Connect:Direct® Secure+ Option for UNIX

Implementation Guide

Version 4.0

Sterling Commerce
An IBM Company

Connect:Direct Secure+ Option for UNIX Implementation Guide
Version 4.0

First Edition

Sterling Commerce, Inc.

4600 Lakehurst Court Dublin, OH 43016-2000 * 614/793-7000

Contents

Chapter 1

About Connect:Direct Secure+ Option for UNIX

The Connect:Direct Secure+ Option for UNIX application provides enhanced security for Connect:Direct and is available as a separate component. It uses cryptography to secure data during transmission. You select the security protocol to use with Secure+ Option.

Tthis section describes:

- Security Concepts
- Secure+ Option Protocol Support
- Secure+ Option Tools
- Before You Begin
- Connect:Direct Secure+ Option for UNIX Documentation

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 implement multiple layers of security. You can select one of three security protocols to secure data during electronic transmission: Transport Layer Security (TLS), Secure Sockets Layer protocol (SSL), or Station-to-Station protocol (STS). Depending on the security needs of your environment, you can also validate certificates using the Sterling External Authentication Server application.

Connect:Direct provides alternative cryptographic solutions depending upon the protocol enabled. The protocols available depend upon the version of Secure+ Option installed. The following table identifies the protocols available in each version of Secure+ Option and the encryption algorithms available for each protocol:

Secure+ Version	Protocol	Encryption Algorithms										
		ECC	IDEA	RC4	DES	Triple DES	AES	HMAC	RSA	DSA	SHS	RNG
3.8	STS	X	X		X	X						
	SSL			X	X	X						
	TLS			X	X	X	X					
4.0	STS	X	X		X	X						
	SSL			X	X	X	X					
	TLS			X	X	X[1]	X[1]	X[1]	X[1]	X[1]	X[1]	X[1]

1 These encryption algorithms are part of the Sterling Crypto-C module, which is FIPS 140-2 certified. When Secure+ Option is operating in FIPS-mode, these algorithms are available for the TLS protocol.

Secure+ Option Protocol Support

Before you configure Connect:Direct Secure+ Option for UNIX, you must determine the protocol that you and your trading partners will use to secure communications sessions. For planning information, see Chapter 2, *Planning Your Implementation of the SSL or TLS Protocol* and Chapter 4, *Planning Your STS implementation for Remote Nodes*.

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

The TLS and SSL protocols use certificates to exchange a session key between the node that initiates the data transfer process (the primary node, or PNODE) and the other node that is part of the communications session (the secondary node, or 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 (self-signed) or issued by a certificate authority (CA). See *Self-Signed and CA-Signed Certificates* on page 18 for details on the differences between self-signed and CA-issued certificates.

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, thus vouching for an entity's identity. A 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 file that includes this certificate and supply the trusted root file to the partner in a connection.

The Sterling External Authentication Server application enables you to validate certificates that are passed during an SSL or TLS session. You can use the Sterling External Authentication Server application to configure certificate chain validation, including the option to validate certificates against one or more Certificate Revocation Lists (CRLs) that are stored on an LDAP server. You can also configure the Sterling Authentication Server application to return attributes associated with the incoming certificate, such as group information, that are stored on an LDAP server. See the *Sterling External Authentication Server Release Notes* for installation information.

For more information on configuring Connect:Direct Secure+ Option for UNIX for external authentication, see the following:

- *Enabling or Disabling External Authentication for a Remote Node* on page 43
- *Configuring External Authentication in the .SEAServer Record* on page 48

FIPS 140-2 Mode for the TLS Protocol

The Sterling Crypto-C module, which is FIPS 140-2 certified, provides you with a FIPS solution for Connect:Direct Secure+ Option for UNIX. FIPS-mode operation is available only for the TLS protocol and the Crypto-C module. For a list of the UNIX platforms supported by the Crypto-C module, see the *Connect:Direct for UNIX Release Notes*.

Station-to-Station Protocol (STS)

The STS protocol is a three-pass variation of the basic Diffie-Hellman protocol. It enables you to establish a shared secret key between two nodes with mutual entity authentication. Nodes are authenticated using digital signatures that sign and verify messages. When you use the STS protocol, you are responsible for generating and managing authentication and signature public keys and exchanging these keys with your trading partners.

In an STS session, each message is signed by the PNODE with its current authentication private key (and possibly its previous authentication private key) and verified by the SNODE using the corresponding public key of the PNODE. Each node uses two session keys to process Connect:Direct control blocks: one for sending and the other for receiving. The encryption algorithms for control blocks and data copying functions are also determined. When strong authentication finishes successfully, control blocks and data are exchanged in an encrypted format for the entire session.

Secure+ Option Tools

Secure+ Option consists of four components: the Secure+ Administration Tool (Secure+ Admin Tool), the parameters file, the access file, and the Secure+ Command Line Interface (Secure+ CLI). The following sections describe these components and their function within Secure+ Option.

Caution: Only one instance of the Secure+ Admin Tool or the Secure+ CLI may be used at a time because they access the same configuration file. Do not open these tools at the same time or multiple copies of the same tool at the same time (two instances of Secure+ Admin or two instances of Secure+ CLI). Only the user who accessed the configuration file first will be able to save updates.

Administration Tool

The Secure+ Admin Tool is a graphical user interface (GUI) that enables you to configure and maintain the Secure+ Option environment. The Admin Tool is the only interface for creating and maintaining the Secure+ Option parameters file; operating system utilities and editing tools cannot be used to create or update this file.

Secure+ Option Parameters File

The Secure+ Option parameters file contains information that determines the protocol and encryption method used during encryption-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 for each trading partner who uses Secure+ Option to perform a secure connection. The local node record defines the most commonly used security and protocol settings for the node at the site. The local node record can also be used as a default for one or more remote node records. Each remote node record defines the specific security and protocol settings used by a trading partner. You should create a remote node record in the Secure+ Option parameters file for each Connect:Direct node that you communicate with even if the remote node does not use Secure+ Option.

Note: The parameters file is not dynamically updated. When multiple users update the parameters file, each user must close and reopen the file to display new records added by all sources.

When you create the parameters file, a record named .SEAServer is automatically added to the file, which enables Connect:Direct Secure+ Option for UNIX to interface with Sterling External Authentication Server during SSL/TLS sessions. External authentication is configured in this record and enabled/disabled in the local and remote node records.

For additional security, the parameters file is stored in an encrypted format. The information used for encrypting and decrypting the parameters file (and private keys) is stored in the Secure+ access file.

Access File

The Secure+ Option access file is generated automatically when you create the Secure+ Option parameters file for the first time. You type a passphrase when you first initialize Secure+ Option.

This passphrase is used to generate the keys necessary to encrypt and decrypt the entries in the Secure+ Option parameters file. The passphrase itself is not retained.

Your Secure+ Option administrator must secure the access file (<*cdinstall*>/ndm/secure+/nodes/.cdspacf).The administrator must have full create and update permissions to update this file. The Connect:Direct server must have read authority. To maintain a secure access file, the general user community should not have access permission. This file can be secured with any available file access restriction tool. Availability of the access file to unauthorized personnel can compromise the security of data exchange.

Secure+ Option Command Line Interface

The Java-based Secure+ Command Line Interface (Secure+ CLI) is provided to enable you to create customized scripts that automate implementing Secure+ Option. Sample UNIX scripts are provided as models for your customized scripts. You can save these scripts with another name, modify them to reflect your environment, and distribute them throughout your enterprise. For more information about using the Secure+ CLI, commands and parameter descriptions, and the scripts, see Chapter 8, *Automating the Setup of Secure+ Option Using the Secure+ CLI*.

Before You Begin

Before you configure the Connect:Direct environment for secure operations, ensure that you complete the following tasks:

- ✦ Identify Expert Security Administrator
- ✦ Assess Security Requirements of Trading Partners
- ✦ Plan Your Implementation of Secure+ Option
- ✦ Complete the Worksheets

Identify Expert Security Administrator

The instructions and information provided to assist you in implementing Secure+ Option assume that you have an expert UNIX security administrator who is familiar with your company's security environment. Identify who this individual is within your company and work with this individual as you plan your Secure+ Option implementation.

Assess Security Requirements of 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 those requirements.Contact your trading partners to gather the information necessary to coordinate your implementation of Secure+ Option.

Plan Your Implementation of Secure+ Option

After you have identified your security administrator and determined the security requirements of your trading partners, review the following information:

- Chapter 2, *Planning Your Implementation of the SSL or TLS Protocol*
- Chapter 3, *Preparing to Set Up Secure+ Option for the TLS or SSL Protocol*
- Chapter 4, *Planning Your STS implementation for Remote Nodes*
- Chapter 5, *Managing STS Keys*

Complete the Worksheets

Before you configure Secure+ Option, complete the worksheets in Appendix A, *Configuration Worksheets*. Use this information to configure the local and remote nodes to use Connect:Direct Secure+ Option for UNIX.

Connect:Direct Secure+ Option for UNIX Documentation

See the Connect:Direct release notes for your platform for a complete list of the product documentation.

About This Guide

The *Connect:Direct Secure+ Option for UNIX Implementation Guide* describes how to implement point-to-point security into Connect:Direct operations with Secure+ Option. This document includes information to plan, install, configure, and use Secure+ Option.

This guide is for programmers and network operations staff who install, configure, and maintain the Secure+ Option product, and assumes knowledge of the Connect:Direct system, including its applications, network, and environment. If you are not familiar with the Connect:Direct system, refer to the Connect:Direct library of manuals.

Task Overview

The following table directs you to the information required to perform the tasks documented in this guide:

Task	For More Information See
Understanding Secure+ Option	Chapter 1, *About Connect:Direct Secure+ Option for UNIX*
Planning for the TLS or SSL Protocol	Chapter 2, *Planning Your Implementation of the SSL or TLS Protocol* and Chapter 3, *Preparing to Set Up Secure+ Option for the TLS or SSL Protocol*
Planning for the STS Protocol	Chapter 4, *Planning Your STS implementation for Remote Nodes* and Chapter 5, *Managing STS Keys*

Task	For More Information See
Starting the Admin Tool	Chapter 6, *Setting Up Secure+ Option*
Populating the Secure+ Option parameters file	Chapter 6, *Setting Up Secure+ Option*
Configuring local and remote node definitions	Chapter 7, *Configuring Nodes for Secure+ Option*
Automating your implementation of Secure+ Option	Chapter 8, *Automating the Setup of Secure+ Option Using the Secure+ CLI* and Appendix D, *Automation Scripts*
Testing Secure+ Option with the STS protocol	Appendix C, *Testing Secure+ Option with the STS Protocol*
Viewing and modifying configuration information	Chapter 9, *Maintaining Secure+ Option*
Viewing Secure+ Option statistics	Chapter 10, *Accessing Secure+ Option Statistics*
Auditing parameters files and certificates for archival purposes	Chapter 11, *Secure+ Option Audits*
Understanding error messages and resolving errors	Chapter 12, *Troubleshooting*

Chapter 2

Planning Your Implementation of the SSL or TLS Protocol

Before you configure Connect:Direct Secure+ Option for UNIX, review the following concepts, requirements, and terms to ensure that you have all the resources and information necessary to implement the Transport Layer Security (TLS) protocol or the Secured Sockets Layer (SSL) protocol. After you have reviewed this information, see Chapter 3, *Preparing to Set Up Secure+ Option for the TLS or SSL Protocol*.

Overview of the TLS Protocol and the SSL Protocol

The TLS and SSL 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, called client authentication, is optional. 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 log of both nodes.

  To perform this level of authentication, the trading partner (SNODE) must have a key certificate file available at its site and the Connect:Direct server (PNODE) must have a trusted root file that validates the identity of either the Certificate Authority (CA) who issued the key certificate or the entity that created the certificate if it is self-signed.
- 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 (CN) 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.

Data Security Provided by TLS and SSL

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

- Authentication—Certificates used in the SSL or TLS session are digitally signed by a CA through an established procedure to validate an applicant's identity or digitally signed by the certificate owner-issuer. The SSL or TLS protocol validates the digital signature of the certificate being used.

  Additional authentication options are available using Sterling External Authentication Server. This application enables you to validate certificates that are passed during an SSL or TLS session. Using the Sterling External Authentication Server application, you can configure certificate chain validation, including the option to validate certificates against one or more Certificate Revocation Lists (CRLs) that are stored on an LDAP server. You an also configure the Sterling Authentication Server application to return attributes associated with the incoming certificate, such as group information, that are stored on an LDAP server. See the *Sterling External Authentication Server Release Notes* for installation information.

  For more information on configuring Connect:Direct Secure+ Option for UNIX for external authentication, see the following:

  - *Enabling or Disabling External Authentication for a Remote Node* on page 43
  - *Configuring External Authentication in the .SEAServer Record* on page 48
- Proof of data origin and data integrity validation—The certificate provides proof of origin of electronic transmission. Message digest (hashing) and encrypting the message digest ensures 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), called "ciphertext," before it is sent to the receiving node. The receiving node then converts the information back into a readable format (decryption), called "plaintext."

Benefits of TLS

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.
- If you are required to have a FIPS 140-2-certified solution, a FIPS-mode of operation is available in Connect:Direct Secure+ Option for UNIX for the TLS protocol.

Summary of Processing Using the SSL or TLS Protocol With Secure+ Option

After you configure Secure+ Option and your trading partners have defined your node in their parameters file, you are ready to exchange data securely with other security-enabled Connect:Direct nodes. Data is securely exchanged between two nodes using the protocol defined in the parameters file.

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.

Authentication

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 two 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. 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

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

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 certificates and digital certificates issued by certificate authorities 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 the 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.
	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 SSL and TLS Certificates

The following table defines the security terms associated with SSL and TLS certificates and communication sessions. The 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 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 who trusts the CA can also trust that the certificate it signs is an accurate representation of the certificate owner.
Certificate signing request (CSR)	Message sent from an applicant to a CA 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.
Cipher suite	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.
Key certificate file	File that contains the encrypted private key and the ID (public key) certificate. This file also contains the certificate common name that can be used to provide additional client authentication.
Passphrase	Passphrase used to access the private key.

Term	Definition
Private key	String of characters used as the private, “secret” part of a complementary public-private key pair. The symmetric cipher of the private key is used to sign outgoing messages and 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 and should never be shared with a trading partner.
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 confirm signatures on incoming messages and 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.
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 software.
Trusted root certificate file (also known as root certificate file	File that contains one or more trusted root certificates used to authenticate ID (public) certificates sent by trading partners during the Secure+ Option protocol handshake.

Chapter 3

Preparing to Set Up Secure+ Option for the TLS or SSL Protocol

After you have completed the planning process for implementing the SSL or TLS protocol with Secure+ Option, you need to complete pre-configuration tasks that will help you set up Secure+ Option to use these protocols.

Preparing to Set Up TLS or SSL

Before you configure the Secure+ Option environment for the TLS or SSL protocol, perform the following setup procedures:

- Complete a configuration worksheet for each trading partner

  Appendix A, *Configuration Worksheets*, provides worksheets to help you keep track of the configuration requirements for each trading partner. Before you begin configuring Secure+ Option, complete a worksheet for the local node record and a worksheet for each trading partner for whom you plan to enable Secure+ Option. Use the *Local Node Security Feature Definition Worksheet* on page 100 to record the settings you plan to enable for the local node. For each trading partner, complete a *Remote Node Security Feature Definition Worksheet* on page 101 and record the settings to enable Secure+ Option for the trading partner.

- Obtain a certificate and generate a key certificate file

  Before configuring Secure+ Option, obtain a certificate and generate a key certificate file. A certificate is created by a trusted certificate authority (CA) or you can create a self-signed certificate. Generate a key certificate file by combining the certificate file and the private key file. Refer to *Obtaining a Certificate and Generating a Key Certificate File* on page 22 in the next section for instructions on obtaining a certificate and generating a key certificate file. Refer to *Understanding the Certificate File Layout* on page 103 for information on certificate files.

- Exchange trusted root certificate files with your trading partners

- Populate the Secure+ Option parameters file

  To communicate with a node running Secure+ Option, the node must have a record in the Connect:Direct network map and the Secure+ Option parameters file. For more information on populating the Secure+ Option parameters file, see *Populating the Secure+ Option Parameters File* on page 34.

Obtaining a Certificate and Generating a Key Certificate File

The TLS and the SSL security protocols use a secure server RSA X.509V3 certificate to authenticate a site for any node that accesses the site.

Obtain a certificate (X509 ASN.1 Base64 format) from a CA or create a self-signed certificate. Create a private key file (ASN.1/Base64 encoded PKCS8/PKCS5 v1.5 or "traditional SSLeay" format) using Sterling Certificate Wizard or any Web server software.

Secure+ Option looks for a key certificate file to authenticate a site. This file combines information from the certificate file and the private key file.

Certificate Wizard is a Sterling Commerce product that provides a way to create the files needed to obtain a certificate and create a key certificate file. It can be used to:

- Generate a certificate signing request (CSR) that you send to the CA to request a certificate.
- Generate a self-signed certificate and act as your own CA.
- Generate a private key file. A private key file is created when you generate the CSR or the self-signed certificate.
- Create a key certificate file that combines the certificate file with the private key file. For information on generating a key certificate file, refer to *Generating a Key Certificate File for a CA Certificate* on page 22 or *Generating a Key Certificate File for a Self-Signed Certificate Using Certificate Wizard* on page 22.

For information on installing Certificate Wizard, refer to the *Sterling Certificate Wizard Release Notes*. For information on using Certificate Wizard, refer to the Certificate Wizard online Help.

Generating 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 Certificate Wizard or any Web server software.
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 Certificate Wizard or a text editor. The key certificate file combines information from the certificate file that you received from the CA and the private key file you generated.

Note: While a key certificate may contain information about its intended use, such as e-mail, Secure+ Option does not use this information. It uses client or server authentication.

Generating a Key Certificate File for a Self-Signed Certificate Using Certificate Wizard

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

1. Generate a self-signed certificate using Certificate Wizard. Certificate Wizard performs the following tasks when it generates a self-signed certificate:
   - Creates a private key called privkey.txt
   - Creates the trusted root file called cert.crt

2. Generate a key certificate file. The key certificate file combines information from the certificate file and the private key file. Certificate Wizard creates a key certificate file called keycert.txt.
3. If necessary, copy the key certificate file to the Connect:Direct server.

FIPS-Mode Considerations

For certificates that will be used for FIPS-mode sessions, the certificate signing algorithm must use a FIPS-approved digest, such as SHA1. If a signature uses, for example, an MD5 digest, it will fail authentication on the FIPS-enabled node. This applies to the entire certificate chain. The certificate is validated in both key certificate files and trusted root certificate files.

You need to ensure that the certificates used by your FIPS-mode trading partners are signed with an approved digest.

If you have private keys generated by Certificate Wizard version 1.2.03 or earlier, they are not encrypted with a FIPS-approved algorithm. Use the OpenSSL utility provided with Connect:Direct for UNIX to convert existing private keys generated using Certificate Wizard to FIPS-approved keys. This utility is located in the CDU 4.0.00/bin directory.The following example shows how to convert an existing private key using the OpenSSL utility:

```
>openssl pkcs8 -topk8 -v2 aes256 -in keycert.pem -out prikey.pem
Enter pass phrase for keycert.pem:
Verifying - Enter Encryption Password:
Loading 'screen' into random state - done
>openssl pkcs8 -topk8 - v2 des3 -in keycert.pem - out prikey.pem
```

The output file, privkey.pem, only contains the private key using the new PBE algorithm. You must rejoin the private key with the certificate by copying the certificate from the keycert.pem file and the private key from prikey.pem to a new key certificate file. You can use Certificate Wizard to generate this new key certificate file.To determine the PBE of a private key, use the asn1parse command. The private key must be the first PEM section in the file, or you must create a file containing only the private key. In the following example, the asn1parse command reveals that the PBE of the private key, prikey.pem, is MD5:

```
>openssl asn1parse -in prikey.pem
0:d=0 h1=4 1=673 cons: SEQUENCE
4:d=1 h1=2 1=27 cons: SEQUENCE
6:d=2 h1=2 1=9 prim: OBJECT     :pbeWithMD5AndDES-CBC...
```

Exchanging Trusted Root Files with Trading Partners

When validating certificates, the trading partner must have a copy of the trusted root certificate file to verify the identity of the entity who issued your certificate, and you must have a copy of the trading partner's trusted root certificate file to validate the entity that issued the trading partner's

certificate file. Obtain a copy of the trusted root file and copy it to the Secure+ Option directory on the Connect:Direct server. You can maintain multiple trusted root files for each trading partner you need to verify, or you can maintain all trusted root file information in one file. If you store all trusted root file information in one file, identify the location of this file in the local node record. If you maintain a separate file for each trading partner, configure the name of each trading partner's trusted root file in the corresponding remote node record.

Note: If the trading partner uses SSL for other secure communications, such as secure e-mail, the trading partner may already have a trusted root file for the CA used in the certificate.

Chapter 4

Planning Your STS implementation for Remote Nodes

If any of your trading partners use the STS protocol, you need to determine how you will manage the STS keys needed for secure communication sessions.

Understanding Key Management for the STS Protocol

When you configure a remote node record to use the STS protocol, you generate unique authentication and signature public keys. In addition, your trading partner generates authentication and signature public keys for that node. In order to communicate with the trading partner, all four keys must be defined in the parameters file for both your configuration and the trading partner's configuration. Therefore, you and your trading partner must exchange keys.

For the initial configuration, you manually exchange keys. You export keys and send them to the trading partner. Then you import the keys you receive from the trading partner into the parameters file. After the initial exchange, you can automate the exchange of key information.

If a remote node uses the STS protocol, you must decide how often to update keys and how to manage key files received from trading partners.

Key Exchange Method

After you exchange keys with a trading partner, both partners should enable the automatic key update feature for easier key management. When automatic key update is enabled, the updated key is sent to the trading partner node during the authentication process and the remote node record is updated with the new key values. Both you and your trading partner must enable automatic key update in order to use this feature.

Key Update Frequency

Decide how frequently to update authentication and signature keys. The more frequently you update key values, the more secure your environment is. When you turn on automated key updates, you can update keys daily, because the updated keys are sent to the trading partners automatically and securely during authentication.

Import Key File Management

Before you begin exchanging key files with a trading partner, you must consider how to manage key files. Secure+ Option names exported key files based on the name of the target node; therefore, new key files that you receive from a trading partner have the same name as the old key file. To avoid overwriting an old key file with a new one, you manage key files in one of the following ways:

- Import the new key file immediately after receiving it from your trading partner and then delete the old key file.
- Rename the key file upon receipt or have your trading partner rename it before sending it.
- Create a directory for each remote node and store each key file separately in the associated directory for use if the configuration is lost or node records are accidently deleted.
- Enable the automatic key update feature. For more information, see *Auto Update Public Keys* on page 28.

It is not necessary to retain key files since the files are stored in the configuration file after you import them. However, saving the key files allows you to reconfigure the parameters file if a configuration is lost or if the node record is accidentally deleted.

Summary of Processing Using the STS Protocol With Secure+ Option

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

STS Secure+ Option Data Exchange

Data exchange consists of three steps: authentication, sending data, and receiving data. The STS protocol data exchange process is described in the following sections. The primary node initiates the data exchange and the secondary node receives the data. The following description of processing depicts the PNODE as sending data and the SNODE as receiving data.

Authentication

The following steps occur during authentication:

1. The PNODE sends a control block to the SNODE. Information for creating an encryption key for the PNODE is included. The SNODE confirms that it has a record defined in the Secure+ Option parameters file for the PNODE. If so, it retains the information for key encryption for processing later. If not, the session fails.
2. The SNODE sends a control block signed with its private authentication key. Information for creating an encryption key is included.
3. The PNODE verifies the signature using the public authentication key of the SNODE.
4. The PNODE returns a control block signed with its private authentication key.

5. The SNODE verifies the signature using the public authentication key of the PNODE.
6. When authentication is successful, each node generates a shared session encryption key for encrypting control blocks.

Sending Customer Data

After communication is authenticated, the PNODE begins transmitting data.

- If data encryption is enabled, information for creating an encryption key is exchanged in the control blocks.
- If digital signature is enabled, the PNODE applies the signature algorithm to the data using its private signature key to ensure that the data was sent by the PNODE and has not been altered.
- If data compression is enabled, the PNODE compresses the data.
- If data encryption is enabled, the PNODE encrypts the data with an encryption algorithm using a shared secret encryption key generated specifically for this transmission. The encryption algorithm is determined at authentication.

Receiving Customer Data

The SNODE receives the data.

- If data is encrypted, the SNODE decrypts the data using the encryption algorithm available for both the PNODE and the SNODE.
- If the data is compressed, the SNODE decompresses it.
- If digital signature is enabled, the SNODE verifies the origin and integrity of the data by applying a verification algorithm using the public digital signature key of the PNODE.

Merging Secure+ Option Settings Using the STS Protocol

When two nodes use the STS protocol to exchange secure data, Secure+ Option settings are exchanged during authentication. These settings are then merged and the resulting value for each security function is used for the Connect:Direct session. The result is based upon the values defined on the primary node (PNODE) and the secondary node (SNODE).

The following sections describe the results of these merged values based on the PNODE and SNODE values.

Digital Signature

When Secure+ Option settings are merged, the most secure setting from either node is used for the digital signature feature. If either node enables the digital signature feature, digital signatures are used for the session. If both nodes disable digital signatures, digital signatures are not used. The following table shows the digital signature setting after the PNODE and SNODE values are merged:

PNODE Value	SNODE Value	Merged Results
Y	Y	Y
Y	N	Y

PNODE Value	SNODE Value	Merged Results
N	Y	Y
N	N	N

Algorithm for Encrypting Control Blocks

The algorithm that encrypts Connect:Direct control blocks used for strong authentication is the first algorithm ID in the PNODE list that is also in the SNODE list. If the nodes do not share a common algorithm, authentication fails.

Auto Update Public Keys

If both nodes enable the auto update function, the authentication and signature public key values are dynamically updated during authentication if the remote node supplies different values. Enabling auto update eliminates much of the work that has to be performed by the Secure+ Option administrator.

Data Encryption

The most secure setting from either node is used for data encryption. If the nodes do not share a common algorithm, the copy operation fails. The following table shows the setting after the PNODE and SNODE values are merged.

PNODE Value	SNODE Value	Merged Results
N	N	N
N	Y	The first algorithm ID in the SNODE list that is in the PNODE list.
N	Algorithm ID	The SNODE algorithm ID if it is in the PNODE list.
Y	N | Y | algorithm ID	The first algorithm ID in the PNODE list that is in the SNODE list.
algorithm ID	N | Y | algorithm ID	The PNODE algorithm ID if it is in the SNODE list.

Overriding STS Functions from the COPY Statement

When you configure a node to use the STS protocol, you can use the COPY statement in Connect:Direct to override the settings in the parameters file, if override is enabled. It is not always possible to disable digital signatures and data encryption. If either node enables these options, the options are used. For more information about using the COPY statement, refer to the Connect:Direct Processes Web site at http://www.sterlingcommerce.com/documentation/processes/processhome.html.

Setting Secure+ Option Function Values from the COPY Statement

The SECURE COPY statement parameter enables you to set data encryption and digital signatures features from the Connect:Direct COPY statement. You can always enable these features from the COPY statement, but you cannot necessarily disable them from the COPY statement.

The SECURE parameter value specified in the COPY statement overrides the value specified in the Secure+ Option remote node record *only* if **Enable Override** is selected in that remote node record. After the security settings are merged between the PNODE and SNODE, the strongest setting is always used. Therefore, the value specified from the COPY statement cannot disable data encryption or digital signatures if the SNODE has enabled them.

If the override function is disabled in that remote node record and the values specified on the COPY statement are different from the values specified in the remote node record, the copy operation fails with a return code of 8 and message ID CSPA011E indicating the error.

The following table describes the SECURE parameters for the Connect:Direct COPY statement.

Parameter Name and Syntax	Parameter Description	Valid Values
secure=(encrypt.data=*valid value*) or secure=(enc=*valid value*)	Enables/disables[1] Copy file encryption	Y | N | algorithm name DESCBC56 | TDESCBC112 | IDEACBC128 Default—Secure+ Option parameters file value
secure=(signature=*valid value*) or secure=(sig=*valid value*)	Enables/disables digital signature creation[1]	Y | N Default—Secure+ Option parameters file value

1 Data encryption and digital signatures cannot necessarily be disabled from the COPY statement.

If both parameters are used from the COPY statement, the syntax is as follows:

```
secure=(enc=y,sig=y)

or

secure=(encrypt.data=y,signature=y)
```

The following sample Process, SECURE, copies the file README.TXT from the PNODE to the SNODE, renames it to VERIFY.TXT, and enables data encryption and digital signatures.

```
step1 copy
    from (
         file="/usr/ndm/bin/readme.txt"
         Pnode
         )
    to (
         file="/usr/ndm/bin/verify.txt"
         Snode
         disp=(rpl)
       )
     secure=(encrypt.data=y,signature=y)
     ckpt=(ckptvalue)
pend
```

Chapter 5

Managing STS Keys

When you define the STS protocol for a remote node, you must exchange keys with the trading partner before using Secure+ Option with that node. To maintain the keys for the STS protocol, you perform the following procedures:

- ✦ Export keys to your trading partner
- ✦ Import keys from your trading partner

Exporting Keys

After you create signature and authentication keys for a node record, you must send this information to the trading partner. Export the information to a file that you can then send to the trading partner. Be sure that you export keys from the server associated with the local node.

Two ways of exporting keys are available. If you define several remote node records to use the STS protocol, you can create export key files for all remote node records at one time. If you want to export the key values for one remote node record, select the remote node record and export the selected remote node record key file. Complete the following procedure to export the authentication and signature public key values for one or more remote node records:

1. Create a specific directory for storing the public key files.
2. If necessary, open the Secure+ Option parameters file. The **Secure+ Admin Tool Main Window** is displayed.
3. Do one of the following:
   - If you want to export the key files for all remote node records, select **Export All Public Keys** from the **Key Management** menu item.
   - If you want to export the key file for one or more remote node records, highlight the node records to export and click the **Export Selection** option of the **Key Management** menu item.

   The **Select Export Destination Directory** dialog box is displayed.
4. Highlight the directory that you created in step 1 and click **Select**.

If you selected **Export All Public Keys**, a file with an **sxp** extension is created for all remote node records. If you exported the keys of a selected remote node record, one file is created for the remote node record. Each file is named after the corresponding remote node record.

Importing Keys

Before you can communicate with a trading partner, you must obtain their key data and import the information into the parameters file. Perform the following steps to import the authentication and signature public key values sent to you by the trading partner administrator.

1. From the **Secure+ Admin Tool Main Window**, highlight the remote record to import the key from.
2. Click the **Import Public Keys** option of the **Key Management** menu item. The **Select Import Source File** dialog box is displayed.
3. Go to the directory that contains the signature and authentication key file for this remote node.
4. Highlight the **sxp** file for this node and click **Import**. The key is imported to the remote node record.
5. Ensure that the remote keys are imported by viewing the **Remote Public Key** box located in the **Authentication Keys** tab and **Signature Keys** tab.

Automating Key Management

The first time that you exchange keys with a trading partner, you must manually export the public keys and send them to the trading partner. After the initial exchange, you can automate the key exchange process. To automatically exchange keys, auto update must be turned on in your parameters file and the trading partner's parameters file. For keys to be exchanged during a session, you must create new keys and also save the old keys as previous. When you save keys as previous, you also define an expiration date. You must complete the key exchange before the previous key pair expiration date occurs.

To activate the automated key management process, complete the following procedure:

1. From the **Secure+ Admin Tool Main Window**, double-click the remote node record to edit.
2. Click the **STS Options** tab.
3. Set **Enable Public Key Auto Updates** to **Yes**.
4. Click **OK**.

Chapter 6

Setting Up Secure+ Option

Before you can configure the node definitions that are necessary for using Secure+ Option, you must complete the following tasks:

- Installing Secure+ Option
- Starting the Secure+ Option Administration Tool
- Populating the Secure+ Option Parameters File

Installing Secure+ Option

You can install Secure+ Option using the Connect:Direct for UNIX installation script. For more information on installing Secure+ Option, see the *Connect:Direct for UNIX Getting Started Guide*.

WARNING: After Secure+ Option is installed, the system administrator is responsible for securing access to the Secure+ Option Administration Tool, Secure+ CLI, and parameters files. The Secure+ administrator and Connect:Direct Server need full permission to the Secure+ directory; no other users require access.

Starting the Secure+ Option Administration Tool

Use the Secure+ Option Administration Tool (Secure+ Admin Tool) or the Secure+ Option Command Line Interface (Secure+ CLI) to set up and maintain a Secure+ Option operation. This section provides instructions on using the Secure+ Admin Tool. Refer to *Chapter 8, Automating the Setup of Secure+ Option Using the Secure+ CLI,* for instructions on using the Secure+ CLI.

To start the Secure+ Admin Tool on a UNIX system, type the following command at the UNIX command prompt from within the ndm/bin directory:

```
spadmin.sh
```

The Secure+ Admin Tool starts and opens the Secure+ Option parameters file for the associated Connect:Direct node.

Note: The parameters file is not dynamically updated. When multiple users update the parameters file, each user must close and reopen the file to display new records added by all sources.

Accessing Secure+ Admin Tool Help

To access the Secure+ Admin Tool Help, from the Secure+ Option Admin Tool **Help** menu, click the **Help Topics** option.

Navigating Help

When you open Help, the table of contents is displayed in the left pane, and the active topic is displayed in the right panel. Click the icons in the following table to navigate the Help.

Icon	Name	Description
	Right Arrow	Moves to the next Help topic.
	Left Arrow	Moves to the previous Help topic.
	Index	Displays a list of index entries; either type or scroll through the list.
	Search	Searches for words or phrases contained in a Help topic.
	Table of Contents	Displays the table of contents.

- In the left frame of the Help window, click the topic, index entry, or phrase to display the corresponding topic in the right frame.
- If + is displayed in front of a topic in the table of contents, click + to expand the available topics.
- To close Help and return to Secure+ Option, click **Close** in the top left corner of the Help window.

Populating the Secure+ Option Parameters File

To communicate with a trading partner using Secure+ Option, you define a node record for that partner in *both* the Connect:Direct network map and the Secure+ Option parameters file. To set up the Secure+ Option environment, you can populate the Secure+ Option parameters file from entries defined in an existing network map.

When you populate the parameters file from the network map, a record is automatically created in the parameters file for each node entry in the network map. Initially, the .Local node record is disabled, and all other records are set to default to local.

Perform the following steps to populate the Secure+ Option parameters file with node entries defined in the Connect:Direct network map:

1. From the **Secure+ Admin Tool Main Window**, click the **Sync with Netmap** option of the **File** menu item.

   The **Available Netmaps** dialog box is displayed.

2. Navigate to the netmap.cfg file located in the *d_dir*/ndm/cfg/*node_name* directory. Select the netmap to open and click **Sync**. The **Select Netmap Entries to Add** dialog box is displayed.

3. Click **Add All.**

   The **Select Parameters File Entries to Delete** dialog box is displayed.

4. Click **Skip** to close the parameters file without deleting any entries.

   The Secure+ Option parameters file is populated and the **Secure+ Admin Tool Main Window** displays remote node records in the parameters file including the records you added from the network map.

Chapter 7

Configuring Nodes for Secure+ Option

Before you begin using Secure+ Option, you must configure nodes for secure operations. This section provides the following information:

- Node Configuration Overview
- Configuring the Secure+ Option .Local Node Record
- Customizing Remote Node Records
- Validating the Configuration
- Configuring External Authentication in the .SEAServer Record
- Using the .Client Record to Prevent Non-Secure API Connections to a Secure+ Option-enabled Server

Node Configuration Overview

When you import the network map records into the Secure+ Option parameters file, Secure+ Option parameters are disabled. To configure the nodes for Secure+ Option, complete the following procedures:

- Configure the Secure+ Option .Local node record

  Define the security options for the local node. Because TLS and SSL provide the strongest authentication with easy-to-maintain keys, configure the local node for one of these protocols. Determine which protocol is used by most trading partners and configure the local node with this protocol.
- Disable remote nodes that do not use Secure+ Option
- Customize a remote node for the following configurations:
  - To use a unique certificate file to authenticate a trading partner
  - To use a different self-signed or CA-signed certificate for client or server authentication
  - To identify a unique cipher suite used by a trading partner
  - To activate common name validation
  - To activate client authentication

- To enable FIPS 140-2 mode
- To activate external authentication

✦ Configure all remote nodes that use a protocol that is not defined in the local node

When you configure the local node, all remote nodes are automatically configured to the protocol defined in the local node. If a trading partner uses a different protocol, you must turn on the protocol in the remote node record. For example, if you activate the TLS protocol in the .Local node record and a trading partner uses the SSL protocol, configure the SSL protocol in the remote node record for the trading partner.

✦ If you want to use Sterling External Authentication Server to validate certificates:

- Update the .SEAServer record with the SEA Server host name and port
- Enable TLS or SSL
- Enable external authentication
- Specify the certificate validation definition to use

✦ If you want to prevent non-secure API connections from communicating with a Secure+ Option enabled server:

- Define a remote node called .Client
- Enable TLS or SSL
- Disable override

Configuring the Secure+ Option .Local Node Record

Configure the .Local node record with the protocol used by most of your trading partners. Because remote node records can use the attributes defined in the .Local node record, defining the .Local node record with the most commonly used protocol saves time. After you define the protocol in the .Local node record, all remote nodes default to that protocol. Also, identify the trusted root file to be used to authenticate trading partners. To configure the local node, refer to the *Local Node Security Feature Definition Worksheet* on page 100 that you completed for the .Local node record security settings and complete the following procedure:

1. From the **Secure+ Admin Tool Main Window**, double-click the **.Local** record. The **Edit Record** dialog box displays the **Security Options** tab, the node name, and the type of node.
2. Set the following **Security Options** for the local node:
   - Enable TLS Protocol or Enable SSL Protocol
   - Enable Override
3. If necessary, change the time out value in the **Authentication Timeout** box.

   Refer to the following table for an explanation of the **Security Options** boxes:

Field Name	Field Definition	Valid Values
Node Name	Specifies the node record name.	.Local This is not an editable field.
Base Record	Specifies the name of the base record. If an alias record is selected, the base record name is displayed in this box.	Name of the local Connect:Direct node.
Type	Specifies the current record type.	Local for a local record and Remote for a remote record. This is not an editable field.
Disable Secure+	Disables Secure+ Option.	Default value is **Disable Secure+**. **Note**: If this option is selected, override is enabled, and no remote node definition exists for the remote node in the Secure+ Option parameters file, Secure+ Option is bypassed.
Enable TLS Protocol	Enables TLS protocol to ensure that data is securely transmitted.	The default value is **Disable Secure+**.
Enable SSL Protocol	Enables SSL protocol to ensure that data is securely transmitted.	The default value is **Disable Secure+**.
Enable STS Protocol	Enables STS protocol to ensure that data is securely transmitted.	The default value is **Disable Secure+**.
Disable Override	Disables the ability to override values in the .Local node record with values in the remote node records.	The default value is **Disable Override**.
Enable Override	Enables override to allow values in the remote node records to override values in the .Local node record.	The default value is **Disable Override**.
Authentication Timeout	Specifies maximum time, in seconds, that the system waits to receive the Connect:Direct control blocks exchanged during the Secure+ Option authentication process. If you specify a value of 0, Connect:Direct waits indefinitely to receive the next message. Specify a time to prevent malicious entry from taking as much time as necessary to attack the authentication process.	A numeric value equal to or greater than 0, ranging from 0 to 3600. The default is **120 seconds**.

4. Click the **TLS/SSL Protocol** tab. The **TLS/SSL Options** dialog box is displayed.
5. Type the location of the trusted root certificate file to use to authenticate trading partners in the **Trusted Root Certificate File** box or click **Browse** and locate the file.

6. Click **Browse** next to the **Certificate File** box, locate the key certificate file and double-click the file to select it. The **Certificate Passphrase** dialog box is displayed.
7. Type the passphrase you specified when you created the certificate in the **Certificate Passphrase** box.
8. Click **OK**. The **Certificate File** box is populated with the certificate file name and location.
9. To enable FIPS 140-2 mode, Click **Yes**.
10. To enable client authentication, click **Yes**.
11. To view information about the trusted root file and the certificate, click **View Certificates**.
12. Highlight the cipher suites to enable in the **Available** list and click **Add**.

Note: Only certain cipher suites are supported in FIPS-mode. For a list of the FIPS-approved cipher suites, see *Special Considerations* in the *Connect:Direct for UNIX Release Notes*.

13. If necessary, use the **Up** and **Down** buttons to reorder the cipher suites. Place the cipher suites in order of preference.
14. Click the **External Authentication** tab. The **External Authentication** dialog box is displayed.
15. Choose one of the following options:
    - To enable external authentication on the remote node, click **Yes** in the **Enable External Authentication** box.
    - To disable external authentication on the remote node, click **No**.
16. Type the **Certificate Validation Definition** character string defined in Sterling External Authentication Server.
17. Click **OK** to close the **Edit Record** dialog box and update the parameters file.

Customizing Remote Node Records

After you configure the .Local node record, Secure+ Option enables the protocol and parameters that you configured for the local node for all remote node records. If all trading partners use the protocol and configuration defined in the .Local node record, you are now ready to begin using Secure+ Option.

However, even when a trading partner uses the same protocol as the one defined in the .Local node record, you may need to customize remote node records for the following configurations:

- Using a unique certificate file to authenticate a trading partner—During a TLS or SSL session, a certificate enables the PNODE to authenticate the SNODE. You identified a certificate in the .Local node record. If you want to use a unique certificate to authenticate a trading partner, you must identify this information in the remote node record.
- Using a self-signed certificate file to authenticate a trading partner—During a TLS or SSL session, a certificate enables the PNODE to authenticate the SNODE. If you want to use a self-signed certificate to authenticate a trading partner, you must identify this information in the remote node record.

- Activating client authentication—Client authentication requires that the SNODE validate the PNODE. If you want to enable client authentication, activate this feature in the remote node record. If you want another layer of security, you can activate the ability to validate the certificate common name.
- Identifying the cipher suite used by a trading partner—When configuring the TLS or SSL protocol, you enable cipher suites that are used to encrypt the transmitted data. When communicating with a trading partner, you and the trading partner must use the same cipher suite to encrypt data. If the trading partner does not enable a cipher suite that is enabled in your configuration, communication fails. If necessary, enable cipher suites in the remote node record.

Note: Only certain cipher suites are supported in FIPS-mode. For a list of the FIPS-approved cipher suites, see *Special Considerations* in the *Connect:Direct for UNIX Release Notes*.

If a trading partner uses a different protocol than the protocol defined in the .Local node record, you need to define the protocol in the remote node record. Refer to *Defining a Protocol for a Remote Node Record* on page 44.

When customizing remote node records, refer to the *Remote Node Security Feature Definition Worksheet* on page 101 you completed for each remote node.

Disabling Secure+ Option for a Remote Node

If a trading partner is not using Secure+ Option, you must disable Secure+ Option for that remote node record. Complete the procedure below to disable Secure+ Option for a remote node record:

1. Open the remote node record for which you want to disable Secure+ Option.
2. Click the **Security Options** tab.
3. Click the **Disable** Secure+ Option.
4. Click the **Disable Override** option.
5. Click **OK** to update the node record.

Adding Certificate Information and Trusted Root Information to a Remote Node Record

You identified a certificate and a trusted root file to use for authentication when you defined the .Local node record. If you want to use a different certificate to authenticate a trading partner or a different trusted root file to validate the identity of the trusted source who issues certificates, you need to add that information to the remote node record. Complete the following procedure to identify unique certificate information or trusted root information for a remote node record:

1. From the **Secure+ Admin Tool Main Window**, double-click the remote node record to open. The **Edit Record** dialog box is displayed.
2. Click the **TLS/SSL Protocol** tab.
3. Type the location of the trusted root certificate in the **Trusted Root Certificate File** box or click **Browse** and locate the file.
4. Click **Clear Certificate** to clear any associated certificate file from the remote node record.

5. Click **Browse** next to the **Certificate File** box, locate the key certificate file, and double-click the file to select it. The **Certificate Passphrase** dialog box is displayed.
6. Type the passphrase you specified when you created the certificate in the **Certificate Passphrase** box.
7. If you want to view trusted root files or key certificate files, click **View Certificates**.
8. Click **OK**.

Adding Self-Signed Certificate Information to a Remote Node Record

If you want to use a self-signed certificate to authenticate a trading partner and this information has not been identified in the .Local node record, you need to add that information to the remote node record. First, obtain a copy of the trading partner's self-signed certificate file and copy this file to a local drive, for example, *d_dir*/ndm/secure+certificates. If you plan to perform client authentication, you must also send a copy of your trusted root certificate file to the trading partner. Then, complete the following procedure to add self-signed certificate information to a remote node record:

1. From the **Secure+ Admin Tool Main Window**, double-click the remote node record to open. The **Edit Record** dialog box is displayed.
2. Click the **TLS/SSL Protocol** tab.
3. Type the location of the trading partner's self-signed certificate file in the **Trusted Root Certificate File** box or click **Browse** and locate the file.
4. Click **OK**.

Enabling or Disabling FIPS Mode

You can enable FIPS 140-2 mode for nodes using the TLS protocol. Complete the following procedure to enable or disable FIPS mode for a TLS remote node record:

1. From the **Secure+ Admin Tool** main window, double-click the remote node record to open. The **Edit Record** dialog box is displayed.
2. Click the **TLS/SSL Protocol** tab.
3. To enable FIPS mode in a remote node record, click **Yes** for the **Enable FIPS 140-2** option.

Note: Only certain cipher suites are supported in FIPS-mode. For a list of the FIPS-approved cipher suites, see *Special Considerations* in the *Connect:Direct for UNIX Release Notes*.

---

4. To disable FIPS mode in the remote node record, click **No** for the **Enable FIPS 140-2** option.
5. Click **OK** to close the **Edit Record** dialog box and update the parameters file.

Enabling or Disabling Client Authentication

Client authentication provides another level of security. If you activate client authentication, two levels of certificate validation occur. First, the PNODE validates the certificate from the SNODE (server authentication). Then, the SNODE performs client authentication by validating the certificate from the PNODE. Within client authentication, you can enable a third level of security. If you provide a certificate common name during setup, the client authentication process first

validates the certificate from the PNODE and then searches for the common name in the certificate file. If the SNODE cannot validate the PNODE certificate or locate the common name, communication fails. You must enable common name checking in the remote node record to use this function. It cannot be activated in the .Local node record.

If client authentication is enabled in the .Local node record, client authentication will be performed for all connections. If you want to disable client authentication for select trading partners, turn off this option in the remote node record for each of those trading partners.

Complete the following procedure to activate or deactivate client authentication for a remote node:

1. From the **Secure+ Admin Tool** main window, double-click the remote node record to open. The **Edit Record** dialog box is displayed.
2. Click the **TLS/SSL Protocol** tab.
3. Choose one of the following:

Note: If client authentication is enabled in the .Local node record, it is automatically enabled in all remote node records.

- To enable client authentication in a remote node record, click **Yes** in the **Enable Client Authentication** box.
- To disable client authentication in the remote node record, click **No** in the **Enable Client Authentication** box.
- To enable an additional level of security, type the certificate common name in the **Certificate Common Name** box.

4. Click **OK** to close the **Edit Record** dialog box and update the parameters file.

Enabling or Disabling External Authentication for a Remote Node

On a node-by-node basis, you can specify whether a remote node uses external authentication or if that remote node defaults to the external authentication setting in the .Local node record.

Complete the following procedure to configure a remote node for external authentication:

1. If necessary, open the remote node record. The **Edit Record** dialog box is displayed.
2. Click the **External Authentication** tab.
3. Choose one of the following options:
   - To enable external authentication on the remote node, click **Yes** in the **Enable External Authentication** box.
   - To disable external authentication on the remote node, click **No**.
   - To default to the external authentication setting defined in the .Local node record, click **Default to Local Node**.

Note: If external authentication is enabled in the .Local node record, it is automatically enabled in all remote node records.

4. Type the **Certificate Validation Definition** character string defined in Sterling External Authentication Server.
5. Click **OK** to close the **Edit Record** dialog box and update the parameters file.

Identifying the Cipher Suite to Use for Data Encryption

Cipher suites encrypt the data you send to a trading partner. To encrypt data, Connect:Direct uses the cipher suite that is available for both the SNODE (trading partner) and the PNODE. If you want to specify a unique cipher suite for a trading partner, you can identify this information in the remote node record. Identifying a cipher suite ensures that the selected cipher suite is used for data encryption. Complete the following procedure to identify the cipher suite used by a remote node.

1. If necessary, open the remote node record. The **Edit Record** dialog box is displayed.
2. Click the **TLS/SSL Protocol** tab.
3. Turn off **Default to Local Node** from the **Cipher Suites** section.
4. Highlight the cipher suite to use in the **Available** list and click **Add**.

Note: Only certain cipher suites are supported in FIPS-mode. For a list of the FIPS-approved cipher suites, see *Special Considerations* in the *Connect:Direct for UNIX Release Notes*.

5. If more than one cipher suite was selected, use the **Up** and **Down** buttons to reorder the cipher suites. Place the cipher suites in order of preference.
6. Click **OK** to close the **Edit Record** dialog box and update the parameters file.

Defining a Protocol for a Remote Node Record

When you configured the .Local node record, all remote node records are automatically configured to use the same protocol. If a trading partner uses a different protocol, you define this protocol in the remote node record. If you have not already done so, complete a *Local Node Security Feature Definition Worksheet* on page 100 for each trading partner who plans to use a protocol that is not defined in the .Local node record.

Configuring a Remote Node Record for the TLS or SSL Protocol

When the remote node record needs to be configured to use a different protocol (TLS or SSL) than is defined in the .Local node record, complete the following procedure:

1. From the **Secure+ Admin Tool Main Window**, double-click the remote node record to edit. The **Edit Record** window displays the **Security Options** tab, the node name, and the type of node.
2. Select the protocol to use by selecting **Enable TLS Protocol** or **Enable SSL Protocol**.
3. If necessary, change the time out value in the **Authentication Timeout** box.
4. Click the **TLS/SSL Protocol** tab.
5. Type the location of the trusted root certificate in the **Trusted Root Certificate File** box or click **Browse** and locate the file.

6. Click **Clear Certificate** to clear any associated certificate file from the remote node record.
7. Click **Browse** next to the **Certificate File** box, locate the key certificate file, and double-click the file to select it. The **Certificate Passphrase** dialog box is displayed.
8. Type the passphrase you specified when you created the certificate in the **Certificate Passphrase** box.
9. If you want to activate client authentication for the node, do the following:
   - Click **Yes** in the **Enable Client Authentication** box.
   - To enable another level of security, type the certificate common name in the **Certificate Common Name** box.
10. To identify the cipher suites that can be used by the remote node, do the following:
    - Turn off **Default to Local Node** from the **Cipher Suites** section.
    - Highlight the cipher suite to use in the **Available** list and click **Add**.
11. Click **OK** to close the **Edit Record** dialog box and update the parameters file.

Configuring a Remote Node for the STS Protocol

When a trading partner uses the STS protocol, configure the remote node record to use the STS protocol, including creating signature and authentication keys and identifying the export version of Secure+ Option being used by the trading partner.

If you have not already done so, complete a *Remote Node Security Feature Definition Worksheet* on page 101 for each trading partner who plans to use the STS protocol. Then use the procedures in this section to configure remote node records for the STS protocol.

After you create the keys for STS-enabled remote node records, you are responsible for managing them. The first time you use the STS protocol, you manually exchange keys with the trading partner. After you exchange keys for the first Secure+ Option enabled communication, you can then turn on the automatic key management function. The **Public Key Auto Update** function enables the public keys to be updated automatically during a communications session. This simplifies key management for ongoing communications.

If you activate the STS protocol for a remote node record, you can use the COPY statement in Connect:Direct Process statements to override settings in the remote node records. Refer to *Overriding STS Functions from the COPY Statement* on page 28 for information about overriding remote node record settings using in Process statements.

Complete the following procedure to configure a remote node for the STS protocol:

1. Double-click the remote node to configure. The **Edit Record** dialog box displays the **Security Options** tab.
2. On the **Security Options** tab, click **Enable STS Protocol**.
3. If you want to allow the COPY statement to override values in the remote node, click **Enable Override**.
4. If necessary, change the value in the **Authentication Timeout** box.
5. Click the **STS Protocol** tab. The **Edit Record** dialog box displays **STS Protocol** information.

6. Set the **STS Options** as desired. Refer to the following table for an explanation of the options:

Field Name	Field Definition	Valid Values
Enable Digital Signatures	Enables or disables digital signatures.	Yes | No | Default to Local Node
Enable Public Key Auto Updates	Enables or disables automatic update of public keys during authentication. If this option is enabled for the PNODE and the SNODE, the public keys of the PNODE and SNDE are automatically exchanged during authentication. Auto update can only occur over a secure connection; therefore, the initial exchange of keys between two nodes must be done manually.	Yes | No | Default to Local Node
Limited Export Version	If a trading partner is using an earlier version of Secure+ Option, identifies the export version of Secure+ Option being used.	Yes | No
Enable Encryption	Enables or disables data encryption during the copy operation. If Yes is selected, data encryption is enabled and the algorithm used depends on the merged value between the PNODE and SNODE. See *Summary of Processing Using the STS Protocol With Secure+ Option* on page 26 for more information about the merged values for data encryption. If No is specified, data encryption is disabled. If Algorithm is specified and encryption is enabled, Algorithm must be populated and **Enable STS Protocol** must be enabled. If the remote node enables encryption, the local node cannot disable it.	Yes | No | Default to Local Node Algorithm name
Encryption Algorithms	Lists acceptable data encryption algorithms when Copy file encryption is requested. List in order of preference, with the most-preferred algorithm first.	Default to Local Node DESCBC56 TDESCBC112 IDEACBC128

7. Click the **Authentication Keys** tab.
8. Click **Generate Key** to create the authentication keys for the node. The **Generate Random Number Seed** dialog box is displayed. If a previous key exists, a message is displayed, prompting you to save the previous key.
9. Do one of the following to generate a key:
   - Type an alphanumeric string at least 32 characters long in the **Random Number Seed** box and click **OK**. Secure+ Option uses the random number seed value to generate the authentication keys.
   - Click **Sample Value**, highlight all of the grid squares, and click **OK** to generate the random number seed.

     The **Authentication Keys** window displays the populated **Local Public Key** box.

10. If you are replacing an older key and you save the old key as the previous key, type the expiration date for the public key in the **Previous Key Pair Expiration Date** box. Refer to the following table for a definition of the **Authentication Keys** boxes.

Field Name	Field Definition	Valid Values
Local Public Key	Public key used for authentication.	Generated by Secure+ Option.
Previous Key Pair Expiration Date	Expiration date for previous authentication public keys. This eliminates the need to update Secure+ Option parameters files across all nodes in the network simultaneously when public keys for the local node are changed.	YYYY/MM/DD HH:MM:SS If time is not specified, 00:00:00 is used.
Remote Public Key	Displays the imported value from the trading partner.	Imported from the trading partner.

11. Click the **Signature Keys** tab.
12. Click **Generate Key** to create the signature keys for the node. The **Generate Random Number Seed** dialog box is displayed. If a previous key exists, a message is displayed, prompting you to save the previous key.
13. Do one of the following to generate a key:
    - Type an alphanumeric string at least 32 characters long in the **Random Number Seed** box and click **OK**. Secure+ Option uses the random number seed value to generate the signature keys.
    - Click **Sample Value**, highlight all of the grid squares, and click **OK** to generate the random number seed.

      The **Signature Keys** window displays the populated **Public Key** box.
14. If necessary, type the expiration date for the public key in the **Previous Key Pair Expiration Date** box.

    Refer to the following table for the name, definition, and valid values for the **Signature Keys** boxes.

Field Name	Field Definition	Valid Values
Local Public Key	Public key used for digital signature.	Generated by Secure+ Option.
Previous Key Pair Expiration Date	Expiration date for previous digital signature public keys. This eliminates the need to update Secure+ Option parameters files across all nodes in the network simultaneously when public keys for the local node are changed.	YYYY/MM/DD HH:MM:SS If time is not specified, 00:00:00 is used.
Remote Public Key	Displays the imported value from the trading partner.	Imported from the trading partner.

15. Click **OK** to close the remote node record and update the parameters file.

Validating the Configuration

Perform this procedure to ensure that the nodes have been properly configured. The validation process checks each node to ensure that all necessary options have been defined and keys have been exchanged. Perform the following steps to validate the parameters file:

1. From the **Secure+ Admin Tool Main Menu**, click **Validate Secure+** from the **File** menu. The **Secure+ Admin Tool - Validation Results** window is displayed.
2. If the parameters file is not correctly configured, warning and error messages are displayed.
3. Go back to the parameters file and make changes to correct each error reported.
4. Read each warning message. If necessary, change the parameters file to correct each warning.

Note: Warning messages do not always mean that the parameters file is configured incorrectly. Some warning messages are informational only.

5. Click **Close** to close the **Validation Results** window.

Configuring External Authentication in the .SEAServer Record

At installation, a record named .SEAServer is created in the parameters file, which enables Connect:Direct Secure+ Option to interface with Sterling External Authentication Server (SEAS) during SSL/TLS sessions to validate certificates. SEAS properties are configured in this record and enabled/disabled in the local and remote node records. Complete the following procedure to configure the server properties that will allow Connect:Direct for UNIX to interface with Sterling External Authentication Server:

Note: The values specified for this procedure must match the values specified in Sterling External Authentication Server.

1. Double-click the record called .SEAServer.
2. Type the **Host Name** for SEAS.
3. Type the **Port Number** where SEAS is listening. The default is **61366**.
4. Click **OK** to update the record.

Using the .Client Record to Prevent Non-Secure API Connections to a Secure+ Option-enabled Server

Connect:Direct servers that use Secure+ Option support secure API connections. Client applications that use the API include Connect Control Center, Sterling Java API, Connect:Direct Requester, and Connect:Direct Browser User Interface. Client applications created using the Sterling Java API support secure API connections; client applications created using the Sterling SDK do not support secure API connections.

If a Connect:Direct server has Secure+ Option enabled, an API configuration that connects to this server uses the .Client node record to determine if it has the authority to connect to the server. The override settings in the .Client node record determine if the client application has the authority to connect to the server. When override is enabled in the .Client record, secure and non-secure API connections are allowed to connect to the Connect:Direct server. When override is disabled in the .Client record, all client connections must use the effective protocol configured in the .Client or .Local node record.

Valid settings for API connections include TLS, SSL, or security disabled. An API configuration follows the same rules as other remote node connections with the following exceptions:

- API connections must use either the SSL or the TLS security protocol
- Certificate-based client authentication is not supported
- The local node is automatically overridden by settings in the .Client node definition

Since the local node is automatically overridden, Secure+ Option configurations that normally disable this setting will not be affected. In order to disable override and prevent API connections that are either nonsecure or that do not use the settings defined in the local node, configure the .Client node record and disable override.

Complete the procedure below to prevent non-secure connections to a Connect:Direct server with Secure+ Option enabled:

1. Double-click the remote node record called .Client.
2. Click the **Disable Override** option.
3. If necessary configure the TLS or SSL protocol in this node, if it is not defined in the .Local node record. Refer to *Configuring a Remote Node Record for the TLS or SSL Protocol* on page 44.

   Note: The TLS or SSL protocol must be configured in either the .Client or the .Local node record.

4. Click **OK** to update the node record.

Chapter 8

Automating the Setup of Secure+ Option Using the Secure+ CLI

The Java-based Secure+ Option Command Line Interface (Secure+ CLI) and sample scripts enable you to create customized scripts that automate creating an initial installation of Secure+ Option, populating the parameters file, and managing node records. You can then distribute these scripts throughout your enterprise to implement the Secure+ Option application. Before you create the scripts for distribution, consider creating an installation of Connect:Direct Secure+ Option for UNIX using the Admin Tool and testing it to verify the results.

Starting and Setting Up the Secure+ CLI

The following sections describe the commands and parameters used to start and set up the command line environment.

Starting the Secure+ CLI

To start the Secure+ CLI:

1. Go to *d_dir*/ndm/bin.
2. Type the following command:

```
spcli.sh
```

3. Press **Enter**.

Controlling the Display of Commands

Set the following parameters to define how error messages are captured:

Parameter	Definition	Values
-li	Switch to enable the display of commands to the terminal.	y | n
-lo	Switch to enable the display of output and error messages to the terminal.	y | n
-le	Switch to enable the display of errors to STDERR.	y | n
-e	Switch to tell the Secure+ CLI to exit when the return code is higher than the specified number. If you do not include this parameter, Secure+ CLI continues to run even after an error has occurred.	0 | 4 | 8 | 16
-p	The full path of the default parameters file directory. The parameters file in this directory is opened automatically.	
-h	Switch to display the usage of the Secure+ CLI.	

Controlling Help

The Help command determines what help information is displayed. You can list all Secure+ CLI commands and display help for individual commands.

Command	Description
help	Displays all the Secure+ CLI commands.
help <command>	Displays help for the specified command.

Specifying Delimiter Characters

Define the following commands to determine how error messages are captured:

Command	Definition	Values
Set begdelim= enddelim=	Defines beginning and ending character to use to enclose keywords that use blanks and other special characters.	Any character The default value is **“** (double quotes).

Using LCU Files to Encrypt Passwords for Use with the Secure+ CLI

The Secure+ CLI displays passwords in plain text. If your company security policy mandates that you use encrypted passwords, you can use the Local Connection Utility (LCU) to create an LCU file that contains non-encrypted information used to encrypt the password and the encrypted password. For more information on creating and using LCU files, see Appendix E, *Using the LCU to Configure Encrypted Passwords*.

Sample Scripts

The following scripts are provided as models for creating custom scripts to define your Secure+ Option environment and automate the implementation of it. To prevent any loss of data, you cannot run the scripts, but you can save them with a different name and modify them to suit your needs. The sample scripts are available in Appendix D, *Automation Scripts*. The scripts are designed to assist you as follows:

UNIX Script	Explanation
spcust_sample1.sh	An example of configuring Secure+ Option for the STS protocol with the Secure+ CLI. The example demonstrates the use of statically generated public keys to facilitate an automated mass implementation of Secure+ Option.
spcust_sample2.sh	An example of configuring Secure+ Option for the STS protocol with the Secure+ CLI. The example demonstrates the use of statically generated public keys to facilitate the automated addition of a remote node to a Connect:Direct network. A script based on this sample is used to configure the existing nodes in the network map. The remote node itself is configured with a script based on spcust_sample1. In order for the operation to be successful, the two scripts must use the same seeds to generate keys.
spcust_sample3.sh	An example of configuring Secure+ Option to use the SSL or TLS protocol with the Secure+ CLI. The example demonstrates the configuration of Secure+ Option with the trusted root file, key certificates, and ciphers.

The following list describes the tasks necessary to write and implement the script for a mass implementation of an initial installation that uses the STS protocol. Excerpts from the spcust_sample1 sample script are provided for reference.

1. Use the Create STSKeyPair command to generate a passphrase-protected file (default name: keypair) in the format that is already in use for importing and exporting public keys. In this key pair file, the private key is encrypted, but the public key is not. The public key string can be cut and pasted into spcust_samplex scripts for setting the remote keys for remote nodes.

2. Distribute the key pair file along with the SPCLI script when you distribute the installation to additional sites.
3. Use the **Update LocalNode** command to set the current key pair to the key pair stored in the key pair file. When using the key pair file to set the local keys in local and remote nodes, you need the key pair file name and its passphrase, as shown in the following example from the spcust_sample1 script:

```
StsAuthlocalkey=set
StsAuthKeyPairFile=keypairfile
StsAuthKeyPairFilePassphrase=secret
StsSiglocalkey=set
StsSigKeyPairFile=keypairfile
StsSigKeyPairFilePassphrase=secret
```

4. For each remote node, use the **Update RemoteNode** command to set the public key that was generated in step 1. Copy the public key string from the key pair file and paste it into the script on the lines where the remote keys of all the remote nodes are set, as shown in this example from the spcust_sample1 script:

```
update remotenode name=*
STSAuthRemoteKey=public key string
STSSigRemoteKey=public key string
```

5. Use the **Update LocalNode** command to generate a new key pair and save the existing key pair as the previous key pair value. For example, in spcust_sample1, after the authentication and signature keys were set to the ones stored in the key pair file generated in step 1, the **Update LocalNode** command was used again, as shown in the following example:

```
update localnode
stsauthlocalkey=gen
stsauthkeyseedType=dynamic
STSPrevAuthKeyExpDateTime=2005:01:01-20:13:44
stssiglocalkey=gen
stssigkeyseedType=dynamic
STSPrevSigKeyExpDateTime=2005:01:01-20:13:45
;
```

Note: The Secure+ CLI script is called spcli.sh. Each command within the script must be terminated with a semicolon (;). Use the **quit** command to exit the spcli.sh script.

Maintaining the Parameters File

The commands in the following table describe how to maintain the parameters file from the command line interface.

Command	Description	Parameter	Values
Init Parmfile	Creates the parameters file. Must be initialized before you can define nodes.	localnode=Name of the local node where the parameters file will be created.	local node name
		path=Location where the parameters file will be created.	directory location For example, *d_dir*/ndm/secure+/*node*
		passphrase=Arbitrary set of characters that encrypts the parameters file.	a string at least 32 characters long
Open parmfile	Opens a parameters file so that you can configure it.	path=Location where the parameters file will be created.	directory location For example, *d_dir*/ndm/secure+/*node*
Close parmfile	Closes the parameters file. After this command is issued, no more updates can be performed on the parameters file.	None	None
Validate parmfile	Validates the parameters file and ensures that it is a valid file.	None	None
Rekey parmfile	Recreates the parameters file if it becomes corrupted.	passphrase=Arbitrary set of characters that encrypts the parameters file.	passphrase, up to 32 characters long

Command	Description	Parameter	Values
Sync netmap	Imports remote node records defined in the Connect:Direct network map.	path=Location and name of the network map file.	location of network map file
		name=Name of the node in the network map. Use wildcard characters to resync more than one node at a time.	node name or wildcard Wildcard values are: Asterisk (*)—any number of characters. Example: kps.* syncs up all nodes with a name that starts with kps. Question mark (?)—a single character. Example: k?s.* syncs up kas.* and kbs.*.

Displaying Information

The following commands are available to display information:

Command	Description	Parameter
display info	Displays information about when the parameters file was last updated.	None
display all	Displays all nodes in the parameters file.	None
display localnode	Displays the values defined in the .Local node record.	None
display remotenode	Displays the values defined in remote node records.	node name or wildcard name—The name of the node to display information about. Use wildcard characters to display information about a group of remote node records. The options are: Asterisk (*)—Indicates any number of characters. For example, kps.* displays all nodes with a name that starts with kps. Question mark (?)—Indicates a single character. For example: k?s.* displays kas.* and kbs.*.
display client	Displays the values defined in the .Client node record.	None
display seaserver	Displays the values defined in the .SEAServer record.	None

Updating the .Local Node Record

The **update localnode** command configures the protocol for the .Local node record. The command has the following parameters:

Command	Parameter	Values
update localnode	protocol=Identifies the protocol to use in the .Local node record.	sts | tls | ssl | disable
	override=Identifies if values in the remote node can override values defined in the .Local node record.	y | n
	AuthTimeout=Specifies the maximum time, in seconds, that the system waits to receive the Connect:Direct control blocks exchanged during the Secure+ Option authentication process.	0–3600 seconds
	StsEnableSig=Enables digital signatures.	y | n
	StsEnableAutoUpdate=Enables automatic update of public keys during authentication.	y | n
	StsLimitExportVersion=Identifies if a trading partner is using an earlier version of Secure+ Option designated as an export version.	y | n
	StsEnableEnc=Enables data encryption during the copy operation.	y | n | algorithm algorithm—the actual text string, “algorithm,” which is not case-sensitive. If y is selected, data encryption is enabled and the algorithm used depends on the merged value between the PNODE and SNODE. If the remote node enables encryption, the local node cannot disable it. If this parameter is set to algorithm, the algorithm must be defined in the StsEnableEncAlg parameter.
	StsEnableEncAlg=Acceptable data encryption algorithms to use when the Copy file encryption is requested. List in order of preference, with most-preferred algorithm first.	DESCBC56 | TDESCBC112 | IDEACBC128

Command	Parameter	Values
	StsEncAlgs=Acceptable data encryption algorithms to use when encrypting control data during secure communications. List in order of preference, with the most-preferred algorithm listed first.	DESCBC56 | TDESCBC112 | IDEACBC128 | all | null all—Makes all valid algorithms available for use. null—Clears any existing values from the node definition. To enable more than one algorithm, use parenthesis to enclose all algorithms listed and a comma to separate each algorithm. For example, to enable two algorithms define the following parameter: StsEncAlgs=(DESCBC56, TDESCBC112)
	StsAuthLocalKey=Generates or clears the authentication keys for the node.	gen | clear | set gen—Generates the authentication key. If this parameter is defined as gen, the StsAuthKeySeed value must be defined. clear—Clears the authentication key value. If this command is defined as clear, the StsAuthKeysToClear must also be specified. set—Sets the value of the authentication key to the value that you define in the StsAuthKeySeed parameter.
	StsAuthKeyPairFile=Key pair file that contains a pair of public and private keys.	key pair file name
	StsAuthKeyPairFilePassphrase=The passphrase specified when the key pair file is created.	alphanumeric string | LCU:filename **Note**: For more information on using LCU files to mask plain text passwords in the Secure+ CLI, see Appendix E, *Using the LCU to Configure Encrypted Passwords*.
	StsAuthPreservePrev=Determines which authentication key values are saved.	y | n The default is **n**.

Command	Parameter	Values
	StsAuthKeySeedType=Identifies what type of key seed will be generated.	dynamic | static | value dynamic—The key seed is generated by the system each time. It is not necessary to define a value for the StsAuthKeySeed parameter. value—The key seed is generated based on the value defined in the StsAuthKeySeed parameter and changes each time the key is generated, even when the StsAuthKeySeed parameter is unchanged. static—the key seed is generated based on the value defined in the StsAuthKeySeed parameter. If the value of StsAuthKeySeed is not changed, the same Auth key is generated.
	StsAuthKeySeed=An alphanumeric string used to generate the authentication keys. This value is required when StsAuthKeySeedType is defined as static or value.	alphanumeric string at least 32 characters
	StsAuthKeysToClear=Identifies the authentication keys to clear.	previous | current | both
	StsPrevAuthKeyExpDateTime=Specifies the date and time that the previous authentication key expires.	yyyy:mm:dd-hh:mm:ss The default value is **30** days.
	StsSigLocalKey=Creates the signature keys for the node.	gen | clear | set gen—Generates the key. If this parameter is defined as gen, the StsSigKeySeed value must be defined. clear—Clears the key value. If this command is defined as clear, the StsSigKeysToClear must also be specified. set—Sets the value of the key to the value that you define in the StsSigKeySeed parameter.
	StsSigKeyPairFile=Key pair file that contains a pair of public and private keys.	key pair file name

Command	Parameter	Values
	StsSigKeyPairFilePassphrase=Passphrase specified when the key pair file is created.	alphanumeric string | LCU:filename **Note**: For more information on using LCU files to mask plain text passwords in the Secure+ CLI, see Appendix E, *Using the LCU to Configure Encrypted Passwords*.
	StsSigPreserveKey=Determines if previous signature key values are saved.	current | previous
	StsSigKeySeed=Alphanumeric string used to generate the signature keys. This value is required when StsSigKeySeedType is defined as static or value.	alphanumeric string at least 32 characters
	StsSigKeySeedType=Identifies what type of key seed will be generated.	dynamic | static | value dynamic—The key seed is generated by the system each time. It is not necessary to define a value for the StsSigKeySeed parameter. value—The key seed is generated based on the value defined in the StsSigKeySeed parameter and changes each time the key is generated, even when the StsSigKeySeed parameter is unchanged. static—The key seed is generated based on the value defined in the StsSigKeySeed parameter. If the value of StsSigKeySeed is not changed, the same Sig key is generated.
	StsSigKeysToClear=Identifies the signature keys to clear.	previous | current | both
	StsPrevSigKeyExpDateTime=The date and time that the previous signature key expires.	yyyy:mm:dd-hh:mm:ss The default value is **30** days
	SslTlsTrustedRootCertFile=The location of the trusted root certificate file to use for client connections.	location | null null—Clears any existing values from the node definition.
	SslTlsCertFile=Identifies the location of the certificate file to use	location | null null—Clears any existing values from the node definition.

Command	Parameter	Values
	SslTlsCertPassphrase=The passphrase specified when the certificate is created.	alphanumeric password assigned when the key was generated | LCU:filename **Note**: For more information on using LCU files to mask plain text passwords in the Secure+ CLI, see Appendix E, *Using the LCU to Configure Encrypted Passwords*.
	SslTlsEnableFIPSMode=Enables FIPS mode for the TLS protocol.	y | n
	SslTlsEnableClientAuth=Enables client authentication in a .Client node record.	y | n
	SslTlsEnableCipher=Specifies the cipher suites enabled **Note:** Only certain cipher suites are supported in FIPS-mode. For a list of the FIPS-approved cipher suites, see *Special Considerations* in the *Connect:Direct for UNIX Release Notes*.	name of a cipher suite | null | all all—Enables all ciphers. null—Clears any existing values from the node definition.
	SslTlsSeaEnable=Enables certificate validation by Sterling External Authentication Server.	y | n
	SeaCertValDef=Character string defined in Sterling External Authentication Server (SEAS).	character string | null null—Clears any existing values from the node definition.

Managing Remote Node Records

This section contains the commands and parameters used to create, update, display, and delete remote node records.

Creating a Remote Node Record

The **create remotenode** command creates a remote node record and configures the protocol settings. The command has the following parameters:

Command	Parameter	Values
create remotenode	model=Name of an existing node to use as a model to copy from.	name of a valid remote node
	Name=Identifies name of the remote node record.	name
	protocol=Specifies protocol to use in the node record.	sts | tls | ssl | disable | DefaultToLN
	override=Identifies if values in the copy statement can override values defined in the remote node record.	y | n | DefaultToLN
	AuthTimeout=Specifies the maximum time, in seconds, that the system waits to receive the Connect:Direct control blocks exchanged during the Secure+ Option authentication process.	0–3600 The default is **120** seconds.
	StsEnableSig=Enables digital signatures.	y | n | DefaultToLN
	StsEnableAutoUpdate=Enables automatic update of public keys during authentication.	y | n | DefaultToLN
	StsLimitExportVersion=Identifies if a trading partner is using an earlier version of Secure+ Option designated as an export version.	y | n The default is **y.**
	StsEnableEnc=Enables data encryption during the copy operation.	y | n | DefaultToLN | algorithm algorithm—the actual text string, “algorithm,” which is not case-sensitive. If y is selected, data encryption is enabled and the algorithm used depends on the merged value between the PNODE and SNODE. If the remote node enables encryption, the local node cannot disable it. If this parameter is set to algorithm, the algorithm must be defined in the StsEnableEncAlg parameter.
	StsEnableEncAlg=Specifies acceptable data encryption algorithms to use when the Copy file encryption is requested. List in order of preference, with most-preferred algorithm first.	DESCBC56 | TDESCBC112 | IDEACBC128

Command	Parameter	Values
	StsEncAlgs=Specifies acceptable data encryption algorithms to use when encrypting control data during secure communications. List in order of preference, with most-preferred algorithm first. ‘	DESCBC56 |TDESCBC112 | IDEACBC128 | All | null all—Makes all valid algorithms available for use. null—Clears any existing values from the node definition. To enable more than one algorithm, use parenthesis to enclose all algorithms listed and a comma to separate each algorithm. For example, to enable two algorithms define the following parameter: StsEncAlgs=(DESCBC56, TDESCBC112)
	StsAuthLocalKey=Generates, clears, or sets the authentication keys for the node.	gen | clear | set gen—Generates the authentication key. If this parameter is defined as gen, the StsAuthKeySeed value must be defined. clear—Clears the authentication key value. If this command is defined as clear, the StsAuthKeysToClear must also be specified. set—Sets the value of the authentication key to the value that you define in the StsAuthKeySeed parameter.
	StsAuthKeySeedType=Identifies what type of key seed will be generated.	dynamic | static | value dynamic—The system generates the key seed each time. It is not necessary to define a value for the StsAuthKeySeed parameter. value—The key seed is generated based on the value defined in the StsAuthKeySeed parameter and changes each time the key is generated, even when the StsAuthKeySeed parameter is unchanged. static—The key seed is generated based on the value defined in the StsAuthKeySeed parameter. If the value of StsAuthKeySeed is not changed, the same Auth key is generated.

Command	Parameter	Values
	StsAuthKeyPairFile=Key pair file that contains a pair of public and private keys for authentication.	key pair file name
	StsAuthKeyPairFilePassphrase= Passphrase specified when the key pair file is created.	alphanumeric string
	StsAuthKeySeed=An alphanumeric string used to generate the authentication keys. This value is required when StsAuthKeySeedType is defined as static or value.	alphanumeric string at least 32 characters long
	StsAuthRemoteKey=The authentication keys used to authenticate the remote node or clear to clear the value of the remote key.	key value | clear key value—The value assigned to the authentication key. clear—Clears the value of the authentication key. If clear is specified, the StsAuthKeysToClear must also be specified.
	StsAuthKeysToClear=Identifies the authentication keys to clear.	previous | current | both
	StsPrevAuthKeyExpDateTime= Identifies the date and time that the previous authentication key expires.	yyyy:mm:dd-hh:mm:ss
	StsSigLocalKey=Creates the signature keys for the node.	gen | clear | set gen—Generates the key. If this parameter is defined as gen, the StsSigKeySeed value must be defined. clear—Clears the key value. If this command is defined as clear, the StsSigKeysToClear must also be specified. set—Sets the value of the key to the value that you define in the StsSigKeySeed parameter.
	StsPrevSigKeyExpDateTime=The date and time that the previous signature key expires.	yyyy:mm:dd-hh:mm:ss

Command	Parameter	Values
	StsSigKeySeedType=Identifies what type of key seed will be generated.	dynamic | static | value dynamic—The key seed is generated by the system each time. It is not necessary to define a value for the StsSigKeySeed parameter. static—The key seed is generated based on the value defined in the StsSigKeySeed parameter and changes each time the key is generated, even when the StsSigKeySeed parameter is unchanged. value—The key seed is generated based on the value defined in the StsSigKeySeed parameter. If the value of StsSigKeySeed is not changed, the same Sig key is generated.
	StsSigKeySeed=An alphanumeric string used to generate the signature keys. This value is required when StsSigKeySeedType is defined as static or value.	alphanumeric string at least 32 characters long
	StsSigRemoteKey=The signature key used to authenticate the remote node or clear to clear the value of the remote key.	key value | clear key value—The value assigned to the key. clear—Clears the value of the key. If clear is specified, the StsSigKeysToClear must also be specified.
	StsSigKeyPairFile=Key pair file that contains a pair of public and private keys.	key pair file name
	StsSigKeyPairFilePassphrase= Passphrase specified when the key pair file is created.	alphanumeric string
	StsSigKeysToClear=Identifies the signature keys to clear.	previous | current | both
	SslTlsTrustedRootCertFile=Location of the trusted root certificate file to use for client connections.	location | null null—Clears any existing values from the node definition.
	SslTlsCertFile=Identifies the location of the certificate file to use	location | null null—Clears any existing values from the node definition.

Command	Parameter	Values
	SslTlsCertPassphrase=Passphrase specified when the certificate is created.	alphanumeric password assigned when the key was generated
	SslTlsEnableClientAuth=Enables client authentication in a .Client node record.	y | n | DefaultToLN DefaultToLN—Defaults to the setting specified in the .Local node record
	SslTlsEnableFIPSMode=Enables FIPS mode for the TLS protocol in a remote node record.	y | n | DefaultToLN DefaultToLN—Defaults to the setting specified in the .Local node record
	SslTlsSeaEnable=Enables certificate validation by Sterling External Authentication Server.	y | n | DefaultToLN DefaultToLN—Defaults to the setting specified in the .Local node record
	SslTlsCertCommonName=The certificate common name defined in the certificate	name | null null—Clears any existing values from the node definition.
	SslTlsEnableCipher=The cipher suites enabled **Note:** Only certain cipher suites are supported in FIPS-mode. For a list of the FIPS-approved cipher suites, see *Special Considerations* in the *Connect:Direct for UNIX Release Notes*.	name of a cipher suite | all | null all—Enable all ciphers. null—Clears any existing values from the node definition.

Updating the Remote Node Record

The **update remotenode** command creates a remote node record and configures the protocol settings. The command has the following parameters:

Command	Parameter	Values
update remotenode	Name=Specifies name for the remote node record.	remote node name | wildcard Use wildcard characters to update a group of remote node records. The options are: Asterisk (*)—Any number of characters. Example: kps.* displays remote nodes with a name that starts with kps. Question mark (?)—Single character. Example: k?s.* displays kas.* and kbs.*.

Command	Parameter	Values
	protocol=The protocol to use in the node record.	sts | tls | ssl disable| DefaultToLN
	override=Identifies if values in the copy statement can override values defined in the remote node record.	y | n | DefaultToLN
	AuthTimeout=Specifies the maximum time, in seconds, that the system waits to receive the Connect:Direct control blocks exchanged during the Secure+ Option authentication process.	0–3600 seconds
	StsEnableSig=Enables digital signatures.	y | n | DefaultToLN
	StsEnableAutoUpdate=Enables automatic update of public keys during authentication.	y | n | DefaultToLN
	StsLimitExportVersion=Identifies If a trading partner is using an earlier version of Secure+ Option designated as an export version.	y | n
	StsEnableEnc=Enables data encryption during the copy operation.	y | n | algorithm | DefaultToLN y—Enable data encryption. If y is selected, data encryption is enabled and the algorithm used depends on the merged value between the PNODE and SNODE. If the remote node enables encryption, the local node cannot disable it. n—Disable data encryption DefaultToLN—Use the values defined in the .Local node record. algorithm—The text string, “algorithm,” which is not case-sensitive. If this parameter is set to algorithm, the algorithm must be defined in the StsEnableEncAlg parameter.
	StsEnableEncAlg—Acceptable data encryption algorithms to use when the Copy file encryption is requested. List in order of preference, with most-preferred algorithm first.	DESCBC56 | TDESCBC112 | IDEACBC128

Command	Parameter	Values
	StsAuthLocalKey=Generates, clears, or sets the authentication keys for the node.	gen | clear | set gen—Generates the authentication key. If this parameter is defined as gen, the StsAuthKeySeed value must be defined. clear—Clears the authentication key value. If this command is defined as clear, the StsAuthKeysToClear must also be specified. set—Sets the value of the authentication key to the value that you define in the StsAuthKeySeed parameter.
	StsEncAlgs=Acceptable data encryption algorithms to use when encrypting control data during secure communications. List in order of preference, with most-preferred algorithm first.	DESCBC56 |TDESCBC112 | IDEACBC128 | All | null All—Sets all valid algorithms available for use. null—Clears any existing values from the node definition. To enable more than one algorithm, use parenthesis to enclose all algorithms listed and a comma to separate each algorithm. For example, to enable two algorithms define the following parameter: StsEncAlgs=(DESCBC56, TDESCBC112)
	StsAuthPreserveKey=Determines if the previous authentication key values are saved.	y |n | current | previous
	StsAuthKeysToClear=Identifies the authentication keys to clear.	current | previous | both
	StsAuthKeySeedType=Identifies what type of key seed will be generated.	dynamic | static | value dynamic—The system generates the key seed each time. It is not necessary to define a value for the StsAuthKeySeed parameter. value—The key seed is generated based on the value defined in the StsAuthKeySeed parameter and changes each time the key is generated, even when the StsAuthKeySeed parameter is unchanged. static—The key seed is generated based on the value defined in the StsAuthKeySeed parameter. If the value of StsAuthKeySeed is not changed, the same authentication key is generated.

Command	Parameter	Values
	StsAuthKeySeed=An alphanumeric string used to generate the authentication keys. This value is required when StsAuthKeySeedType is defined as static or value.	an alphanumeric string at least 32 characters long
	StsAuthRemoteKey=Authentication keys used to authenticate the remote node or clear to clear the value of the remote key.	key value | clear key value—Value assigned to the authentication key. clear—Clear the value of the authentication key. If clear is specified, the StsAuthKeysToClear must also be specified.
	StsPrevAuthKeyExpDateTime=Date and time that the previous authentication key expires.	yyyy:mm:dd-hh:mm:ss The default value is 30 days
	StsAuthKeyPairFile=Key pair file that contains a pair of public and private keys.	key pair file name
	StsAuthKeyPairFilePassphrase= Passphrase specified when the key pair file is created.	alphanumeric string
	StsSigLocalKey=Creates the signature keys for the node.	gen | clear | set gen—Generates the key. If this parameter is defined as gen, the StsSigKeySeed value must be defined. clear—Clears the key value. If this command is defined as clear, the StsSigKeysToClear must also be specified. set—Sets the value of the key to the value that you define in the StsSigKeySeed parameter.
	StsSigPreserveKey=Determines if previous signature key values are saved.	current | previous

Command	Parameter	Values
	StsSigKeySeedType=Identifies what type of key seed will be generated.	dynamic | static | value dynamic—The key seed is generated by the system each time. It is not necessary to define a value for the StsSigKeySeed parameter. value—The key seed is generated based on the value defined in the StsSigKeySeed parameter and changes each time the key is generated, even when the StsSigKeySeed parameter is unchanged. static—The key seed is generated based on the value defined in the StsSigKeySeed parameter. If the value of StsSigKeySeed is not changed, the same Signature key is generated.
	StsSigKeySeed=Alphanumeric string used to generate the signature keys. This value is required when StsSigKeySeedType is defined as static or value.	alphanumeric string at least 32 characters long
	StsPrevSigKeyExpDateTime=Date and time that the previous signature key expires.	yyyy:mm:dd-hh:mm:ss The default value is **30 days**
	StsSigRemoteKey=Signature key used to authenticate the remote node or clear to clear the value of the remote key.	key value | clear key value—The value assigned to the key. clear—Clears the value of the key. If clear is specified, the StsSigKeysToClear must also be specified.
	StsSigKeysToClear=Identifies the signature keys to clear.	previous | current | both
	StsSigKeyPairFile=Key pair file that contains a pair of public and private keys.	key pair file name
	StsSigKeyPairFilePassphrase= Passphrase specified when the key pair file is created.	alphanumeric string
	SslTlsTrustedRootCertFile=Location of the trusted root certificate file to use for client connections.	location | null null—Clears any existing values from the node definition.

Command	Parameter	Values
	SslTlsCertFile=Identifies the Location of the certificate file to use.	location | null null—Clears any existing values from the node definition.
	SslTlsCertPassphrase=Passphrase specified when the certificate is created.	alphanumeric password assigned when the key was generated.
	SslTlsEnableFIPSMode=Enables FIPS mode for the TLS protocol in a remote node record.	y | n | DefaultToLN DefaultToLN—Defaults to the setting specified in the .Local node record
	SslTlsEnableClientAuth=Enables client authentication in a .Client node record.	y | n | DefaultToLN DefaultToLN—Defaults to the setting specified in the .Local node record
	SslTlsSeaEnable=Enables certificate validation by Sterling External Authentication Server.	y | n | DefaultToLN DefaultToLN—Defaults to the setting specified in the .Local node record
	SslTlsCertCommonName=Certificate common name defined in the certificate	name | null null—Clears any existing values from the node definition.
	SslTlsEnableCipher=Cipher suites enabled **Note:** Only certain cipher suites are supported in FIPS-mode. For a list of the FIPS-approved cipher suites, see *Special Considerations* in the *Connect:Direct for UNIX Release Notes*.	name of a cipher suite | null | all all enables all ciphers. null—Clears any existing values from the node definition.

Displaying a Remote Node Record

The **display remotenode** command displays information about one or more remote node records. The command has the following parameter:

Command	Parameter	Values
display remotenode	name=Name of the remote node record to display information about.	node name | wildcard value To display information about more than one remote node record, use wildcard characters. Use wildcard characters to display information about a group of remote node records. The options are: Asterisk (*)—Any number of characters. Example: kps.* displays remote nodes with a name that starts with kps. Question mark (?)—A single character. Example: k?s.* displays kas.* and kbs.*.

Deleting a Remote Node Record

The **delete remotenode** command deletes one or more remote node records. The command has the following parameter:

Command	Parameter	Values
delete remotenode	name=Name of the remote node record to display information about. Use wildcard characters to delete a group of remote node records.	remote node name | wildcard value To display information about more than one remote node record, use wildcard characters. Use wildcard characters to display information about a group of remote node records. The options are: Asterisk (*)—Any number of characters. Example: kps.* displays remote nodes with a name that starts with kps. Question mark (?)—A single character. Example: k?s.* displays kas.* and kbs.*.

Updating the .Client Node Record

The **update client** command creates a .Client node record and configures the protocol settings. The command has the following parameters:

Command	Parameter	Values
update client	protocol=Protocol to use in the node record.	sts | tls | ssl disable | DefaultToLN
	override=Identifies if values in the copy statement can override values defined in the remote node record.	y | n | DefaultToLN
	AuthTimeout=Specifies the maximum time, in seconds, that the system waits to receive the Connect:Direct control blocks exchanged during the Secure+ Option authentication process.	0–3600 seconds
	SslTlsTrustedRootCertFile=Location of the trusted root certificate file to use for client connections.	location | null null—Clears any existing values from the node definition.
	SslTlsCertFile=Location of the certificate file to use.	location | null null—Clears any existing values from the node definition.
	SslTlsCertPassphrase=Passphrase specified when the certificate is created.	alphanumeric password assigned when the key was generated.

Command	Parameter	Values
	SslTlsEnableClientAuth=Enables client authentication in a .Client node record.	y | n | DefaultToLN
	SslTlsCertCommonName=Certificate common name defined in the certificate.	name | null null—Clears any existing values from the node definition.
	SslTlsEnableCipher=Cipher suites enabled.	name of a cipher suite | all | null all enables all ciphers. null—Clears any existing values from the node definition.

Maintaining the Sterling External Authentication Server Record

This section contains the commands and parameters used to update and display the .SEAServer record.

Updating the Sterling External Authentication Server Record

The **update seaserver** command configures properties for Sterling External Authentication Server (SEAS) in the .SEAServer record that is created at installation. The command has the following parameters:

Command	Parameter	Values
update seaserver	Protocol=Protocol to use in the .SEAServer record.	sts | tls | ssl disable | DefaultToLN
	SeaHost=External authentication host name defined in SEAS.	host name | null null—Clears any existing values from the node definition
	AuthTimeout=Specifies the maximum time, in seconds, that the system waits to receive the Connect:Direct control blocks exchanged during the Secure+ Option authentication process.	0–3600 seconds
	Override=Identifies if values in the copy statement can override values defined in the remote node record.	y | n | DefaultToLN
	SeaPort=External authentication server port number (listening) defined in SEAS.	port number | 61366

Displaying the Sterling External Authentication Record

The **display SEAServer** command displays information about the .SEAServer record. This command has no parameters.

Maintaining STS Keys

The commands and parameters in this section describe how to create, export, and import STS keys.

Creating an STS Key Pair

The **create STSKeypair** command creates a key pair of public and private keys. The only required parameter is passphrase. The command has the following parameters:

Command	Parameter	Values
Create STSKeypair	KeyPairFile= File in which to store the encrypted key pair.	valid file name. The default value is **keypair**.
	passphrase=Arbitrary set of characters.	a string | LCU:filename **Note**: For more information on using LCU files to mask plain text passwords in the Secure+ CLI, see Appendix E, *Using the LCU to Configure Encrypted Passwords*.
	KeySeedType=Type of key seed to generate.	dynamic | static dynamic—The key seed is generated by the system each time. static—The key seed is generated based on the value defined in the KeySeed parameter. If the value of KeySeed is not changed, the same key pair is generated.
	KeySeed=Alphanumeric string at least 32 characters long used to generate the key pair. This parameter is required when static is specified for KeySeedType.	key seed value, at least 32 characters long
	StsLimitExportVersion=Identifies If a trading partner is using an earlier version of Secure+ Optionn designated as an export version.	y | n The default is **y**.

Exporting STS Keys

The **export STSKeys** command allows you to export STS keys. The command has the following parameters.

Command	Parameter	Values
export STSKeys	dir=Directory in which to export the key file.	valid directory name
	name=Name of the remote node record to export keys for. Use wildcard characters to export information about a group of remote node records.	remote node name | wildcard Wildcard options are: Asterisk (*)—Any number of characters. For example, kps.* exports all nodes with a name that starts with kps. Question mark (?)—A single character. For example, k?s.* exports kas.* and kbs.*.

Importing STS Keys

The **import STSKeys** command allows you to import STS keys. The command has the following parameter:

Command	Parameter	Values
import STSKeys	path=The directory and name of the file that contains the public keys.	valid path value

Automatically Creating STS Keys for Remote Node Records

Before you can begin using Secure+ Option and the STS protocol for secure communications, you must exchange keys with the trading partner. To facilitate automatic configuration, you can now generate unique keys for a trading partner without the initial exchange of keys. For information on using scripts to automatically update STS keys, see *Sample Scripts* on page 53 and Appendix D, *Automation Scripts*.

Perform the following procedure to create unique keys for the STS protocol and eliminate the need to exchange keys manually with a trading partner:

1. In the .Local node record for both locations, configure the same previous authentication key and the same previous digital signature key.
2. In the .Local node records, configure unique keys in the current authentication and current digital signature keys.

3. Configure every remote node record with the following information:
   a. The same current authentication public key
   b. The same current signature public key
4. Turn on auto update of public keys.

   Turning on auto update causes the public keys in the remote node records to be automatically updated to the value in the current authentication field and the current signature public key field from the remote node.

Completing the procedure results in the following activity:

- When the first connection between the local and remote node occurs, STS authentication is performed based on the values defined in the previous authentication key.
- If authentication is successful, enabling auto update causes each node to automatically import the public authentication key and public signature key from the remote node record from control information exchanged on the communications link. It also updates the remote node record in the local parameters file.
- For subsequent connections, the newly imported keys will be used for authentication.

Chapter 9

Maintaining Secure+ Option

After you set up the Secure+ Option environment, you perform additional maintenance tasks as needed:

- Displaying the Secure+ Option Node Information
- Viewing Information about the Secure+ Option Parameters File
- Modifying a Secure+ Option Configuration
- Modifying Secure+ Option Keys

Displaying the Secure+ Option Node Information

After you set up node records in Secure+ Option, you can view all of the nodes and their attributes from the **Secure+ Admin Tool Main Window**. To display a Secure+ Option node record, open it by double-clicking the node record name.

Node List Field Descriptions

Below is a description of all the fields displayed in the Node Name List:

Field Name	Field Definition	Values
Node Name	Displays the node record name.	.Local | remote node name | .client
Type	Displays the current record type.	L | R L—Local record R—Remote record
Secure+	Displays the status of Secure+ Option.	N | TLS | SSL | STS | * N—Disabled TLS—TLS protocol SSL—SSL protocol STS—STS protocol *—Default to local node

Field Name	Field Definition	Values
Override	Displays the status of override. Enable override in the local node to allow remote node records to override the settings in the local node record. If the STS protocol is enabled, enabling override in a remote node allows the values in the COPY statement to override key values in the remote node record.	Y | N | * Y—Enabled N—Disabled *—Default to local node
CipherSuites	Displays the TLS or SSL cipher suites that are enabled for the node record.	Varies, based on the cipher suites enabled.
ClientAuth	Displays the status of client authentication. If the TLS or SSL protocol is used, enabling client authentication means the SNODE verifies the identity of the PNODE.	Y | N | * Y—enabled N—Disabled *—Default to local node
Encryption	Indicates if encryption is enabled in the STS protocol.	Y | N Y—Enabled N—Disabled
Signature	Identifies if digital signature is enabled in the STS protocol.	Y | N | * Y—Enabled N—Disabled *—Default to local node
LimExpr	Identifies if the Limited Export version is being used by a remote node.	Y | N | * Y—Enabled N—Disabled *—Default to local node
AutoUpdate	Indicates if the option to automatically update key values during communication is enabled.	Y | N | * Y—enabled N—disable *—default to local node
Base Record	Displays the name of the base record for the alias records.	There are no parameter values.

Viewing Secure+ Option Node Record Change History

Perform the following steps to view the history of changes to a Secure+ Option node record.

1. From the **Secure+ Option Admin Tool Main Window**, double-click the node record name.
2. Click the **Security Options** tab.

   The history of changes is displayed in the **Update History** field.

Viewing Information about the Secure+ Option Parameters File

Perform the following steps to view information about the Secure+ Option parameters file:

1. Open the Secure+ Admin Tool.
2. On the **File** menu option of the **Secure+ Admin Tool Main Window,** click **Info**. The **File Information** dialog box is displayed.

   Refer to the following table for an explanation of the fields.

Field Name	Field Definition
Current File	Displays the name of the parameters file opened.
Number of Records	Lists the number of nodes defined in the parameters file.
Number of Updates	Displays how many times the parameters file has been updated.
Last 3 Updates	Displays the name of the last three nodes updated.

3. Click **OK** to close the **File Information** dialog box.

Modifying a Secure+ Option Configuration

After using Secure+ Option, it may be necessary to modify a configuration. This section provides the following procedures for modifying Secure+ Option information:

- ✦ Disabling Secure+ Option
- ✦ Deleting a Secure+ Option remote node record
- ✦ Resecuring the Secure+ Option parameters file and access file
- ✦ Changing the cipher suites
- ✦ Changing the encryption algorithms suites for the STS protocol

Disabling Secure+ Option

You can use this procedure to disable all nodes in a configuration or one remote node. Perform the following steps to disable Secure+ Option:

1. Do one of the following:
   - To disable all nodes in a configuration, open the local node record.
   - To disable one node, open the remote node record for that node.
2. Click the **Security Options** tab.

3. Click the **Disable Secure+**.
4. Click **OK** to update the node record.

Note: In order to continue Connect:Direct operations with Secure+ Option disabled, both trading partners must disable Secure+ Option.

Deleting a Secure+ Option Remote Node Record

If a remote node record is no longer defined in the network map, you can remove it from the parameters file. The following procedure deletes nodes that are defined in the Secure+ Option parameters file but not in the selected network map:

1. From the **Secure+ Admin Tool Main Window**, click the **Sync with Netmap** of the **File** menu.
2. Click the network map to use from the pulldown list.
3. Click **OK**.
4. Click **Skip** to move through the Select Netmap Entries to the **Add** dialog box.
5. Do one of the following to delete node records:
   - To delete selected node records, highlight the remote nodes to delete and click **Delete Selection**.
   - To delete all remote node records that are not found in the network map, click **Delete All**.

Caution: Do *not* delete the remote node record that is named for the Connect:Direct node. It is the base record for the .Local node record. You cannot delete the .Local node record.

Resecuring the Secure+ Option Parameters File and Access File

Routinely, or if your access file is compromised, perform the following steps to resecure Secure+ Option:

1. From the **Secure+ Admin Tool Main Window**, click **Rekey Secure+** from the **File** menu. The **Rekey Secure+** dialog box is displayed.
2. Type an alphanumeric string at least 32 characters long in the **Passphrase** field. Secure+ Option uses the passphrase to re-encrypt the Secure+ Option parameters and access files. You do not have to remember this passphrase value.
3. Click **OK** to accept the new passphrase. The Secure+ Option decrypts and re-encrypts the parameters file and access file.

WARNING: Do not type a new passphrase if an error occurs.

If an error occurs while you are resecuring the files, restore the node records from the ACFSave directory. This directory is created after the **Rekey Secure+** feature is executed.

Changing Cipher Suites

When you activate the TLS or SSL protocol for a node, cipher suites are used to encrypt the data being transmitted. The same cipher suite must be used at both ends of the transmission. Secure+ Option searches the enabled cipher suite list at both nodes and locates the first cipher suite that is common for communications at both the SNODE and the PNODE.

You enable the cipher suites in the preferred order, or enter them randomly and order them using the following procedure:

1. If necessary, open the Secure+ Option Admin Tool.
2. Double-click the node that you want to edit.
3. Click the **TLS/SSL Protocol** tab.
4. If you are configuring a remote node, turn off **Default to Local Node** in the Cipher Suites section.
5. Modify cipher suites as follows:
   - To enable a cipher suite, highlight the item in the **Available** list and click **Add**.
   - To remove a cipher suite, highlight the item in the **Enabled** list and click **Remove**.
   - To change the priority of a cipher suite, highlight the item and click **Up** or **Down** until the item is in the correct priority position.
6. Click **OK**.

Changing the STS Protocol Encryption Algorithms

When you activate the STS protocol for a node, it uses algorithms to encrypt the data being transmitted. The same algorithms must be used at both ends of the transmission.

Secure+ Option searches the enabled algorithm list and locates the first algorithm that is common for communications at both ends. You can enable the algorithms in the preferred order, or enter them randomly and order them using the following procedure:

1. If necessary, start the Secure+ Admin Tool.
2. Double-click the node that you want to edit.
3. Click the **STS Options** tab.
4. Turn off **Default to Local Node** in the Encryption Algorithms section.
5. Modify algorithms as follows:
   - To enable an algorithm, highlight the item in the **Available** list and click **Add**.
   - To remove an algorithm, highlight the item in the **Enabled** list and click **Remove**.
   - To change the priority of an algorithm, highlight the item and click **Up** or **Down** until the item is in the correct priority position.
6. Click **OK**.

Modifying Secure+ Option Keys

After using Secure+ Option, it may be necessary to update keys and clear keys when using the STS protocol. This section provides the following procedures for modifying key files in Secure+ Option:

- Updating keys in STS-configured node records
- Clearing keys in STS-configured node records

Updating Keys in STS-Configured Node Records

To maintain communications with a trading partner when you update your keys, you must maintain a copy of the previous keys until your trading partner receives the updated keys. Perform the following steps to update signature and authentication keys:

1. Open the Secure+ Admin Tool.
2. Double-click the node record to update.
3. Click the **STS Protocol** tab.
4. Click the tab for the type of new keys you want to generate: **Authentication Keys** or **Signature Keys**.
5. Click **Generate Key** to create new public keys. A message asks if you want to save the current key pair as previous.
6. Click **Yes** to save the current key pair as the previous. The **Generate Random Number Seed** dialog box is displayed.
7. Do one of the following to generate a key:
   - Type an alphanumeric string at least 32 characters long in the **Random Number Seed** field and click **OK**. Secure+ Option uses the random number seed value to generate the authentication keys.
   - Click **Sample Value**, highlight all of the grid squares, and click **OK** to generate the random number seed.

   The **Public Keys** window displays the populated **Public Key** field.
8. If necessary, type the expiration date in the **Previous Key Pair Expiration Date** field.
9. Click **OK** to save the changes to the parameters file.

   If you and your trading partner both enable the Public Key Auto Updates feature in the parameters file, Secure+ Option uses the existing keys to establish a session and then creates new keys during authentication. The parameters files for you and your trading partner are automatically updated with the new key values.

Clearing Keys in STS-Configured Node Records

Perform the following steps to clear the keys in node records.

1. Open the Secure+ Admin Tool.
2. Double-click the node record to update with key information.
3. Click the **STS Protocol** tab.

4. Click the tab for the type of new keys you want to clear: **Authentication Keys** or **Signature Keys**.
5. Click **Clear Key** to clear the keys.
6. Click the appropriate button for the type of keys to clear: **Current**, **Previous**, **Both**, or **Cancel**.

   The keys are cleared.

Chapter 10

Accessing Secure+ Option Statistics

Connect:Direct logs statistics for Connect:Direct Process activity. Connect:Direct statistics include Secure+ Option information for a Process.

Secure+ Option Statistics Record Information

Fields are included in Connect:Direct Process statistics records to provide Secure+ Option information about the Process. Secure+ Option information is included in the Process statistics information only when you attach to a Secure+ Option server. For information on viewing statistics, refer to the *Connect:Direct Browser User Interface User's Guide*. When you use the **select statistics** function to view information about a Connect:Direct Process, statistics information about a particular Process is displayed. If Secure+ Option is enabled, Secure+ Option fields are also displayed.

The Secure+ Option fields and values available using the **select statistics** function are shown in the following table:

Field Name	Field Description	Values
Secure+ Enabled	Specifies whether Secure+ Option is enabled.	Y | N
Secure+ Protocol	Specifies which protocol is enabled.	SSL 3.0 | STS 1.0 | TLS 1.0
Cipher Suite	Displays the cipher suite used during a session.	cipher suite name For example: SSL_RSA_EXPORT_WITH_RC4_40_MD5
PNode Cipher List	Specifies the encryption algorithms available for the PNODE during the session.	IDEACBC128 | TDESCBC112 | DESCBC56

Field Name	Field Description	Values
PNode Cipher	Specifies the preferred data encryption as specified in the Secure+ Option parameters file of the PNODE.	Y | N | algorithm name
SNode Cipher List	Specifies the encryption algorithms available for the SNODE during the session as specified in the Secure+ Option parameters file of the SNODE.	IDEACBC128 | TDESCBC112 | DESCBC56
SNode Cipher	Specifies the preferred data encryption algorithm as defined in the Secure+ Option parameters file of the SNODE.	Y | N | algorithm name
Control Block Cipher	Specifies the algorithm used for encrypting control blocks. This value is determined during authentication when the PNODE and SNODE are merged.	IDEACBC128 | TDESCBC112 | DESCBC56
Copy Data Cipher	Specifies the encryption method used for encrypting data. The value is determined after the values in the SNODE and the PNODE are merged.	IDEACBC128 | TDESCBC112 | DESCBC56
PNODE Signature Enabled	Indicates whether digital signatures are enabled for the PNODE. This value is obtained from the Secure+ Option parameters file settings. If the COPY statement overrides the Secure+ Option parameters file, the value from the COPY statement is used.	Y | N
SNODE Signature Enabled	Indicates whether digital signatures are enabled for the SNODE. This value is obtained from the Secure+ Option parameters file settings.	Y | N

Field Name	Field Description	Values
Signature Enabled	Identifies the digital signature value used for a copy operation. In the Session Start record, this value is the result of the merged value between the PNODE and SNODE. In the Copy Termination record, if the COPY statement overrides the parameters file value, the merged value depends on the value supplied in the COPY statement. (The unprocessed value from the COPY statement is recorded in the Signature Enabled field of the PNODE).	Y | N
Current Signature Verified	Indicates whether the current digital signature was verified.	Y | N
Previous Signature Verified	Indicates whether the previous digital signature was verified.	Y | N

Connect:Direct CLI Select Statistics Detail

When you use the CLI **select statistics** function to view the information about a Connect:Direct Process, you see statistics information about a particular Process. Connect:Direct Secure+ Option fields are shown in bold in the following samples. The Connect:Direct field names, descriptions, and valid values are shown in *Secure+ Option Statistics Record Information* on page 85. For more information on Connect:Direct certificate auditing, see Chapter 11, *Secure+ Option Audits*.

Session Start (SSTR) Record

The following sample Session Start Record (SSTR) displays the output of an SSL session:

```
Record Id         => SSTR
Process Name      =>                    Stat Log Time  => 15:23:21
Process Number    => 0                  Stat Log Date  => 10/16/2004
Submitter Id      =>

Start Time        => 15:23:20           Start Date     => 10/16/2004
Stop Time         => 15:23:21           Stop Date      => 10/16/2004

SNODE             => JKTIB8100
Completion Code   => 0
Message Id        => LSMI004I
Message Text      => PNODE session started - remote node &NODE
Secure+ Protocol  => SSL 3.0
SSL Cipher Suites => ssl_RSA_WITH_RC4_128_MD5
------------------------------------------------------------------------
```

Copy Termination (CTRC) Record

The Copy Termination Record (CTRC) sample below uses the SSL protocol:

```
Record Id         => CTRC
Process Name      => XX              Stat Log Time  => 15:26:32
Process Number    => 195             Stat Log Date  => 10/16/2004
Submitter Id      => user1
Start Time        => 15:23:47        Start Date     => 10/16/2004
Stop Time         => 15:26:32        Stop Date      => 10/16/2004
SNODE             => DLAS8100
Completion Code   => 0
Message Id        => SCPA000I
Message Text      => Copy operation successful.
COPY DETAILS: Ckpt=> Y Lkfl=> N Rstr=> N XLat=> N Scmp=> N Ecmp=> N
From node         => S
Src File          => D:\long path
Dest File         => D:\long path
Src CCode         => 0               Dest CCode       => 0
Src Msgid         => SCPA000I        Dest Msgid       => SCPA000I
Bytes Read        => 23592960        Bytes Written    => 23592960
Records Read      => 1024            Records Written  => 1024
Bytes Sent        => 23791420        Bytes Received   => 23791420
RUs Sent          => 30721           RUs Received     => 30721
Secure+ Protocol =>SSL 3.0
SSL Cipher Suites =>SSL_RSA_WITH_RC4_128_MD5
------------------------------------------------------------------------
```

Connect:Direct CLI Select Process Detail

When you use the CLI **select process** command to view information about a Connect:Direct Process, you see statistics about a Process. The following sample Select Process Detail uses the STS protocol. If Secure+ Option is not enabled, no Secure+ information is displayed:

```
Process Name      => XX              Class          => 32
Process Number    => 197             Priority       => 10
Submitter Node    => DALLAS          PNode          => DALLAS
Submitter         => user1           SNode          => DALLAS
Retain Process    =>N

Submit Time       => 15:55:55        ScheduleTime   =>
Submit Date       => 10/19/2001      ScheduleDate   =>

Queue             => EXEC
Process Status    => EX
Message Text      =>
Function          => COPY
Step Name         => TWO
Type              => Send
File Bytes        => 3202560         File Recs      => 0
Xmit Bytes        => 3247926         Xmit Buffers   => 0
Secure+ Protocol  => STS 1.0
Ctrl Block Cipher => TDESCB112
Copy Data Cipher  => IDEACBC128
Signature Enabled => Y
------------------------------------------------------------------------
```

Chapter 11

Secure+ Option Audits

Secure+ Option provides auditing of parameters files and certificates for archival purposes.

Secure+ Option Parameters File Auditing

The Secure+ Option Administration Tool (Secure+ Admin Tool) and the Secure+ Option Command Line Interface (Secure+ CLI) log changes made to the Secure+ Option parameters file. The following events are logged:

- Application Startup
- Init Parmfile
- Open Parmfile
- Sync Netmap
- Rekey Parmfile
- Create Node
- Update Node
- Delete Node

The parameters file logging feature has the following operational characteristics:

- The logging feature is always enabled and cannot be disabled.
- If errors occur when the log is being updated, the application terminates.
- Each log entry contains a timestamp, user ID, and a description of the action/event.
- When an existing node is updated, any changed fields are reported.
- When a node is created or deleted, the values of all non-empty fields are reported.
- Any commands that modify a node are logged.

Note: The certificates used by Secure+ Option are individual files that can be stored anywhere on the system. As a result, the logging feature cannot detect when existing certificate files are modified. Secure+ Option only stores the certificate path name and detects changes to this field only.

Accessing Parameters File Audit Logs

The parameters file audit logs are stored in a dedicated directory, ...\secure+\log. The log file naming convention is SP[YYYY][MM][DD].001 (using local time), and the contents of a log file are limited to a single calendar date. You can view these log files using any text editor. Log files are not deleted by Secure+ Option.

Parameters File Audit Log Entries

Each audit log has the following header:

```
[YYYYMMDD][HH:MM:SS:mmm][userid]
```

When a parameter file is created or opened, an ID is generated that associates the change with the node being updated as shown in the following:

```
[YYYYMMDD][HH:MM:SS:mmm][userid][ParmFileID]
```

The following fields may appear in a **create**, **update**, or **delete** audit record.

Field Name	Description
Name	Name of the node
BaseRecord	Name of the base record
Type	Record type of local, remote, or alias
Protocol	Enables Secure+ Option protocol
Override	Enables overriding the current node
AuthTimeOut	Authentication timeout
StsEnableSig	Enable STS signatures
StsEnableAutoUpdateh	Enable STS auto update remote public keys
StsEnableEnc	Enable STS copy encryption stream
StsEncAlgs	Enable STS control block encryption streams
StsAuthLocalKey	STS authentication local public key
StsAuthRemoteKey	STS authentication remote public key
StsPrevAuthKey	STS authentication previous private key (masked)
StsPrevAuthKeyExpDateTime	STS authentication previous private key expiration date
StsSigLocalKey	STS signature local public key
StsSigRemoteKey	STS signature remote public key

Field Name	Description
StsPrevSigKey	STS signature previous private key (masked)
StsPrevSigKeyExpDateTime	STS signature previous private key expiration date
SslTlsTrustedRootCertFile	Pathname to trusted roots file
SslTlsCertFile	Pathname to key certificate file
SslTlsCertPassphrase	Key certificate passphrase (masked)
SslTlsEnableClientAuth	Enable client authentication
SslTlsCertCommonName	Common name of the remote certificate to verify
SslTlsEnableCipher	List of SSL/TLS cipher suites
SslTlsSeaEnable	Enable external authentication
SeaCertValDef	External authentication validation definition
SeaHost	External authentication host name
SeaPort	External authentication port number

Parameters File Audit Log Error Reporting

Errors are reported for the following logging functions: **open log**, **write log**, and **lock log**. If an error occurs during one of these functions, an error message is displayed, and the application is terminated. The lock function times out after 30 seconds. Typically, Secure+ Admin Tool or the Secure+ CLI hold the lock for less than one second per update.

Secure+ Option Certificate Auditing

In an SSL/TLS session, audit information about the identity certificate and its signing certificate is logged in the statistics log in the Session Start (SSTR) and Copy Termination (CTRC) records. The audit information is included in the response data from a **select statistics** command in the SSTR and CTRC records. In an SSL/TLS session, the PNODE (client) always logs the audit information. The SNODE (server) only logs the information when client authentication is enabled. For logging to occur, the session handshake must succeed and progress to the point of logging the SSTR and CTRC records.

Certificate Audit Log Entries

The audit consists of the subject name and serial number of the identity and its signing certificate. The identity certificate also contains an issuer attribute, which is identical to the signing certificate subject name. Although many signing certificates may exist between the identity and final root certificate, the audit includes only the last two certificates in a chain: an intermediate certificate and an end certificate.

In the SSTR and CTRC records, the CERT contains the common name and serial number of the key certificate, and the CERI contains the common name of the issuer and the serial number of an intermediate or root CA. They may also contain the certificate serial number, for example:

```
CERT=(/C=US/ST=MA/L=Marshfield/O=test.org/OU=Dev/CN=Test ID/SN=99c0ce01382e6c83)|
CERI=(/C=US/ST=MA/L=Marshfield/O=test.org/CN=root CA/SN=da870666bbfb5538)
```

Secure+ Option certificate audits may contain the following fields:

Field Name	Abbreviation	Max Lengths (RFC 2459)
Common Name	CN	64
Country	C	2
Locality	L	128
State	ST	128
Organization	O	64
Organization Unit	OU	64
Email Address	emailAddress	128
Serial Number	SN	128 (estimated)

Accessing Certificate Audit Logs

Certificate audit information located in the SSTR and CTRC records cannot be accessed directly using Connect:Direct Requester or Connect:Direct Browser User Interface. To access certificate information, you can issue a query directly to the database or use an SDK-based or JAI-based program to issue a **Select Statistics** command. The response to the **Select Statistics** command contains the **AuditInfo** field of the statistics records, including the SSTR and CTRC records. This field contains certificate audit information.

The following example was generated using a database query. The certificate audit information is highlighted in bold.

```
'2007-05-21 14:50:27', 2, 'SSTR', 'CAEV', '', 0, '2007-05-21 14:50:26', '2007-05-21
14:50:27', '', '', 'JLYON-XP.4400', 0,
'MSGI=LSMI004I|SBST=(&NODE=JLYON-XP.4400)|PNOD=JLYON-XP.4400|CSPE=Y|CSPP=TLSv1|CSPS=
TLS_RSA_WITH_AES_256_CBC_SHA|
CERT=(/C=US/ST=MA/L=Marshfield/O=test.org/OU=Dev/
CN=Example Test ID/SN=a9febbeb4f59d446)|
CERI=(/C=US/ST=MA/L=Marshfield/O=test.org/OU=Dev/CN=Example
IntermediateCA/SN=a69634a8a7830268)|STSD=2|TZDI=-14400|'

'2007-05-21 14:50:28', 2, 'CTRC', 'CAPR', 'SAMPLE', 1, '2007-05-21 14:50:27',
'2007-05-21 14:50:28', 'JLYON-XP.4400', 'jlyon', 'JLYON-XP.4400', 0,
'MSGI=SCPA000I|LCCD=0|LMSG=SCPA000I|OCCD=0|OMSG=SCPA000I|PNAM=SAMPLE|PNUM=1|SNAM=STE
P1|SBND=JLYON-XP.4400|SBID=jlyon|PNOD=JLYON-XP.4400|SNOD=JLYON-XP.4400|LNOD=P|FROM=P
|XLAT=N|ECZI=N|ECMP=N|SCMP=N|OERR=N|CKPT=Y|LKFL=N|RSTR=N|RUSZ=65535|PACC=|SACC=|PPMN
=|SFIL=C:\Program Files\Sterling Commerce\Connect Direct
v4.4.00\Server\Process\Sample.html|SDS1= |SDS2= |SDS3=
|SFSZ=0|SBYR=861|SRCR=1|SBYX=863|SRUX=1|SNVL=-1|SVOL=|DFIL=C:\Program Files\Sterling
Commerce\Connect Direct v4.4.00\Server\Process\Verify.html|PPMN=|DDS1=R|DDS2= |DDS3=
|DBYW=861|DRCW=1|DBYX=863|DRUX=1|DNVL=0|DVOL=|CSPE=Y|CSPP=TLSv1|CSPS=TLS_RSA_WITH_AE
S_256_CBC_SHA|CERT=(/C=US/ST=MA/L=Marshfield/O=test.org/OU=Dev/CN=Example Test
ID/SN=a9febbeb4f59d446)|CERI=(/C=US/ST=MA/L=Marshfield/O=test.org/OU=Dev/
CN=Example Intermediate CA/SN=a69634a8a7830268)
|PCRC=N|ETMC=60|ETMK=10|ETMU=0|STSD=2|TZDI=-14400|'
```

Certificate Audit Log Error Reporting

If an error occurs when the subject name is extracted from the identity (CERT) or issuer's (CERI) certificates, the following message ID is logged:

```
CERT=(MSGI=CSPA310E)|CERI=(MSGI=CSPA310E)
```

Only the message ID is displayed with the CERT or CERI tokens; the standard Connect:Direct error function is not used. After the error occurs, the session continues.

Chapter 12‘

Troubleshooting

Use the following table to help troubleshoot problems with Secure+ Option:

Problem	Possible Cause	Solution
Secure+ Option features are enabled in the parameters file, but the statistics record indicates that these functions are disabled.	The Connect:Direct network maps do not contain entries for the PNODE and SNODE.	Verify that the network map entries for both the PNODE and the SNODE exist.
Running a Process with a remote node fails with an authentication error.	Unique public/private key pairs are generated for the remote node record and the .Local node record is set to Enable Override=N.	Change the .Local node record to Enable Override=Y.
The Secure+ Option parameter, ENCRYPT.DATA, specified from the COPY statement causes the copy step to fail with error message CSPA080E.	The algorithm name used in the COPY statement is not in the supported algorithm list for both nodes.	Verify that the algorithm name in the COPY statement is in the supported algorithm list for both nodes.
Secure+ Option is installed, but error message CSPA001E occurs on non-Secure+ Option transfers.	Remote node records do not exist.	◆ A remote node record must exist for every node in the netmap. Use the Sync with Netmap feature to create any missing nodes. ◆ Disable Secure+ Option by clicking Disable Secure+ in the .Local node record.
Signature verification fails with error message CSPA002E.	Configuration settings missing or incorrect.	◆ If this is a non-Secure node, make sure the remote node record has Disable Secure+ selected. ◆ Check the Secure+ Option settings for the node.

Problem	Possible Cause	Solution
Signature verification fails with error message CSPA003E, CSPA004E, or CSPA005E.	◆ Configuration settings missing or incorrect. ◆ A security attack in progress.	◆ Correct the STS settings for the remote node. ◆ Execute standard operating procedure for investigating security violation.
Signature verification fails with error message CSPA007E.	Expired Signature Previous Key Pair. Date exceeded or keys have been changed.	If Auto Update is disabled, check the expiration date for the signature key pair for both nodes. Check the update history log on both nodes for the last change to the record. Verify that the signature public key is correct for both nodes.
Running a Process with a remote node fails with an authentication error, CSPA008E.	Authentication Previous Key Pair Expiration Date exceeded or keys have been changed.	If Auto Update is disabled, check the authentication previous key pair expiration date for both nodes. Check the update history log on both nodes for the last change to the record. Verify the authentication public key is correct for both nodes.
Strong authentication fails with the error, CSPA010E.	◆ The time allowed for strong authentication expired. ◆ A security attack in progress.	◆ Increase the timeout value. ◆ Execute standard operating procedure for investigating security violation.
Secure+ Option session fails with the error, CSPA011E.	An illegal attempt to override Secure+ Option parameters.	◆ Turn on Enable Override in the remote node record to allow the COPY statement to override the node settings. ◆ Check the COPY statement and remove the override statements.
Secure+ Option session fails with the error, CSPA014E.	Secure+ Option cannot read the remote node definition.	Check the remote node definition settings.
Secure+ Option session fails with the error, CSPA016E.	Secure+ Option is not enabled in the local node definition.	Make sure Secure+ Option is enabled for the local node.
Secure+ Option session fails with the error, CSPA019E.	Error generating digital signature.	◆ Resubmit the Process. ◆ Call Sterling Commerce Customer Support.
Secure+ Option session fails with the error, CSPA077E.	The COPY statement requested Secure+ Option parameters but Secure+ Option is not configured.	Remove the SECURE= parameter from the COPY statement.

Problem	Possible Cause	Solution
Secure+ Option session fails with the error, CSPA079E.	Invalid encryption algorithm identified in COPY statement.	Change the ENC.DATA parameter and specify one of the following values: Y, N, IDEACBC128, TDESCBC112, or DESCBC56 and resubmit the Process.
Secure+ Option session fails with the error, CSPA080E.	No common algorithms are available for both nodes.	Verify the algorithm list for both nodes contains at least one common algorithm name.
Secure+ Option session fails with the error, CSPA091E.	Session attempted but remote node is not configured.	Make sure both nodes are defined and Enable STS Protocol is selected for the remote node record.
Secure+ Option session fails with the error, CSPA200E.	Both nodes are not configured for the same protocol.	Check the protocol setting at both sites and verify that the same protocol is configured at each site. If necessary, edit the remote node record and activate the STS protocol.
Secure+ Option session fails with the error, CSPA202E.	SSL or TLS protocol handshake failed.	Edit the cipher suite list and add a cipher suite used by the trading partner.
Secure+ Option session fails with the error, CSPA203E or CSPA204E.	The SSL or TLS protocol could not validate the server's certificate.	Make sure the certificate information is typed into the node record.
Secure+ Option session fails with the error, CSPA205E.	A trading partner is not using TCP/IP for communication.	Make sure that both ends of the communication use TCP/IP.
Secure+ Option session fails with the error, CSPA206E.	The SSL or TLS protocol could not validate the server's certificate.	Make sure the certificate information is entered into the node record.
Secure+ Option session fails with the error, CSPA208E.	The common name in the certificate received does not match the Secure+ Option configuration.	Make sure the certificate common name is spelled correctly and uses the same case as that in the certificate.
Secure+ Option session fails with the error, CSPA209E.	The certificate has expired or is invalid.	Obtain a new certificate and reconfigure the node record.
Secure+ Option session fails with the error, CSPA210E.	The COPY statement attempts to override settings in the SSL or TLS protocol.	◆ The system continues to operate. ◆ If desired, change the Process statement and remove the COPY override options.
Secure+ Option session fails with the error, CSPA211E.	The remote trading partner failed to send a certificate.	Notify the trading partner that a certificate is required.
Secure+ Option session fails with the error, CSPA280E.	The trusted root certificate could not be loaded.	Check the local node configuration and make sure the location of the trusted root certificate is correctly identified.

Problem	Possible Cause	Solution
Secure+ Option session fails with the error, CSPA281E.	The trusted root certificate is empty.	Check the local node configuration and make sure the location of the trusted root certificate is correctly identified.
Secure+ Option session fails with the error, CSPA282E.	The user certificate file cannot be loaded.	Check the local node configuration and make sure the location of the user certificate file is correctly identified.
Secure+ Option session fails with the error, CSPA303E.	The parameters files have not been initialized.	Run the Admin Tool to initialize the parameters files.
Secure+ Option session fails with the error, CSPA309E.	The SSL library failed during the handshake.	Examine all related errors to determine the cause of the failure.
Secure+ Option session fails with the error, CSPA311E.	Certificate validation failed.	Verify that the root certificate is properly configured. An alternate certificate may be required.

Appendix A

Configuration Worksheets

Use the worksheets to record the configuration information for Secure+ Option.

- The *Local Node Security Feature Definition Worksheet* on page 100 is a record of the security functions defined for the local Connect:Direct node.
- The *Remote Node Security Feature Definition Worksheet* on page 101 is a record of the security functions defined for remote nodes. For each trading partner, define a remote node record. Make a copy of the blank *Remote Node Security Feature Definition Worksheet* for each remote node that you are configuring for Secure+ Option operations.

Local Node Security Feature Definition Worksheet

Record the security feature definition for the Secure+ Option .Local node record on this worksheet.

Local Node Name: _____

Configured Security Functions

Enable TLS protocol: Yes _____ No _____

Enable SSL protocol: Yes _____ No _____

Authorization Timeout: _____

Trusted Root Certificate File location: _____

Key Cert File: _____

Certificate Passphrase: _____

Cipher Suite(s) Enabled: _____

External Authentication

Enable External Authentication Yes _____ No _____

Certificate Validation Definition _____

Enable FIPS 140-2 mode Yes _____ No _____

Remote Node Security Feature Definition Worksheet

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

Remote Node Name: _____

Security Options

Protocol defined in the .Local node record: TLS or SSL _____

Is the remote node using the protocol defined in the .Local node record? Yes _____ No _____

If you answered No to the question above, identify the protocol to use for the Remote Node:

Note: If you do not enable the override option, Secure+ Option generates an error message.

Enable TLS protocol: Yes _____ No _____

Enable SSL protocol: Yes _____ No _____

Enable STS protocol: Yes _____ No _____

If you want to use the same protocol defined in the local node, select Default to Local Node.

Enable Override: Yes _____ No _____

Note: If you want to use the COPY statement to override settings in the parameters file for STS-enabled nodes, set Enable Override for the remote node. The COPY statement cannot override settings in SSL-enabled or TLS-enabled remote nodes.

Authorization Timeout: _____

TLS or SSL Protocol Functions

Trusted Root Certificate File location: _____

Certificate File: _____

Certificate Passphrase: _____

Cipher Suite(s) Enabled: _____

Enable Client Authentication: Yes _____ No _____ Default to local node _____

Certificate Common Name: _____

Note: If you want to add a second level of security, enable client authentication for the remote node and type the certificate common name.

External Authentication

Enable External Authentication Yes _____ No _____ Default to local node _____

Certificate Validation Definition _____

Enable FIPS 140-2 mode Yes _____ No _____

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 obtain a certificate from a certificate authority or you can create a self-signed certificate. When you obtain a certificate file, a trusted root certificate file and key file are created. This appendix describes the layout of the trusted root certificate file 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.

A sample trusted root certificate file called trusted.txt is installed in the Secure+\certificates directory when you install Secure+ Option. Use any text editor to add or delete certificate information to this file. In simple configurations, only one trusted root certificate file is used. In more sophisticated configurations, you may associate individual trusted root files with one or more node records.

When you use a certificate signing request (CSR) tool, such as the Certificate Wizard, you do not need to change the contents of the key certificate file. This is created for you by the Certificate Wizard.

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-----
```

Appendix C

Testing Secure+ Option with the STS Protocol

Before using the STS protocol in a production environment, test the installation to ensure Secure+ Option is operational. The parameters file is initially populated with two node records: the .Local record *and* a remote record for the associated Connect:Direct node. The remote record for this local node is created to support loopback (PNODE=SNODE) processing to test Secure+ Option. This section provides information for setting the options for the initial PNODE=SNODE testing of Secure+ Option for use with the STS protocol.

Setting Up the Local and Remote Node Record for Testing

To test the Secure+ Option installation, you need to configure the sample .Local node record and remote node record. To set up the local and remote node records for testing, complete the following procedure:

1. On a UNIX system, type the following command at the UNIX command prompt from within the ndm/bin directory:

   ```
   spadmin.sh
   ```

2. To configure the .Local node record, double-click **.Local**.

   The **Edit Record** dialog box opens to the **Security Options** tab.

3. Change the **Security Options** values (for the local or remote node record) as follows:

Field	Local Node Setting	Remote Node Setting
Secure+ Protocol	Enable STS Protocol	Enable STS Protocol
Node or Copy Statement Override	Enable Override	Default to Local Node
Authentication Timeout	90	90

4. Click the **STS Protocol** tab. The **Edit Record - STS Protocol** window displays the STS Options.
5. On the **STS Options** tab, click the following values:

Field	Local Node Setting	Remote Node Setting
Enable Digital Signatures	Yes	Default to Local Node
Enable Public Key Auto Updates	No	Default to Local Node
Enable Encryption	Yes	Default to Local Node

6. Click the **Authentication Keys** tab.
7. Generate the authentication keys as follows:
   a. Click the **Generate Key** button. The **Generate Random Number Seed** dialog box is displayed.
   b. Type an alphanumeric string at least 32-characters long in the **Random Number Seed** field and click **OK**.
8. Click the **Signature Keys** tab.
9. Generate the signature keys as follows:
   a. Click the **Generate Key** button. The **Generate Random Number Seed** dialog box is displayed.
   b. Type an alphanumeric string at least 32-characters long in the **Random Number Seed** field and click **OK**.

      The Secure+ Option Admin Tool generates the Signature keys and populates the **Local Public Key** field.
10. Click **OK** to save the changes.

Exchanging Public Keys

To complete the setup for testing the STS protocol, you need to exchange public keys. In a production environment, you create and configure a remote node for each trading partner. Then you export keys from the remote node and send this information to the trading partner. Finally, you import keys that the trading partner sends to you into the remote node record.

Exporting Keys

The first step to exchange keys is to export the keys of the remote node. Exporting keys copies the public keys to a file that can be sent to the trading partner. Perform the following steps to export the public keys for the remote node entry:

1. Create a specific directory for storing the public key files.

2. From the **Secure+ Admin Tool Main Window**, highlight the remote node record. Then click **Export Selection** from the **Key Management** menu. The **Select Export Destination Directory** dialog box is displayed, for example, *d_dir*/ndm/Secure+ export.
3. Click **Select** and save the key file to the directory that you created in step 1. A file with an **sxp** extension is created for the remote node entry for this local node. The file is named after the local node.

Importing Keys

Now you are ready to import keys. In a production environment, your trading partner sends you this file. However, for test purposes you use the information you exported from the preceding process. Perform the following steps to import the public keys:

1. From the **Secure+ Admin Tool Main Window**, highlight the remote node record and click **Import Public Keys** from the **Key Management** menu. The **Select Import Source File** dialog box is displayed.
2. Go to the directory where you saved the **sxp** file in *Exporting Keys* on page 108.
3. Highlight the **sxp** file for this node and click **Import**. The key is imported to the remote node record.

Ensure that the remote keys are imported by viewing the **Remote Public Key** field located in the **Authentication Keys** tab and **Signature Keys** tab.

Note: The remote node public keys are not displayed in the .Local node record, but are displayed in the base remote node record.

Validating the Configuration

Perform this procedure to ensure that the nodes have been properly configured. The validation process checks each node to ensure that all necessary options have been defined and keys have been exchanged. Perform the following steps to validate the parameters file:

1. From the **Secure+ Admin Tool Main Menu**, click **Validate Secure+** from the **File** menu. The **Secure+ Admin Tool - Validation Results** window is displayed.
2. If the parameters file is not correctly configured, warning and error messages are displayed.
3. Read each warning message. To correct each warning or error reported, go back to the parameters file and make changes as needed.

Note: Warning message do not always mean that the parameters file is incorrectly configured. Some warning messages are informational only.

4. Click **Close** to close the **Validation Results** window.

Exchanging Data and Verifying Results

Submit the sample Process provided with Connect:Direct. To verify the success of the sample Process and to review the Secure+ Option statistics for the session, refer to the *Connect:Direct for UNIX Getting Started Guide*.

Appendix D

Automation Scripts

The following scripts are provided as models for creating custom scripts to define your Secure+ Option environment and automate the implementation of it. To prevent any loss of data, you cannot run the scripts, but you can save them with a different name and modify them to suit your needs.

Automating Secure+ Option Implementation for the STS Protocol

The two sample scripts in this section illustrate using the Secure+ CLI to automate a mass implementation of Secure+ Option and add a remote node, respectively. Both scripts illustrate using statically generated public keys.

Using Statically Generated Public Keys for Automated Mass Implementation

The spcust_sample1 script demonstrates a mass implementation of Secure+ Option using statically generated public keys.

```
REM spcust_sample1.sh contains an example of configuring
REM Secure+ with the Secure+ CLI. The example demonstrates
REM the use of statically generated public keys to facilitate
REM an automated mass rollout of Secure+.
REM
REM Variables
REM The return code.
REM spcli.sh returns the highest return code of the commands
REM it executed. Possible return codes and their meanings are
REM      0    success
REM      4    warning
REM      8    error
REM     16    fatal error
set cdInstallDir=
set spDir=%cdInstallDir%\Server\Secure+
pushd "%spDir%"
REM  Main script
REM
echo.
echo This script has been prevented from running because it will alter
echo The configuration of Secure+. Before removing this warning and its
echo exit call, please modify the script so that it carries out only
echo desired modifications to the configuration of Secure+.
echo.
goto :EOF
all :initCustom
call :invokeCLI
call :terminateCustom
REM End of main script
goto :EOF
REM
REM Functions
REM
REM Custom initialization logic written by customer.
REM
:initCustom
REM Customer adds custom initialization code here.
echo Init custom...
echo.
REM del /F "%spDir%\Nodes"
REM End ofinitCustom
goto :EOF
REM Invoke CLI to configure Secure+.
:invokeCLI
   set tempFile=clicmds.txt
```

```
   echo      ;
 >%tempFile%
    echo    ; -- Create a new parmfile
>>%tempFile%
    echo    ;
 >>%tempFile%
    echo    init parmfile
>>%tempFile%
    echo        localnode=KCHEN-TW-4200
 >>%tempFile%
    echo        path="%spDir%\Nodes"
 >>%tempFile%
    echo        passphrase=12345678901234567890123456789012
 >>%tempFile%
    echo    ;
>>%tempFile%
    echo    display info
>>%tempFile%
   echo    ;
>>%tempFile%

  echo    ;
  >>%tempFile%
    echo    ; -- Synch with netmap
>>%tempFile%
    echo    ;
 >>%tempFile%
    echo    sync netmap
  >>%tempFile%
    echo        path=v4.3.00\KCHEN-TW-4200
 >>%tempFile%
    echo        name=*
 >>%tempFile%
    echo    ;
>>%tempFile%
    echo    ;
 >>%tempFile%
     echo    ; -- Update localnode, enable protocol and add keys.
>>%tempFile%
     echo    ;
  >>%tempFile%
     echo    update localnode
 >>%tempFile%
    echo         protocol=sts
>>%tempFile%
    echo         override=y
 >>%tempFile%
    echo         STSEnableSig=y
 >>%tempFile%
    echo         STSEnableEnc=y
```

```
>%tempFile%
    echo         STSEnableAutoUpdate=y
  >>%tempFile%
   echo         stsAuthlocalkey=set
>>%tempFile%
    echo         StsAuthKeyPairFile=keypairfile
 >>%tempFile%
    echo         StsAuthKeyPairFilePassphrase=secret
>%tempFile%
    echo         stsSiglocalkey=set
>>%tempFile%
    echo         StsSigKeyPairFile=keypairfile
 >>%tempFile%
    echo         StsSigKeyPairFilePassphrase=secret
>>>%tempFile%
      echo     ;
  >>%tempFile%
    echo     ;
>>%tempFile%
   echo     ; -- Display localnode
 >>%tempFile%
   echo     ;
 >>%tempFile%
    echo     display localnode
>>%tempFile%
    echo     ;
 >>%tempFile%
    echo     ;
>>%tempFile%
    echo     ; -- Update remotenode records
>>%tempFile%
    echo     ;
 >>%tempFile%
    echo     update remotenode
>>%tempFile%
    echo         name=*
 >>%tempFile%
   echo
STSAuthRemoteKey=0200.5F8E.F736.1E38.85C7.2BD0.E312.BD4D.FBC5.A2D3.8E1F
>>%tempFile%
```

```
   echo
 STSSigRemoteKey=0200.5F8E.F736.1E38.85C7.2BD0.E312.BD4D.FBC5.A2D3.8E1F
 >>%tempFile%
     echo     ;
 >>%tempFile%
     echo     ;
  >>%tempFile%
     echo     ; -- Verify localnode does not have remote keys
  >>%tempFile%
     echo     ;
  >>%tempFile%
     echo     display localnode
  >>%tempFile%
     echo     ;
 >>%tempFile%
     echo     ;
 >>%tempFile%
     echo     ; -- Verify all remote nodes have remote pub keys set
 >>%tempFile%
     echo     ;
 >>%tempFile%
     echo     display remotenode
 >>%tempFile%
     echo         name=*
  >>%tempFile%
     echo     ;
  >>%tempFile%
    echo     ;
 >>%tempFile%
    echo     ; -- Update localnode, enable protocol and add keys.
 >>%tempFile%
    echo     ;
 >>%tempFile%
     echo     update localnode
  >>%tempFile%
     echo         stsauthlocalkey=gen
  >>%tempFile%
     echo         stsauthkeyseedType=dynamic
  >>%tempFile%
     echo         STSPrevAuthKeyExpDateTime=2005:01:01-20:13:44
>>%tempFile%
```

```
   echo         stssiglocalkey=gen
 >>%tempFile%
    echo         stssigkeyseedType=dynamic
 >>%tempFile%
    echo         STSPrevSigKeyExpDateTime=2005:01:01-20:13:45
  >>%tempFile%
    echo     ;
  >>%tempFile%
        echo     display localnode;
 >>%tempFile%
     echo     ; -- Validate parmfile
  >>%tempFile%
    echo     ;
 >>%tempFile%
    echo     validate parmfile
  >>%tempFile%
    echo     ;
 >>%tempFile%
   echo     quit;
 >>%tempFIle%
       call "%spDir%\spcli.sh" -e 8 -li y < %tempFile%
       set RC=%ERRORLEVEL%
       del %tempFile%
 REM End of invokeCLI
 goto :EOF
 REM Custom termination logic written by customer.
 REM
 :terminateCustom
 REM Customer adds custom termination code here.
 REM For example, E-mail standard out log for review.
 REM Send error messages to system monitoring facility.
   echo.
   echo Custom Terminating with errorlevel of %RC%
   echo.
 REM End of terminateCustom
 goto :EOF
popd
```

Adding a Remote Node to a Connect:Direct Network

The spcust_sample2 script illustrates how to automate adding a remote node to a Connect:Direct network.

```
@echo off
REM spcust_sample2.sh contains an example of configuring
REM Secure+ with the Secure+ CLI. The example demonstrates
REM the use of statically generated public keys to
REM facilitate the automated addition of a new node to a
REM Connect:Direct network. In this example, the name of
REM the new node is cd-ultra3600.
REM     A script based on this sample is used to
REM configure the existing nodes in the new node's Netmap.
REM The new node itself is configured with a script based
REM on spcust_sample1.sh. In order for the addition to be
REM successful, the two scripts must use the same seeds
REM to generate keys.
REM
REM Variables
REM The return code.
REM spcli.sh returns the highest return code of the commands
REM it executed. Possible return codes and their meanings are
REM      0    success
REM      4    warning
REM      8    error
REM     16    fatal error
set cdInstallDir=
set spDir=%cdInstallDir%\Server\Secure+
pushd "%spDir%"
REM  Main script
echo This script has been prevented from running because it will alter
echo The configuration of Secure+. Before removing this warning and its
echo exit call, please modify the script so that it carries out only
echo desired modifications to the configuration of Secure+.
goto :EOF
call :initCustom
call :invokeCLI
call :terminateCustom
REM End of main script
goto :EOF
REM Functions
REM Custom initialization logic written by customer.
:initCustom
REM Customer adds custom initialization code here.
echo Init custom...
echo.
REM End ofinitCustom
goto :EOF
REM Invoke CLI to configure Secure+.
```

```
invokeCLI
    set tempFile=clicmds.txt
    echo    ;
 >%tempFile%
    echo    ;
 >>%tempFile%
    echo    ; -- Synch with netmap
 >>%tempFile%
    echo    ;
 >>%tempFile%
    echo    sync netmap
>>%tempFile%
    echo        path=v4.3.00\KCHEN-TW-4200
 >>%tempFile%
    echo        name=*
 >>%tempFile%
    echo    ;
>>%tempFile%
    echo    ;
>>%tempFile%
    echo    ; -- Update remotenode records
 >>%tempFile%
    echo    ;
>>%tempFile%
    echo    update remotenode
 >>%tempFile%
    echo        name=salmon3600
 >>%tempFile%
    echo        stsAuthlocalkey=set
  >>%tempFile%
    echo        StsAuthKeyPairFile=keypairfile
 >>%tempFile%
    echo        StsAuthKeyPairFilePassphrase=secret
>>%tempFile%
    echo        stsSiglocalkey=set
>>%tempFile%
    echo        StsSigKeyPairFile=keypairfile
 >>%tempFile%
    echo        StsSigKeyPairFilePassphrase=secret
>>%tempFile%
    echo
STSAuthRemoteKey=0200.5F8E.F736.1E38.85C7.2BD0.E312.BD4D.FBC5.A2D3.8E1F
>>%tempFile%
    echo
STSSigRemoteKey=0200.5F8E.F736.1E38.85C7.2BD0.E312.BD4D.FBC5.A2D3.8E1F
 >>%tempFile%
    echo    ;
 >>%tempFile%
   echo    display remotenode name=salmon3600
```

```
  >>%tempFile%
     echo    ;
  >>%tempFile%
     echo    validate parmfile
   >>%tempFile%
     echo    ;
  >>%tempFile%
    echo    quit;
 >>%tempFIle%
        call "%spDir%\spcli.sh" -p "%spDir%\Nodes" -e 8 -li y < %tempFile%
        set RC=%ERRORLEVEL%
        del %tempFile%
 REM End of invokeCLI
 goto :EOF
 REM Custom termination logic written by customer.
 :terminateCustom
 REM Customer adds custom termination code here.
 REM For example, E-mail standard out log for review.
 REM Send error messages to system monitoring facility.
     echo Custom Terminating with errorlevel of %RC%
 REM End of terminateCustom
 goto :EOF
popd
```

Configuring Secure+ Option to Use the SSL or TLS Protocol

The spcust_sample3 script demonstrates using the Secure+ CLI to configure Secure+ Option to use the SSL or TLS protocol with the trusted root file, key certificates, and ciphers.

```
@echo off
REM spcust_sample3.sh contains an example of configuring
REM Secure+ to use SSL or TLS protocols with the Secure+ CLI.
REM The example demonstrates the configuration of Secure+
REM with the trusted root and key certificates and ciphers
REM Variables
REM The return code.
REM spcli.sh returns the highest return code of the commands
REM it executed. Possible return codes and their meanings are
REM      0    success
REM      4    warning
REM      8    error
REM     16    fatal error
  echo  ; -- Create a new parmfile
>>%tempFile%
    echo  ;
 >>%tempFile%
    echo    init parmfile
>>%tempFile%
    echo        localnode=KCHEN-TW
>>%tempFile%
    echo        path="%spDir%\Nodes"
>>%tempFile%
    echo        passphrase=12345678901234567890123456789012
>>%tempFile%
    echo    ;
 >>%tempFile%
    echo    display info
>>%tempFile%
    echo    ;
>>%tempFile%
    echo    ;
>>%tempFile%
    echo    ; -- Synch with netmap
 >>%tempFile%
set cdInstallDir=
set spDir=%cdInstallDir%\Server\Secure+
pushd "%spDir%"
REM  Main script
 This script has been prevented from running because it will alter
 The configuration of Secure+. Before removing this warning and its
 exit call, please modify the script so that it carries out only
 desired modifications to the configuration of Secure+.
goto :EOF
call :initCustom
call :invokeCLI
call :terminateCustom
REM End of main script
goto :EOF
REM Functions
REM Custom initialization logic written by customer.
:initCustom
```

```
REM Customer adds custom initialization code here.
echo Init custom...
echo.
REM del /F "%spDir%\Nodes"
REM End ofinitCustom
goto :EOF
REM Invoke CLI to configure Secure+.
REM
:invokeCLI
    set tempFile=clicmds.txt
    echo  ;
 >%tempFile%
  echo    ;
>>%tempFile%
    echo    sync netmap
>>%tempFile%
    echo        path=v4.3.00\KCHEN-TW-4200
 >>%tempFile%
    echo        name=*
>>%tempFile%
    echo    ;
>>%tempFile%
    echo    ;
 >>%tempFile%
    echo    ; -- Update localnode, enable protocol and add keys.
 >>%tempFile%
    echo    ;
  >>%tempFile%
    echo    update localnode
>>%tempFile%
    echooverride=n
 >>%tempFile%
    echo    protocol=tls
>>%tempFile%
    echoSSLTLSTRUSTEDROOTCERTFILE="%spDir%\Certificates\trusted.txt"
 >>%tempFile%
    echo           SSLTLSCERTFILE="%spDir%\Certificates\keycert.txt"
>>%tempFile%
    echoSSLTLSCERTPASSPHRASE=athena
 >>%tempFile%
    echoSSLTLSENABLECIPHER=all
 >>%tempFile%
    echo    ;
 >>%tempFile%
    echo    ;
 >>%tempFile%
    echo    ; -- Display localnode
 >>%tempFile%
    echo    ;
>>%tempFile%
     echo    display localnode
 >>%tempFile%
    echo    ;
 >>%tempFile%
    echo    validate parmfile
>>%tempFile%
```

```
    echo    ;
   >>%tempFile%
    echo    quit;
   >>%tempFIle%
         call "%spDir%\spcli.sh" -e 8 -li y < %tempFile%
         set RC=%ERRORLEVEL%
         del %tempFile%
  REM End of invokeCLI
  goto :EOF
  REM Custom termination logic written by customer.
  :terminateCustom
  REM Customer adds custom termination code here.
  REM For example, E-mail standard out log for review.
  REM Send error messages to system monitoring facility.
      echo.
      echo Custom Terminating with errorlevel of %RC%
      echo.
  REM End of terminateCustom
  goto :EOF
 popd
```

Appendix E

Using the LCU to Configure Encrypted Passwords

The Secure+ CLI displays passwords in plain text. If you need to encrypt passwords for use with the Secure+ CLI, use the Local Connection Utility (LCU) to create an LCU file that contains non-encrypted information used to encrypt the password and the encrypted password, such as a keycert passphrase. You can then refer to this file when prompted for passwords.

LCU Files

The following example shows how to specify when an LCU file is used in place of a plain-text password:

```
SPCLI> Create STSKeyPair
  KeyPairFile="C:\My STSKeys\keypair"
  Passphrase="LCU:C:\My STSKeys\MyPwd.lcu";
SPCG6701 rc=0 Create stskeypair command successful.

SPCLI> Update RemoteNode
  Name=*
  StsAuthLocalKey=SET
  StsAuthKeyPairFile="C:\My STSKeys\keypair"
  StsAuthKeyPairFilePassphrase="LCU:C:\My STSKeys\MyPwd.lcu"
  StsSigLocalKey=SET
  StsigKeyPairFile="C:\My STSKeys\keypair"
  StsKigKeyPairFilePassphrase="LCU:C\My STSKeys\MyPwd.lcu";
  SCPG4701 rc=0 Update remote node "JLYON-XP.4400" command successful
SPCLI Update LocalNode
  SslTlsCertFile=C:\Certs\iden_keycert.pem
  SslTlsCertPassphrase="LCU:C:\My STSKeys\MyPwd.lcu";
SPCG3601 rc=0 Update local node command successful.
```

The use of the LCU syntax "LCU:" indicates that what follows is an LCU filename and not a passphrase. The pathname of the LCU file can be a relative path, a relative path to the bin directory, or a full path. If **LCU:filename** contains spaces, it must be enclosed in quotation marks:

"LCU:filename". The default name of the LCU file is cddef.bin. After the cddef.bin file is created, you can rename it as needed.

LCU files can be used to provide encrypted passwords for the following commands and parameters:

Command	Parameter
Update LocalNode	StsAuthKeyPairFilePassphrase StsSigKeyPairFilePassphrase SslTlsCertPassphrase
Create RemoteNode	StsAuthKeyPairFilePassphrase StsSigKeyPairFilePassphrase SslTlsCertPassphrase
Update RemoteNode	StsAuthKeyPairFilePassphrase StsSigKeyPairFilePassphrase SslTlsCertPassphrase
Create STSKeyPair	Passphrase
Update Client	SslTlsCertPassphrase
Update SEAServer	SslTlsCertPassphrase

Creating an LCU File

Complete the following procedure to create an LCU file:

1. Type the following command to run the LCU utility:

```
lcu.sh
```

2. As you are prompted, enter values for the following parameters:
   - Node
   - API Address
   - API Port
   - User Name
   - Password
   - Confirm Password

   The cddef.bin file is created.

Glossary

A

Access File

A file that is generated automatically when you create the Secure+ Option parameters file for the first time and contains the Secure+ Option passphrase to encrypt and decrypt the private keys in the Secure+ Option parameters file. Your Secure+ Option administrator must secure the access file. This file can be secured with any available file access restriction tools. Availability of the access file to unauthorized personnel can compromise the security of data exchange.

Administration Tool (Admin Tool)

The Secure+ Option tool that enables configuring and maintaining the Secure+ Option environment. This is the only tool you can use to configure and maintain Secure+ Option.

Asymmetric Keys

A separate but integrated user key pair comprised of one public key and one private key. Each key either encrypts information or decrypts information but does not perform both functions.

Authentication

The process of verifying that a particular name really belongs to a particular entity and assurance that a message is not modified in transit or storage.

C

Certificate

A document obtained from a certificate authority 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 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.

Certificate Authority

A company responsible for verifying and processing certificate requests, and issuing and managing certificates. The CA you choose should be one that your trading partners trust. You must meet the requirements for the CA you choose.

Certificate Revocation List

A list of certificates that have been revoked.

Certificate Signing Request

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

Cipher Suite

A cryptographic algorithm that enables you to encrypt and decrypt files and messages.

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 security that requires the client or SNODE to authenticate its identity to the server by sending its certificate. The SNODE must request a certificate before the client sends it.

Common Criteria

An internationally recognized set of guidelines (ISO 15408), which define a common framework for evaluating security features and capabilities of Information Technology security products.

Configuration File

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

D

Data Confidentiality

Ensuring that data remains private during transmission.

Data Integrity

Ensuring that information is not altered during transmission.

Decryption

Any process to convert cipher text back into plain text.

Digital Certificate

A specifically formatted document that allows you to authenticate or identify yourself to a Web browser, an E-mail reader, a secure server, or a client. It contains information on who you are, your relevant details, and who issued the certificate. A certificate can be tied to an E-mail address, a Web server or a company, and in each case the certificate is used for different things. A basic E-mail certificate allows you to prove that you are who you say you are. It also allows you to store more information about yourself such as your place of work or telephone contact details. The certificate also contains your public key.

Digital Signature

Processing 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 who signs a physical document. Because a private key is unique to each person, a value encrypted using the sender's private key and subsequently decrypted using the sender's public key authenticates the sender's identity.

E

Encryption

Any process that converts plain text into cipher text.

Encryption Algorithm

The set of mathematical logic that encrypts or decrypts data.

I

Integrity

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

J

Java

A programming language that allows development of applications that can run from any kind of device or machine: a PC, a Macintosh computer, a network computer, the Internet, or a mobile phone. The Java language makes it possible to develop software that is portable, modular, and secure.

JDK

The Java Development Kit (JDK) contains the software and tools that developers need to compile, debug, and run applets and applications written using the Java programming language.

JRE

The Java Runtime Environment (also known as the Java Runtime or JRE) consists of the Java virtual machine, the Java platform core classes, and supporting files. It is the runtime part of the Java Development Kit and provides no compiler, debugger, or tools. The JRE is the smallest set of executable files that constitute the standard Java platform.

K

Key Certificate File

A file stored on the client that contains an encrypted message to identify the client and enable client/server authentication during secure connections.

Keys

A collection of bits, usually stored in a file, which encrypts or decrypts a message.

L

License Management Key

A file containing definitions that Connect:Direct and Secure+ Option use to enable the software. You must have a license key to use either of these applications.

.Local Node Record

The base record in a parameters file that defines the Secure+ Option server. It includes the most commonly used settings at a site and is the central node through which all communication is filtered. Depending upon how each remote node record is configured, trading partner node records may use settings that are defined in the .Local node record.

N

Network Map (Netmap)

The file that identifies all valid Connect:Direct nodes in a network including a .Local node record and a remote node record for each trading partner. The network map also defines the rules or protocols used by each node when communicating with the local Connect:Direct node.

Nonrepudiation

Providing undeniable proof of origin of transmitted data.

P

Passphrase

Similar to a password but can be any characters, including spaces. A passphrase is stronger than a password, although not many programs support the use of a passphrase.

Password

A character-limited word or phrase that establishes identity to allow access to a system. Generally, a password is composed of letters, numbers, or both.

Primary Node (PNODE)

The node that submits the Connect:Direct Process to the secondary node (SNODE). In every communication, you must have a PNODE and an SNODE.

Private Key

The secret key of a public-private key cryptography system. This key enables you to *sign* outgoing messages and decrypt incoming messages.

Proof of Data Origin

A method of verifying the identity of the sender and that information is not altered during an electronic exchange.

Public Key

The public key of a public-private key cryptography system. This key confirms *signatures* on incoming messages or encrypts a file or message so that only the holder of the private key can decrypt the file or message. A public key is disseminated freely to clients and servers via certificates signed by a certificate authority (CA).

R

Remote Node Record

An entry in the parameters file that defines the security settings used to communicate with a trading partner. A remote node record must be defined for every trading partner you communicate with.

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 Certificate

A self-generated 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 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.

Station-to-Station Protocol (STS)

A three-pass variation of the basic Diffie-Hellman protocol. It allows you to establish a shared secret key between two nodes with mutual entity authentication. Nodes are authenticated using digital signatures to sign and verify messages or control blocks.

T

Third-Party Certificate

A certificate, other than those that are preconfigured for the application, that identifies an organization. If third-party certificates are used by the server, the corresponding trusted certificate must be installed in the client manually.

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.

Both the SSL protocol and the TLS protocol manage secure communication in a similar way. However, TLS provides a more standard, more secure method for managing authentication and exchanging messages. TLS uses Key-Hashing for Message Authentication Code (HMAC), to ensure that a record cannot be altered while traveling over an open network such as the Internet. TLS defines the Enhanced Pseudorandom Function (PRF), used to generate key data, with the HMAC and uses two hash algorithms to guarantee security. 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 exposed.

Trusted Root Certificate File

A file stored in a local directory on the client that contains signed certificates from trusted sources. During secured connections, the client compares the server certificate, or vice versa, to the trusted root certificate file to determine if the server certificate was signed by a trusted source. The client can establish a secured connection if a trusted source signed the server certificate. Optionally, if client authentication is used, the server compares the client certificate to its own trusted root certificate file to determine if the client certificate was signed by a trusted source.

U

Unsecure Connection

A 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

Symbols

A

C

D

E

F

G

H

I

K

L

M

N

O

P

R

S

T

U

V

W