpAddress:` The URL of the iPass software update server
	■ `RetryDelay:` The time delay, in minutes, before retrying a server that recently failed a connection request. When a connection fails to a server, it is reordered to the end of the list. Once the `RetryDelay` expires, that server is brought back to the top of the list. The default value is 15 minutes. Valid range is: >=0.
	■ `FailureThreshold:` Once the failure count exceeds the `FailureThreshold`, the server is reordered to the end of the list. The default value is 0.
	
	Refer to iPass NetServer Documentation for more details.

`UsePolicyFile`	`UsePolicyFile=y/n` This property determines if the server uses policy file for authentication. Default value for this property is set to n. This is an advanced setting. The server may not function properly if this value is set incorrectly
`ZipLogFilesEnabled`	`ZipLogFilesEnabled=true/false.` Determines whether or not trace and log files are zipped. Default is set to `true`.

Configuration Options

This section discusses some RoamServer configurable options in detail. For more information on setting properties, see the Property Glossary on page 22.

Policy File

A Policy File allows you to filter the requests being sent to your authentication server. RoamServer will validate all users against this file before contacting your authentication server. This feature may be helpful if you wish RoamServer to authenticate from a large user database, but only want a small group of those users to be able to roam, or conversely, if you only wish to deny roaming access to a small group.

The Policy Tool, rs_policy.csh, located in your <RS_Home>/bin directory, is an application used for creation and maintenance of a Policy File. Although the Policy File is a text file, iPass recommends you use the Policy Tool for creating, editing and maintaining your Policy File. This will ensure proper formatting and correct policy criteria.

To create a policy file:

1. In the **<RS_Home>/bin** directory, run the file **rs_policy.csh**
2. If the tool detects that no Policy File exists, it will create one in the default directory.

To enable use of a Policy File:

1. Run **ipassconfig.csh -conf**
2. Enter **Yes** when prompted with the following **Do you wish to use the PolicyFile during authentication?**
3. Enter the path and name of your policy file, or press **Enter** to accept the default.

To edit or manage your policy file:

1. In the policy tool, choose your option from the menu:
   - Add a rule
   - Remove a rule
   - Edit a rule
   - Explain an existing rule
   - List the rules
   - Save the rules
   - List Country Code
   - Quit
2. When done, **enter 8 to quit the Tool**. You must restart RoamServer so that it can read a newly edited Policy File.

Policy File Pattern Matching

The policy file pattern matching is from most specific to the least, as follows:

#class of service	auth_domain	user_id	country_code
1	1	1	1

1	1	1	0
1	1	0	1
1	1	0	0
1	0	1	1
1	0	1	0
1	0	0	1
1	0	0	0
0	1	1	1
0	1	1	0
0	1	0	1
0	1	0	0
0	0	1	1
0	0	1	0
0	0	0	1
0	0	0	0

All rules are read and the most specific rule to match a given request is used. For example, these entries in a policy file would block all wireless access, except in the US.

#class of service	auth_domain	user_id	country_code	Allow access
WIRELESS	*	*	*	N
WIRELESS	*	*	US	Y

Because the policy file is written with permissions of root/admin, lowering the privileges required to run the policy tool will cause the tool to fail. Accordingly, you may wish to do one of the following to ensure policy file permissions are valid:

- Reset policy file permissions everytime the policy tool is run.
- Set up a cron job to periodically reset the file permission regardless of when policy tool is run.

Policy File Mapping

This table shows the mappings of NAS port type numbers to the class of service.

Nas-port-type	Class of service
0	DIAL-UP
1	DIAL-UP
2	DIAL-UP-ISDN
3	DIAL-UP-ISDN
4	DIAL-UP
5	DIAL-UP-PHS
6	DIAL-UP
7	DIAL-UP
8	DIAL-UP
9	DIAL-UP
10	WIRED
11	WIRED
12	WIRED
13	WIRED
14	WIRED
15	WIRED
16	WIRED
17	WIRELESS
18	WIRELESS
19	WIRED
20	WIRED

21	MOBILEDATA
22	MOBILEDATA
23	MOBILEDATA
24	WIRELESS
25	WIRED
ALL OTHERS	DIAL UP

Failover Configuration

Failover

If the primary server is unreachable, RoamServer can fail over to one or more secondary authentication or accounting servers. This feature works with RADIUS, LDAP and TACACS authentication protocols.

> *Your secondary servers do not have to be of the same type as your primary server. For example, if you had both a RADIUS server and an LDAP server, you could designate your RADIUS server as primary and your LDAP server as secondary.*

To configure RoamServer to failover to a secondary authentication server:

1. Run **ipassconfig.csh -conf**
2. Enter **Yes** when the following prompt appears **Do you wish to add new AuthServer?**
3. Enter the properties for the new authentication server as described under authentication Servers above.
4. If you are using RADIUS or TACACS, you must make sure that the shared secrets are the same for each server. In addition, if using RADIUS, you must make sure that RoamServer is entered as a client of the Secondary RADIUS as well as with the Primary.
5. Restart RoamServer. RoamServer will now be able to fail over to the secondary authentication server in the case of a power, hardware, or software failure happen to primary authentication server.

There is no limit to the number of secondary authentication servers you can specify. You can repeat the above to specify more authentication servers, by incrementing the number for each new server (AuthServer1, AuthServer2, etc.). However, in the ipassRS.properties file, you must ensure that servers are listed in numerical order such as:

`AuthServer1, AuthServer2, AuthServer3`, or failover will not occur.
In addition, you may not skip any numbers in the sequence when specifying servers. (For example, AuthServer1, AuthServer2 and AuthServer4 would not be an acceptable sequence.)

To configure the RoamServer to fail over to a secondary accounting server:

1. Run **ipassconfig.csh -conf**
2. Enter **Yes** when the following prompt appears **Do you wish to add new AcctServer?**
3. Enter the properties for the new accounting server as described under Accounting Servers, above.
4. If you are using RADIUS or TACACS, you must make sure that the shared secrets are the same for each server. In addition, if using RADIUS, you must make sure that RoamServer is entered as a client of the Secondary RADIUS as well as with the Primary.
5. Restart RoamServer. RoamServer should now be able to fail over to the secondary accounting server in the case of a power, hardware, or software failure happen to primary accounting server.

UNIX and Site Failover

Since there will always be a response from the local server, if you set one of your failover servers to UNIX or Site, there is no need to set any further servers in the sequence.

Trace Log File Configuration

RoamServer can be configured to write information about access attempts to a log file for debugging purposes. If

enabled, debugging information is output to a local log file, named `roamserver.trace`, which is found in the `<RS_Home>/logs` directory. The amount of debugging output can be controlled by changing the `DebugLevel` setting. The range for this value is 0 to 5 (inclusive), where 0 produces the least amount of output, and 5 produces the highest.

RoamServer can log information about both access attempts and accounting transactions. When placed into debug mode, RoamServer will log transactional information into a local file that can be used in troubleshooting. In addition, the software can be configured to log accounting data to either a local file or to forward it to a remote accounting server.

If your `DebugLevel` value is set to any value greater than 0, you will need to customize the log file rotation and backup process so that the logs do not build up unnecessarily. A `DebugLevel` of *5* produces a great deal of output.

Ascend Data Filters for Non-VPN Access

Some network providers on the iPass network filter port 25 traffic (SMTP), in an effort to prevent the distribution of spam mail on their networks. Although traffic through port 25 is blocked from these providers, they do allow traffic to pass to a limited number of IP addresses to allow users to send SMTP mail to valid mail servers. The IP addresses to which port 25 traffic is allowed is communicated by the Ascend Data Filter attributes, which are sent when the user successfully authenticates. These attributes are configured in ipassRS.properties. (The format is similar to how a RADIUS users file would be configured to return those attributes.)
If users will be connecting through a VPN, this property can be ignored with no effects. If users will not be connecting through a VPN, then iPass strongly recommends you configure these settings to reflect your SMTP servers.

Sample Settings

```
AscendDataFilter1=ip in forward tcp est
AscendDataFilter2=ip in forward dstip xxx.xxx.xxx.xxx/yy
AscendDataFilter3=ip in drop tcp dstport=25
AscendDataFilter4=ip in forward
```

xxx.xxx.xxx.xxx/yy would be replaced by an IP mask identifying the customer's mail server IP addresses (for example, 218.239.99.139/32). Note that most providers only allow masks ranging from 24 to 32.

For example, if your SMTP servers' public IP address is 236.14.5.70, then the settings would look like this:

```
AscendDataFilter1=ip in forward tcp est
AscendDataFilter2=ip in forward dstip 236.14.5.70/32
AscendDataFilter3=ip in drop tcp dstport=25
AscendDataFilter4=ip in forward
```

Note that either a single IP address (236.14.5.70/32) or a range of IP addresses (236.14.5.0/24) can be specified.

In this second example, there are two entries. The first is a single SMTP server, and the second is a network range. Port 25 traffic will be allowed to the single IP address specified in AscendDataFilter2, as well as the entire network specified in AscendDataFilter3.

```
AscendDataFilter1=ip in forward tcp est
AscendDataFilter2=ip in forward dstip 236.14.5.70/32
AscendDataFilter3=ip in forward dstip 236.16.6.0/24
AscendDataFilter4=ip in drop tcp dstport=25
AscendDataFilter5=ip in forward
```

Up to 17 different IP addresses or range strings can be specified in this manner.

Log File Deletion

Log files and accounting files can grow to unmanageable sizes. To control this, you can set log files to be deleted after a specified period of time by setting `LogDirFileDeletionAge` to an appropriate value. The default is 90 days.

Route-by-Realm

Route-by-Realm allows routing requests to specific AAA servers, based on the user's realm or domain. Routing can also be done by routing prefix.

This allows you to use different types of authentication server, if necessary. For example, you could use both a RADIUS server and an LDAP server simultaneously. Requests from one domain, or with one prefix, can be directed to one server while requests from another domain or with another prefix can be directed to a second server.

To enable Route-by-Realm:

Set **RouteByRealm** to **YES**. If Route-by-Realm is enabled, you will also need to set other properties to specify your other AAA servers, including RoutingRealm, AuthServer, and AcctServer.

Sample Settings

```
AuthServer1=protocol=RADIUS,ipaddress=10.10.0.1,port=1812,sharedsecret=foo,Attempts=3,Idl
eTimeout=5000,IncludePrefix=No,IncludeDomain=No,IncludeDomainAsWinPrefix=No
AuthServer2=protocol=LDAP,ipaddress=10.10.0.2,port=389,LdapConfigFile=C:/ipass/roamserver
/6.1.0/ipassLDAP.properties,IdleTimeout=10000,enableSsl=No

AcctServer1=protocol=RADIUS,ipaddress=10.10.0.1,port=1813,sharedsecret=foo,Attempts=3,Idl
eTimeout=5000,IncludePrefix=No,IncludeDomain=No,IncludeDomainAsWinPrefix=No
AcctServer2=protocol=AcctFile,localAcctFileName=C:/ipass/roamserver/6.1.0/logs/acct.log,I
ncludeDomainAsWinPrefix=No
```

```
RouteByRealm=YES
RoutingRealm1=Realm=mydomain.com,AuthServer1=AuthServer1,AcctServer1=AcctServer1
RoutingRealm2=Realm=XY,AuthServer1=AuthServer2,AcctServer1=AcctServer2
RoutingRealm3=Realm=DEFAULT,AuthServer1=AuthServer1,AcctServer1=AcctServer1
```

In this sample there are two Authentication Servers defined. AuthServer1 is a RADIUS. AuthServer2 is an LDAP.

If the customer logs in using the realm mydomain.com, the line which begins, "RoutingRealm1" is in play. It defines the primary Authentication Server as AuthServer1. The rule is written as AuthServer1=AuthServer1. This translates as "The primary authentication server for this realm is the line above that starts with 'AuthServer1='"

But if the customer logs in with the realm XY, the line which begins, "RoutingRealm2" is in play. It defines the primary Authentication Server as AuthServer2. The rule is written as AuthServer1=AuthServer2. This translates as "The primary authentication server for this realm is the line above that starts with 'AuthServer2='"

You'll notice Route-by-Realm also directs traffic to Accounting Servers. When addressing the Accounting Server the key AuthServer becomes AcctServer. Otherwise, the logic of how Route-by-Realm works is the same.

In the examples, when routing realm mydomain.com is used, AcctServer1 is employed to send the accounting records to the RADIUS.

When routing realm XY is used, AcctServer2 is used. Please note, in this case, a text file is being populated because LDAP does not record accounting records.

The final line, where the realm is DEFAULT is required to catch any requests that contain malformed realms. This line gives the RoamServer an avenue to forward the request. Not having this line can cause the RoamServer to crash if a malformed realm is used.

ipassLDAP.properties

In the AuthServer property of **ipassRS.properties**, you can specify a path to a file containing special LDAP settings named **ipassLDAP.properties**. This section explains configuration of this file.

User-Configurable Options

This table summarizes the configurable options in **ipassLDAP.properties**. When an **ipassLDAP.properties** file is not present, or if an option is not specified, the default values will be used.

Property	Default Value	Comments

`LdapBaseDn`	NULL	Specifies base DNs to be used during LDAP authentication. When configured, it will be appended to the `LdapExactMatchRdn` during exact match bind and used as a search base during the LDAP search operation. Any variables supplied in the format of `$VARIABLE` will be replaced with the actual value of that variable. The current variables supported are `$USERID`, `$PREFIX` and `$DOMAIN`.
		If no `LdapBaseDn` is configured, then no anonymous bind and search will be performed.
		Multiple base DNs (more than one line) are permitted in the ipassLDAP.properties file. When multiple base DNs are configured, the authentication process will use them in the order they appear in the ipassLDAP.properties file. If authentication fails using the first `LdapBaseDn`, authentication will be re-attempted using the second `LdapBaseDn` and so on.
		Since a base DN is added on to the login name when an exact match bind is performed, if a user logs on using a full DN (`uid=Joe,ou=people,o=example.com`), `LdapBaseDn` should not be because performance will be reduced.
		Examples:
		`LdapBaseDn=ou=people,o=example.com`
		`LdapBaseDn=o=example.com`
		`LdapBaseDn=dc=company,dc=com`

`LdapBindDn`	NULL	For LDAP servers that do not support anonymous binds, this configuration will set a specific DN to be used for binding to the LDAP server, before performing a search operation.

		When anonymous binds are supported, omit this configuration and the default value of `NULL` will be used. Example: `LdapBindDn=uid=bindmaster,ou=people,o=example.com`
`LdapBindPasswd`	NULL	For LDAP servers that do not support anonymous binds, this configuration will set a password to be used for binding to the LDAP server before performing a search operation. When anonymous binds are supported, omit this configuration and the default value of `NULL` will be used. Example: LdapBindPassword=bindUserPassword
`LdapBindPasswdEncr`	NULL	This property has been added to keep the bind password of LDAP files in encrypted format. After server startup, the LdapBindPassword attribute will change to LdapBindPasswdEncr attribute with password encrypted.
`LdapCompareAttr`	NULL	Configuration to enable comparison of user passwords against a specific user attribute in the LDAP directory as a means of authentication. The user attribute specified must contain a password saved in clear text in the LDAP directory for `LdapCompareAttr` to work. This compare replaces the final user bind to authenticate the user. The user bind authenticates against the standard password attribute (usually `userpassword`), which may or may not be encrypted in the LDAP directory. Example: `LdapCompareAttr=roamingPassword`
`LdapDetectBaseDn`	YES	When `LdapDetectBaseDn` is enabled, and no `LdapBaseDn` is configured, it will detect all the available `BaseDn` (a.k.a. `namingContexts`) of the LDAP server. Valid values: YES or NO.
`LdapDoExactMatch`	NO	Disables or enables binding directly to the LDAP server for user authentication using only the user's login id, password, and any base DN by the `LdapBaseDn` configuration. Accepted values are `YES` or `NO`. Example: `LdapDoExactMatch=YES`
`LdapExactMatchRdn`	uid=$USERID	The DN used for the exact match bind is comprised of two parts: the relative DN (RDN) and the base DN. The base portion can be

		specified by the LdapBaseDn configuration. The relative DN format can be specified by the LdapExactMatchRdn. The RDN is by default uid=$USERID, where the variable $USERID is replaced by the username specified at login time. The current variables supported are $USERID and $DOMAIN. For example: User *joe* exists in a LDAP tree with a DN of uid=joe,ou=people,o=example.com, and he logs in as *joe@example.com*. For a successful exact match bind, leave the LdapExactMatchRdn as default and set the LdapBaseDn=ou=people,o=example.com. User *Mary* exists in a LDAP tree with a DN of cn=Mary,dc=company,dc=com, and she logs in as *Mary@example.com*. For a successful exact match bind, set the LdapExactMatchRdn=cn=$USERID and set the LdapBaseDn=dc=company,dc=com. The exact match bind can be disabled by setting LdapDoExactMatch=NO. Only one LdapExactMatchRdn (one line) is allowed in the ipassLDAP.properties file. Examples: LdapExactMatchRdn=cn=$USERID LdapExactMatchRdn=$USERID
LdapExcludeWildcards	NULL	This property is used to exclude wildcards (special characters) from the default set of LDAP wildcards &|~=^!<>*()+. These wildcards could be used for LDAP Blind Injection, so excluding them is not recommended. Default is none. Example: LdapExcludeWildcards=& or LdapExcludeWildcards=&|~=^!<>*()+
LdapGroupDepth	3	Can be used in conjunction with LdapMemberOfGroup to limit the depth of the search for nested groups. Valid values are from 1 to 10. A value of 1 would avoid any nested group search and only look for direct group memberships.

LdapIgnoreExpiredAdPassword	NO	If set to YES, RoamServer will allow access by ignoring expired Active Directory (AD) passwords.
LdapMemberOfGroup	NULL	This property will enable verification that a user is a member of a given group in Active Directory. RoamServer compares the given group DN to the attribute and any subsequent nested groups, up to a maximum depth of 10 nested groups. Example: LdapMemberOfGroup=CN=CompanyUsers,CN=Users,DC=CorporateHQ,DC=company,DC=com
LdapSearchFilter	uid=$USERID	Specifies a custom filter when searching an LDAP server for a user. If this option is not set, the default filter (uid=$USERID) will be used. When an exact match bind is disabled or is unsuccessful, an anonymous bind and search will be used. A custom filter may be supplied for the search. Any variables supplied in the format of $VARIABLE will be replaced with the actual value of that variable. The current variables supported are $USERID, $PREFIX and $DOMAIN. Only one filter (one line) is presently allowed in the ipassLDAP.properties file. The variables' values are taken from the user's login. For example if someone logs in as *joe@example.com*, the variable $USERID would be replaced by *joe* (that is, everything to the left of the leftmost @-sign, not including any prefix such as *iPass/*). The variable $DOMAIN would be replaced by *example.com* (that is, everything to the right of the leftmost @-sign). For example: if the search filter is (&(mail=$USERID@$DOMAIN)(dialup=true)), when joe from example.com logs on, the search filter will be converted to (&(mail=joe@example.com)(dialup=true)) Examples: LdapSearchFilter=uid=$USERID LdapSearchFilter=mail=$USERID@$DOMAIN LdapSearchFilter=(&(uid=$USERID)(dialup=true)) Class_of_service_str can also be used as a valid attribute for the search query. Valid values for this attribute are: DIAL-UP, DIAL-UP-ISDN, DIAL-UP-PHS, WIRED, WIRELESS, MOBILEDATA.

		Example: `LdapSearchFilter=(&(sAMAccountName=$USERID)(memberOOf=CN=$(class_of_service_str),CN=Users,DC=company,DC=com))`
`LdapSearchMoreServers`	NO	Uncomment and customize the `LdapSearchMoreServers` line to enable/disable searching other LDAP servers when the user is not found on the current LDAP server. Valid values are YES or NO. Default value is NO. Note to Active Directory (AD) users: you will, in most cases, need this enabled to YES.
`LdapSearchScope`	2	Determines the scope of the LDAP search. Valid values are: 0=Object Scope, 1=One Level Scope , 2=Subtree Scope
LdapUacAttr	userAccountControl=512,544,66048,66080,262656,262688,328192,328224	To enable ACA support with LDAP. Given default value only applicable for Active Directory. For LDAP, configure your customized user account control attribute with value associated for active users. example: LdapUacAttr=ipassStatus=active or LdapUacAttr=userStatus=enable or LdapUacAttr=userEnabled=true

Suggested Configuration

Example 1 (Most common)

For companies with an LDAP directory structure where roaming users are stored in different directories:

```
uid=user1,ou=development,o=example.com
uid=user2,ou=finance,o=example.com
uid=user3,ou=marketing,o=example.com
```

Performing a search for the user might be a simpler approach. Therefore, the exact match bind step can be skipped all together. If all users login with the format of user1@example.com, then only do an anonymous bind and search of the LDAP directory.

Set the following in the `ipassLDAP.properties` file:

```
LdapBaseDn=o=example.com
LdapDoExactMatch=no
LdapSearchFilter=uid=$USERID
```

Example 2

For companies with an LDAP directory structure where all roaming users are stored in the same directory:

```
uid=user1,ou=people,o=example.com
uid=user2,ou=people,o=example.com
uid=user3,ou=people,o=example.com
```

All users are in the ou=people,o=example.com directory. If all users log in with the format of user1@example.com, then to bind to the LDAP server on the first try with the exact match bind.

Set the following in the ipassLDAP.properties file:

```
LdapBaseDn=ou=people,o=example.com
```

Example 3

For companies whose roaming users login with a full Distinguished Name (DN) such as: `uid=user1,ou=development,o=example.com@example.com`, the user id portion (which is everything to the left of the leftmost @-sign) is the full DN of the user.

Only the exact match bind is needed.

Set the following in the ipassLDAP.properties file:

```
LdapExactMatchRdn=$USERID
LdapDoExactMatch=Yes
```

Appendix I: Error Messages

This section lists error messages that can be returned by the RoamServer at Debug Levels 0, 1 and 2. Although other debug levels are possible, they are used only for packet dumps and no error messages are associated with them. Variables denoted in the list by + (for example, `+ioe.getMessage()`) will be replaced at runtime with specific data.

Feature	Debug Level	Message
Tacacs+		
	1	Error occurred while trying to communicate to the TACACS+
	1	Failed to convert TACACS+ packet to bytes
	1	"Failed to open TCP socket to TACACS+ server: IO Error, "+ioe.getMessage()
	1	"Failed to open TCP socket to TACACS+ server:
	1	Failed to send packet to TACACS+ server" +ioe.getMessage()
	1	Unexpected NULL clientSocket, socket could be closed.
	1	Timed Out reading packet from TACACS+ server "
	1	"Failed to read packet from TACACS+ server "
	1	Cannot parse raw TACACS+ packet
	1	"Error closing socket to TACACS+ " +ioe.getMessage()
	1	"ERROR parsing header of packet received from TACACS+
	1	"Unsupported reply packet type " +this.hdr_type +" received from
	1	"ERROR decrypting TACACS+ packet"
	1	"ERROR: missing TACACS+ packet type"
	1	"parse() not supported for this reply packet type " +pktType
	1	"ERROR: missing TACACS+ packet type"
	1	"ERROR: toBytes() not supported for packet type " +pktType
	1	"ERROR encrypting TACACS+ packet"
	1	"CHAP challenge conversion failed."
	1	"CHAP password conversion failed."
	1	"ERROR encrypting TACACS+ packet"
	2	Error or Timeout in getting reply from TACACS+ server
	2	Password is NULL, TACACS+ Minor Version 0 does not support CHAP
	2	Error/Timeout getting first auth reply from TACACS+ server
LDAP		
	0	"Server's LDAP Info is Missing "
	0	"Unexpected return code (" +rc +")"
	0	"Internal Error: LDAP server address not set"
	1	"Illegal LDAP Configuration: Must configure an "+LdapInfo.LDAP_BASE_DN +" or Enable "+LdapInfo.LDAP_DO_EXACT_MATCH
	1	"Error creating RDN from ldapExactMatchRdn"
	1	"ExactMatchBind failed " +ne.getMessage()
	1	"Error creating Search Filter."
	1	"LDAP Authentication failed " +reason
	1	"Error, LDAP search found multiple matches "+entryCount +" found
	1	"LDAP Search found multiple matches for this user " +slee.getMessage()

	1	"LDAP Search exceeded " +searchTimeout +" millisecond time limit:
	1	"LDAP Search Error: " +ne.getMessage()
	1	"LDAP Compare of (" +name +") attribute with password failed."
	1	"LDAP Compare of (" +name +") attribute failed: " +ne.getMessage()
	1	"Unexpected NULL ldap context"
	1	"Invalid attribute name: "+attrName+", in line: "+origString
	1	"Could not authenticate user at this LDAP server"
	1	"TIMEOUT while talking to LDAP server after " +sInfo.NumRetry +"
	2	"Error while closing connection to LDAP server" +ne.getMessage()
SSLPost		
	0	fileDesc+fileName+" does not exist"
	0	"Cannot read "+fileDesc+filename
	0	"Failed to instanciate SSLPostCommunicator: "+cce.getMessage()
	0	"Could not instantiate SSLSocketImpl"
	0	"ERROR: Missing IpassDictionaryEntry"
	1	"Socket receive timed out"
	1	"Failed to receive data from server: " + serverInfoRec.IpAddress
	1	"IOEXCEPTION: while talking to server: " + serverInfoRec.IpAddress + ":" +
	1	"received null Communicator object"
	1	"received null serverInfoRec"
	1	"received null requestPkt"
	1	"received null replyPkt"
	1	"Could not create sslSocket: doHandshake failed"
	1	"Could not create sslSocket: Instantiation failed"
	1	"sslSocket null for ServerSide communicator"
	1	"Could parse post packet: " +replyStr
	1	Malformed Post Packet
	1	"Malformed post packet header"

	1	"Unexpected NULL sslSocket."
	1	"Error parsing MultiInstance attribute "+name+", of type " +de.getType()
	1	"Error parsing attribute "+name+", of type " +de.getType()
	1	"Error in converting the packet to bytes: " + e.toString()
	1	"Error for attribute "+name+ ": "+i.getMessage()+" Ignoring it"
	1	"Dropping attribute for ipassCode " +ipassCode+" value "+value+",
	1	Base64 Decode ERROR: Dropping OBJECT of ipassCode " +ipassCode+"
	1	"Dropping OBJECT of ipassCode " +ipassCode+" value "+value+", OptionalDataException: "+o.getMessage()
	1	"Dropping OBJECT of ipassCode " +ipassCode+" value "+value+",
	1	"Dropping OBJECT of ipassCode " +ipassCode+" value "+value+", IOException: "+i.getMessage()
	1	"Dropping attribute for ipassCode " +ipassCode+" value "+value+",
	2	"NULL sslServerSocket, listener socket could be closed."
	2	"SSL handshake failed, closing accepted socket."
	2	"Listeners are shutdown, closing accepted socket."
	2	"Rejecting packet from: " +sslSocket.getHost()
	2	"Error: No ipassPkt to send"
	2	"Unexpected NULL sslSocket, socket could be closed."
	2	"Could parse post packet: " +packetStr
	2	"Error parsing IpassPostPkt: Unknown URI/request type " +uri
	2	"Error parsing IpassPostPkt: missing empty string."
	2	"Error parsing IpassPostPkt."
	2	"Unknown PostPkt attribute (" +name +"): ignoring it."
Handlers		
	0	"Software update failed"
	0	"Download failed"
	0	"Error occurred while trying to instantiate RSPolicyRules: " + i.getMessage()
	0	"Error occurred while adding policy rule: An entry with the same rule:" + id + " exists!"
	0	"File "+policyFile+" not found"

	0	"Failed to Shutdown due to policy errors as the TransactionController is null"
	0	"Failed to Shutdown due to policy errors as the TransactionContext is null"
	0	Cannot find TRANSACTION CONTROLLER
	0	Cannot find exceptionHandler
	0	Could not get LOCAL_HOST_IP
	0	Error occurred while trying to instantiate " + s.toString()
	0	Error occurred while trying to send the reply packet
	0	No Server found for the following transaction type: "+ reqTypeName
	0	No valid handler found for the request of type "+type);
	0	ERROR occurred while trying to save the acct record in a file: "+i.getMessage())
	0	Error occurred while trying to instantiate RSAcctReqHandler: " +
	0	Unexpected ERROR: "+Config.FAILED_ACCT_LOG_DIR+" property not set!
	0	Could not create directory \"" + failedDirPath + "\" to store
	0	ERROR, expected "+Config.FAILED_ACCT_LOG_DIR+" to be a directory, got \"" + failedDirPath + "\" instead.
	1	"Software Update Failed due to failure to load the Server's Version Table."
	1	"Unable to copy "+this.serverJarFileName +" to "+this.updatefilesJarFileName
	1	"User " + user_id + " is denied access based on the policy rule:
	1	"IO error in loading policy File "+policyFile
	1	"Error loading the policy file"
	1	"Cannot get SSLPOST listener port, defaullting to:" + UNKNOWN_PORT
	1	"Failed to handle Heartbeat message!"
	1	"Failed to load RS Policy Rules: "+se.getMessage()
	1	"Policy Restriction. Verify Policy Failed."
	1	"Authentication Rejected: Invalid Reply Packet"
	1	"ERROR: list lock is NULL. Cannot check for duplicates in our
	1	"exception ocurred: " + e.toString()
	1	"ERROR: list lock is NULL. Cannot add entry to our accessList"
	1	"No such hashing alrorithm error: "+nsae.getMessage()

	1	handleRequest-Communicator object is null
	1	Error: File: " + fileName + " does not exist on the server
	1	Error: File: " + fileName + " content is empty!
	1	failed to get file contents
	1	Invalid Request: Failed to get the path of the file: " + fileName
	1	Invalid Request: Cannot return the files in the keys folder!
	1	Invalid Request: filename is not from the $ipass.server.home: " +
	1	Invalid Request: File:" + fileName + " does not exist on the server!
	1	"Invalid Request: File name not specified!
	1	handleRequest-Communicator object is null
	1	Failed to reload the new config file, reverted to the old config file...
	1	Invalid request, Failed to Reload the new config file, and failed
	1	Invalid request, Failed to Reload the new config file, and failed to find the " + fileName + ".bak in order to
	1	Invalid request, Failed to Reload the new config file, and failed to delete it.\nPlease copy the " + fileName + ".bak
	1	Failed to rename " + fileName + " to " + fileName + ".bak"
	1	Failed to delete " + fileName + ".bak"
	1	Error, Config Filename could not be obtained!
	1	source Ip is null, not a valid CTRL_MSG_IP
	1	netSourceIp +" is not a valid/configured CTRL_MSG_IP
	1	Invalid Request: File contents are empty!
	1	Invalid Request: Failed to load the config changes: " + e.getMessage()
	1	Protocol is not supported by current version of software: Server ID=" + serverInfoRec.ServerInfoId + ", Server
	1	ERROR: Cannot get communicator for server IP: " + serverInfoRec.IpAddress + ", of Protocol: " +
	1	"No Servers found: Null returned from getRoute()"
	2	netSourceIp +" is not a valid/configured CTRL_MSG_IP");
RADIUS		
	0	Failed to open DatagramSocket

	0	Cannot get LOCAL_HOST_IP, unable to set NAS_IP in RADIUS packet
	0	IOException on listener for port "+serverPort+": "+e.getMessage();
	0	IOException on listener for port is due to RADIUS Listeners being
		shutdown
	0	ERROR creating the UDP socket at port "+port+". (Port may be in use)");
	0	Failed to instanciate SharedSSLPostCommunicator
	1	Unexpected NULL socket, socket could be closed
	1	IOException on DatagramSocket
	1	Error occurred while trying to talk to AAA server
	1	Failed to communicate with radius server after " +sInfo.NumRetry
	1	RADIUSPkt parsing errors
	1	Input not a byte array
	1	Empty RADIUS data
	1	Illegal type in RADIUS packet
	1	Missing identifier in the RADIUS packet
	1	Missing Length in the RADIUS packet
	1	Missing authenticator in the RADIUS packet
	1	Missing code in the RADIUS packet
	1	Missing length in the RADIUS packet
	1	ERROR: Invalid CHAP_PASSWD length of "+dataLen
	1	ERROR: Invalid MESSAGE_AUTHENTICATOR length of "+dataLen
	1	Missing IpassDictionaryEntry for radius code " + code
	1	Illegal data type
	1	Malformed radius packet (When data length is longer than the packet header specified)
	1	ERROR: missing MESSAGE_AUTHENTICATOR to validate EAP-Message
	1	ERROR: missing Request Authenticator to validate EAP-Message
	1	ERROR: failed to re-calculate Message-Authenticator"
	1	ERROR: Invalid Message-Authenticator

	1	ERROR: missing Request Authenticator
	1	ERROR: failed to generate test Authenticator
	1	ERROR: missing Response Authenticator
	1	ERROR: Invalid Response Authenticator
	1	No such algorithm
	1	Digest Exception
	1	No valid RADIUS code for Ipass Packet Type "+getPktType()+" Status "+status
	1	Missing IDENTIFIER header attribute, using value of "+ident+"
	1	Error: CHAP Identifier missing from packet
	1	CHAP password conversion failed.
	1	CHAP challenge conversion failed.
	1	ERROR: missing Shared Secret to calculate the Message Authenticator
	1	ERROR: when calculating HMAC digest of Message Authenticator
	1	ERROR: Request Authenticator is missing.
	1	Unsupported encoding exception
	1	NoSuchAlgorithmException
	1	Exception: " + e.toString());
	1	ERROR: missing Shared Secret
	1	ERROR: Base64 Decode of iPass Attribute " +ipassAttrCode +" Failed
	1	WARNING: Unable to get Dictionary entry for iPass Attribute
	1	ERROR: UTF8 conversion of iPass Attribute " +ipassAttrC
	1	ERROR: Base64 Decode of iPass Attribute " +ipassAttrCode +" failed
	1	ERROR: Base64 Decode Vendor Specific Attribute " +vendorId+":"+vendorType +"
	1	ERROR: Invalid Vendor Specific Attribute format
	1	Vendor ID missing from Vendor Specific Attribute
	1	Vendor Type missing from Vendor Specific Attribute (VendorID="+vendorId+
	1	Vendor Length missing from Vendor Specific Attribute (VendorID="+vendorId+", VendorType="+vendorType+
	1	Value missing from Vendor Specific Attribute (VendorID="+vendorId+", VendorType="+vendorType

	1	Value from Vendor Specific Attribute is corrupted. (VendorID="+vendorId+", VendorType="+vendorType
	1	expected len was "+vendorValueBytes.length
	1	Cannot convert attribute "+attr +", RADIUSType type of IPADDRESS
	1	Cannot convert attribute "+attr +", RADIUSType of Integer to iPassType " +iPassType
	1	Unsupported iPass attribute " +attr +", with radius value " +radiusValue
	1	NULL input: key is null
	1	NULL input: text is null
	1	Hashing error
	1	No such hashing algorithm error
	2	Cannot parse raw packet
	2	Receive timeout set to " +sInfo.IdleTimeout milliseconds
	2	RADIUSBufferSize error
	2	NULL serverSocket, listener socket could be closed.
	2	Started RADIUS Listener "+i +" on port "+listenerThreads[i].getServerPort());
	2	Cannot convert attribute "+attr +", RADIUSType of TEXT to iPassType " +iPassType
	2	Unsupported String Encoding: " +attr +", with radius Type " +radiusType
	2	Cannot convert attribute "+attr +", RADIUSType of String to iPassType " +iPassType
	2	Cannot convert to Integer: "+attr +", with radius Type " +radiusType
	2	Cannot convert attribute "+attr +", RADIUSType Time to iPassType
	2	Cannot convert attribute "+attr +", RADIUSType BYTEARRAY to iPass type " + iPassType
	2	Illegal data type " + radiusType
Site		
	0	Failed to load SiteCommunicator library
	1	Error occurred while trying to do Site file authentication
	2	Failed talking to SITE server
Unix		
	0	Failed to load UnixCommunicator library
	1	Error occurred while trying to do UNIX authentication

	2	Failed talking to Unix server
NT and NT RAS		
	2	Received authentication accept packet from Windows Server
	2	Received authentication reject packet from Windows Server
AcctFile		
	1	Failed to write to local AcctFile
	1	Error occurred while trying to talk to Windows server
	1	Failed talking to Windows server
	2	Received unexpected null packet when writing to local AcctFile

Appendix II: RADIUS Attributes

When using RoamServer with RADIUS authentication, check your RADIUS logs to verify your RFC attributes. If an attribute is not shown in the tables here, then you need to re-configure your RADIUS to eliminate the attribute.

RADIUS Authentication Attributes

This table shows which attributes may be found in which kinds of packets, and in what quantity. On the table:\

- **0:** This attribute must not be present in the packet.
- **0+:** Zero or more instances of this attribute may be present in the packet.
- **0-1:** Zero or one instance of this attribute may be present in the packet.
- **1:** Exactly one instance of this attribute must be present in the packet.

Request	Accept	Reject	Challenge	#	Attribute	Notes
0-1	0-1	0	0	1	User-Name	
0-1	0	0	0	2	User-Password	An Access-Request must contain either a User-Password or a CHAP-Password or State. An Access-Request must *not* contain both a User-Password and a CHAP-Password. If future extensions allow other kinds of authentication information to be conveyed, the attribute for that can be used in an Access-Request instead of User-Password or CHAP-Password.
0-1	0	0	0	3	CHAP-Password	An Access-Request must contain either a User-Password or a CHAP-Password or State. An Access-Request must *not* contain both a User-Password and a CHAP-Password. If future extensions allow other kinds of authentication information to be conveyed, the attribute for that can be used in an Access-Request instead of User-Password or CHAP-Password.
0-1	0	0	0	4	NAS-IP-Address	An Access-Request must contain either a NAS-IP-Address or a NAS-Identifier (or both).
0-1	0	0	0	5	NAS-Port	
0-1	0-1	0	0	6	Service-Type	An Access-Request must contain either a NAS-IP-Address or a NAS-Identifier (or both).
0-1	0-1	0	0	7	Framed-Protocol	
0-1	0-1	0	0	8	Framed-IP-Address	
0-1	0-1	0	0	9	Framed-IP-Netmask	
0	0-1	0	0	10	Framed-Routing	
0	0+	0	0	11	Filter-Id	
0-1	0-1	0	0	12	Framed-MTU	
0+	0+	0	0	13	Framed-Compression	

0+	0+	0	0	14	Login-IP-Host	
0	0-1	0	0	15	Login-Service	
0	0-1	0	0	16	Login-TCP-Port	
0	0+	0+	0+	18	Reply-Message	
0-1	0-1	0	0	19	Callback-Number	
0	0-1	0	0	20	Callback-Id	
0	0+	0	0	22	Framed-Route	
0	0-1	0	0	23	Framed-IPX-Network	
0-1	0-1	0	0-1	24	State	An Access-Request must contain either a User-Password or a CHAP-Password or State. An Access-Request must *not* contain both a User-Password and a CHAP-Password. If future extensions allow other kinds of authentication information to be conveyed, the attribute for that can be used in an Access-Request instead of User-Password or CHAP-Password.
0	0+	0	0	25	Class	
0+	0+	0	0+	26	Vendor-Specific	
0	0-1	0	0-1	27	Session-Timeout	
0	0-1	0	0-1	28	Idle-Timeout	
0	0-1	0	0	29	Termination-Action	
0-1	0	0	0	30	Called-Station-Id	
0-1	0	0	0	31	Calling-Station-Id	
0-1	0	0	0	32	NAS-Identifier	
0+	0+	0+	0+	33	Proxy-State	
0-1	0-1	0	0	34	Login-LAT-Service	
0-1	0-1	0	0	35	Login-LAT-Node	
0-1	0-1	0	0	36	Login-LAT-Group	
0	0-1	0	0	37	Framed-AppleTalk-Link	
0	0+	0	0	38	Framed-AppleTalk-Network	
0	0-1	0	0	39	Framed-AppleTalk-Zone	
0-1	0	0	0	60	CHAP-Challenge	
0-1	0	0	0	61	NAS-Port-Type	
0-1	0-1	0	0	62	Port-Limit	
0-1	0-1	0	0	63	Login-LAT-Port	

0-1	0	0	0	77	Connect-Info	
0+	0+	0+	0+	79	EAP-Message	
0-1	0-1	0-1	0-1	80	Message-Authenticator	
0	0-1	0	0	85	Acct-Interim-Interval	

RADIUS Accounting Attributes

This table shows the attributes found in Accounting-Request packets. No attributes should be found in Accounting-Response packets except Proxy-State and possibly Vendor-Specific. On the table:

- **0:** This attribute must not be present in packet.
- **0+:** Zero or more instances of this attribute may be present in packet.
- **0-1**: Zero or one instance of this attribute may be present in packet.
- **1**: Exactly one instance of this attribute must be present in packet.

#	Attribute	Notes
0-1	User-Name	
0	User-Password	
0	CHAP-Password	
0-1	NAS-IP-Address	An Accounting-Request must contain either a NAS-IP-Address or a NAS-Identifier (or both).
0-1	NAS-Port	
0-1	Service-Type	
0-1	Framed-Protocol	
0-1	Framed-IP-Address	
0-1	Framed-IP-Netmask	
0-1	Framed-Routing	
0+	Filter-Id	
0-1	Framed-MTU	
0+	Framed-Compression	
0+	Login-IP-Host	
0-1	Login-Service	
0-1	Login-TCP-Port	
0	Reply-Message	
0-1	Callback-Number	
0-1	Callback-Id	
0+	Framed-Route	
0-1	Framed-IPX-Network	
0	State	
0+	Class	
0+	Vendor-Specific	
0-1	Session-Timeout	
0-1	Idle-Timeout	
0-1	Termination-Action	
0-1	Called-Station-Id	
0-1	Calling-Station-Id	
0-1	NAS-Identifier	An Accounting-Request must contain either a NAS-IP-Address or a NAS-Identifier (or both).

0+	Proxy-State	
0-1	Login-LAT-Service	
0-1	Login-LAT-Node	
0-1	Login-LAT-Group	
0-1	Framed-AppleTalk-Link	
0-1	Framed-AppleTalk-Network	
0-1	Framed-AppleTalk-Zone	
1	Acct-Status-Type	
0-1	Acct-Delay-Time	
0-1	Acct-Input-Octets	
0-1	Acct-Output-Octets	
1	Acct-Session-Id	
0-1	Acct-Authentic	
0-1	Acct-Session-Time	
0-1	Acct-Input-Packets	
0-1	Acct-Output-Packets	
0-1	Acct-Terminate-Cause	
0+	Acct-Multi-Session-Id	
0+	Acct-Link-Count	
0	CHAP-Challenge	
0-1	NAS-Port-Type	
0-1	Port-Limit	
0-1	Login-LAT-Port	
0-1	Acct-Input-Gigawords	
0-1	Acct-Output-	
0-1	Event-Timestamp	
0+	Connect-Info	