Radia Client Automation Enterprise

For the Windows® operating system

Software Version: 9.00

SSL Implementation Guide

Document Release Date: April 2013

Software Release Date: June 2013

PERSISTENT

Legal Notices

Warranty

The only warranties for products and services are set forth in the express license or service agreements accompanying such products and services. Nothing herein should be construed as constituting an additional warranty. Persistent Systems shall not be liable for technical or editorial errors or omissions contained herein. The information contained herein is subject to change without notice.

Restricted Rights Legend

Confidential computer software. Valid license from Persistent Systems or its licensors required for possession, use or copying. No part of this manual may be reproduced in any form or by any means (including electronic storage and retrieval or translation into a foreign language) without prior agreement and written consent from Persistent Systems.

Copyright Notice

Trademark Notices

Microsoft®, Windows®, Windows® XP, and Windows Vista® are U.S. registered trademarks of Microsoft Corporation.

Java is registered trademark of Oracle and/or its affiliates.

Acknowledgements

This product includes software developed by the Apache Software Foundation (http://www.apache.org/).

This product includes cryptographic software written by Eric Young (eay@cryptsoft.com).

This product includes software developed by the OpenSSL Project for use in the OpenSSL Toolkit (http://www.openssl.org/).

This product includes software written by Tim Hudson (tjh@cryptsoft.com).

This product includes software written by Daniel Stenberg (daniel@haxx.se).

This product includes OVAL language maintained by The MITRE Corporation (oval@mitre.org).

Documentation Updates

The title page of this document contains the following identifying information:

- Software Version number, which indicates the software version.
- Document Release Date, which changes each time the document is updated.
- Software Release Date, which indicates the release date of this version of the software.

To check for recent updates or to verify that you are using the most recent edition of a document, go to:

http://support.persistentsys.com/

This site requires that you register for a Persistent Passport and sign in. Register online at the above address.

For more details, contact your Persistent sales representative.

Support

Persistent Software support provides customer self-solve capabilities. It provides a fast and efficient way to access interactive technical support tools needed to manage your business. As a valued support customer, you can benefit by being able to:

- Search for knowledge documents of interest
- Submit and track support cases and enhancement requests
- Submit enhancement requests online
- Download software patches
- Look up Persistent support contacts
- Enter into discussions with other software customers
- Research and register for software training

To access the Self-solve knowledge base, visit the Persistent Support home page.

Note: Most of the support areas require that you register as a Persistent Support user and sign in. Many also require an active support contract. More information about support access levels can be found on the Persistent Support site.

To register for a Persistent Support ID, go to: Persistent Support Registration.

Contents

Chapter 1

Introduction to SSL, Certificates, and Private Keys

Overview

Caution: If your environment uses Core and Satellite servers, first read the *Radia Client Automation Enterprise User Guide* as the installation, configuration, and troubleshooting information in that guide may override the information in this guide.

This chapter provides an introduction to some of the important components, concepts, and terms that are relevant to SSL encryption. This chapter also discusses SSL in the context of RCA, including:

- SSL cipher-suite information
- SSL encryption requirements
- A list of the RCA products that can be configured to use SSL

An Introduction to SSL Encryption

Secure Sockets Layer (SSL) is a cryptographic protocol that enables software applications to communicate securely across a network. SSL is designed to prevent eavesdropping, tampering, and message forgery. It is based on a principle named mutual authentication, which ensures that both parties in a conversation know precisely with whom they are communicating.

This section describes some of the components, concepts, and terms that are part of SSL encryption.

Public Key Cryptography

SSL implements mutual authentication by using public key cryptography. A key is simply a binary code, encoded and served in a text file, and associated with a particular user or software application.

SSL uses two keys—a public key and a private key—to encrypt and decrypt messages that are sent over the network. The public key is given freely to interested parties, but the associated private key remains private and is possessed only by the owner of the certificate. Data encrypted with the public key can be decrypted only with the private key.

In the context of RCA, SSL public key cryptography includes:

- A private-public key pair on the server
- A certificate from a trusted Certificate Authority (CA). This is typically already present on each system.

Each of these requirements is described in the next sections.

The Key Pair

SSL encryption uses a **key pair** to encrypt a transmission. The key pair consists of a **private key** and a **public key**.

- **Private Key**: In the context of RCA, a key pair must be generated for each server. The server retains the **private key** and must keep it secure.
- **Public Key**: The **public key** is passed to the client by the server. The client must trust that the public key that it receives is truly from the server that the client thinks it is communicating with. Certificate Authorities sign public keys so that the keys can be trusted.

Certificates

A certificate is an electronic document that contains the server's public key, the server name, and a signature from a CA. A certificate authority is a trusted third party who attests that the public key in the certificate belongs to the party – in this case, the server – named in the certificate. It is the responsibility of the CA to verify the credentials of any party that applies for a certificate. This allows others to trust the information in the certificates issued by this CA.

There are independent CAs, such as Thawte and Verisign, who charge a fee for their services. There are free CAs also.

A client is configured with certificates from the CAs that it trusts. As long as the server's certificate has been signed by a CA that the client trusts, the server's certificate is considered "trusted," and SSL communications between the server and client can be initiated.

Certificates and Your Environment

This section discusses certificate generation in production and test environments.

Production Situations

It is best to generate a Certificate Signing Request that can be signed by a trusted Certificate Authority.

Test Situations

You can provide either:

- A **self-signed certificate**
  In this case, you must configure the client to trust each server's certificates.

- A **private CA-signed certificate**
  In this case, you can sign each server's certificate quickly—because you are the signing authority—and you only need to configure your clients to trust the private CA certificate that you generated.

Ciphers and Hash Functions

A cipher is a method of encrypting information. A hash function is a method of compressing information that transforms data into a short, fixed-length string that serves as a digital "fingerprint." Hash functions used in cryptography create a unique fingerprint for every input and work in one direction only; in other words, you cannot derive the original data from the fingerprint. Ciphers and hash functions are used by SSL.

SSL can use a number of ciphers and hash functions to encrypt messages. It uses two ciphers and one hash function for each connection. Together, the two ciphers and hash function are known as the cipher suite, and they are used to establish and protect that connection.

Keystores and Truststores

For two-way SSL communication, the server and each client must have a truststore and a keystore.

- A keystore is a database that stores your private keys. It also contains certificates for trusted CAs.
- A truststore stores the public keys that you trust.

The keystore and the truststore are typically implemented as files. A keystore file is protected by a password. A truststore file needs no password because it contains no private information.

Setting up SSL

To set up SSL on each device that is authenticated:

1. Locate (or create) a keystore.
2. Generate a public-private key pair.
3. If this new key pair is not yet trusted—in other words, if the public key that you generated is not yet in your keystore—follow these steps:
   a. Generate a **Certificate Signing Request** (CSR) from the key pair.
   b. Send the CSR signed to a trusted CA.
   c. When the CA issues a signed certificate in response to your request, import the signed certificate that they send you into the keystore.
4. Configure the client and server to use the public-private key pair certificates.

How SSL Establishes a Secure Connection

A client and a server establish a secure connection by performing a handshake operation. The handshake accomplishes the following:

- The client and server agree on a cipher suite to use for the connection.
- The server sends its certificate—including public key, server name, and CA—to the client. The client can then contact the CA to verify the server's identity. If mutual authentication is required, the server will also request a certificate from the client.
- The client and server generate session keys that will be used for the duration of this connection.
- The client encrypts a random number with the server's public key, and sends the result to the server.
- The server decrypts the random number with its private key, which hides the session keys from third parties, since only the server and the client can access this data.
- The client and server generate session keys that they use for encryption and decryption.

SSL in Radia Client Automation Environment

This section presents information required to set up and use SSL in Radia Client Automation environment. It provides an overview of the protocols that are used to secure the various RCA server-RCA agent communications.

Note: RCA supports up to five levels of CA certificates. This means up to five CAs can be used in RCA environment.

SSL Requirements

To ensure that SSL encryption works with the RCA products, the following requirements must be met.

- RCA servers must have a public key, a private key, and a Certificate Authority public key.
- RCA agents must have a Certificate Authority public key.

SSL Cipher Suite and Encryption Information

The ciphers can be configured for the Apache Server, Configuration Server, and TCL based web servers.

Configuring Ciphers for the Apache Server

Follow these steps to configure ciphers for the Apache Server:

1. Use a text editor to open the `httpd-ssl.conf` file available at the location: `<InstallDir>\ApacheServer\conf\extra`.
2. Modify the parameter `SSLCipherSuite` and set it according to your enterprise requirements. The default value for this parameter is set as `ALL:!ADH:!EXP:!NULL:+HIGH:+MEDIUM:-LOW`.

Configuring Ciphers for the Configuration Server

Follow these steps to configure ciphers for the Configuration Server:

1. Use a text editor to access the `edmprof.dat` file, located in the `bin` folder of the Configuration Server directory.
2. Under section `MGR_SSL`, set the `SSL_CIPHERS` according to your enterprise requirements. The default value for `SSL_CIPHERS is ALL:!ADH:!EXP:!NULL:+HIGH:+MEDIUM:-LOW`.

Configuring Ciphers for the TCL Based Web Server

Follow these steps to configure ciphers for the TCL Based Web Servers:

1. Use a text editor to access the respective configuration file.
2. Set the parameter `set tls::defaults(-ciphers)` according to your enterprise requirements. The default value for `set tls::defaults(-ciphers)` is `ALL:ADH:RC4+RSA:+HIGH:+MEDIUM:+LOW:+SSLv2:+EXP:+eNULL`.

SSL Version Parity

To communicate using SSL, RCA servers and agents must use the same version of SSL. This version of RCA supports version 3.0 of the SSL protocol and version 1.0 of the TLS protocol.

Abbreviations and Variables

Abbreviations Used in this Guide

Abbreviation	Definition
RCA	Radia Client Automation
Core and Satellite	RCA Enterprise environment consisting of one Core server and one or more Satellite servers.
CSDB	Configuration Server Database

Variables Used in this Guide

Variable	Description	Default Values
InstallDir	Location where the RCA server is installed	For a 32-bit OS: `C:\Program Files\Hewlett-Packard\HPCA` For a 64-bit OS: `C:\Program Files(x86)\Hewlett-Packard\HPCA`
SystemDrive	Drive label for the drive where the RCA server is installed	C:

Summary

- Secure Sockets Layer (SSL) is a cryptographic protocol that enables *secure communications* across a network.

- SSL implements mutual authentication by using *public key cryptography*.
- SSL uses a *public key* and a *private key* to encrypt and decrypt messages.
- A certificate must be obtained from a *Certificate Authority*. It contains:
  - The *server's public key*,
  - The *server name*, and
  - A *signature* from a Certificate Authority.
- SSL uses *ciphers* and *hash functions* to encrypt messages.
- Two-way SSL communication requires each server and each client to have a *truststore* and a *keystore*.
- To ensure SSL encryption viability with the RCA products:
  - RCA servers must have a *public key*, a *private key*, and a *CA public key*.
  - RCA agents must have a *CA public ke*y.
- SSL ciphers can be configured for the Apache Server, Configuration Server, and TCL based web servers.

Chapter 2

Setting up Certificates for SSL

> **Note:** If you are already creating certificates in your environment with existing tools, skip to the next chapter "Configuration and Use" on page 21

The Certificate Generation Utility

For testing, a **Certificate Generation Utility** has been provided. This utility makes it easy to create self-signed certificates for testing. It will be used in this chapter to demonstrate the process for setting up SSL for RCA.

> **Caution:** This utility is intended for testing purposes *only* and should **not** be used in a production environment.

Before using this utility, consider the following:

- The Certificate Generation Utility is **not** a supported Radia Client Automation product.
- The Certificate Generation Utility is provided *free of charge*.
- The Certificate Generation Utility is used at *your own discretion*; Technical Support **will not** address any issues regarding its use or functionality.

> **Note: Certificate Generation Utility** can generate certificates on Windows platforms only. However, after the certificates are generated on a Windows system, the same can be copied over to and used on Linux platforms.

Locating the Certificate Generation Utility

The Certificate Generation Utility can be found at the following location:

`<InstallDir>\Tools`

Setting up a Certificate

The first step required to set up SSL is to make sure that each system that will be authenticated has a private key and a signed certificate. If you already have a private key for this system, you can use it to generate a certificate.

Using an Existing Private Key

If you already have a private key in PEM file format, follow these steps:

1. In the `Tools\servers` directory, create a new directory named `hostname`.
   In this case, hostname is the name of the server for which a signed certificate is to be created.
   For example:

   `Tools\servers\cmserver1`

2. Copy your private key `PEM` file into the directory that you just created.
3. Rename the `PEM` file that you just copied as follows: *`hostname`*`-prvkey.pem`
   For example:

   `Tools\servers\cmserver1\cmserver1-prvkey.pem`

Generating a Signed Certificate

This section provides instructions for creating signed certificates that will be used for SSL configuration. There are three ways that you can generate a signed certificate:

- *Self-signing*
  This is the most convenient but least secure of the three options. Use this strictly for testing.
- Via a *generated* CA
  This option is more secure than self-signing, but not secure for a production environment. It creates a new CA with the parameters that you specify, and this new CA signs the certificate.
- Via a *trusted* CA
  This is the most secure option of the three. In a production environment, be sure to use certificates that have been signed by a trusted CA.

This task will use the Certificate Generation Utility to demonstrate each of these three options.

Server Names

When you generate a certificate or a certificate request, you must specify the server name. You can use the simple host name (for example, `cmserver1`) or the fully qualified host name (for example, `cmserver1.mycorp.com`).

The name that you specify should match the name that will be used in the URL when this server is accessed.

Option 1: Generating a Self-Signed Certificate

1. From the `certificate_mgmt` directory, run the following command:
   `cert_mgr create self`
2. Provide the following information at the prompts:

Parameter	Example
Server name (becomes the CN)	`cmserver1`
Country name	`US`
State or province name	`California`

Parameter	Example
Locality Name	`Sacramento`
Organization name	`Mycompany`
Organizational unit name	`IT`

This information is used to create the **Distinguished Name (DN)** for the certificate. The DN is a unique identifier that is used to provide a name that is unique to the certificate. The DN is derived from the **Common Name (CN)** of the server and the other parameters that you specify.

The components of the DN, including the CN, are visible in the `cert.txt` file, as shown here:

```
Subject: C=US, ST=CA, L=Sacramento, O=MyCompany, OU=IT,
CN=cmserver1
```

Note: It is important that the CN part of the certificate's DN be the same as the server's host name. This is vital to the client trusting that it is communicating with the expected host.

After the utility finishes, the server certificate, private key, and related files are located in the `certificate_mgmt\servers\`*`hostname`* directory. For example, if the server name entered is `cmserver1`, the following files are generated:

```
certificate_mgmt\servers\cmserver1\cmserver1-cert.pem
certificate_mgmt\servers\cmserver1\cmserver1-cert.txt
certificate_mgmt\servers\cmserver1\cmserver1-prvkey.pem
certificate_mgmt\servers\cmserver1\cmserver1-signer.pem
certificate_mgmt\servers\cmserver1\cmserver1-signer.txt
certificate_mgmt\servers\cmserver1\cmserver1-cert.rnd
certificate_mgmt\servers\cmserver1\cmserver1-keystore.txt
certificate_mgmt\servers\cmserver1\cmserver1-keystore.jks
certificate_mgmt\servers\cmserver1\cmserver1-truststore.txt
certificate_mgmt\servers\cmserver1\cmserver1-truststore.jks
```

Note: The `keystore` and `truststore` files are generated only when the JAVA_HOME environment variable points to a Java runtime environment (JRE).

After you have verified that your files were correctly generated, proceed to "Configuration and Use" on page 21.

Caution: Self-signed certificates are adequate for testing purposes *only*; they should **not** be used in a production environment.

Option 2: Generating a Re-usable Certificate Signed by a Generated CA

1. From the `certificate_mgmt` directory, run the following command.
   `cert_mgr create signed`
2. Provide the following information at the prompts.

Parameter	Example
Server name (becomes the CN)	`cmserver1`
Country name	`US`
State or province name	`California`
Locality Name	`Sacramento`
Organization name	`Mycompany`
Organizational unit name	`IT`

Note: The first time you run this command, you will also be prompted for information about the certificate authority (CA) that you are generating. On subsequent runs, it will not prompt you.

The utility generates two sets of files in this case.

- The first set consists of the server certificate and related files, which are located in the `certificate_mgmt\servers\`*`hostname`* directory (see the description of the files under "Option 1: Generating a Self-Signed Certificate" on page 14.
- The second set consists of the CA files. These are located in the `certificate_mgmt\ca` directory.

For example, if the server name is specified as `cmserver1`, the following files are generated:

```
certificate_mgmt\servers\cmserver1\cmserver1-cert.pem
certificate_mgmt\servers\cmserver1\cmserver1-cert.txt
certificate_mgmt\servers\cmserver1\cmserver1-prvkey.pem
certificate_mgmt\servers\cmserver1\cmserver1-signer.pem
certificate_mgmt\servers\cmserver1\cmserver1-signer.txt
certificate_mgmt\servers\cmserver1\cmserver1-cert.rnd
certificate_mgmt\servers\cmserver1\cmserver1-keystore.txt
certificate_mgmt\servers\cmserver1\cmserver1-keystore.jks
certificate_mgmt\servers\cmserver1\cmserver1-truststore.txt
certificate_mgmt\servers\cmserver1\cmserver1-truststore.jks
certificate_mgmt\ca\ca.rnd
```

```
certificate_mgmt\ca\ca-cert.pem
certificate_mgmt\ca\ca-prvkey.pem
certificate_mgmt\ca\ca-index.txt
certificate_mgmt\ca\ca-index.txt.attr
certificate_mgmt\ca\ca-index.txt.old
certificate_mgmt\ca\ca-serial
certificate_mgmt\ca\ca-serial.old
```

Note: When you use the signed option, the Signing Authority Certificate is copied from the `certificate_mgmt\ca` directory. If the `certificate_mgmt\ca\ca-cert.pem` file already exists, that file will be used. Otherwise, it will be created on the first run and used for generating subsequent certificates.

Note: The `keystore` and `truststore` files are generated only when the JAVA_HOME environment variable points to a Java runtime environment (JRE).

After you have verified that your files were correctly generated, proceed to "Configuration and Use" on page 21.

Caution: This method of generating signed certificates is adequate for testing purposes only and should not be used in a production environment.

Option 3: Generating a Certificate Signed by a Trusted CA

These steps show you how to use the Certificate Generator Utility to generate a private key and certificate request that you can then send to a trusted CA. This might be an external CA, such as Verisign or Thawte, or a CA that your company or institution owns and administers.

1. From the `certificate_mgmt` directory, run the command
   `cert_mgr create request`
2. Provide the following information at the prompts:

Parameter	Example
Server name (becomes the CN)	`cmserver1`
Country name	`US`
State or province name	`California`
Locality Name	`Sacramento`
Organization name	`Mycompany`

Parameter	Example
Organizational unit name	`IT`

After the utility finishes, the certificate request, private key, and related files are located in the `certificate_mgmt\servers\hostname` directory. For example, if the server name is specified as `cmserver1`, the following files are generated:

```
certificate_mgmt\servers\cmserver1\cmserver1.rnd
certificate_mgmt\servers\cmserver1\cmserver1-prvkey.pem
certificate_mgmt\servers\cmserver1\cmserver1-request.pem
certificate_mgmt\servers\cmserver1\cmserver1-request.txt
```

3. Request a signed certificate by sending the *hostname*`-request.pem` to your signing authority.

   Note: Be sure that the server certificate that is purchased is a **base-64 encoded x.509** certificate. This is typical for certificates that are generated for the Apache Freeware (ModSSL or OpenSSL) Server.

4. When you receive this signed certificate from your signing authority, paste it into the `servers\`*hostname*`\`*hostname*`-cert.pem` file.
5. Paste the Signing Authority Certificate (must be in PEM format) into the `servers\`*hostname*`\`*hostname*`-signer.pem file`.

You now have a private key, a signed certificate, and the signing authority certificate files that are needed for product configuration.

Additional Information about Certificates and Keys

This section provides information about installing the private key file.

Installing the Private Key File

The Certificate Generation Utility also generates a private key in the form of the following PEM file:

hostname`-prvkey.pem`.

To install the private key, place this file in the appropriate directory on the server. See "Configuration and Use" on page 21 for the specific location of this directory for each type of RCA server.

If you open the private key file with a text editor, the contents will look similar to the following:

```
-----BEGIN RSA PRIVATE KEY-----
Proc-Type: 4,ENCRYPTED
DEK-Info: DES-CBC,6EC0947550541AAB

1MV8Y4rkywlYn30yUB5ULtKLfj0YSzX+KZvxCeuw+9x95x1Ikvej4b8iBDuEOaTR
MIIDZTCCAs6gAwIBAgIBADANBgkqhkiG9w0BADQFADCBhDELMAkGA1UEBhMCVVMx
MzMxNDJaMIGEMQswC7YDVQQGEwJVUzELMAkGA1UECBMCQ0ExEDAOBgNVBAcTB0Zy
```

```
ZW1vbnQxDzANBgNVBAoTBlN5Z2F0ZTEQMA4GA1UECxMHQWx0dmlldzEQMA4GA1UE
H1OkihMe0Ny94uj8a6ccMJ+1kRj2grVmaw8tJi+6G76NXhvZvwumfHZMtnhKUKth
Mf3XLTukzlz5LqoVJzuDoLQVcm7Ddx0iff+FLwRhsjl53KQqoRYucLOopirXYc6R
8T+XMo3tkd4q=
-----END RSA PRIVATE KEY-----
```

In order to maintain compatibility with industry standards, The RSA crypto-system method of obtaining certificate requests has been adopted. The RSA crypto-system is a public key crypto-system that offers encryption and digital signatures (authentication). In the private key shown above the key type (**RSA**) is indicated at the beginning and end of the file.

Summary

- A *Certificate Generation Utility* has been provided that makes it easy to create self-signed certificates.
- There are three ways to generate a signed certificate:
  - *Self-signing*
  - Via a *generated CA*
  - Via a *trusted CA*

Chapter 3

Configuration and Use

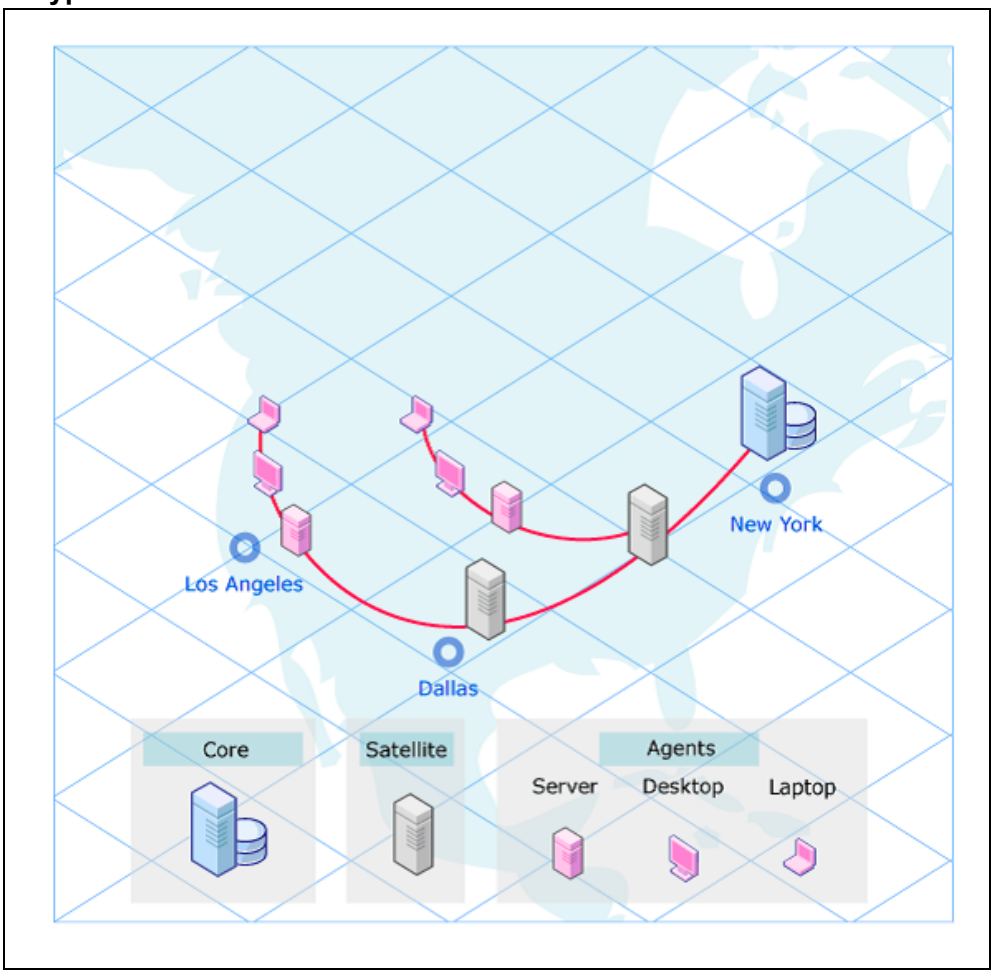
Caution: If your environment uses Core and Satellite servers, first read the *Radia Client Automation Enterprise User Guide* as the installation and configuration information in that guide may override the information in this guide.

Core and Satellite Overview

This section describes how to implement SSL functionality in your Core and Satellite RCA environment to secure the communications between RCA servers and RCA agents.

The consolidated Core and Satellite configuration simplifies configuration of SSL certificates.

A Typical RCA Environment



Configuring Core and Satellite with SSL Certificates

A single screen is used to configure SSL certificates in the RCA Core and Satellite environment. Follow these steps to enable and configure SSL for the RCA Server:

1. In the rCA Console, click **Configuration** tab.
2. In the left pane, click **Infrastructure Management** > **SSL**.
3. In the SSL Server area, select the **Enable SSL** check box.
4. Select whether to **Use existing certificates** or **Upload new certificates**.
   If Upload new certificates is selected, click **Browse** to navigate to and select Private Key File name and Server Certificate File name.
5. Click **Save**.

RCA Agents

Secure (SSL) communications are supported on the following RCA agents.

- Radia Client Automation Application Manager agent (Application Manager)
- Radia Client Automation Application Self-service Manager agent (Application Self-service Manager), see "RCA Application Self-service Manager Agent" below.
- Radia Client Automation Inventory Manager agent (Inventory Manager)
- Radia Client Automation Patch Manager agent (Patch Manager agent)

To enable SSL communications with a Configuration Server for these RCA agents, pass `SSLMGR` and `SSLPORT` with the appropriate values on a `RADSKMAN` command line, as in:

```
Radskman sslmgr=host,sslport=443
```

RCA Application Self-service Manager Agent

For the Application Self-service Manager, setup `sslmanager` and `sslport` tags in the `ARGS.XML` file, as in:

```
<SSLMANAGER>localhost</SSLMANAGER>

<SSLPORT>443</SSLPORT>
```

Configuring Two-Way SSL

RCA enables you to use two-way authentication to ensure secure communication between the server and client devices. The client devices authenticate themselves to the server devices and server devices authenticate themselves to the client devices.

Complete the following tasks to enable two-way SSL in your environment:

1. Enable two-way SSL on Core or Satellite server.
2. Enable two-way SSL on Configuration server.
3. Enable two-way SSL on the RCA agent.

Enabling Two-Way SSL on Core or Satellite Server

Complete the following steps to enable two-way SSL on Core or Satellite server to which the agent connects:

1. Navigate to the `<InstallDir>\ApacheServer\conf\extra` directory.
2. Using a text editor, open the `httpd-ssl.conf` file.
3. Uncomment the following parameters:
   - `SSLVerifyClient require`
   - `SSLVerifyDepth 10`
4. Save and close the file.

Enabling Two-Way SSL on Configuration Server

Complete the following steps to enable two-way SSL on the Configuration server to which the agent connects:

1. Navigate to the `<InstallDir>\ConfigurationServer\bin` directory.
2. Using a text editor, open the `edmprof.dat` file.
3. Locate the `MGR_SSL` section.
4. Add the parameter, `VERIFY_CLIENT` and set it to `Y`.
5. Save the file.
6. Restart the **ConfigurationServer** service.

Enabling Two-Way SSL on RCA Agent

Complete the following steps to enable two-way SSL on RCA agent:

1. Verify that the CA root certificate exists in the `<InstallDir>\Agent\CACertificates` as `cacert.pem`.
   If it does not exist, navigate to the `<InstallDir>\ApacheServer\conf\ssl.crt` directory on Core server, copy the CA root certificate, `ca-bundle` to the `\CACertificates` directory, and rename it to `cacert.pem`.
2. Create a new certificate for client on the Core server. By default, the certificate is created in the `<InstallDir>\Tools\servers` directory.
3. Create a new directory, `Certificates` in `IDMSYS` directory on the RCA agent.
4. Copy the client signed certificate and the private key from the Core server to the `Certificates` directory on RCA agent.

5. When running the agent connect, add the following parameters to the command line:
   - cert=<*name of client signed certificate file*>
   - key=<*name of private key file*>
   - sslmgr=<*IP address of the SSL manager*>
   - sslport=<*port number of the SSL manager*>

   For example:

```
radntfyc localhost radskman
cat=prompt,uid=$machine,sname=POWERPOINTVIEW,log=connect_
Soft-
ware.log,logsize=2048000,cop=y,cert=hp1.pem,key=hp1key.pem,sslmgr=
<IP address>,sslport=444
```

Summary

- You can configure Core and Satellite with SSL certificates.
- You can enable secure communication on RCA agents.
- You can configure two-way SSL for secure communication between server and agent.

Appendix A

Command Line Options for the Certificate Generation Utility

The tables Options for cert_mgr create and Options for cert_mgr import describe the options that can be used with the `cert_mgr create` and `cert_mgr import` commands that are described in Chapter 2, "Setting up Certificates for SSL" on page 13.

Options for cert_mgr create

Option	Description	Default
`-hostname`	Host name of the server for which you will create the certificates.	Simple host name of the system on which you are running `cert_mgr`.
`-trustpass`	The password for the truststore.	changeit
`-rndbytee`	Size of the random bytes when creating the random file that will be used to create the private key for the server certificate.	2048 bytes
`-keysize`	Size of the server's private key in bits.	1024 bits
`-keypass`	The password for the server's certificate when it is added to the keystore.	secret
`-days`	The number of days the server's certificate will be valid.	9999 days
`-carndbytes`	Same as `rndbytes`, but for the CA.	2048 bytes
`-cakeysize`	Same as `keysize`, but for the CA.	1024 bytes
`-cadays`	Same as `days`, but for the CA.	9999 days

Options for cert_mgr import

Option	Description	Default
`-hostname`	Host name of the server for which you will create the certificates.	Simple host name of the system on which you are running `cert_mgr`.
`-sig-nedcert`	The fully qualified path and file name of the signed certificate that was returned by the CA.	

Option	Description	Default
`-sign-ercert`	The fully qualified path and file name to the certificate of the signing CA. Used when importing a certificate via the Certificate Generation Utility.	

We appreciate your feedback!

If an email client is configured on this system, by default an email window opens when you click here.

If no email client is available, copy the information below to a new message in a web mail client, and then send this message to radiadocfeedback@persistent.co.in.

Product name and version: Radia Client Automation Enterprise, 9.00

Document title: SSL Implementation Guide

Feedback: