Connect:Direct® Secure+ Option for OpenVMS

Implementation Guide

Version 3.5

Sterling Commerce
An IBM Company

Connect:Direct Secure+ Option for OpenVMS Implementation Guide
Version 3.5

First Edition

CDS+OVMSIG1011

Contents

Chapter 1

About Connect:Direct Secure+ Option for OpenVMS

Secure+ Option, which is part of Connect:Direct for OpenVMS, provides enhanced security for Connect:Direct using cryptography to secure data during transmission. You select the security protocol (SSL or TLS) to use with the Secure+ Option product.

Security Concepts

Cryptography is the science of keeping messages private. A cryptographic system uses encryption keys between two trusted communication partners. These keys encrypt and decrypt information so that the information is known only to those who have the keys.

There are two kinds of cryptographic systems: symmetric-key and asymmetric-key. Symmetric-key (or secret-key) systems use the same secret key to encrypt and decrypt a message. Asymmetric-key (or public-key) systems use one key (public) to encrypt a message and a different key (private) to decrypt it. Symmetric-key systems are simpler and faster, but two parties must somehow exchange the key in a secure way because if the secret key is discovered by outside parties, security is compromised. Asymmetric-key systems, commonly known as public-key systems, avoid this problem because the public key may be freely exchanged, but the private key is never transmitted.

Cryptography provides information security as follows:

- Authentication verifies that the entity on the other end of a communications link is the intended recipient of a transmission.
- Non-repudiation provides undeniable proof of origin of transmitted data.
- Data integrity ensures that information is not altered during transmission.
- Data confidentiality ensures that data remains private during transmission.

Secure+ Option enables you to select one of two security protocols to use to secure data during electronic transmission: Transport Layer Security (TLS) or Secure Sockets Layer protocol (SSL).

Secure Sockets Layer Protocol (SSL) and Transport Layer Security Protocol (TLS)

The SSL and the TLS protocols use certificates to exchange a session key between the node that initiates the data transfer (the primary node, or PNODE) and the node with which the PNODE wants to set up a secure connection (the secondary node, or the SNODE). A certificate is an electronic document that associates a public key with an individual or other entity. It enables you to verify the claim that a given public key belongs to a given entity. Certificates can be self-issued or issued by a certificate authority (see *Self-Signed and CA-Signed Certificates* on page 14 for details). When a CA receives an application for a certificate, the CA validates the applicant's identity, creates a certificate, and then digitally signs the certificate. A certificate authority (CA) issues and revokes CA-issued certificates. Self-signed certificates are created and issued by the owner of the certificate, who must export the certificate in order to create a trusted root for the certificate and supply the trusted root of the self-signed certificate to the partner in a connection.

For more information on how Secure+ Option uses certificate information, see Chapter 2, *Set Up Secure+ Option to Use Certificates,* and review *Example of Setting Up and Processing Certificates* on page 18.

Connect:Direct Secure+ Option for OpenVMS Tools

Connect:Direct Secure+ Option for OpenVMS consists of two components: the Secure+ Administration tool and the Secure+ Option parameters file. The following sections describe these components and their function within Connect:Direct Secure+ Option for OpenVMS.

Administration Tool

The SPADMIN Tool enables you to configure and maintain the Connect:Direct Secure+ Option for OpenVMS environment. The SPADMIN Tool is the only interface for creating and maintaining the Secure+ Option parameters file (ndm$$directory:secnetfile.dat); operating system utilities and editing tools do not work. You can run the SPADMIN tool as an interactive script and choose options from a main menu or as a command line task. For more information, see Chapter 3, *Use the Secure+ Admin Tool and Populate the Secure+ Option Parameters File* and Chapter 6, *Use CLI to Maintain the Secure+ Option Parameters File*.

Secure+ Option Parameters File

The Secure+ Option parameters file (ndm$$directory:secnetfile.dat) contains information that determines the protocol and encryption method used during security-enabled Connect:Direct operations. To configure Secure+ Option, each site must have a parameters file that contains one local node record and at least one remote node record. The local node record defines the most commonly used security and protocol settings at the site and can be used as a default for one or more remote node records. Each remote node record defines the specific security and protocol used by a trading partner. You should create a remote node record in the Secure+ Option parameters file for each (adjacent) remote node defined in your network map to keep these two files in sync. For more information including detailed step-by-step procedures on how to configure the local node record and remote node records, see Chapter 4, *Configure Nodes for Secure+ Option*.

For additional security, the Secure+ Option parameters file is stored in an encrypted format.

Use Certificates to Authenticate Trading Partners

The SSL and TLS protocols provide three levels of authentication:

- During the first level of authentication called server authentication, the site initiating the session (PNODE) requests a certificate from its trading partner (SNODE), during the initial handshake. The SNODE returns its ID certificate (read from its key certificate file) and the PNODE authenticates it using one or more trusted root certificates stored in a trusted root certificate file (the name and location of which are specified in the remote node record for that specific trading partner in the PNODE's Secure+ Option parameters file). Root certificates are signed by a trusted source—either a public certificate authority, such as Thawte, or by the trading partner acting as its own CA. If the ID certificate from the SNODE cannot be validated using any root certificate found in the trusted certificate file, or if the root certificate has expired, the PNODE terminates the session. Connect:Direct writes entries to the statistics logs of both nodes, and the session is aborted.
- The second level of authentication is optional and is called client authentication. If this option is enabled in the SNODE's Secure+ Option parameters file definition for the PNODE, the SNODE will request a certificate from the PNODE, and authenticate it using the information in its trusted root certificate file. If this authentication fails, the SNODE terminates the session and Connect:Direct writes information about the failure to the statistics logs of both nodes.
- The third authentication level is also optional and consists of validating the PNODE's certificate common name. When the security administrator enables client authentication, they can also specify the common name contained in the PNODE's ID certificate. During client authentication, the SNODE compares the common name it has specified for the PNODE in its Secure+ Option Parameters file with the common name contained in the certificate sent by the PNODE. If the compare fails, that is, the information is not identical, the SNODE terminates the session, and Connect:Direct writes information about the failure to the statistics logs of both nodes.

The SSL and TLS protocols provide data security in the following areas:

- Authentication—Because the CA went through an established procedure to validate an applicant's identity, users who trust the CA can be sure the key is held by the owner. The CA prevents impersonation, and provides a framework of trust in associating an entity with its public and private keys.
- Proof of data origin and data integrity validation—The certificate provides proof of origin of electronic transmission and encryption validates data integrity. Message digest (hashing) and encrypting the message digest ensure that the data is not altered.
- Data confidentiality—Cipher suites encrypt data and ensure that the data remains confidential. The sending node converts sensitive information to an unreadable format (encryption) before it is sent to the receiving node. The receiving node then converts the information back into a readable format (decryption).

Both the SSL protocol and the TLS protocol manage secure communication in a similar way. However, TLS provides a more secure method for managing authentication and exchanging messages, using the following features:

- While SSL provides keyed message authentication, TLS uses the more secure Key-Hashing for Message Authentication Code (HMAC) to ensure that a record cannot be altered during transmission over an open network such as the Internet.
- TLS defines the Enhanced Pseudorandom Function (PRF), which uses two hash algorithms to generate key data with the HMAC. Two algorithms increase security by preventing the data from being changed if only one algorithm is compromised. The data remains secure as long as the second algorithm is not compromised.
- While SSL and TLS both provide a message to each node to authenticate that the exchanged messages were not altered, TLS uses PRF and HMAC values in the message to provide a more secure authentication method.
- To provide more consistency, the TLS protocol specifies the type of certificate that must be exchanged between nodes.
- TLS provides more specific alerts about problems with a session and documents when certain alerts are sent.

Summary of Processing Using the TLS or SSL Protocol

After you configure Secure+ Option, you are ready to exchange datSSL protocol:data exchangea securely with other security-enabled Connect:Direct nodes. Your node must also be defined in the Secure+ Option parameters file (secnetfile.dat) of your trading partner. Data is securely exchanged between two nodes using the protocol defined in the parameters files.

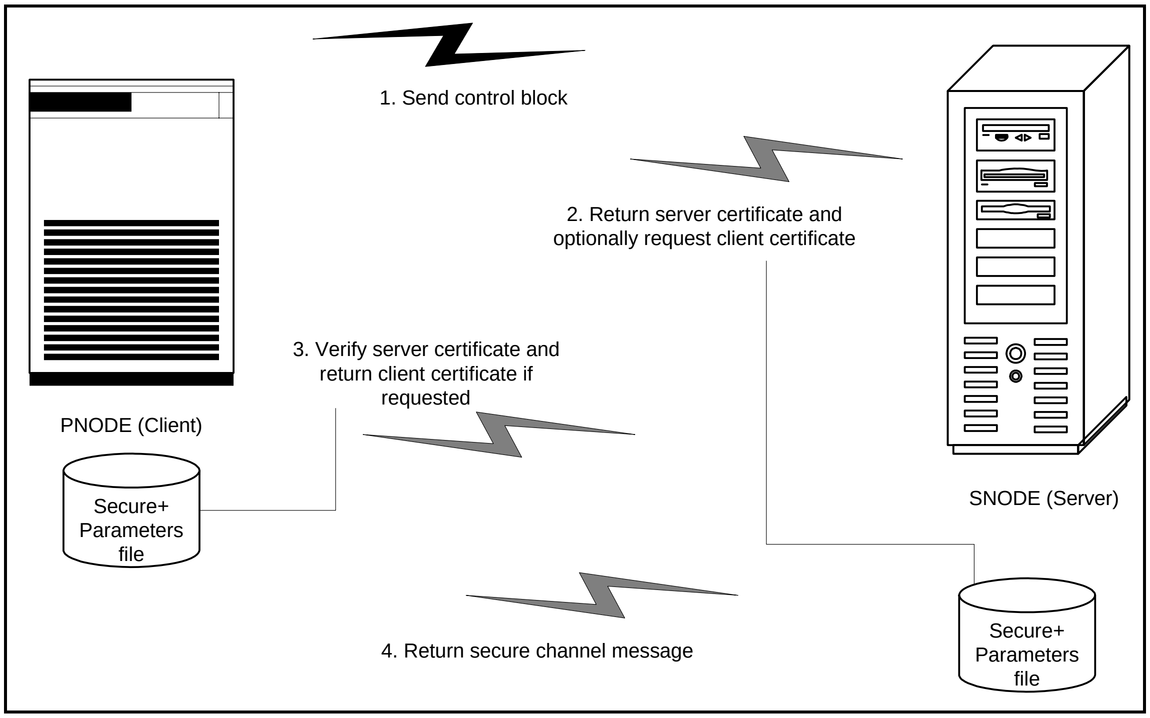
The following sections describe what happens during a data exchange between two Connect:Direct nodes using Secure+ Option with the TLS or SSL protocol.

Secure+ Option Data Exchange

Data exchange consists of two processes: authentication and sending/receiving data. The TLS or SSL protocol data exchange process is described in the following sections. In the illustration below, the primary node (PNODE) is sending data while the secondary node (SNODE) is receiving it.

Authentication

The following figure illustrates the authentication process using the TLS or SSL protocol:


1. Send control block
2. Return server certificate and optionally request client certificate
3. Verify server certificate and return client certificate if requested
PNODE (Client)
Secure+ Parameters file
SNODE (Server)
4. Return secure channel message
Secure+ Parameters file

The following steps occur during authentication:

1. The PNODE (client) sends a control block containing protocol (TLS or SSL) and cipher information to the SNODE (server). The SNODE confirms that it has a record defined in its Secure+ Option parameters file for the PNODE, and determines if a common cipher can be found and used for secure communication. Cipher suites are used to encrypt the data being sent between nodes. If the SNODE finds a record for the PNODE in its Secure+ Option parameters file and verifies it has a cipher defined in common with the PNODE, a common cipher is negotiated and the session continues.
2. The SNODE sends its ID certificate to the PNODE who confirms that it has a record defined in the Secure+ Option parameters file. Information for creating a public key is included. The PNODE verifies the ID certificate of the SNODE using the trusted root certificate file defined in its Secure+ Option parameters file, and generates a session key.
3. If client authentication is enabled on the SNODE, the SNODE requests an ID certificate from the PNODE. The PNODE sends its ID certificate defined in its Secure+ Option parameters file to the SNODE for verification against the trusted root certificate file specified in the SNODE’s Secure+ Option parameters file. If a common name was also specified in the Secure+ Option parameters file for the PNODE, this name is used to verify the common name field of the PNODE's certificate.
4. The SNODE confirms that a secure environment is established and returns a secure channel message.

Send/Receive Customer Data

Once a Secure+ session has been established, all control blocks and customer data transmitted between the PNODE and SNODE are encrypted using the negotiated cipher.

Connect:Direct Documentation

See *Connect:Direct for OpenVMS Release Notes* Version 3.4 for a complete list of the product documentation.

About This Guide

Connect:Direct Secure+ Option for OpenVMS Implementation Guide is for programmers and network operations staff who install, configure, and maintain the Connect:Direct Secure+ Option for OpenVMS product.

This guide assumes knowledge of the OpenVMS operating system, including its applications, network, and environment. If you are not familiar with the OpenVMS operating system, refer to the OpenVMS library of manuals.

Task Overview

The following table guides you to the information required to perform Secure+ Option tasks:

Task	**Reference**
Understanding Secure+ Option	Chapter 1, *About Connect:Direct Secure+ Option for OpenVMS*
Planning the Secure+ Option implementation	Chapter 2, *Set Up Secure+ Option to Use Certificates* Chapter 3, *Use the Secure+ Admin Tool and Populate the Secure+ Option Parameters File* Appendix A, *Configuration Worksheets*
Obtaining a certificate and generating a key certificate file to use with Secure+ Option	Chapter 2, *Set Up Secure+ Option to Use Certificates* Appendix B, *Understanding the Certificate File Layout*
Exchanging public keys or trusted root certificate files with your trading partners	*Exchange Certificate Information with Trading Partners* on page 17
Populating the Secure+ Option parameters file with node information and becoming familiar with the Secure+ Admin tool	Chapter 3, *Use the Secure+ Admin Tool and Populate the Secure+ Option Parameters File*
Configuring the Secure+ Option local node record and remote node records	Chapter 4, *Configure Nodes for Secure+ Option*
Performing routine Secure+ Option maintenance tasks such as viewing and deleting node records	Chapter 5, *Maintain Secure+ Option Nodes*
Using the Command Line Interface to manually maintain Connect:Direct Secure+ Option for OpenVMS	Chapter 6, *Use CLI to Maintain the Secure+ Option Parameters File*
Viewing Secure+ Option statistics	Chapter 7, *Secure+ Option Statistics*

Chapter 2

Set Up Secure+ Option to Use Certificates

As part of the planning process, you must determine the type of certificates your trading partners are using and what to use for your system. In addition, you must plan which method to use for generating and validating certificates for nodes that use the SSL or TLS protocol for secure communications. Before you configure Connect:Direct Secure+ Option, review the following concepts, requirements, terms, and tool descriptions to ensure that you have all the resources and information necessary to implement the Transport Layer Security (TLS) protocol or the Secure Sockets Layer (SSL) protocol.

Before You Begin

Identify Expert Security Administrator

The instructions and information provided to assist you in implementing the SSL/TLS protocol assume that you have an expert OpenVMS security administrator who is familiar with the terms associated with digital certificates and has experience using the tools required to generate and manage certificates, including:

- System security applications
- Security terminology associated with digital certificates (see *Terminology and Security Applications for SSL and TLS Certificates* on page 23)
- Working knowledge of the Connect:Direct application and its environment

Assess Security Requirements of Your System and Trading Partners

Security planning is a collaborative effort between you and your trading partners. You must know the expectations of your trading partners and plan your security implementation to meet these requirements. Chapter 2, *Set Up Secure+ Option to Use Certificates*, contains information about the different types of certificates and the information they contain. This chapter also explains what files you need to store in your default Connect:Direct ndm$$directory to enable secure connections.

Consider the following guidelines for configuring communications sessions using the SSL or TLS protocol:

- ✦ You must acquire the certificates before you configure Secure+ Option.
- ✦ Determine whether you and your trading partner will use self-signed certificates or certificates signed by a Certificate Authority.
- ✦ Determine whether to use client authentication.

For more general information on certificates, see Appendix B, *Understanding the Certificate File Layout*.

Self-Signed and CA-Signed Certificates

Determining the type of certificate to use for secure communications sessions and the method to generate the certificate is challenging. Self-signed and CA-issued digital certificates offer advantages and disadvantages. You may also be required to use both types of certificates, depending on the security requirements of your trading partners. The following table compares the advantages and disadvantages of self-signed and CA-signed certificates.

Type of Certificate	**Advantages**	**Disadvantages**
Self-signed certificate	No cost	Requires you to distribute your certificate, minus its private key, to each trading partner in a secure manner.
	Easy to generate	Difficult to maintain; anytime the certificate is changed, it must be distributed to all clients.
	Self-validated	Not validated by a third-party entity
	Efficient for small number of trading partners	Inefficient for large number of trading partners
CA-signed certificate	Eliminates having to send your certificate to each trading partner	Trading partners must download digital CA-signed certificate used to verify the digital signature of trading partner public keys only if the CA certificate is not available
	No changes are required on the trading partner's system if you recreate the CA digitally signed certificate using the same CA	Must be purchased from third-party vendor

Terminology for Authentication for SSL and TLS Protocols

The following table defines the security terms associated with SSL and TLS certificates and applies them to communications sessions between a Connect:Direct PNODE (client) and SNODE (server). All terms are listed in alphabetical order.

Term	Definition
CA-Signed Certificate	Digital document issued by a certificate authority that binds a public key to the identity of the certificate owner, thereby enabling the certificate owner to be authenticated. An identity certificate issued by a CA is digitally signed with the private key of the certificate authority.
Certificate (also known as digital certificate, public key certificate, digital ID, or identity certificate)	Signed certificate that is obtained from a certificate authority by generating a certificate signing request (CSR). It typically contains: (1) distinguished name and public key of the server or client; (2) common name and digital signature of the certificate authority; (3) period of validity (certificates expire and must be renewed); and (4) administrative and extended information. The certificate authority analyzes the CSR fields, validates the accuracy of the fields, generates a certificate, and sends it to the requester. A certificate can also be self-signed and generated by any one of many tools available, such as Certificate Wizard or OpenSSL. These tools can generate a digital certificate file and a private key file in PEM format, which you can combine using any ASCII text editor to create a key certificate file.
Certificate Authority (CA)	An organization, such as VeriSign, that issues digitally-signed certificates. The certificate authority authenticates the certificate owner's identity and the services that the owner is authorized to use, issues new certificates, renews existing certificates, and revokes certificates belonging to users who are no longer authorized to use them. The CA digital signature is assurance that anybody that trusts the CA can also trust that the certificate that it signs is an accurate representation of the certificate owner.
Certificate Signing Request (CSR)	Message sent from an applicant to a certificate authority in order to apply for a digital identity certificate. Before creating a CSR, the applicant first generates a key pair, keeping the private key secret. The CSR contains information identifying the applicant (such as a directory name in the case of an X.509 certificate), and the public key chosen by the applicant. The CSR may be accompanied by other credentials or proofs of identity required by the certificate authority, and the certificate authority may contact the applicant for further information. Many tools are available to create a CSR to send to a CA.
Cipher Suites	A cryptographic key exchange algorithm that enables you to encrypt and decrypt files and messages with the SSL or TLS protocol.
Client Authentication	A level of authentication that requires the client to authenticate its identity to the server by sending its certificate.

Term	**Definition**
Key certificate file	File that contains the encrypted private key and the ID (public key) certificate. This file also contains the certificate common name which can be used to provide additional client authentication.
Passphrase	Passphrase used to access the private key.
Private Key	String of characters used as the private, “secret” part of a complementary public-private key pair. The asymmetric cipher of the private key is used to decrypt data that is encrypted with its complementary public key. Data that is encrypted with a Public Key can only be decrypted using its complementary Private Key. The private key is never transmitted.
Protocol	The rules determining the format and transmission of data. SSL and TLS are cryptographic protocols which provide secure communications on the Internet
Public Key	String of characters used as the publicly distributed part of a complementary public-private key pair. The asymmetric cipher of the public key is used to encrypt data for the session key that is exchanged between server and client during negotiation for an SSL/TLS session. The public key is part of the ID (public key) certificate. This information is stored in the key certificate file and read when authentication is performed.
Root Certificate file	See *Trusted Root Certificate File (also known as Root Certificate file)*.
Self-Signed Certificate	Digital document that is self-issued, that is, it is generated, digitally signed, and authenticated by its owner. Its authenticity is not validated by the digital signature and trusted key of a third-party certificate authority. To use self-signed certificates, you must exchange certificates with all your trading partners.
Session Key	Asymmetric cipher used by the client and server to encrypt data. It is generated by the SSL or TLS software.
Trusted Root Certificate File (also known as Root Certificate file)	File which contains one or more trusted root certificates used to authenticate ID (public) certificates sent by trading partners during the Secure+ protocol handshake.

Obtain and Generate Certificates

The TLS and the SSL security protocols use a secure server RSA X.509V3 certificate to authenticate an entity for any security-enabled node that accesses the entity. Obtain a server certificate from a CA or create a self-signed server certificate. Create a private key using Certificate Wizard or any other tool, such as OpenSSL. For more information on certificates and to see a sample certificate, see Appendix B, *Understanding the Certificate File Layout*. Secure+ Option uses a key certificate file and a trusted root certificate file to authenticate an entity.

Generate a Key Certificate File for a CA Certificate

Complete the following steps to generate a key certificate file from a certificate generated by a CA:

1. Generate a certificate signing request (CSR) and a private key using OpenSSL, Certificate Wizard, or any other tool.
2. Send the CSR to the CA to request a certificate.
3. When you receive the certificate from the CA, generate a key certificate file using OpenSSL, Certificate Wizard, or a text editor. The key certificate file combines information from the certificate that you received from the CA and the private key you generated.

   Note: While a key certificate file may contain information about its intended use, such as e-mail, Secure+ Option does not use this information. Connect:Direct Secure+ Option uses only the common name information from the certificate when the client authentication option is enabled.

4. After you generate the key certificate file, copy it to the OpenVMS system using FTP in ASCII mode.

Generate a Key Certificate File for a Self-Signed Server Certificate

Complete the following steps to generate a key certificate file for an entity that is authenticated with a self-signed server certificate:

1. Generate a self-signed server certificate including a private key using a tool such as OpenSSL or Certificate Wizard.
2. Generate a key certificate file, which combines information from the self-signed certificate and the private key.
3. After you generate the key certificate file, copy it to the OpenVMS system using FTP in ASCII mode.

Exchange Certificate Information with Trading Partners

If you are using third-party CAs, you and your trading partners must exchange trusted root certificate files. The trusted root files are used to verify the identity of the CA who issued your certificate, and you must have a copy of the trading partner’s trusted root certificate file to validate the CA who issued the trading partner’s certificate file. Obtain a copy of the trusted root certificate file, which can usually be downloaded from the CA’s server) and copy it to the desired directory on the OpenVMS system. You can maintain multiple trusted root certificate files for each trading partner you need to verify, or you can maintain all trusted root certificate file information in one file. If you maintain a separate file for each trading partner, specify the name of each trading partner’s trusted root certificate file in the corresponding remote node record.

If you use a self-signed certificate, exchange certificates with your trading partners. If you use VeriSign or Thawte as the CA to obtain a certificate, exchange the trusted root certificate file with your trading partners.

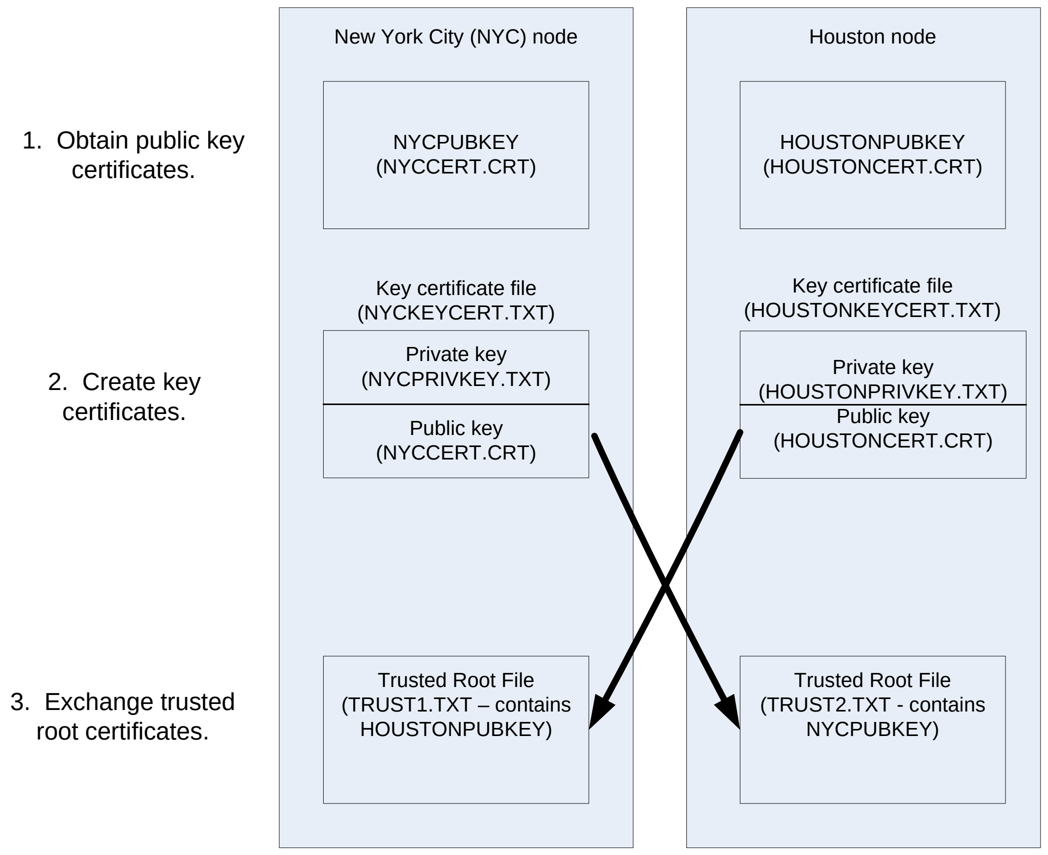
Example of Setting Up and Processing Certificates

In this example two nodes, New York City and Houston are running Connect:Direct for OpenVMS with Secure+ Option using the TLS protocol. The two nodes want to set up a secure connection between them and need to enter node, protocol, certificate, and cipher information about each other in their Secure+ Option parameters files. The example showing how the two nodes configure the necessary information in Secure+ Option is shown in *Example of Configuring Nodes and How a Secure Connection is Processed* on page 31.

During the planning phase, each node created self-signed server certificates and while doing so generated three pieces of information:

- A certificate common name (which is used when client authentication is enabled)
- A private key consisting of a random string of typed characters (used when a secure connection is being established but not entered in Secure+ Option)
- The passphrase for accessing the private key (which is also used when configuring nodes in Secure+ Option)

In this example, the two nodes exchange root certificate files so that they can authenticate the ID certificate of the other node. Each trading partner stores the other’s root certificate in its trusted root certificate file—TRUST1.TXT for the NYC node and TRUST2.TXT for the Houston node.


New York City (NYC) node
Houston node
1. Obtain public key certificates.
NYCPUBKEY (NYCCERT.CRT)
HOUSTONPUBKEY (HOUSTONCERT.CRT)
Key certificate file (NYCKEYCERT.TXT)
Key certificate file (HOUSTONKEYCERT.TXT)
2. Create key certificates.
Private key (NYCPRIVKEY.TXT)
Public key (NYCCERT.CRT)
Private key (HOUSTONPRIVKEY.TXT)
Public key (HOUSTONCERT.CRT)
3. Exchange trusted root certificates.
Trusted Root File (TRUST1.TXT – contains HOUSTONPUBKEY)
Trusted Root File (TRUST2.TXT - contains NYCPUBKEY)

To configure Secure+ Option, each node must define itself as the local node and the other trading partner as a remote node both in its network map (netfile.dat) and in its Secure+ Option parameters file (secnetfile.dat). For more information on how to set up the network map, see *Connect:Direct for OpenVMS Installation and Administration Guide*. To see an example of how to customize these node records, see *Example of Configuring Nodes and How a Secure Connection is Processed* on page 31.

Chapter 3

Use the Secure+ Admin Tool and Populate the Secure+ Option Parameters File

Use the following information to familiarize yourself with the Secure+ Option administration tool and the procedure to populate the Secure+ Option parameters file. Instructions for populating the Secure+ Option parameters file assume that you have populated the Connect:Direct network map.

Secure+ Option Administration Tool

The Secure+ Option Administration (SPADMIN) Tool is used initially to set up and maintain Secure+ Option operations. You can run this tool either as an interactive script or a command line task. This guide focuses on the interactive script, but if you want to use the SPADMIN tool as a command line task, see Chapter 6, *Use CLI to Maintain the Secure+ Option Parameters File*.

Note: Three SPADMIN functions are not available using CLI—viewing a node record, listing all node records, and synchronizing the Connect:Direct network map with the Secure+ Option parameters file. You must use the SPADMIN tool as an interactive script to perform these functions.

Start the Secure+ Option SPADMIN Tool as an Interactive Script

To run the Connect:Direct Secure+ Option OpenVMS SPADMIN Tool as an interactive script, at the command line set the default directory to ndm$$directory, type the following command, and press **Enter**:

```
SET DEF NDM$$DIRECTORY
$SPADMIN
```

The SPADMIN main menu is displayed.

```
TSC5->spadmin

        1. Add a Connect Direct node to the Secure+ Parmfile
        2. Change an existing Connect Direct node's parameters
        3. Delete an existing Connect Direct node from the Secure+ Parmfile
        4. Get an existing Connect Direct node's parameters
        5. List all Connect Direct nodes in the Secure+ Parmfile

        8. Sync with Netmap
        9. Quit this procedure

    Type '?' at any prompt for a description of the information
    requested.
1,2,3,4,5,8,9
```

All options on the SPADMIN main menu except Options 8 and 9 are discussed in Chapter 4, *Configure Nodes for Secure+ Option*, and Chapter 5, *Maintain Secure+ Option Nodes*. You can access help on any option by typing **?** and pressing **Enter**.

Stop the Secure+ Option SPADMIN Tool as an Interactive Script

To exit and return control to the DCL prompt, type **9** and press **Enter** to quit using the SPADMIN tool.

Populate the Secure+ Option Parameters File Using Sync with Netmap

You must configure Secure+ Option before you begin using it for secure communications.

Note: Before you populate the Secure+ Option parameters file, be sure you restrict access to it—allowing access to the Secure+ Option parameters file (ndm$$directory:secnetfile.dat) to unauthorized personnel may compromise the security of your data exchange. When planning the Secure+ Option implementation, ensure that the access attributes of these files remain as *PUBLIC *EXCLUDE.

The most efficient way to use Secure+ Option is to create and save a Secure+ Option parameters file using the Sync with Netmap option. The Sync with Netmap option creates a single local node definition that is used as a default and a remote node definition for each external trading partner defined in your network map. When you synchronize the node information in your Secure+ Option parameters file with the node information in your network map file, whether or not all the nodes require a secure connection, you minimize errors and overhead both in terms of performance and statistical record keeping.

Note: You cannot perform the Sync with Netmap operation using the command line interface. You must go through the SPADMIN main menu and use the interactive interface.

Your trading partners, too, must create their own Secure+ Option parameters file. The network map contains the nodes and their names and communications information, such as TCP port numbers, while the Secure+ Option parameters file contains the same node names along with protocol, certificate, and cipher information. Together these two files enable secure communications.

After you install Connect:Direct for OpenVMS, register the product using a license key, and configure it for your environment, the Secure+ Option component is installed and an empty Secure+ Option parameters file is created with one local node record that is set up as a non-secured node.

The Sync with Netmap option enables you to populate the Secure+ Option parameters file by importing the node name information from the Connect:Direct network map. After you initially populate the Secure+ Option parameters file, you can use this option to synchronize the network map file and Secure+ Option parameters file after you add nodes to the network map.

Note: The Sync with Netmap option only adds records for nodes in the network map to the Secure+ Option parameters file. It does not delete node records from the Secure+ Option parameters file or copy node records from it to the network map. Node records in the network map and the Secure+ Option parameters file must be deleted manually.

To populate the Secure+ Option parameters file from the Connect:Direct network map:

1. With the SPADMIN main menu options on display, type **8** to select Sync with Netmap and press **Enter**.

2. The location and name of the network map and Secure+ Option parameter files are displayed along with the information for all nodes added to the Secure+ Option parameter file.

```
1,2,3,4,5,8,9  8
Netmap file is NDM$$DIRECTORY:NETFILE.DAT
Parm file is NDM$$DIRECTORY:SECNETFILE.DAT
Added L node DALLAS.VMS.V3400
Added R node HOUSTON.HPNS.V3400
Added R node DUBLIN.OS400.V3400
Added R node BOSTON.WINDOWS.V3400
Added R node NEWYORK.UNIX.V3400
Added R node ELKTON.ZOS.V3400
Added R node CHESTERTOWN.OVMS.V3400
Added R node CAMBRIDGE.ZOS.V3400
Added R node DENTON.WINDOWS.V3400
Added R node HOUSTON.OVMS.V3400
```

3. Update the local and remote node records as necessary using the following procedures:
   - *Configure the Local Node Record* on page 27
   - *Customize a Remote Node Record* on page 30

Chapter 4

Configure Nodes for Secure+ Option

Before you begin using Secure+ Option, you must configure the local node and remote node records for secure operations. Use the following information as a guide:

- *Scenarios for Configuring Nodes* on page 25
- *Configure the Local Node Record* on page 27
- *Customize a Remote Node Record* on page 30

In addition, the following two configuration examples are provided:

- *Example of Configuring Nodes and How a Secure Connection is Processed* on page 31 explains how two different Connect:Direct systems have their Secure+ Option parameter files configured and how the local node record can be used to define all parameters not specified in a remote node record. The server and client authentication processes are also described.
- *Example of Configuring Secure Communications between Connect:Direct for z/OS and Connect:Direct for OpenVMS* on page 34 provides a detailed example for defining a remote node record in both a Connect:Direct for z/OS Secure+ Option parameters file and a Connect:Direct for OpenMVS Secure+ Option parameters file to set up a secure connection between the two nodes.

Scenarios for Configuring Nodes

After you install Connect:Direct for OpenVMS and the Secure+ Option parameters file has been created, you decide how to configure the records for your trading partners. In the Protocol field, specify the most commonly used protocol, or specify **N** to disable Secure+ Option in the local node record. Enabling the TLS or SSL protocol in the local node record configures remote nodes to default to the settings in the local node record if you create the remote node records using the Sync with Netmap option or specify **D** (default to local node) in the Protocol field in the remote node records later. Use the following table to help you decide how to configure your remote node records.

Scenario	How to Configure Node Records
Most trading partners do not require a secure connection.	◆ After you define all nodes in the network map, use the Sync with Netmap option to populate the Secure+ parameters file. The value of the Protocol parameter is **D** (default to local node) for all remote node records. Because the value of the Protocol parameter in the local node record defaults to **N**, which disables Secure+ Option, Secure+ Option is disabled for all remote nodes. See *Populate the Secure+ Option Parameters File Using Sync with Netmap* on page 22. ◆ To manually configure the remote nodes that require a secure connection, see *Customize a Remote Node Record* on page 30. ◆ If you add a remote node record later, use the Sync with Netmap option, and if the remote node requires a secure connection, manually configure it.
Most trading partners require a secure connection.	◆ After you define all nodes in the network map, use the Sync with Netmap option to populate the Secure+ parameters file. ◆ Configure the local node record and specify the protocol used most often (SSL or TLS) and other parameters, such as ciphers and certificate information. See *Configure the Local Node Record* on page 27. ◆ Remote nodes that use the same values specified in the local node record are enabled for secure communications because the Protocol parameter is set to **D** (default to the local node) after you perform the Sync with Netmap operation. See *Customize a Remote Node Record* on page 30 to make any of the following changes to a remote node record: ◆ To use a unique certificate file to authenticate a remote node ◆ To use a different self-signed server certificate for client or server authentication ◆ To identify a unique cipher suite used by a trading partner ◆ To activate common name validation and client authentication ◆ To use a protocol that is different from the protocol defined in the local node record ◆ To disable Secure+ Option for a node that does not require a secure connection

Note: The following node configuration procedures use the SPADMIN Tool interactive script, which enables you to customize node records one by one. You can also use the SPADMIN Tool command line task to configure multiple records at one time. For more information on using the SPADMIN Tool as a command line task, see Chapter 6, *Use CLI to Maintain the Secure+ Option Parameters File*.

Configure the Local Node Record

The values defined in the local node record can be used as the default settings for remote node records. Determine which protocol is used by most trading partners and configure this protocol in the local node record, which makes it easy to configure the remote node records.

To configure the local node record:

1. With the SPADMIN main menu options on display, type **2** to select Change an Existing Connect:Direct node's parameters and press **Enter**.
2. Type the node name exactly as it is specified in the network map and press **Enter**.
3. For node type, type **L** and press **Enter**.

   The system validates the existing node and displays the current settings for the local node, as shown in the following example:

```
TSC5->spadmin

        1. Add a Connect Direct node to the Secure+ Parmfile
        2. Change an existing Connect Direct node's parameters
        3. Delete an existing Connect Direct node from the Secure+ Parmfile
        4. Get an existing Connect Direct node's parameters
        5. List all Connect Direct nodes in the Secure+ Parmfile

        8. Sync with Netmap
        9. Quit this procedure

    Type '?' at any prompt for a description of the information
    requested.
1,2,3,4,5,8,9  2
Enter node name test2
Is test2 local or remote? [L/R][R] L
                 Node Name:    DALLAS.VMS.3400
                 Node type:    L
 1.               Protocol:    N
 2.  Client Authentication:    N
 3. Authentication timeout:    300
 4.Certificate common name:
 5.  Root Certificate file:
 6.   Key Certificate file:
 7.             Passphrase:
 8.          Cipher suites:

        1. Change the protocol
        2. Change client authentication
        3. Change Authentication timeout
        4. Change Certificate common name
        5. Change Root certificate
        6. Change Key Certificate
        7. Change Passphrase
        8. Change Cipher suites
       14. Show data
       15. Exit
1,2,3,4,5,6,7,8,14,15
```

4. Type the number of the parameter you want to change and press **Enter**. For example, to change the protocol, type **1** and press **Enter**.
5. Type the new value for the parameter and press **Enter**.

   Refer to the following table for a description of the parameters and their values.

Term	Definition
Protocol	Protocol used to set up a secure connection between two nodes. Secure+ Option has four values for this field: ◆ S for SSL ◆ T for TLS ◆ N for no (disable Secure+ Option for this node) ◆ D to default to the local node (for remote nodes only) This field defaults to N (no Secure+ Option) but you can change it to TLS or SSL when configuring the local node. For remote nodes, this field defaults to D to match the protocol specified for the local node.
Client Authentication	A level of authentication that requires the PNODE (client) to send the server its ID certificate so that the server can authenticate the client’s identity. Specify Yes in this field to enable client authentication in the remote node records.
Authentication timeout	Maximum number of seconds that the system waits to receive the Connect:Direct control blocks exchanged during the Secure+ Option authentication process. Valid values = 0 to 300 Default = 300
Certificate Common Name	Part of the ID certificate that can be used to provide an additional form of client authentication when client authentication is enabled for a remote node in your Secure+ Option parameters file. The value that you specify in the Certificate common name field is validated against the common name that is in the PNODE certificate presented for client authentication. The information in this field is case-sensitive and must match exactly what was defined when the certificate was originally created.
Root Certificate file	File which contains one or more trusted root certificates used to authenticate ID (public) certificates sent by trading partners during the SSL/TLS protocol handshake. Type the fully qualified name of the root certificate file.

Term	Definition
Key certificate file	File that contains the encrypted private key and the ID (public key) certificate. This file also contains the certificate common name, which can be used to provide additional client authentication. Type the fully qualified name of the key certificate file.
Passphrase	Passphrase used to access the private key. You must enter this passphrase when configuring a secure node although Connect:Direct Secure+ Option for OpenVMS does not validate it. This passphrase is used to decrypt the private key before the SSL/TLS protocol handshake. Type this information exactly as originally entered when the certificate was created because it is case-sensitive.
Cipher Suites	A cryptographic key exchange algorithm that enables you to encrypt and decrypt files and messages with the SSL or TLS protocol. Secure+ Option supports the following ciphers: 1. NULL_MD5 2. NULL_SHA 3. EXP_RC4_MD5 4. RC4_MD5 5. RC4_SHA 6. EXP_RC2_CBC_MD5 7. IDEA_CBC_SHA 8. EXP__DES_CBC_SHA 9. DES_CBC_SHA 10. DES_CBC3_SHA 11. AES128_SHA 12. AES256_SHA You must select one or more ciphers when configuring a secure node by taking one of the following actions: ◆ Typing a list of the ciphers in order of preference, separated by commas and pressing **Enter**. For example, 12,10 places AES256_SHA as the first cipher choice followed by DES_CBC3_SHA. ◆ Typing **all** to include all secure ciphers and pressing **Enter**. The all option includes cipher #3 through cipher #12. **Note:** Configure the remote node record to contain at least one cipher in common with the trading partner.

6. When you are finished configuring the local node record, type **15** and press **Enter**.

A list of all parameters specified for the node is displayed. At the **Param values OK?** prompt, you have two choices:

- If the values are correct, press **Enter**. A message is displayed confirming that the node record was updated.
- If you need to change a value, type **N** and press **Enter**. Change the necessary parameters.

Caution: You should configure only one local node record in your Secure+ Option parameters file, although the system does not prevent you from creating more than one local node record. When Connect:Direct Secure+ Option for OpenVMS attempts to initiate a secure connection, it uses the name of the local node record that matches the local node defined in the network map.

Customize a Remote Node Record

After you have populated the Secure+ Option parameters file, configured the local node record, and synchronized with the network map, you can modify the remote node records to reflect the requirements of your trading partners.

To configure a remote node record:

1. With the SPADMIN main menu options on display, type **2** to select Change an Existing Connect:Direct node’s parameters and press **Enter**.

   Tip: Use option 5, List all Connect:Direct nodes in the Secure+ Parmfile, to enable you to copy the node name and paste it in the node name prompt.

2. Type the node name exactly as it is specified in the network map and press **Enter**.
3. For node type, press **Enter** to accept the default remote node type (R).

   The system displays the current settings for the remote node. In the following example, fields not defined in the remote node record use values defined in the local node record because D has been specified for the Protocol parameter.

   Note: If you specify TLS or SSL as the protocol, you must define all values to use to set up the secure connection.

```
1,2,3,4,5,8,9  2
Enter node name test2
Is test2 local or remote? [L/R][R]
                 Node Name:    BOSTON.WINDOWS.V3400
                 Node type:    R
 1.               Protocol:    D
 2.  Client Authentication:    N
 3. Authentication timeout:    100
 4.Certificate common name:
 5.  Root Certificate file:
 6.   Key Certificate file:
 7.             Passphrase:
 8.          Cipher suites:    AES256_SHA

        1. Change the protocol
        2. Change client authentication
        3. Change Authentication timeout
        4. Change Certificate common name
        5. Change Root certificate
        6. Change Key Certificate
        7. Change Passphrase
        8. Change Cipher suites
       14. Show data
       15. Exit
1,2,3,4,5,6,7,8,14,1
```

4. Type the number of the parameter you want to change and press **Enter**. For example to change the protocol, type **1** and press **Enter**.

   Type the new value for the parameter and press **Enter**. These fields are described in the table on page 28.

5. When you are finished configuring the remote node record, type **15** and press **Enter**.

   A list of all parameters specified for the node is displayed. At the **Param values OK?** prompt, you have two choices:

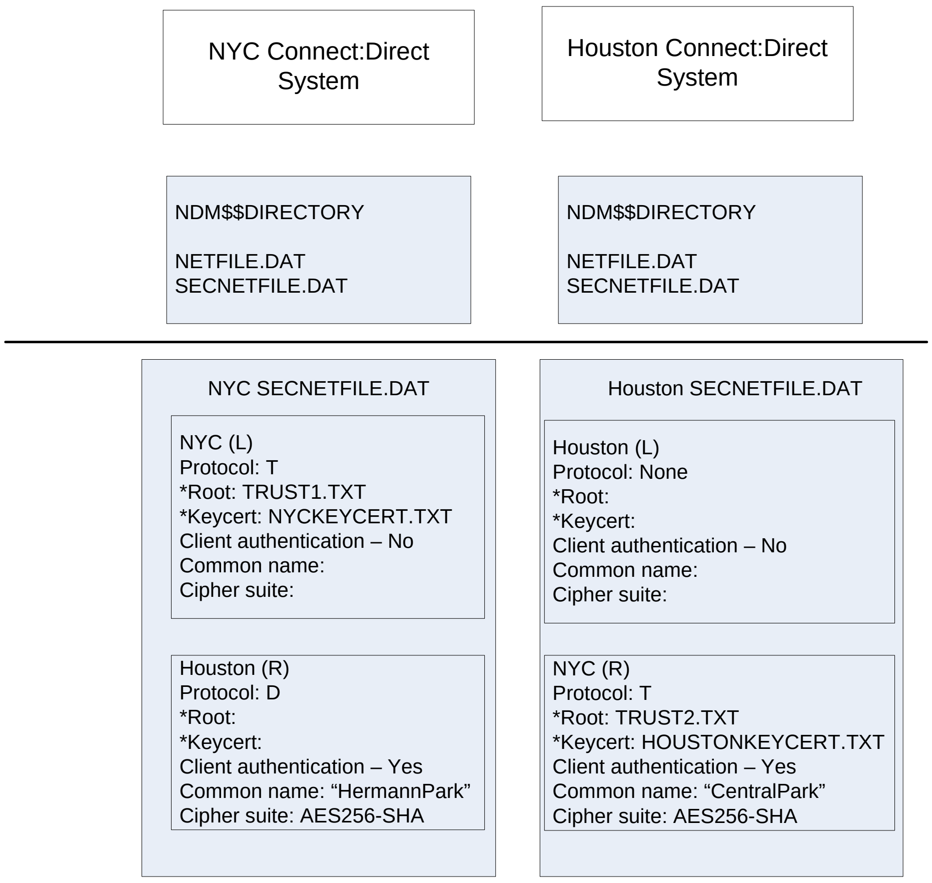
   - If the values are correct, press **Enter**. A message is displayed confirming that the node record was updated.
   - If you need to change a value, type **N** and press **Enter**. Change the necessary parameters.

Example of Configuring Nodes and How a Secure Connection is Processed

In this example, two offices, New York City and Houston, have set up their Connect:Direct systems the same way: their default Connect:Direct directory (ndm$$directory) contains the files necessary to run Connect:Direct and set up secure connections. (Most of the necessary data and command files, such as the Initialization Parameters file and Startup command file, are not shown in the following diagram but reside in the same directory as the network map file and the Secure+ Option parameters file.)

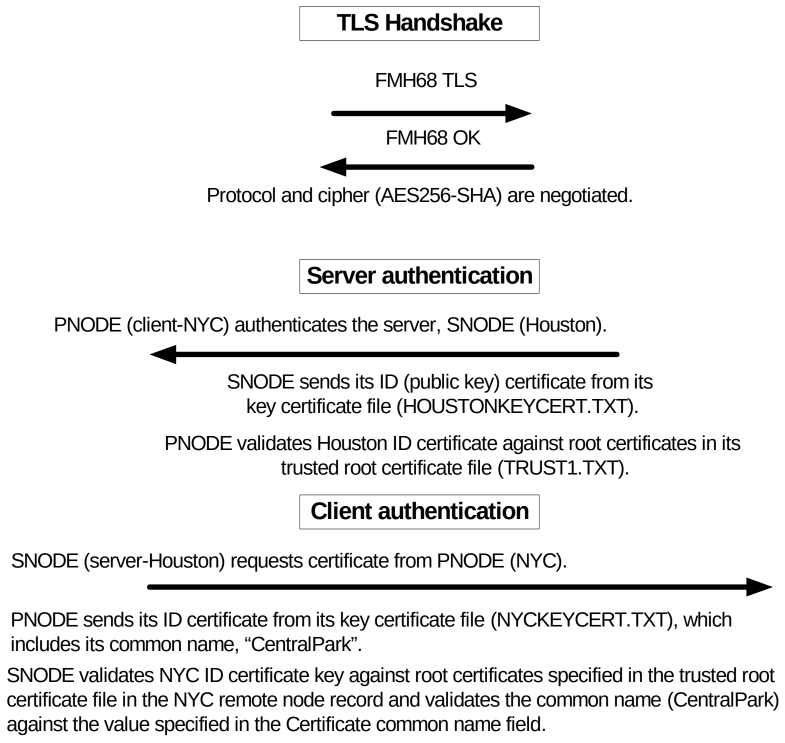
The bottom portion of the following diagram illustrates the values defined for the local node record in the NYC and Houston Secure+ Option parameters files, respectively, and the remote node record that each created for the other. The NYC Secure+ Option local node record defines the TLS protocol and some (but not all) parameters for secure connections. The NYC remote node record for Houston defines the Protocol field to default to the local node for all parameters not specified. For example, when NYC initiates a session with Houston, NYC uses the key certificate file specified in the local node record (NYCKEYCERT.TXT) because no value is specified in the Keycert field of the remote node record for Houston. The local node record for the Houston system disables Secure+ Option, so all Secure+ Option parameters are defined in the remote node records for its trading partners.

Note: An asterisk (*) beside a field label (Root and Keycert) indicates that the file name specified must be fully qualified.



Connect:Direct uses communications information contained in the network map file, such as the TCP/IP host name and port number, to initiate the connection, as shown in the TLS handshake in the following diagram. In this example, the PNODE (NYC) sends a control block to the SNODE (Houston). The SNODE confirms that it has a record defined for the PNODE in its Secure+ Option parameters file, which also defines the requested protocol (TLS) and cipher (AES256_SHA). If the handshake is successful and a common cipher can be negotiated, the server and client authentication processes continue.

Note: After the protocol and common cipher have been negotiated, Secure+ Option uses this cipher to encrypt data before sending it. If more than one cipher is enabled, the preferences defined in the server Secure+ Option parameters file determine the cipher used for the SSL protocol and the preferences defined in the client Secure+ Option parameters file determine the cipher used for the TLS protocol.


TLS Handshake
FMH68 TLS
FMH68 OK
Protocol and cipher (AES256-SHA) are negotiated.
Server authentication
PNODE (client-NYC) authenticates the server, SNODE (Houston).
SNODE sends its ID (public key) certificate from its key certificate file (HOUSTONKEYCERT.TXT).
PNODE validates Houston ID certificate against root certificates in its trusted root certificate file (TRUST1.TXT).
Client authentication
SNODE (server-Houston) requests certificate from PNODE (NYC).
PNODE sends its ID certificate from its key certificate file (NYCKEYCERT.TXT), which includes its common name, “CentralPark”.
SNODE validates NYC ID certificate key against root certificates specified in the trusted root certificate file in the NYC remote node record and validates the common name (CentralPark) against the value specified in the Certificate common name field.

Server authentication always occurs; that is, the PNODE, which initiates the connection as the client, always requests an ID certificate from the SNODE acting as the server. In this example, the Houston node sends its ID certificate, which NYC verifies against the trusted root certificates in the root certificate file specified in the local node record because the NYC remote node record for Houston has the Protocol field set to D and no value specified in the Root certificate file field.

Client authentication is optional and is enabled in Houston’s remote node record for NYC. Therefore, Houston, the server, requests that NYC send its ID certificate, which includes its common name. The server (Houston) validates NYC’s ID certificate against certificates stored in the trusted root certificates file specified in the Root certificate file field of the NYC remote node record and also validates that the value specified in the Common name field for the NYC remote node matches the common name value in its ID certificate.

Example of Configuring Secure Communications between Connect:Direct for z/OS and Connect:Direct for OpenVMS

In this example, two nodes have set up records in their respective Secure+ Option parameters files:

- ✦ The Connect:Direct for z/OS node is named Q1A.ZOA.V4600 and is defined in a remote node record in the Connect:Direct for OpenVMS Secure+ Option parameters file and in the Connect:Direct for OpenVMS network map.
- ✦ The Connect:Direct for OpenVMS node is named Q1A.ITAN.V3400 and is defined in a remote node record in the Connect:Direct for z/OS Secure+ Option parameters file and in the Connect:Direct for z/OS network map.

The Secure+ Option records are defined to allow each node to act as either the client (PNODE) or the server (SNODE), depending on which one initiates the session.

Settings for the z/OS Local Node Record and the OpenVMS Remote Node Record in the z/OS Secure+ Option Parameters File

In the Connect:Direct for z/OS Secure+ Option parameters file, the local node record has the following settings:

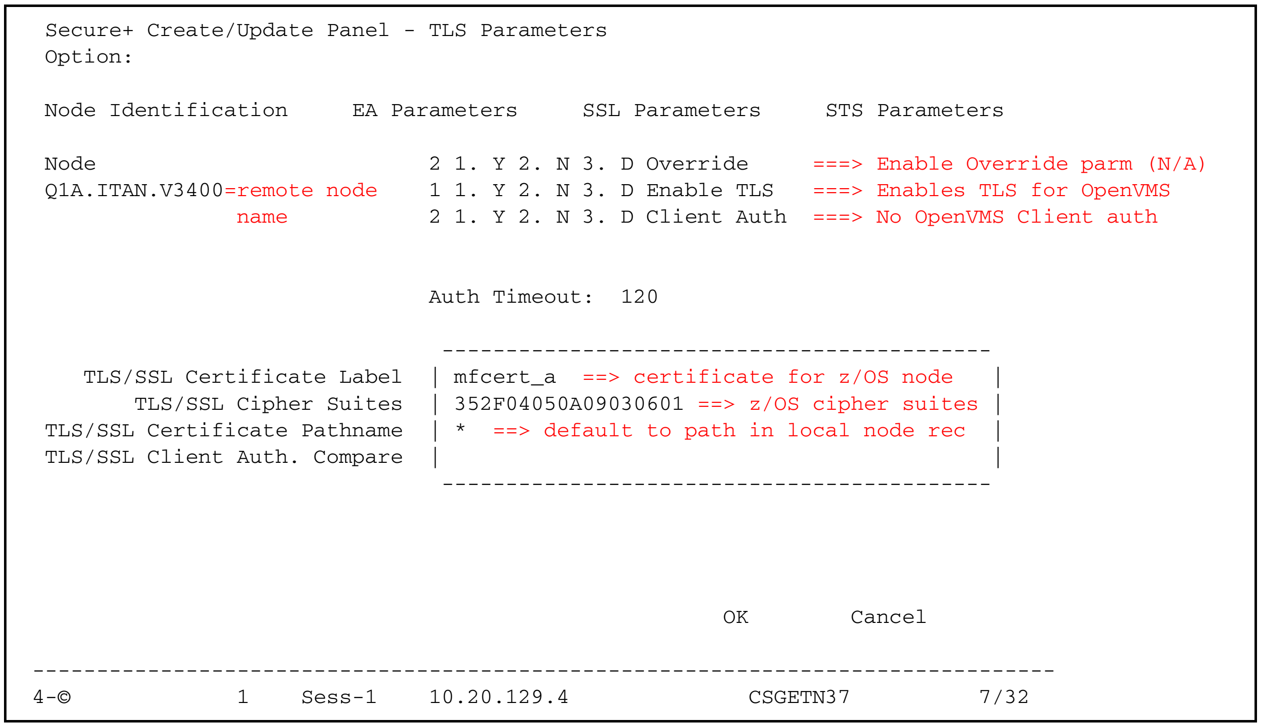
- ✦ **Y** in the Override field
- ✦ **N** in the Enable TLS (or SSL) field
- ✦ **N** in the Client Auth field

The settings for the local node record have the following effects: Disabling Secure+ Option in the local node record means that the protocol and other settings for secure connections must be defined in each remote node record; enabling the Override parameter allows settings in remote node records to override those in the local node record; client authentication is not enabled for all remote nodes.

The remote node record defined for the OpenVMS node named Q1A.ITAN.V3400 in the z/OS Secure+ Option parameters file has the following settings:

- ✦ Node Identification is Q1A.ITAN.V3400. This value must correspond to the node name specified in the Connect:Direct for z/OS network map.
- ✦ Override is not applicable in the remote record and defaults to N.
- ✦ The TLS protocol is enabled for sessions to connect to this node.
- ✦ This OpenVMS node will not request client authentication of z/OS nodes with which it communicates.
- ✦ Auth Timeout is set to the two-minute default to identify the maximum time that the system waits to receive Connect:Direct control blocks exchanged during the authentication protocol.

The following Secure+ Create/Update Panel - TLS Parameters panel for Secure+ Option for z/OS illustrates the settings for the OpenVMS node named Q1A.ITAN.V3400 and commentary on the values set for the parameters.



```
Secure+ Create/Update Panel - TLS Parameters
Option:

Node Identification     EA Parameters     SSL Parameters     STS Parameters

Node                          2 1. Y 2. N 3. D Override     ===> Enable Override parm (N/A)
Q1A.ITAN.V3400=remote node    1 1. Y 2. N 3. D Enable TLS   ===> Enables TLS for OpenVMS
               name           2 1. Y 2. N 3. D Client Auth  ===> No OpenVMS Client auth


                              Auth Timeout:  120

                               -------------------------------------------
   TLS/SSL Certificate Label  | mfcert_a  ==> certificate for z/OS node   |
       TLS/SSL Cipher Suites  | 352F04050A09030601 ==> z/OS cipher suites |
TLS/SSL Certificate Pathname  | *  ==> default to path in local node rec  |
TLS/SSL Client Auth. Compare  |                                           |
                               -------------------------------------------



                                                   OK        Cancel

--------------------------------------------------------------------------------
4-©             1     Sess-1     10.20.129.4              CSGETN37          7/32
```

The information in the bottom half of the screen pertains to the key certificate for the z/OS node. The OpenVMS remote node record for the z/OS node has enabled client authentication, as shown in *Settings for the z/OS Remote Node Record in the OpenVMS Secure+ Option Parameters File* on page 36. Therefore, when the z/OS node initiates the session, the OpenVMS node (the server) requests that the client send its ID certificate so that the OpenVMS node can authenticate the client by validating the key certificate defined on this panel (mfcert_a) against the key certificate specified in the Root Certificate file field (mfcert_a.txt) of the z/OS remote node record in the OpenVMS Secure+ Option parameters file, as illustrated on page 36. When the z/OS node is the server, it must send its public key, which is stored in the mfcert_a file, to the OpenVMS node during server authentication.

In this example, the z/OS key certificate resides in the default key database defined for the local node (indicated by *). If the certificate location does not default to the local node, the remote node definition must point to the absolute path. Definitions for the default key database are stored in the local node record. Certificate information identifying the z/OS node to remote nodes and remote nodes to the z/OS node is stored in the GSKKYMAN database. When certificates are exchanged, trading partners send the ID certificate portion of their keys to each other. In the z/OS system, this information must be imported into the GSKKYMAN database.

Note: In the OpenVMS system, fully qualified paths are always required for file locations.

The TLS ciphers previously selected are shown using the standard two-byte IBM convention for displaying ciphers (352F04050A09030601). The systems negotiate a cipher suite common to both the z/OS and OpenVMS nodes to encrypt information during the handshake and when actual data is being transmitted.

Settings for the z/OS Remote Node Record in the OpenVMS Secure+ Option Parameters File

The following example shows the remote node record that defines the Connect:Direct for z/OS node named Q1A.ZOA.V4600. The OpenVMS network map contains an adjacent (remote) node record with the exact same name.

```
                Node Name:    Q1A.ZOS.V4600
                Node type:    R
1.               Protocol:    T
2.  Client Authentication:    y
3. Authentication timeout:    100
4.Certificate common name:    mfsscert_a
5.  Root Certificate file:    disk$data:[qaitan.q1a]mfcert_a.txt
6.   Key Certificate file:    disk$data:[qaitan.q1a]2048sskeycert.txt
7.             Passphrase:    ****
8.          Cipher suites:  EXP_RC4_MD5,RC4_MD5,RC4_SHA,EXP_RC2_CBC_MD5,IDEA_CBC_SHA,
                              EXP_DES_CBC_SHA,DES_CBC_SHA,DES_CBC3A
```

When the OpenVMS node is the server, it requests that the client authenticate itself (Client Authentication = Y) and send its certificate common name (mfsscert_a) for an extra layer of authentication. The public key information for the z/OS node is stored in the Root Certificate file named mfcert_a.txt; its location is specified (disk$data:[qaitan.qla]).

The key certificate file contains the information that identifies the OpenVMS node to other nodes (disk$data:[qaitan.q1a]2048sskeycert.txt). In order for the OpenVMS system to access its private key to send information to the other node, the passphrase must be entered as well. The z/OS node validates this key certificate information against the information stored in its GSKYYMAN database.

The cipher suites are listed in the order of preference, and the first one that matches a cipher suite defined for the other node is used to establish a session.

Chapter 5

Maintain Secure+ Option Nodes

After you have set up your Connect:Direct Secure+ Option for OpenVMS system, perform additional maintenance tasks as needed. This section provides procedures for performing the following Secure+ Option maintenance tasks:

- ✦ *Add a Node Record* on page 37
- ✦ *Delete a Node Record* on page 38
- ✦ *View an Existing Node Record* on page 38
- ✦ *Get a List of All Configured Nodes* on page 39

When you manually add a node to the Secure+ Option parameters file, you must also insert the new node in the network map to define it in the system. It is also a good idea to delete a node no longer being used in the network map and from the Secure+ Option parameters file.

Add a Node Record

You can manually add a node record to the Secure+ Option parameters file. The name of the node must match the name of an existing node in the network map for Secure+ Option to make the connection to that node. The fields in this procedure are described in detail in the field table on page 28.

1. With the SPADMIN main menu options on display, type **1** to select Add a Connect:Direct node to the Parmfile and press **Enter**.
2. Type the node name exactly as it is specified in the network map and press **Enter**.
3. For node type, press **Enter** to accept the default remote node type (R) or type **L** for the local node record and press **Enter**.
4. For the protocol, press **Enter** to accept the default protocol (No to disable Secure+ Option for this node) or choose one of the following options and press **Enter**:
   - **S** for SSL
   - **T** for TLS

- **D** for default (for a remote node to default to the local node’s protocol and other parameters)

5. For client authentication, type **Y** to enable client authentication or **N** to disable it and press **Enter**.
6. For the authentication timeout, type the number of seconds that the system waits to receive the Connect:Direct control blocks exchanged during the Secure+ Option authentication process.
7. For the root certificate file, type the fully qualified name of the root certificate file containing the trusted root certificate of the node with whom you want to have a secure connection and press **Enter**.
8. For the key certificate file, type the fully qualified name of the file containing the private key and the ID (public key) certificate and press **Enter**.
9. For the passphrase, type the passphrase used to access the private key and press **Enter**.
10. For the list of one or more ciphers to use during the authentication process, take one of the following actions:
    - Type a list of the ciphers in the order you want to use them, separated by commas and press **Enter**. For example, 12, 10 uses AES256_SHA as the first cipher choice followed by DES_CBC3_SHA.
    - Type **all** to include all secure ciphers and press **Enter**. The all option tries all secure options from cipher #3 through cipher #12.
11. At the Param values OK prompt, type **Y** and press **Enter** to add the node and its parameters.

Delete a Node Record

You can delete a node record in the Secure+ Option parameters file. You will want to do this if you have deleted a node record in the network map to keep these two files in sync. Complete the following procedure to delete an existing node record:

1. With the SPADMIN main menu options on display, type **3** to select the Delete an existing Connect:Direct node from the Parmfile option and press **Enter**.
2. Type the node name exactly as it is specified in the network map and press **Enter**.
3. For node type, press **Enter** to accept the default remote node type (R) or type **L** for the local node record and press **Enter**.

   If it exists, the node record is deleted.

View an Existing Node Record

Before you make changes to a existing node record, you may want to view that node’s parameters prior to updating its node record. Complete the following procedure to view an existing node’s parameters:

1. With the SPADMIN main menu options on display, type **4** to select the Get an existing Connect:Direct node’s parameters option and press **Enter**.
2. Type the node name exactly as it is specified in the network map and press **Enter**.
3. For node type, press **Enter** to accept the default remote node type (R) or type **L** for the local node record and press **Enter**.

   The information for that particular node is displayed, as shown in the following example:

```
TSC5->spadmin

        1. Add a Connect Direct node to the Secure+ Parmfile
        2. Change an existing Connect Direct node's parameters
        3. Delete an existing Connect Direct node from the Secure+ Parmfile
        4. Get an existing Connect Direct node's parameters
        5. List all Connect Direct nodes in the Secure+ Parmfile

        8. Sync with Netmap
        9. Quit this procedure

    Type '?' at any prompt for a description of the information
    requested.
1,2,3,4,5,8,9  4
Enter node name S7.TEST1.35
Is S7.TEST1.35 local or remote? [L/R][R]
                 Node Name:    S7.TEST1.35
                 Node type:    R
 1.               Protocol:    T
 2.  Client Authentication:    n
 3. Authentication timeout:    30
 4.Certificate common name:    sterling
 5.  Root Certificate file:    [ndm.3400.certs]wizcert
 6.   Key Certificate file:    [ndm.3400.certs]wizkey
 7.             Passphrase:
 8.           Cipher suites:    AES256_SHA
```

Get a List of All Configured Nodes

You can get a list of all the node entries in the parameters file for record keeping purposes or to research node names before updating existing nodes or adding new nodes. Complete the following procedure to get a list of all configured nodes:

1. With the SPADMIN main menu options on display, type **5** to select the List all Connect:Direct nodes in the Parmfile option and press **Enter**.

   A list of all nodes including their types, names, and protocol is displayed, such as in the following example:

```
TSC5->spadmin

        1. Add a Connect Direct node to the Secure+ Parmfile
        2. Change an existing Connect Direct node's parameters
        3. Delete an existing Connect Direct node from the Secure+ Parmfile
        4. Get an existing Connect Direct node's parameters
        5. List all Connect Direct nodes in the Secure+ Parmfile

        8. Sync with Netmap
        9. Quit this procedure

    Type '?' at any prompt for a description of the information
    requested.
1,2,3,4,5,8,9  5
Local     TSC5.TEST1.332   NO
Remote    CSG.PROD390      NO
Remote    IT.TEST2.34      NO
Remote    TEST.ALL-TW      NO
Remote    TEST.ALL2ND      NO
Remote    K2.TEST1.343     NO
Remote    NSTOP.TEST1.343  NO
Remote    NSTOP.TEST1.350  TLS
Remote    QA.OS390.V4400   NO
Remote    S7.TEST1.343     NO
Remote    S7.TEST1.35      TLS
Remote    TSC5.TEST1.331   NO
Remote    TSC5.TEST1.332   NO
```

Chapter 6

Use CLI to Maintain the Secure+ Option Parameters File

You can run the Secure+ Option Administration (SPADMIN) tool as a command line interface task. This chapter describes the commands and their syntax for performing the following SPADMIN functions:

- ✦ Add a Node Record
- ✦ Update a Node Record
- ✦ Delete a Node Record

Note: You cannot perform the following functions using the command line interface; you must go through the SPADMIN main menu and use the interactive interface:

- ✦ Get an existing Connect Direct node's parameters
- ✦ List all Connect Direct nodes in the Secure+ Parmfile
- ✦ Sync with Netmap

General Syntax for the Command Line Interface

To run the Connect:Direct Secure+ Option for OpenVMS SPADMIN Tool as a command line interface, at the command line, change the default directory to ndm$$directory. Type the maintenance command you wish to use followed by the required NODEname and TYPE parameters and any additional optional parameters. For CLI commands, you can use the full command word or just the first three letters. For example, for the Delete command, you can type DELETE or DEL. For all required and optional parameters, you can use the full parameter or just the first four letters. For example, for the key certificate file parameter, you can type KEYCERT_FILE or KEYC. Optional parameters are enclosed in brackets.

```
SPADCLI ADD|CHAnge|DELete| NODEname /TYPE=L|R [/PROTocol=S|T|N|D] [/CLTImeout=300]
[/CAUTh=Y|N] [/CIPH_suite=1,2,3...] [/COMMon_name=”xxx”]
[/KEYCert_file=[directory]<filename>] [/TRUSted_root=[directory]<filename>]
[/PASSphrase=”xxxx”]
```

The following table describes the parameters used on the command line to perform the maintenance functions:

Parameter	Field Description	Valid Values
ADD|CHANGE|DELETE	Specifies the action you want to take, that is, to add, change, or delete a node definition.	ADD CHANGE DELETE
NODEname	Specifies the name of the node being configured.	Node name. You can enter up to 16 characters. Enter the nodename as a command parameter, not the parameter=value format.

TYPE	Specifies the record type.	L = Local node record
		R = Remote node record

PROTocol	Specifies the protocol used for the node.	SSL = SSL is enabled.
		TLS = TLS is enabled.
		N = No (Secure+ Option is disabled)
		D = Default to Local Node protocol and parameters (for remote node only)

CLTImeout	Specifies the maximum time, in seconds, that the system waits to receive the Connect:Direct control blocks exchanged during the Secure+ Option authentication process.	A numeric value equal to or greater than 0, ranging from 0 to 300. The default is **300** seconds.
CAUTh	Enables client authentication in addition to server authentication. If you enable client authentication, you can also specify the common name of the certificate file in the COMMON_NAME parameter to add another level of authentication validation.	Y = Client authentication is enabled. N = Client authentication is disabled.

Parameter	Field Description	Valid Values
CIPH_SUITE	Identifies a list of ciphers used by trading partners to encrypt transmitted data. Connect:Direct uses the first cipher that is common to both trading partners (SNODE) and the PNODE. If you want to specify a unique cipher for a trading partner, you can identify this information in the remote node record.	1–12 = The numbers of the individual cipher to use (separated by commas, for example, 1,2,3): 1. NULL_MD5 2. NULL_SHA 3. EXP_RC4_MD5 4. RC4_MD5 5. RC4_SHA 6. EXP_RC2_CBC_MD5 7. IDEA_CBC_SHA 8. EXP_DES_CBC_SHA 9. DES_CBC_SHA 10. DES_CBC3_SHA 11. AES128_SHA 12. AES256_SHA ALL = All ciphers from 3–12 are included, in that order.
COMMon_name	Specifies the common name of the certificate file, which can be used to provide additional client authentication. To enable this option, specify CLIENT_AUTHENTICATION=Y.	Common name specified on the certificate. You can enter up to 256 characters. You must enclose the common name in quotation marks because this field is case-sensitive.
KEYCert_file	Specifies the name of the key certificate file, which contains information used to authenticate a trading partner.	[directory]<filename>
TRUSted_root	Specifies the name of the file which contains one or more trusted root certificates used to authenticate ID (public) certificates sent by trading partners during the Secure+ protocol handshake. Enter the fully qualified name of the root certificate file.	[directory]<filename>
PASSphrase	Identifies the passphrase used to access the private key.	You must enclose the passphrase in quotation marks because this field is case-sensitive.

Add a Node Record

To add a node record in the Secure+ Option parameters file, change the default directory to ndm$$directory at the command line. Type the following command, specifying the node name, node type, protocol, and other parameter values you want to include, and press **Enter**.

```
SPADCLI ADD NODEname /TYPE=L|R [/PROTocol=S|T|N|D] [/CLTImeout=300] [/CAUTh=Y|N]
[/CIPH_suite=1,2,3...|ALL] [/COMMon_name=”XXX”]
[/KEYCert_file=[directory]<filename>] [/TRUSted_root=[directory]<filename>]
[/PASSphrase=”xxxx”]
```

The following example shows a remote node record added which requires the TLS protocol, client authentication, and verification of the common name in the PNODE's certificate.

```
tsc3$@spadmin_cli.com add sun36/type=R/prot=T/cltim=20/cauth=y/ciph=all
/comm=”vms34mc”/keyc=disk$axp:[dal3.certs]vkeycert.txt
/trus=disk$axp:[dal3.certs]vtrusted.txt /pass=”vms34mc”
Connect Direct Node added successfully
```

Update a Node Record

To update a node record in the Secure+ Option parameters file, change the default directory to ndm$$directory at the command line. Type the following command, specifying the node name, node type, and parameter values you want to modify, and press **Enter**.

```
SPADCLI CHAnge NODEname /TYPE=L|R [/PROTocol=S|T|N|D] [/CLTImeout=300] [/CAUTh=Y|N]
[/CIPH_suite=1,2,3...|ALL] [/COMMon_name=”XXX”]
[/KEYCert_file=[directory]<filename>] [/TRUSted_root=[directory]<filename>]
[/PASSphrase=”xxxx”]
```

The following example shows the result of the security administrator changing a remote node called CD41 to default to the protocol and parameters defined in the local node record.

```
$@spadmin_cli.com CHANGE CD41/type=R/prot=D
Changing cd node CD41
Successfully Updated Connect Direct node CD41
```

Delete a Node Record

To delete a node record in the Secure+ Option parameters file, change the default directory to ndm$$directory at the command line. Type the following command, specifying the name and type of the node you want to delete, and press **Enter**.

```
SPADCLI DELete NODEname /TYPE=L|R
```

The following example shows a remote node called SUN36 being deleted:

```
$@spadmin_cli.com delete sun36/type=r
Deleting cd node SUN36
Deleted cd node SUN36
```

Chapter 7

Secure+ Option Statistics

Connect:Direct for OpenVMS logs statistics for Connect:Direct Process activity, including Secure+ Option information for a session.

The following samples of Connect:Direct Process statistics records contain information for Secure+ Option support. For information about viewing statistics for Connect:Direct for OpenVMS Process statistics, refer to the *Connect:Direct for OpenVMS User's Guide*.

Step End Statistics Record

When you use the Show Statistics command to view the information for a session with the TLS protocol enabled, you see output similar to the following. The Secure+ Option fields are in bold. A description for the fields follows the examples.

```
==============================================================================
3.4                      S E L E C T    S T A T I S T I C S
==============================================================================
Date    => 12.15.2007 Time      => 11:49:30.37         PROCESS - STEPEND
Pnumber => 67          Xlate     =>                   Start Date=> 12.15.2007
Pname   => LOOPBACK    Compress  => NO                Start Time=> 11:49:28.33
Msgid   => SCPA000I    Restart   => NO                End Date  => 12.15.2007
Rtncd   => 0           Link Stat => OK                End time  => 11:49:29.94
FDBK    => 0           Snode     => NSTOP.TEST1.350   Direction => SEND
Step    => STEP01      Submitter => TSC5.TEST1.332    TEST1
RUsize  => 4096
  From PNODE DSN= DKA100:[TEST1.NDM_W00]INITPARMS.DAT
           I/O Bytes=> 1414        Xmit Bytes=> 1486
           I/O Recs => 36          Xmit RUs  => 1          Comp%=> -5.09
  To   SNODE DSN= $DEV.TSTDATA.VMSTEST
           I/O Bytes=> 1414        Xmit Bytes=> 1486
           I/O Recs => 36          Xmit RUs  =>            Comp%=> -5.09
Protocol=> TLSv1
Cipher  => AES256-SHA
SCPA000I:   Feedback: 0    Reply: 0  Function: DMCOPYRT
Copy operation successful.
A copy operation completed successfully.
SYSTEM ACTION: Execution continued with the next CD process step.
RESPONSE: None.
```

The following statistics are displayed for the Copy Termination, Stepend, and Process End Statistics records.

Field	Description	Valid Values
Protocol	Specifies whether TLS or SSL is used.	TLSv1 SSLv3
Cipher	Specifies the ciphers used in the session.	Any valid cipher

Appendix A

Configuration Worksheets

Use the worksheets in this appendix to record the configuration information for Connect:Direct Secure+ Option for OpenVMS. The local node record defines the most commonly used security and protocol settings for the site. Each remote node record defines the specific security and protocol used by a trading partner.

- ✦ Use the *Local Node Record Security Feature Definition Worksheet* to define the protocol, certificate information, and ciphers for the local node. The local node record can be used as a default for all remote records or can be ignored, and each remote record individually configured.
- ✦ The *Remote Node Record Security Feature Definition Worksheet* is a record of the defined security functions for connections with remote nodes. Make a copy of the blank *Remote Node Record Security Feature Definition Worksheet* for each remote node record that you are configuring for Connect:Direct Secure+ Option for OpenVMS operations.

For a complete description of all parameters and their settings, see *Terminology for Authentication for SSL and TLS Protocols* on page 14. For instructions on how to configure the local node record and remote node records, see Chapter 4, *Configure Nodes for Secure+ Option*.

Local Node Record Security Feature Definition Worksheet

Record the security feature definitions for the Connect:Direct Secure+ Option for OpenVMS local node record on this worksheet. Note that the certificate common name and passphrase are case-sensitive.

This information is used for remote node definitions that specify the D(efault) option in the protocol field.

Local Node Name: ____

Protocol: SSL _____TLS_____ No_____

Client authentication: Yes ______ No _____

Authentication timeout (in seconds): ________ (A numeric value equal to or greater than 0 sec., from 0–300)

Certificate Common Name:
(enclose in quotation marks
if using CLI:) ____

Root certificate file:
(fully-qualified) ____

Key Certificate file:
(fully qualified) ____

Passphrase for the Private Key:
(enclose in quotation marks
if using CLI) ____

Cipher Suite(s): Circle the ciphers you want to use:

1. NULL_MD5
2. NULL_SHA
3. EXP_RC4_MD5
4. RC4_MD5
5. RC4_SHA
6. EXP_RC2_CBC_MD5
7. IDEA_CBC_SHA
8. EXP__DES_CBC_SHA
9. DES_CBC_SHA
10. DES_CBC3_SHA
11. AES128_SHA
12. AES256_SHA

Remote Node Record Security Feature Definition Worksheet

Record the security feature definitions for a remote node record on this worksheet. Make a copy of this worksheet for each remote node defined in the Connect:Direct Secure+ Option for OpenVMS parameters file that you are configuring for Connect:Direct Secure+ Option for OpenVMS operations.

If you enable client authentication, you can also enter a certificate common name for an additional level of authentication. Note that both the certificate common name and passphrase for the private key are case-sensitive. Include a fully-qualified path in the file names for the root certificate file and key certificate file.

Remote Node Name: ____

Protocol: SSL _____TLS_____ No_____ Default (to Local node protocol)_____

Client authentication Yes ______ No _____

Authentication timeout (in seconds): ________ (A numeric value equal to or greater than 0 sec., from 0–300)

Certificate Common Name: (enclose in quotation marks if using CLI:) ____

Root certificate file: ____

Key Certificate file: ____

Passphrase for Encrypted Private Key (enclose in quotation marks) if using CLI: ____

Cipher Suite(s): Circle the ciphers you want to use:

1. NULL_MD5
2. NULL_SHA
3. EXP_RC4_MD5
4. RC4_MD5
5. RC4_SHA
6. EXP_RC2_CBC_MD5
7. IDEA_CBC_SHA
8. EXP__DES_CBC_SHA
9. DES_CBC_SHA
10. DES_CBC3_SHA
11. AES128_SHA
12. AES256_SHA

Appendix B

Understanding the Certificate File Layout

The SSL and TLS security protocols use a secure server RSA X.509V3 certificate to authenticate your site to any client that accesses the server and provides a way for the client to initiate a secure session. You can obtain a certificate from a certificate authority or you can create a self-signed certificate. When you obtain a certificate, a trusted root certificate and private key are created. This appendix describes the layout of the trusted root certificate and the key certificate file.

Certificate Files

Secure+ Option uses two certificate files to initiate TLS or SSL sessions: a trusted root certificate file and a key certificate file.

When you obtain a root certificate from a certificate authority, you receive a trusted root certificate file. To configure Secure+ Option, add the name and location of the trusted root certificate file to the node record using the Secure+ Admin Tool. You can use any text editor to add or delete certificates to this file.

If you set up your own PKI infrastructure, you may chain more than two certificates, including a CA root certificate, one or more intermediate CA certificates, and an identity certificate. You can create chained certificates using one of the following methods:

- ✦ Using the Local Key Certificate File—In a chain of two certificates, the local key certificate file contains a private key and an identity certificate. In a longer chain, the key certificate file contains the private key and the identity key, followed by the intermediate CA certificates.
- ✦ Using the Remote Trusted File— In a chain of two certificates, the remote trusted file contains the CA root certificate. In a longer chain, the remote trusted file contains the CA root certificate and all the intermediate CA certificates.

Formats

The formats discussed in this section apply to the certificate files used with Secure+ Option.

General Object Format

All objects are formatted in the Privacy Enhanced Mail (PEM) style, beginning with a line in the format. Below is a sample object format:

```
-----BEGIN <object>-----
```

and end with:

```
-----END <object>-----
```

In this sample, <object> is a placeholder for the name of the object type: CERTIFICATE or ENCRYPTED PRIVATE KEY.

Certificate Format

A certificate is encoded as a general object with the identifier string CERTIFICATE or X.509 CERTIFICATE. The base64 data encodes a Bit Error Rate (BER)-encoded X.509 certificate. This is the same format used for PEM. Anyone who provides or understands PEM-format certificates can accommodate the certificate format. For example, VeriSign commonly fulfills certificate requests with certificates in this format, SSLeay supports them, and SSL servers understand them. Both Netscape and Microsoft support this format for importing root CA certificates.

Private Key Format

A private key is encoded as a general object with the identifier string ENCRYPTED PRIVATE KEY. The base64 data encodes a BER-encoded PKCS#8 Private Key object. The passphrase associated with the Private Key is required for Secure+ Option and is stored in the Secure+ Option parameters file. Additional encryption is used to prevent the passphrase from being discovered.

Sample Certificate Files

In the sample user certificate below, a private key is followed by the server certificate, which is followed by the root certificate.

```
Sample User Certificate

-----BEGIN ENCRYPTED PRIVATE KEY-----
MIICCDAaBgkqhkiG9w0BBQMwDQQIIfYyAEFKaEECAQUEggHozdmgGz7zbC1mcJ2r
.
.
.
IGpupStY5rLqqQ5gwLn45UWgzy6DM96CQg6+Dyn0N9d1M5lIg2wlnUwE8vI=
-----END ENCRYPTED PRIVATE KEY-----

User/Server Certificate
-----BEGIN CERTIFICATE-----
MIICUDCCAdoCBDaM1tYwDQYJKoZIhvcNAQEEBQAwgY8xCzAJBgNVBAYTAlVTMRMw
.
.
.
iKlsPBRbNdq5cNIuIfPS8emrYMs=
-----END CERTIFICATE-----

// Final Root Certificate (optional)
-----BEGIN CERTIFICATE-----
MIICUDCCAdoCBDaM1tYwDQYJKoZIhvcNAQEEBQAwgY8xCzAJBgNVBAYTAlVTMRMw
.
.
.
iKlsPBRbNdq5cNIuIfPS8emrYMs=
-----END CERTIFICATE-----
```

In the sample root certificate below, the trusted.txt file contains a list of trusted root certificates.

```
Sample Root Certificate

RSA Commercial CA - exp. Dec 31, 2003
-----BEGIN CERTIFICATE-----
MIICUDCCAdoCBDaM1tYwDQYJKoZIhvcNAQEEBQAwgY8xCzAJBgNVBAYTAlVTMRMw
.
.
.
iKlsPBRbNdq5cNIuIfPS8emrYMs=
-----END CERTIFICATE-----

RSA Commercial CA - exp. Dec 31, 2010
-----BEGIN CERTIFICATE-----
MIICUDCCAdoCBDaM1tYwDQYJKoZIhvcNAQEEBQAwgY8xCzAJBgNVBAYTAlVTMRMw
.
.
.
iKlsPBRbNdq5cNIuIfPS8emrYMs=
-----END CERTIFICATE-----
```

Glossary

A

Asymmetric key encryption

Process using two separate, but closely integrated, encryption keys: one public and one private. Each key is one way in that something encrypted using the public key can only be decrypted using the private key and vice versa. Something encrypted using the public key cannot be decrypted using the same public key.

Asymmetric keys

A separate, but integrated, user key pair comprised of one public key and one private key. Each key is one way meaning that a key used to encrypt information cannot be used to decrypt information (e.g., data encrypted with the public key can only be decrypted using the private key of that particular key pair).

Authentication

The process of ensuring the identity of various connecting user or device participants exchanging electronic data. Verifies the person or server at either end of a message is in fact who they/it claim to be and not an impostor. Provides assurance that participants are legitimate persons or devices with appropriate information access permissions.

C

CA-Signed Certificate

A digital document issued by a certificate authority that binds a public key to the identity of the certificate owner, thereby enabling the certificate owner to be authenticated. An identity certificate issued by a CA is digitally signed with the private key of the certificate authority.

Certificate

A document obtained from a certificate authority (CA) by generating a certificate signing request (CSR) that contains specific information in a specific format about the requester. It typically contains (1) distinguished name and public key of the server or client; (2) distinguished name and digital signature of the CA; (3) period of validity (certificates expire and must be renewed); and (4) administrative and extended information. The CA analyzes the CSR fields, validates the accuracy of the fields, generates a certificate, and sends it to the requester. Also referred to as a digital certificate, public key certificate, digit ID, or identity certificate.

Certificate Authority

An organization that issues digitally-signed certificates. The certificate authority authenticates the certificate owner's identity and the services that the owner is authorized to use, issues new certificates, renews existing certificates, and revokes certificates belonging to users who are no longer authorized to use them. The CA digital signature is assurance that anybody that trusts the CA can also trust that the certificate that it signs is an accurate representation of the certificate owner.

Certificate Common Name

A part of the certificate, which can be used in client authentication to enable the remote node (SNODE) to verify the identify of the PNODE when a secure connection is being established.

Certificate Revocation List

A list of certificates that have been revoked.

Certificate Signing Request (CSR)

An output file sent through E-mail to a certificate authority to request an X.509 certificate.

Cipher

A cryptographic key exchange algorithm that enables you to encrypt and decrypt files and messages with the SSL or TLS protocol.

Cipher Text

Data that is encrypted. Cipher text is unreadable until it is converted into plain text (decrypted) with a key.

Client

The entity that initiates a communication session. See also Primary Node.

Client Authentication

A level of authentication that requires the client to authenticate its identity to the server by sending its certificate. The server must request a certificate before the client sends it.

Configuration File

A file that contains instructions and definitions upon which the system bases its processing.

Commands

Connect:Direct commands initiate and monitor activity within the Connect:Direct system.

Confidentiality

Assurance that data is not read or accessed by unauthorized persons.

D

Decryption

The process of transforming encrypted data back into meaningful information.

Digital signature

Process using public and private keys to verify participant identity in the exchange of electronic information. A digital signature uniquely authenticates the person "signing" an electronic document much like a human signature uniquely identifies the person signing their name to a physical document. Because a private key is unique to each person, a value encrypted using the senders private key and subsequently decrypted using the senders public key authenticates the senders identity.

E

Encryption

The process of converting meaningful data into a meaningless form to protect the confidentiality of sensitive information.

Encryption algorithm

The set of mathematical logic that converts (encrypts/decrypts) data.

I

Integrity

Assurance that data is not modified (by unauthorized persons) during storage or transmittal.

K

Key

A unique numerical value which feeds into an encryption algorithm, setting the encryption or decryption process into motion.

Key Certificate File

A file that contains the encrypted private key and the ID (public key) certificate, which is used to authenticate an entity.

N

Network map

The file (netfile.dat) that identifies all valid Connect:Direct nodes in the network. One network map file is associated with each Connect:Direct local node. The network map has one entry for each of the other Connect:Direct nodes to which the local Connect:Direct node communicates. The network map entries also contain the rules or protocol that the nodes adhere to when communicating.

P

Primary Node (PNODE)

The Connect:Direct node that initiates a session and is the controlling node. Every Process has one primary node and one secondary node.

Private key

String of characters used as the private, “secret” part of a complementary public-private key pair. The asymmetric cipher of the private key is used to decrypt data that is encrypted with its complementary public key. Data that is encrypted with a Public Key can only be decrypted using its complementary Private Key. The private key is never transmitted.

Protocol

The rules determining the format and transmission of data. SSL and TLS are cryptographic protocols which provide secure communications on the Internet.

Public key

The publicly available component of an integrated asymmetric key pair.

R

Root Certificate File

File which contains one or more trusted root certificates used to authenticate ID (public) certificates sent by trading partners during the Secure+ protocol handshake.

S

Secondary Node (SNODE)

The Connect:Direct node that interacts with the primary node (PNODE) during Connect:Direct Process execution and is the non-controlling node. Every Process has one secondary node and one primary node.

Secure Sockets Layer (SSL)

A protocol that provides secure communications with transport protocols, including FTP, over TCP/IP. It is an open, non-proprietary Internet protocol that is widely adopted as standard. SSL ensures point-to-point security, meaning that the data is secured as it is transmitted across a single socket.

Self-Signed Server Certificate

A self-generated server certificate that identifies your organization. It is often used during the period between your request and receipt of a certificate from a public certificate authority. If self-signed server certificates are used, the trusted root signing certificate must be installed in the client manually.

Server

The location that receives communication from a client.

Session Key

Cryptography key intended to encrypt data for a limited period of time, typically only for a single communications session between a pair of entities. When the session is over, the key is discarded and a new one established when a new session takes place.

T

Transport Layer Security (TLS)

A protocol that provides secure communications with transport protocols, including FTP, over TCP/IP. It is an open, non-proprietary Internet protocol that is widely adopted as standard. TLS ensures point-to-point security, meaning that the data is secured as it is transmitted across a single socket.

Trusted Root Certificate File

See *Root Certificate File*.

U

Unsecure Connection

An FTP connection that has no security.

X

X.509 Certificate

Public key certificate specification developed as part of the X.500 directory specification, and often used in public key systems.

Index