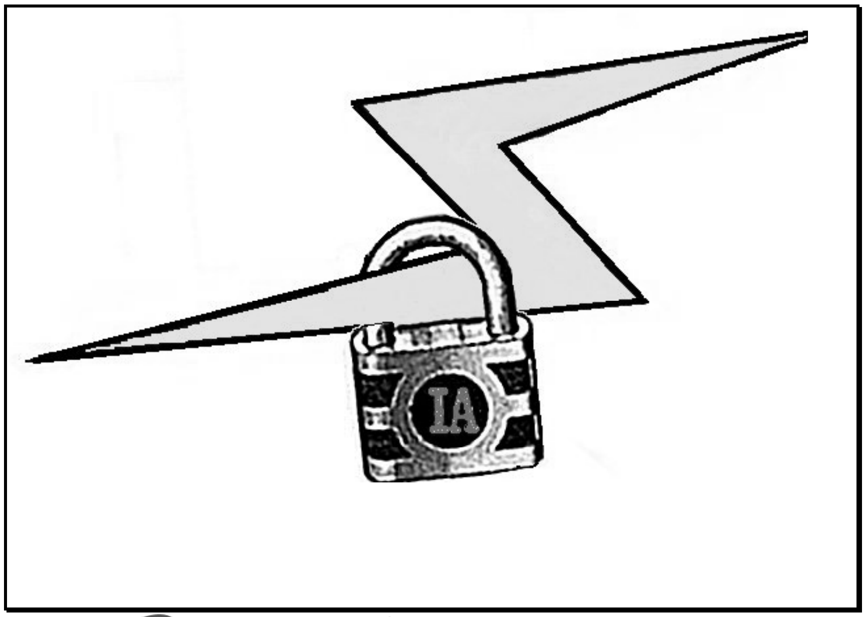


IAgent

TCIF 98-006 Issue 2 Interactive Agent Users Guide

Version 2.3.2

Lymeware Corporation
www.lymeware.com

IAgent

TCIF 98-006 Issue 2 Interactive Agent User's Guide

Version 2.3.2

Lymeware Corporation
19 Halls Road, Suite 216
Old Lyme, Connecticut USA

2003-IAUG232-0011

2003-IAUG232-0011

Contents

List of Examples

List of Figures

List of Tables

Preface

Software Version

This guide is published in support of Lymeware Corporation's **IAgent** software product, version 2.3.2. It may also be pertinent to later releases – please consult the release notes accompanying the software for further details.

Readership

This manual is intended for administrators who need to setup and manage the IAgent system. Knowledge of TCIF 98-006 Issue 2, EDI formats and TCIF EDI formats, ASC X.12 EDI standards, system administration, and basic cryptography concepts is required.

Scope of this Guide

This manual describes the configuration, operation and management of the IAgent system as described by TCIF98-006 Issue 2. This manual is divided into five parts, and several appendixes. Part 1 covers operation concepts of the IAgent system and introduces IA and SSL concepts. Part 2 covers how to install the IAgent system with all required components. Part 3 covers how to configure the IAgent system with manually generated configuration tables. Part 4 covers how to integrate the IAgent system with backend systems (BESs) using the provided Integration Application Program Interfaces (Integration APIs) . Part 5 is a reference section that also covers advanced topics and describes management tools used to maintain the IAgent system and maintenance issues.

The following is a brief outline of the contents of the manual:

Part 1 - Introduction

Chapter 1 Contains general introductory material to Interactive Agent (IA) systems, IA protocols and the different components of IA messaging and communication.

Chapter 2 Introduces the IAgent product and components.

Chapter 3 Contains basic introductory material to the SSL protocol, and the security components of SSL messaging and communication.

Part 2 – IAgent Installation

Chapter 4 Outlines how to install and configure the IAgent system for production use.

Chapter 5 Describes pre-installation tasks.

Chapter 6 Describes how to install the IAgent system.

Chapter 7 Describes post installation tasks.

Part 3 – IAgent Configuration

Chapter 8 Describes how to generate a server key and server certificate request, and how to obtain, test and install the certificate.

Chapter 9 Describes how to configure the IAgent system with configuration files.

Chapter 10 Describes how to add a Trading Partner to the IAgent system.

Chapter 11 Describes how to test connectivity and messaging to a Trading Partner.

Chapter 12 Multiple Trading Partner issues

Chapter 13 IA receipt overview

Chapter 14 IA receipt issues

Part 4 – Integration API Technical Reference

Chapter 15 Integration Overview

Chapter 16 Outbound (Client) Formats

Chapter 17 Inbound (Server) Formats

Chapter 18 File & Directory Integration

Chapter 19 Named Pipe Integration

Chapter 20 Socket Integration

Chapter 21 Tibco Message Integration

Chapter 22 Testing Integration Configuration

Chapter 23 Example Integration API Test Programs

Chapter 24 Testing Integration with Backend Systems

Part 5 – IAgent Administration

Chapter 25 Operations issues

Chapter 26 Maintenance issues

Chapter 27 Troubleshooting issues

Chapter 28 Error Messages

Chapter 29 Security issues

Chapter 30 How to get help

Manual Revision History

Version 2.3.2	August 2003
Version 1.3.1	January 2003
Version 1.3.0	April 2002
Version 0.3.0	November 2001
Version 0.2.6	April 2001
Version 0.2.5b	September 2000
Version 0.2.4b	March 2000
Version 0.2.3b	January 2000
Version 0.2 (“Quick Readme” version)	November 1999

Support Questions and Bug Reporting

Several e-mail addresses are available for customer support, technical support and sales questions or to report a potential bug in the software or documentation. If your product was purchased from Lymeware, please use the following addresses:

Service@lymeware.com for all account related inquires and issues, including those relating to licenses. If customers are unsure which address to use then they should send to this address. This address is monitored daily, and all messages will be responded to. This address also has the alias CustomerService@lymeware.com

Support@lymeware.com for all technical inquires and problem reports, including documentation issues from customers with support contracts. Customers should include relevant contact details, including company name and phone number, in initial message to speed processing. Messages that are continuations of an existing problem report should include the problem report ID in the subject line. Customers without support contracts with Lymeware Corporation should not use this address, but should contact their distributor directly. This address is monitored daily, and all messages will be responded to. See *Chapter 30* for more support information and options.

Sales@lymeware.com for all sales related inquires and similar communication. This address is monitored daily, and all messages will be responded to.

Bugs@lymeware.com for bug reports and documentation problems.

Bug reports on software releases are always welcome. These may be sent by any means, but e-mail to the bug reporting address listed above is preferred. Please send proposed fixes and successful workarounds with the report if possible. Additional useful information would include IAgent software version, hardware description, operating system version and patches, screen dumps, relevant sections of logs and configuration files, and failed messages files. Any reports will be acknowledged, but further action is not guaranteed. Any changes resulting from bug reports may be included in future releases.

Support for products purchased from our distributors:

For all products purchased from our distributors, all questions (with the notable exception of bug reports) should be sent directly to the distributor. If your distributor does not offer a service contract then one may be obtained from Lymeware Corporation. Please send all such inquiries to Sales@lymeware.com.

Documentation Conventions

Typographic conventions

This table describes typographic conventions used within this document.

Convention	Use
Italics	*Italics* type is used for titles of other manuals and documents, and for files and file extensions. **Example** *iagent.conf* file
Bold	Bold type is used for key terms the first time they are used within a chapter. **Example** The **Common Name** is
`Curier New Font`	This font is used for commands to be typed by the user or for system commands or utilities. **Example** Type: `iagent –m F –I dir`
<Angle brackets>	Angle brackets indicate variable information, such as input file names created by the user. **Example** <inputfilename>*.edi*

Table 1. Typographic conventions

Symbols used within syntax statements

This table describes symbols used within program syntax statements used within this document.

Convention	Use
<>	Substitute a value for any term that appears within the angle brackets. Do not enter the angle brackets unless specifically instructed to do so. **Example** rm <filename> means that you should type the name of the file you wish to delete.
{ }	Braces indicate a required part of a statement. Do not enter the braces. **Example** {-f <filename>} means that you must enter the –f parameter followed by a required filename.
[ ]	Square Brackets indicate an optional part of a statement. Do not enter the brackets. **Example** [-f <filename>] means you have the option of entering the –f parameter followed by a filename.
…	An ellipse indicates that the immediately preceding item can be repeated indefinitely. Do not enter the ellipse. **Example** -f … means you can repeat the–f parameter with other values.
( )	Parentheses should be entered as shown. They are part of the syntax of a statement and are not special symbols. **Example** -f (filename) means you should enter a filename enclosed by Parentheses.

Table 2. Symbols used within syntax statements

Part 1 - Introduction

Part 1 contains a general description of TCIF Interactive Agents and details of the components of such systems. This section also introduces the Lymeware IAgent Product, detailing product features and system requirements. Lastly this section includes an introduction to SSL.

CHAPTER 1 – INTRODUCTION TO TCIF INTERACTIVE AGENT SYSTEMS

In March 1999, The Telecommunications Industry Forum (TCIF) created an Electronic Interactive Agent (IA) standard, (TCIF 98-006), which allows secure EDI transactions over relatively insecure networks (i.e. The Internet), via TCP/IP and SSLv3. This standards body was chartered with increasing the ease and use of secure EDI ordering in the Telecommunications industry, an industry that has seen increasing growth due to the open competition created by the FCC 1994 Telecommunications Act.

The IA Specification describes and specifies a data interface to provide interoperability between trading partners in the Telecommunications Industry. The initial purpose of the data exchange was to support Local Service Ordering, but any EDI transaction or type of ordering is supported.

The intent of the IA was to improve upon the performance of the conventional "store and forward" method of messaging used by Value Added Networks (VANs). When using a VAN for connectivity, the sending trading partner drops off an EDI transaction message. Later, perhaps in minutes or hours, the VAN delivers the EDI transaction message to the receiving trading partner. This "batch" mode and the associated delays, preclude the use of a VAN for "near-real-time" EDI transaction processing. The trading partners using the IA for connectivity, with either a direct connection (frame relay, ATM, etc.), or a public data network (the Internet, the AXIS network, etc.) have no built-in delays and will support "near real-time" transaction processing.

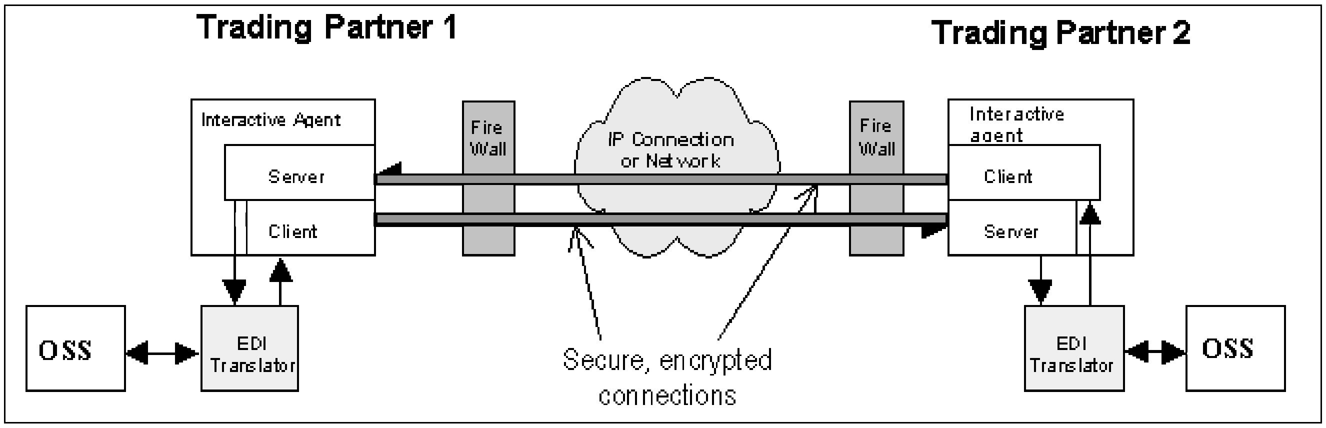


Figure 1. An Interactive Agent (IA) system for EDI communications

The IA design provides:

- Secured data transactions over unsecured networks (including the Internet), using Secure Socket Layer (SSLv3) protocol, a de facto Internet standard developed by Netscape.
- Sender and Receiver Trading Partner authentication (identification and validation) via Industry standard X.509 certificates issued by a valid third-party Certificate Authority.
- Digital non-repudiation by both the sending and receiving Trading Partners.

- Supports current and emerging Public Key Infrastructure (PKI) security standards (including RSA PKCS and IETF PKIX standards support).
- Supports transaction receipts with multiple levels of validation, customized by trading partner.
- Supports multiple levels of message validation on each transaction, customized by trading partner.

The following cipher suites are supported by the IA standard:

- RSA_NULL_SHA1 - Basic authentication without encryption,
- RSA_DES_CBC_SHA1 - Authentication with DES encryption, and
- RSA_3DES_CBC_EDE_SHA1 - Authentication with triple DES encryption

More information on supported cipher suites can be found in TCIF-98-009.

All of the following message types are internally sent and received in ASN.1 DER format. Specific details of the DER representation may be found in the TCIF standard. The following are fully supported:

Message types:

- Basic EDI with Security (SSL encryption only)
- Enhanced EDI Security with message integrity support (SSL encryption and digest)
- Enhanced EDI Security with Non-Repudiation (includes SSL encryption and signed message integrity)

Receipt types:

- No receipt
- Basic Receipt with Security (SSL encryption only)
- Enhanced Receipt Security with message integrity support (SSL encryption and digest)
- Enhanced Receipt Security with Non-Repudiation (includes SSL encryption and signed message integrity)

Status messages:

- A defined **TEST** IA status message.
- A defined **STOP WAN** IA status message.
- A defined **STOP OSS** IA status message.
- A custom 32 bit IA status message.

Additional protocol details, including required operational details and ASN.1 messages specific components of the TCIF Interactive Agent can be found in the Telecommunications Industry Forum

(TCIF) standards documents, including the TCIF 98-006, and TCIF 98-016. These may be acquired via the Alliance for Telecommunications Industry Solutions (www.atis.org).

CHAPTER 2 – INTRODUCTION TO THE IAGENT PRODUCT

The IAgent product is Lymeware Corporation's commercial-class secure messaging solution for the Telecommunications Industry. The IAgent product is a conformal implementation of the TCIF/ECIC Interactive Agent (IA v2), which allows secure EDI transactions over relatively insecure networks, via TCP/IP and SSLv3.

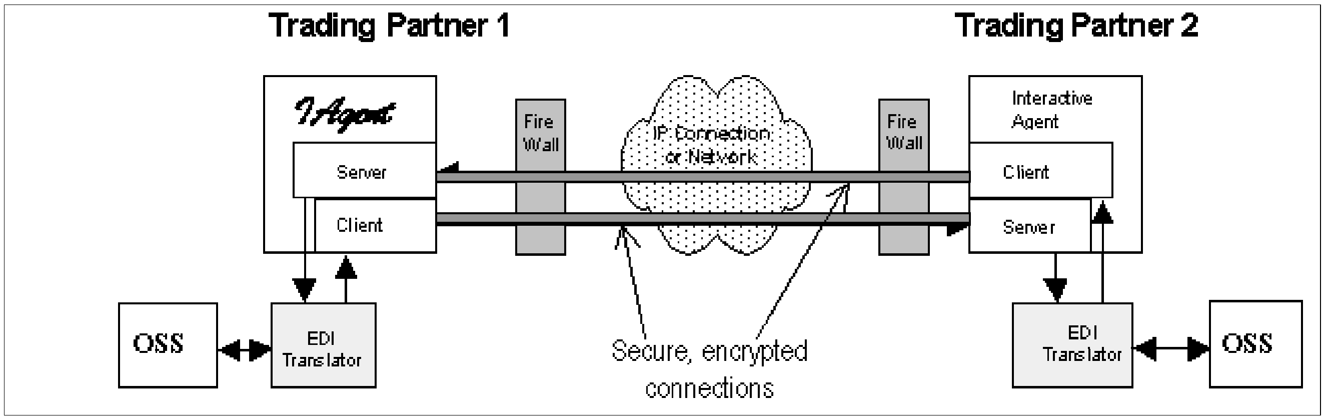


Figure 2. Using an IAgent system for IA trading partner communications

The IAgent product supports:

- Secured data transactions over unsecured networks (including the Internet), using Secure Socket Layer (SSLv3) protocol, a de facto Internet standard developed by Netscape.
- Sender and Receiver Trading Partner authentication (identification and validation) via Industry standard X.509 certificates issued by a valid third-party Certificate Authority.
- Digital non-repudiation by both the sending and receiving Trading Partners.
- Supports current and emerging Public Key Infrastructure (PKI) security standards (including RSA PKCS and IETF TLS and PKIX standards support).
- Supports transaction receipts with multiple levels of validation, customized by trading partner.
- Supports multiple levels of message validation on each transaction, customized by trading partner.

IAgent product benefits:

- Client / Server Architecture with Public Standards, including TCIF, IETF and RSA support
- Near Real- time High Volume Performance (up to 30 transactions a minute)
- Multi-threaded and Scaleable Design
- Customer-centric Security and Auditing
- Elimination of EDI Value Added Network (VAN) fees

- •Simple Operational Support System (OSS) Integration via Multiple Application Program Interfaces (Integration APIs)
- •Web-based Monitoring
- •Optionally supports the Tibco™ TIB/Rendezvous API for BES support
- •Uses the popular OpenSSL cryptography toolkit

Simple Integration is the key

The underlying structure of the TCIF IA standard and the IAgent design is a symmetrical client/server configuration where both the client and server functions are required at each implementation. Integrating these client and server components into an existing OSS or backend system is key to performance, scalability and deployment time. IAgent provides **Integration APIs**, designed for simple integration with common application interfaces. The four supported integration APIs (for internal message input and output transfer) include: the File/Directory interface, the Named Pipe interface, the IP Socket interface and the Tibco^{tm} TIB/Rendezvous message bus interface.

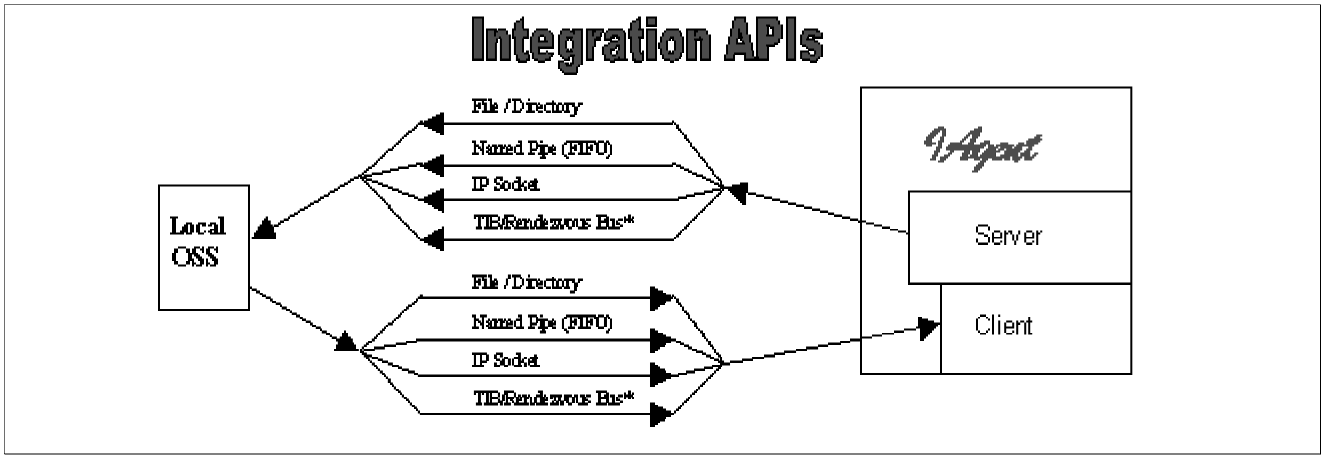


Figure 3. IAgent Integration APIs

- •The File/Directory interface atomically reads and writes ASCII text files (EDI messages in standard ASC X.12 formats).
- •The Named Pipe interface reads from and writes to specific user defined named pipes (or FIFOs).
- •The IP Socket interface reads from a single user defined input socket and writes to a single output socket. The customer processes reading/writing to this interface do not have to reside on the same platform as the IAgent system.
- •The (optional) Tibco^{tm} TIB/Rendezvous message bus interface reads input Tibco messages with a single subscriber and writes output Tibco messages with multiple publishers, supporting multiple platforms.

User defined connectivity

Connectivity: IA Trading partners must share TCP/IP connectivity either through a direct private connection between both trading partners (e.g. Frame Relay) or over a shared/public network (such as the Internet). Each trading partner must provide the other partner their unique IP addresses and port assignments for the location to send and receive IA messages (e.g. EDI Pre-Order and Ordering, or other message types). IAgent uses standard SSLv3 to provide the secured transport over the TCP/IP connection. The specific agreed method to connect with TCP/IP and other IA implementation issues, such as levels of message and receipt security, supported cipher suites, certificate requirements and valid Certificate Authorities, are typically addressed in a Joint Implementation Agreement between trading partners. See *Appendix A* for necessary Trading Partner information and worksheets.

How does it work?

The heart of the IAgent system is the Public Key Infrastructure (PKI), comprised of X.509 client certificates and private RSA keys, X.509 server certificates and private RSA keys, and one or more Certificate Authority (CA) X.509 certificates. See *Chapter 2* for more information on Certificates and Public Key Encryption.

Here is how it works in a nutshell. The IAgent client has its own X.509 certificate, with a SSL unique Common Name (as supplied by the Trading Partner) and the certificate's matching private RSA key. The client also has one or more CA (usually root) certificates. These certificates will be used during the IA handshake (between the client and the server) to verify the validity of the server's certificate.

Similarly, on the other side of the fence, the IAgent server has its own X.509 certificate and private key. The server also has one or more CA certificates that it uses to validate the client's certificate.

A very simplified version of the IA / SSLv3 handshake (with emphasis on certificate transfers and validation) would look something like this:

IAgent Client	IAgent Server
Connect --->	
	<----------------- Server Hello (handshake start)
Client Hello -------------------------------------->	
	<------------------- Server sends own certificate
Client validates Server certificate with correct CA certificate (if local copy exists). If valid then Client sends own certificate -------------->	
	Server validates Client certificate with correct CA certificate (if local copy exists). If valid then handshake completes and secures connection created
Client sends IA message ------------------------>	Server reads IA message
Client send ends successfully --------------------->	
Client closes socket connection	Server detects client socket close and closes secure connection.

Table 3. IAgent Handshake Sequence

Each Trading Partner's IA consists of two components: an outbound client and an inbound server.

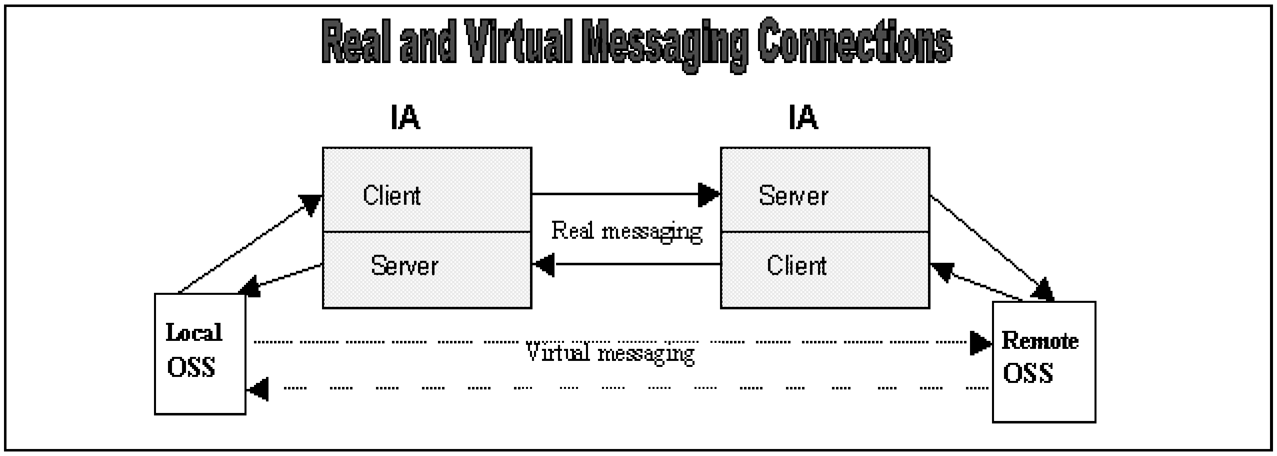


Figure 4. Real and Virtual Messaging Connections

The IA Client reads an input message from a "backend" EDI system. This message is in a raw ASC X.12 EDI format. The Client processes the message, reading the header (also called the ISA header), then the body of the EDI message. Depending on the destination-trading partner (read in the ISA header), the Client will format the correct IA message with a specific level of detail and security. The Client will then connect to the destination trading partner's own IA server via a TCP/IP connection, either a dedicated connection (Frame Relay, ATM, etc.) or a public connection such as the Internet. After the initial TCP/IP connection, the Client sets up a validated secure connection to the destination trading partner's IA Server, via SSL, and sends the IA message in a shared encrypted form. The Client then breaks down the secure connection and disconnects from the Server.

The IA Server waits for a remote IA Client connection request. When such request is received, the Server sets up a validated secure connection and reads the encrypted IA message sent by the remote Client. The Server parses and validates the message contents, and if valid sends the body of the message (in EDI format) to a “backend” EDI system (BES) for further processing.

CHAPTER 3 – INTRODUCTION TO SSL

As an introduction this chapter is aimed at readers who are familiar with the Web, HTTP, and TCP/IP, but are not security experts. It is not intended to be a definitive guide to the SSL protocol, nor does it discuss specific techniques for managing certificates in an organization. Rather, it is intended to provide a common background for IAgent users by pulling together various concepts, definitions, and examples as a starting point for further exploration. For more detailed information on SSL and the cryptographic components used in SSL see *Appendix H – References*.

Cryptographic Techniques

Understanding SSL requires an understanding of cryptographic algorithms, message digest functions (also called. one-way or hash functions), and digital signatures. These techniques are the subject of entire books and provide the basis for privacy, integrity, and authentication.

Cryptographic Algorithms

Suppose Alice wants to send a message to her bank to transfer some money. Alice would like the message to be private, since it will include information such as her account number and transfer amount. One solution is to use a **cryptographic algorithm**, a technique that would transform her message into an encrypted form, unreadable except by those it is intended for. Once in this form, the message may only be interpreted through the use of a secret key. Without the key the message is useless: good cryptographic algorithms make it so difficult for intruders to decode the original text that it isn't worth their effort.

There are two categories of cryptographic algorithms: conventional and public key.

Conventional cryptography, also known as **symmetric cryptography**, requires the sender and receiver to share a **key**: a secret piece of information that may be used to encrypt or decrypt a message. If this key is secret, then nobody other than the sender or receiver may read the message. If Alice and the bank know a secret key, then they may send each other private messages. The task of privately choosing a key before communicating, however, can be problematic.

Public key cryptography (PKI), also known as **asymmetric cryptography**, solves the key exchange problem by defining an algorithm that uses two keys, each of which may be used to encrypt a message. If one key is used to encrypt a message then the other must be used to decrypt it. This makes it possible to receive secure messages by simply publishing one key (the public key) and keeping the other secret (the private key). The most widely supported key pair algorithm is the **RSA** algorithm, from RSA Security®, now in the public domain.

Anyone may encrypt a message using the public key, but only the owner of the private key will be able to read it. In this way, Alice may send private messages to the owner of a key-pair (the bank), by encrypting it using their public key. Only the bank will be able to decrypt it.

Message Digests

Although Alice may encrypt her message to make it private, there is still a concern that someone might modify her original message or substitute it with a different one, in order to transfer the money to themselves, for instance. One way of guaranteeing the **integrity** of Alice's message is to create a concise summary of her message and send this to the bank as well. Upon receipt of the message, the bank creates its own summary and compares it with the one Alice sent. If they agree then the message was received intact.

A summary such as this is called a **message digest** or hash function. Message digests are used to create short, fixed-length representations of longer, variable-length messages. Digest algorithms are designed to produce unique digests for different messages. Message digests are designed to make it too difficult to determine the message from the digest, and also impossible to find two different messages which create the same digest -- thus eliminating the possibility of substituting one message for another while maintaining the same digest.

Another challenge that Alice faces is finding a way to send the digest to the bank securely; when this is achieved, the integrity of the associated message is assured. One way to do this is to include the digest in a digital signature.

Digital Signatures

When Alice sends a message to the bank, the bank needs to ensure that the message is really from her, so an intruder does not request a transaction involving her account. A digital signature, created by Alice and included with the message, serves this purpose.

Digital signatures are created by encrypting a digest of the message, and other information (such as a sequence number) with the sender's private key. Though anyone may decrypt the signature using the public key, only the signer knows the private key. This means that only they may have signed it. Including the digest in the signature means the signature is only good for that message; it also ensures the integrity of the message since no one can change the digest and still sign it.

To guard against interception and reuse of the signature by an intruder at a later date, the signature contains a unique sequence number. This protects the bank from a fraudulent claim from Alice that she did not send the message -- only she could have signed it (**non-repudiation**).

Certificates

Although Alice could have sent a private message to the bank, signed it, and ensured the integrity of the message, she still needs to be sure that she is really communicating with the bank. This means that she needs to be sure that the public key she is using corresponds to the bank's private key. Similarly, the bank also needs to verify that the message signature really corresponds to Alice's signature.

If each party has a certificate which validates the other's identity, confirms the public key, and is signed by a trusted agency, then they both will be assured that they are communicating with whom they think they are. Such a trusted agency is called a **Certificate Authority**, and certificates are used for authentication.

Certificate Contents

A certificate associates a public key with the real identity of an entity, known as the subject. Information about the subject includes identifying information (the distinguished name), and the public key. It also includes the identification and signature of the Certificate Authority that issued the certificate, and the period of time during which the certificate is valid. It may have additional information (or extensions) as well as administrative information for the Certificate Authority's use, such as a serial number.

Certificate Authorities

By first verifying the information in a certificate request before granting the certificate, the Certificate Authority assures the identity of the private key owner of a key-pair. For instance, if Alice requests a personal certificate, the Certificate Authority must first make sure that Alice really is the person the certificate request claims.

Certificate Chains

A Certificate Authority may also issue a certificate for another Certificate Authority. When examining a certificate, Alice may need to examine the certificate of the issuer, for each parent Certificate Authority, until reaching one which she has confidence in. She may decide to trust only certificates with a limited chain of issuers, to reduce her risk of a "bad" certificate in the chain.

Creating a Root-Level CA Certificate

As noted earlier, each certificate requires an issuer to assert the validity of the identity of the certificate subject, up to the top-level Certificate Authority (CA). This presents a problem: Since this is who vouches for the certificate of the top-level authority, which has no issuer? In this unique case, the certificate is "self-signed", so the issuer of the certificate is the same as the subject. As a result, one must exercise extra care in trusting a **self-signed certificate**. The wide publication of a public key by the root authority reduces the risk in trusting this key -- it would be obvious if someone else publicized a key claiming to be the authority.

Certificate Management

Establishing a Certificate Authority is a responsibility that requires a solid administrative, technical, and management framework. Certificate Authorities not only issue certificates, they also manage them -- that is, they determine how long certificates are valid, they renew them, and they keep lists of certificates that have already been issued but are no longer valid (**Certificate Revocation Lists**, or CRLs). Say Alice is entitled to a certificate as an employee of a company. Say too, that the certificate needs to be revoked when Alice leaves the company. Since certificates are objects that get passed around, it is impossible to tell from the certificate alone that it has been revoked. When examining certificates for validity, therefore, it is necessary to contact the issuing Certificate Authority to check CRLs although this is not usually an automated part of the process.

Secure Sockets Layer (SSL)

The **Secure Sockets Layer** (SSL) protocol is a protocol layer that may be placed between a reliable connection-oriented network layer protocol (e.g. TCP/IP) and the application protocol layer (e.g. HTTP, or IAP). SSL provides for secure communication between client and server by allowing mutual authentication, the use of digital signatures for integrity, and encryption for privacy.

The protocol is designed to support a range of choices for specific algorithms used for cryptography, digests, and signatures. This allows algorithm selection for specific servers to be made based on legal, export or other concerns, and also enables the protocol to take advantage of new algorithms. Choices are negotiated between client and server at the start of establishing a protocol session.

There are a number of versions of the SSL protocol, primarily version 2.0 and 3.0. One of the benefits in SSL 3.0 is that it adds support of certificate chain loading. This feature allows a server to pass a server certificate along with issuer certificates to the browser. Chain loading also permits the browser to validate the server certificate, even if Certificate Authority certificates are not installed for the intermediate issuers, since they are included in the certificate chain. SSL 3.0 is the basis for the Transport Layer Security (TLS) protocol standard, currently in development by the Internet Engineering Task Force (IETF).

Session Establishment

The SSL session is established by following a handshake sequence between client and server. This sequence may vary, depending on whether the server is configured to provide a server certificate or request a client certificate. Though cases exist where additional handshake steps are required for management of cipher information, this article summarizes one common scenario: see the SSL specification for the full range of possibilities.

Once an SSL session has been established it may be reused, thus avoiding the performance penalty of repeating the many steps needed to start a session. For this the server assigns each SSL session a unique session identifier which is cached in the server and which the client can use on forthcoming connections to reduce the handshake (until the session identifier expires in the cache of the server).

The elements of the handshake sequence, as used by the client and server, are listed below:

Negotiate the Cipher Suite to be used during data transfer

Establish and share a session key between client and server

Optionally authenticate the server to the client (required for IA)

Optionally authenticate the client to the server (required for IA)

The first step, Cipher Suite Negotiation, allows the client and server to choose a Cipher Suite supportable by both of them. The SSL3.0 protocol specification defines 31 Cipher Suites. A Cipher Suite is defined by the following components:

Key Exchange Method

Cipher for Data Transfer

Message Digest for creating the Message Authentication Code (MAC)

These three elements are described in the sections that follow.

Key Exchange Method

The key exchange method defines how client and server will agree upon the shared secret symmetric cryptography key used for application data transfer. SSL 2.0 uses RSA key exchange only, while SSL 3.0 supports a choice of key exchange algorithms including the RSA key exchange when certificates are used, and Diffie-Hellman key exchange for exchanging keys without certificates and without prior communication between client and server.

One variable in the choice of key exchange methods is digital signatures -- whether or not to use them, and if so, what kind of signatures to use. Signing with a private key provides assurance against a man-in-the-middle-attack during the information exchange used in generating the shared key.

Cipher for Data Transfer

SSL uses the conventional cryptography algorithm (symmetric cryptography) described earlier for encrypting messages in a session. There are eight choices, including the choice to perform no encryption:

- No encryption (NULL) (supported by IA Protocol)
- Stream Ciphers:
  - RC4 with 40-bit keys
  - RC4 with 128-bit keys
- CBC Block Ciphers
  - RC2 with 40 bit key
  - DES with 40 bit key
  - DES with 54 bit key (supported by IA Protocol)
  - Triple-DES with 168 bit key, and (supported by IA Protocol)
  - Idea™ (128 bit key)

Here **CBC** refers to Cipher Block Chaining, which means that a portion of the previously encrypted cipher text is used in the encryption of the current block. **DES** refers to the Data Encryption Standard, which has a number of variants (including DES40 and 3DES_EDE). DES has recently been supplanted by AES, the new NIST standard. **Idea™** is not supported by IAgent, but is very popular in Europe, and is patented by MediaCrypt AG. **RC2** is a proprietary algorithm from RSA Security®.

Digest Function

The choice of digest function determines how a digest is created from a record unit. SSL supports the following:

No digest (Null choice)
MD5, a 128-bit hash
Secure Hash Algorithm (SHA-1), a 160-bit hash (supported by IA Protocol)

The message digest is used to create a **Message Authentication Code** (MAC) which is encrypted with the message to provide integrity and to prevent against replay attacks.

Handshake Sequence Protocol

The handshake sequence uses three protocols:

The SSL Handshake Protocol
for performing the client and server SSL session establishment,

The SSL Change Cipher Spec Protocol
for actually establishing agreement on the Cipher Suite for the session, and

The SSL Alert Protocol
for conveying SSL error messages between clients and servers.

These protocols, as well as application protocol data, are encapsulated in the SSL Record Protocol. An encapsulated protocol is transferred as data by the lower layer protocol, which does not examine the data. The encapsulated protocol has no knowledge of the underlying protocol.

The encapsulation of SSL control protocols by the record protocol means that if an active session is renegotiated the control protocols will be transmitted securely. If there were no session before, then the Null cipher suite is used, which means there is no encryption and messages have no integrity digests until the session has been established.

Data Transfer

The SSL Record Protocol is used to transfer application and SSL Control data between the client and server, possibly fragmenting this data into smaller units, or combining multiple higher level protocol data messages into single units. It may compress, attach digest signatures, and encrypt these units before transmitting them using the underlying reliable transport protocol (Note: currently all major SSL implementations lack support for compression).

The foregoing has only been an introduction, if you really want to see the protocol details then read the Secure Socket Layer protocol standard:

The SSL Protocol Version 3, November 18, 1996 (Netscape Communications)
http://wp.netscape.com/eng/ssl3/

Part 2 – IAgent Installation

Part 2 of this manual contains a step-by-step description of IAgent product installation, including pre-installation and post-installation tasks.

CHAPTER 4 – INSTALLATION OVERVIEW

The installation and configuration of the IAgent system is rather complex. Essentially the following actions are required:

Generate an RSA key pair (public and private) and Certificate Signing Request (CSR),

Obtain a personalized X.509 certificate (signed by a valid certificate authority),

Install the IAgent product, support utilities and associated products,

Configure IAgent to your local machine and trading partner identity,

Add one or more (remote) trading partners,

Modify and test network connectivity to each trading partner,

Configure and integrate to the appropriate "back-end" systems (BESs)

Required Documentation

This list of documentation may be needed for reference during IAgent installation:

Instructions for using the operating system on your target machine

Documents for your EDI Translator, Operation Support System (OSS), or Gateway products (whatever system will act as the IAgent back-end system / interface)

Instructions for using the `vi` or other text editor (for configuration file modification)

The following three chapters cover all the tasks required to install the IAgent product. Subsequent chapters cover specific tasks in detail.

CHAPTER 5 – PRE-INSTALLATION TASKS

Pre Installation Task Instructions

IAgent pre-installation consists of five tasks:

1. Completing the pre-install worksheet
2. Acquiring root access on the target machine
3. Acquiring the correct software for your machine
4. Requesting the license file
5. Hardware connectivity issues

These tasks should be completed in order, since subsequent tasks may require information from previous tasks. If you encounter any problems with these tasks or instructions, refer to *Chapter 30: How To Get Help*.

1. Completing the pre-install worksheet

See *Appendix A* for the **IAgent Pre-Installation Worksheet** and instructions for completion. Once completed continue to the next task.

2. Acquiring root access on the target machine

Root or super-user access will be required to install the IAgent product on your host machine. No additions of special users or groups are required. A CD-ROM drive is required for standard installation as the IAgent product is supplied on an ISO-9660 format CD-ROM. Please contact Lymeware or your distributor if a different form of media is required.

Also sufficient disk storage of 50 Megabytes should be available on a local (not network, AFS, DFS or NFS) drive. Network drive performance could adversely impact the performance of the installed IAgent system. At this time the security suggestions (see *Chapter 29*) should be addressed and implemented.

3. Acquiring the correct software for your machine

Be sure that the version of the IAgent product is correct both for the hardware and operating system of your target machine. For example, if your target machine is an Intel platform running Sun Solaris 2.6, then the IAgent Solaris product is not appropriate since this product only supports Solaris running on SPARC platforms. If you have any question about which version of IAgent product supports your specific machine and operating system, please contact the Lymeware sales team or your distributor for assistance.

Insure that the target machine has all vendor-suggested patches, packages, updates and/or service packs applied. A full list of known required updates for supported platforms is listed on the Lymeware web site (www.lymeware.com)www.lymeware.com).

Install all optional packages and products (including the Tibco^{tm} TIB/Rendezvous product) at this time.

The Perl scripting language (at least version 5.003) is required for several of the optional utilities, located in `/opt/iagent/tools`. It should be installed at this time.

More information about the Tibco requirements may be found in *Chapter 21*.

4. Requesting the license file.

A specific license data file will be required to run the IAgent system on your target machine. Lymeware or your distributor will supply this license file if a **License Request Form** (see *Appendix A*) is completed and returned to Lymeware or your distributor.

The license file will be delivered to the Contact E-Mail Address within 5 business days.

5. Hardware connectivity issues

Network (TCP/IP) connectivity is required to successfully communicate with other trading partners and IA systems. At this time the **IAgent Connectivity Worksheet** (see *Appendix A*) should be completed. This may require information and assistance of network or LAN personnel.

Note: The IAgent product expects, at a minimum, a direct network (TCP/IP) connection to a peer IA machine, firewall or router and does not support proxy servers or services.

Also at this time any required connectivity installation (of hardware, circuits, and connections) should be completed, provisioned and tested prior to IAgent installation.

CHAPTER 6 – PRODUCT INSTALLATION

The actual installation of the IAgent product should take no more than one hour. The target machine should not be in general use and may have to be re-booted after installation, depending on the target machine's operating system.

Please refer to the platform-specific INSTALL file (`install.txt`) in the IAgent distribution for your specific platform and environment specific installation instructions.

The specific tasks in the INSTALL file should be completed in order, since subsequent tasks may require information from previous tasks. If you encounter any problems with these tasks or instructions, refer to *Chapter 30: How To Get Help*.

NOTE: If you are upgrading from a previous version of IAgent please complete the following steps prior to installation:

1. Shutdown the current IAgent system,
2. Copy all configuration files (and any other files you have modified, including license files) to a new location (not under `/opt/iagent`),
3. Backup all transactions, pending receipts and log files to a new location.

You are now ready to install the latest version "on top of" your last installed version. Please refer to the INSTALL file (`install.txt`) in the IAgent distribution for platform and environment specific update instructions. If the updated version is a newer version number than currently installed (if the update is a patched or bug-fix release it may have the same version number), then a new `license.dat` file (with the correct matching version number) will be required.

CHAPTER 7 – POST-INSTALLATION TASKS

Post Installation Task Instructions

IAgent post-installation consists of five tasks:

1. Installing the license file
2. Installing local certificates and keys
3. Creating the IAgent configuration file
4. Verifying PRNGD installation and configuration
5. Creating the Trading Partners configuration file

These tasks should be completed in order, since subsequent tasks may require information from previous tasks. If you encounter any problems with these tasks or instructions, refer to *Chapter 30: How To Get Help*.

1. Installing the license file.

At this time the supplied license file should be installed. The license file will be delivered to the Contact E-Mail Address as was supplied in the License Request Form. The license file must be installed at `/opt/iagent` as `license.dat` and should be owned by root.

2. Installing local certificates and keys.

At a minimum, a single X.509 Server certificate and private key pair is required to operate the IAgent system. A minimum of the signing CA Certificate will also be required.

Chapter 8 of this manual covers Certificate Signing Request (CSR) and Key generation and installation in detail.

3. Creating the IAgent configuration file

Chapter 9 of this manual covers IAgent configuration management in detail.

4. Verifying PRNGD installation and configuration (Sun/Solaris versions only)

Cryptographic software needs a source of unpredictable data to work correctly. Due to requirements of the underlying cryptography toolkit (OpenSSL) this version of IAgent requires the use of a secure pseudo-random number generator to seed and supply entropy to specific cryptographic functions. Because all target systems do not supply this facility as a standard system service, additional software is provided to perform these services.

This software generates pseudo-random entropy from collecting data and reading logs from several system processes and then digests the data into some form of randomness. Lymeware provides

PRNGD and EGD as solutions to this requirement. **Note**: It is required that one of the two entropy programs is installed, configured, and running prior to starting the IAgent product.

Sun Solaris only
During the Sun Solaris install process (pkgadd) the **prngd** program was installed, configured, and added to the target machine's `/etc/rc3.d` scripts for automatic startup on reboot. Correct operation may be tested by running the `test_prngd.pl` utility found in `/opt/iagent/tools`.

5. Creating the Trading Partners configuration file

Chapter 10 of this manual covers Trading Partner administration in detail.

6. Testing Integration API Connectivity

Chapter 22 of this manual covers Integration API connectivity testing in detail.

7. Testing Trading Partner Connectivity

Chapter 11 of this manual covers Trading Partner connectivity testing in detail.

8. Post Install Security Tasks

Chapter 29 of this manual covers Post Install security tasks and suggestions in detail.

Part 3 – IAgent Configuration

Part 3 of this manual contains detailed configuration information for certificate, key, trading partner and configuration file administration. This section also covers receipt use and issues.

CHAPTER 8 – CERTIFICATE AND KEY GENERATION AND INSTALLATION

Certificate and Key generation and configuration

IAgent certificate and key generation and installation consists of six steps:

1. Selecting a third party Certificate Authority (CA)
2. Completing the certificate request worksheet
3. Creating a Certificate Signing Request file and private key file
4. Sending the Request to your chosen CA.
5. Installing local certificates and keys.
6. Installing local Certificate Authority certificates.

These tasks should be completed in order, since subsequent tasks may require information from previous tasks. If you encounter any problems with these tasks or instructions, refer to *Chapter 30: How To Get Help*.

At a minimum, a single X.509 Server certificate and private key pair is required to operate the IAgent system. The signing CA Certificate from the Certificate Authority which generated your server certificate will also be required.

1. Selecting a third party Certificate Authority (CA)

Selection of a third party Certificate Authority is usually a mutual decision between trading partners. It is suggested that an agreement of a list of acceptable CAs be made prior to any subsequent tasks.

2. Completing the certificate request worksheet

See *Appendix A* for a **Certificate Request Worksheet** and instructions for completion. Once completed continue to the next task.

3. Creating a Certificate Signing Request file and private key file

A certificate signing request (CSR) must be created and sent to your chosen Certificate Authority. Using your completed Certificate Request Worksheet, use the included `gen_csr` utility to create private keys and CSRs.

See *Appendix B* for a certificate overview and a `gen_csr` tutorial.

4. Sending the Request to your chosen CA

How to send the CSR to the Certificate Authority (CA) and what forms of proof of identity will be required depend on the CA chosen. Refer to the CA's support or sales staff for further instructions.

Typically the CA will return the signed certificate via e-mail. Upon receipt of the signed certificate proceed to the next task.

5. Installing local certificates and keys

A directory (`/opt/iagent/client_certs`) is provided to store all your local X.509 certificates. It is assumed that all certificates will contain the Trading Partner Common Name (see *Chapter 10*) as the Certificate Common Name and be in PEM (or base-64) format.

A directory (`/opt/iagent/client_keys`) is provided to store all your local private RSA keys. It is assumed that all private keys will be in PEM (or base-64) format. It is suggested that all private keys be kept in this directory.

6. Installing local Certificate Authority certificates

A directory (`/opt/iagent/ca_certs`) is provided to store all CA X.509 certificates. Typically your signing CA certificate and any CA certificates used by your Trading Partners should be stored here. It is assumed that all certificates will be in PEM (or base-64) format.

CHAPTER 9 – IAGENT SYSTEM CONFIGURATION ADMINISTRATION

The System (or IAgent) configuration file

The system configuration file (`iagent.conf`, by default) contains option and setting entries, which override any system defaults (however, command line arguments will override any configuration file settings!). All IAgent options and settings should be in this file

IAgent configuration consists of setting all of the following types of configuration options:

Basic system-wide configuration options,

Process Specific configuration options,

Certificate and Key location options,

Integration API settings and options, and

Advanced and special options

The configuration options may appear on the command line or appear in the system configuration file (`iagent.conf`). It is recommended that the configuration file be used when operating the IAgent in a production system, and that additional command line options may be used during testing and troubleshooting.

The easiest way to build a system configuration file is to make a copy from the example supplied in the `/opt/iagent/examples` directory. Copy this example file to `/opt/iagent` as a new file called `iagent.conf`. Now open the new file with your text editor of choice. We will now tackle each group of configuration options available in the following sections.

Basic system-wide configuration options

These options set the IAgent configuration file and Trading Partner file names, the integration interface to support and the mode in which to run the IAgent system.

```
cachesize = ENTRIES
             Server session cache size (in session entries).

receiptmode = MODE
             Receipt mode supported. May be either internal or external.

config = FILENAME
             IAgent configuration file name - default is iagent.conf.

interface = TYPE
             Input (client) or output (server) interface supported. May be
              any of the following: file, dir, pipe, socket, or (optionally)
              tibco.

mode = MODE
             IAgent mode: either F (full), S (standalone server only)
              or C (standalone client only).
```

```
TPconfig = FILENAME
             IAgent trading partner configuration file name -
              default is tpartner.conf.
```

Process Specific configuration options

These options set the IAgent Server ports, (both standard and hi priority ports).

```
hipriport = PORT
             High priority server port (for Hipri Server process only).

stdport = PORT, or port = PORT
             Standard server port (for Std Server process only).
```

Certificate and Key location options

These options describe the location of all certificates and private keys in the IAgent system.

```
key = FILENAME
             Own local private key file (in PEM format).

cert = FILENAME
             Own local certificate file (in PEM format).

CApath = PATH
             Path to Certificate Authority (CA) certificates (in PEM format).

CAfile = FILENAME
             Certificate Authority (CA) file (in PEM format).
```

Integration API settings and options

These options describe the setting for the Integration API Interface used (and set) above. All other options (for other Integration API Interfaces) will be ignored, if they exist.

File Interface Options

```
file = FILENAME
             Input file name (for File interface only).
```

Directory Interface Options

```
dirpolldelay = SECONDS
             Client directory poll delay in seconds
              (for Directory interface only).

indir = PATH
             Input directory name (for Directory interface only).
```

```
outdir = PATH
             Output directory name (for Directory interface only).
```

Named Pipe Options

```
inpipe = PIPENAME
             Input pipe name (for Pipe interface only).

outpipe = PIPENAME
             Output pipe name (for Pipe interface only).
```

Socket Interface Options

```
insocket = SOCKETPORT
             Input socket port (for Socket interface only).

outsocket = SOCKETADDRESS:PORT
             Output socket address and port (for Socket interface only).
```

Optional Tibco Interface Options

```
intibco = MESSAGE_NAME
             Input tibco message subject name (for Tibco interface only).

outtibco = MESSAGE_NAME
             Output tibco message subject name (for Tibco interface only).
```

Advanced and special options

These options are not required for standard operation of your IAgent system and are usually used only during testing.

```
sloppyallow = 0/1
             Flag to allow receipt of any type of messages from
              active trading partners (for testing purposes only).

sloppyinactive = 0/1
             Flag to allow receipt of message from inactive trading
              partners (for testing purposes only).

ignoredupreceipts = 0/1
             Flag to allow duplicate receipts without triggering errors
              (for testing purposes only).

loopbackdelay = SECONDS
             Debug loopback delay in seconds (for testing purposes only).

debug = DEPTH, or debuglevel = DEPTH
             Debug display depth (for testing purposes only).
```

```
entropy = FILENAME
             Entropy source (file) required for cryptographic calculations.
             For Solaris product versions only.

vhack = 0/1
             Self-signed issuer certificate verify hack flag.

sendstatus = 0/1
             Flag to force automatic sending of IAstatus messages when
              internal errors occur.

verbose = 0/1
             Flag to force verbose (more messages to display and logs).

oneshot = 0/1
             Flag to force exit after a single IA message is sent or received.
              For debug only.

raw = 0/1
             Flag to force sending of raw IA EDI messages.  For debug only.

packetmode = 0/1
             Flag to force sending of IA packet-mode messages only.
              For debug only.

savereceipts = 0/1
             Flag to force storage of all inbound and outbound receipts.
```

An example of a directory interface configuration file and a socket interface configuration file may be found in *Appendix C*.

IAgent command line arguments

The second way to modify the way the IAgent system operates is with command line arguments. These are usually only used during testing and are not used in the automatic startup during system boot (see *Chapter 25* for more information on IAgent startup options).

The following table lists all the available IAgent command line arguments. Each argument may also appear in the system configuration file (`iagent.conf`) , but will be overridden by any identical argument on the command line.

```
-A, --sloppyallow
             Flag to allow receipt of any type of messages from
              active trading partners (for testing purposes only).

-B, --sloppyinactive
             Flag to allow receipt of message from inactive trading
              partners (for testing purposes only).

-C ENTRIES, --cachesize=ENTRIES
             Server session cache size (in session entries).

-D PATH, --outdir=PATH
             Output directory name (for Directory interface only).

-F FILENAME, --file=FILENAME
             Input or output file name (for File interface only).

-I, --ignoredupreceipts
             Flag to allow duplicate receipts without triggering errors
              (for testing purposes only).

-L SECONDS, --loopbackdelay=SECONDS
             Debug loopback delay in seconds (for testing purposes only).

-O, --oneshot
             Flag to force processing of only a single transaction.

-P PIPENAME, --outpipe=PIPENAME
             Output pipe name (for Pipe interface only).

-R MODE, --receiptmode=MODE
             Receipt mode supported. May be either internal or external.

-S SOCKETADDRESS:PORT, --outsocket=SOCKETADDRESS:PORT
             Output socket address and port (for Socket interface only).
```

Table 4. IAgent Command Line Arguments

```
-T MESSAGE_NAME, --outtibco=MESSAGE_NAME
             Output tibco message subject name (for Tibco interface only).

-V, --version
             Prints the version of IAgent - then exits.

-Y, --savereceipts
             Flag to force storage of all inbound and outbound receipts.

-Z, --packetmode
             Forces IA Input packet format support only.

-a PATH, --indir=PATH
             Input directory name (for Directory interface only).

-b PIPENAME, --inpipe=PIPENAME
             Input pipe name (for Pipe interface only).

-c FILENAME, --config=FILENAME, --config-file=FILENAME
             IAgent configuration file name - default is iagent.conf.

-d DEPTH, --debug=DEPTH, --debuglevel=DEPTH
             Debug display depth (for testing purposes only).

-e FILENAME, --entropy=FILENAME
             Entropy source (file) required for cryptographic calculations.

-g SOCKETPORT, --insocket=SOCKETPORT
             Input socket port (for Socket interface only).

-h PORT, --hipriport=PORT
             High priority server port (for Hipri Server process only).

-i TYPE, --interface=TYPE
             Input (client) or output (server) interface supported. May be
              any of the following: file, dir, pipe, socket, or tibco.

-j MESSAGE_NAME, --intibco=MESSAGE_NAME
             Input tibco message subject name (for Tibco interface only).

-k FILENAME, --key=FILENAME, --key-file=FILENAME
             Own local private key file (in PEM format).

-m MODE, --mode=MODE
             IAgent mode: either F (full), S (standalone server only)
              or C (standalone client only).
```

Table 5. IAgent Command Line Arguments (continued)

```
-n SECONDS, --dirpolldelay=SECONDS
             Client directory poll delay in seconds
              (for Directory interface only).

-p PORT, --stdport=PORT, --port=PORT
             Standard server port (for Std Server process only).

-q, --vhack  Self-signed issuer certificate verify hack flag.

-r, --raw    Flag to force RAW text (not ASN1 message) transfer.

-t FILENAME, --TPconfig=FILENAME, --tpartner=FILENAME
             IAgent trading partner configuration file name -
              default is tpartner.conf.

-u, --usage, --help
             Displays this usage page.

-v, --verbose
             Flag to force A LOT of logging to stdout and log files.

-w, --sendstatus
             Flag to force automatic sending of IAstatus messages when
              internal errors occur.

-x FILENAME, --cert=FILENAME, --cert-file=FILENAME
             Own local certificate file (in PEM format).

-y PATH, --CApath=PATH
             Path to Certificate Authority (CA) certificates (in PEM format).

-z FILENAME, --CAfile=FILENAME
             Certificate Authority (CA) file (in PEM format).
```

Table 6. IAgent Command Line Arguments (concluded)

Default IAgent configuration values and settings

The IAgent product uses default values for most of the IAgent configuration file and command line arguments. The following table lists all current default configuration values and settings, all of which can be overridden with either configuration file values or command line values.

```
IAgent configuration option Default display
sloppyallow  =(off)
sloppyinactive  =(off)
cachesize = ENTRIES =(128)
outdir = PATH =(out_dir)
file = FILENAME =(test_file)
ignoredupreceipts  =(off)
loopbackdelay = SECONDS =(0)
oneshot  =(off)
receiptmode = MODE =(internal)
outsocket = SOCKETADDRESS:PORT =(127.0.0.1)
outtibco = MESSAGE_NAME =(iagent.tibco.server.message.version1.1)
version  =(off)
savereceipts  =(off)
packetmode  =(off)
indir = PATH =(in_dir)
config , config-file = FILENAME =(iagent.conf)
debug , debuglevel = DEPTH =(0)
entropy = FILENAME =(/tmp/entropy)
insocket = SOCKETPORT =(6500)
hipriport = PORT =(6998)
interface = TYPE =(dir)
intibco = MESSAGE_NAME =(iagent.tibco.client.message.version1.1)
key , key-file = FILENAME =(own_key.pem)
mode = MODE =(F)
dirpolldelay = SECONDS =(5)
stdport , port = PORT =(6999)
vhack  =(off)
raw  =(off)
TPconfig , tpartner = FILENAME =(tpartner.conf)
usage , help  =(off)
verbose  =(off)
sendstatus  =(off)
cert , cert-file = FILENAME =(own_cert.pem)
CApath = PATH =(./ca_certs)
CAfile = FILENAME =(ca_cert.pem) (CFG_DEFAULT)CHAR: mode = S
```

Table 7. Default IAgent configuration values

CHAPTER 10 – TRADING PARTNER ADMINISTRATION

The IAgent system communicates only with other (remote) IA systems that have a trading partner identity defined within the IAgent system. An identity is defined when a matching trading partner record exists. A trading partner record consists of the following:

The Trading Partner ID,
The Trading Partner Common Name (CN),
The Trading Partner's server IP address,
The Trading Partner's standard port,
The Trading Partner's hi priority port,
The Trading Partner's SSL session cache timeout,
The Trading Partner's sending IA message type,
The Trading Partner's receiving IA message type,
The Trading Partner's sending IA receipt type,
The Trading Partner's receiving IA receipt type,
The Trading Partner's IA receipt timeout,
and the status of the Trading Partner server.

It is expected that the local machine will also have a trading partner entry, so, at a minimum, the trading partner configuration file should have two entries (one local and one remote).

The easiest way to build a trading partner configuration file is to make a copy from the example supplied in the `/opt/iagent/examples` directory. Copy this example file to `/opt/iagent` as a new file called `tpartners.conf`. Now open the new file with your text editor of choice. We will now tackle trading partner maintenance in the following sections.

Trading Partner Maintenance

There are only three actions for trading partner maintenance:

Addition of a new trading partner,

Removal of a current trading partner, and

Modification of a current trading partner.

Each of these actions is described below. It is expected that all of these actions will be made against the trading partner configuration file (usually `tpartners.conf`) and will be activated upon the re-starting of the IAgent system.

Addition of a new trading partner

Adding a new trading partner is very straightforward and consists of only four steps:

1. Complete a **Trading Partner Configuration Worksheet**, as found in *Appendix A*,
2. Copy a new section, as found in the example file to the end of your trading partner file,

3. Add the new Trading Partner id to the **trading_partners** variable**.**
   Note: this is a comma delimited list of all trading partners in your IAgent system.
4. Modify the new section to reflect the values of the **Trading Partner Configuration Worksheet**.

Removal of a current trading partner

Removing a current trading partner is trivial and consists of only two steps:

1. Remove the current Trading Partner id from the **trading_partners** variable**.**
   Note: this is a comma delimited list of all trading partners in your IAgent system.
2. Delete the section for this trading partner, as found in your current trading partner file.

Modification of a current trading partner

Using a text editor, edit the trading partner file (`tpartner.conf`, by default) and modify the setting for each trading partner as required. When finished the IAgent system must be stopped and re-started for the changes to take effect. See below for a description of each field.

The Trading Partner configuration file

The Trading Partner configuration file (by default `tpartner.conf`) consists of several entries per trading partner and is processed prior to any SSL communications. All trading partners to be supported in the IAgent system (clients and/or servers) will require an entry in the configuration file. An entry for the local trading partner is also required.

Each of the trading partner entries will need to have a separate section in the configuration file (as shown below). Each trading partner section should contain the following fields:

Configuration Field	**Description**
partner_id	This is a internal trading partner identifier (and does not reflect any EDI value)
local_partner_id	This is the local trading partner identifier as found in the EDI ISA05 or ISA06 segment, 15 characters maximum as per ANSI X.12
remote_partner_id	This is the remote trading partner identifier as found in the EDI ISA05 or ISA06 segment, 15 characters maximum as per ANSI X.12
certificate_common_name	This is the CommonName (CN) as found in the Client Certificate sent from the remote Trading Partner
ip_address	The trading partner's IP address as a string (AAA.BBB.CCC.DDD)

Table 8. Trading Partner Configuration Fields

standard_ip_port	The trading partner's standard IA server IP port address (on same IP address as above)
hipri_ip_port	The trading partner's high priority IA server IP port address (on same IP address as above)
session_timeout	Connection session cache timeout in seconds. If 0 then no resumable sessions supported for this trading partner.
ia_message_from_tp	The IA EDI Message Type supported from this Trading Partner. May be any of the following: **BASIC_EDI –** Basic EDI message **INTEGRITY_EDI –** Integrity (with digest) EDI message **SIGNED_EDI –** Signed (with digital signature) EDI message
ia_message_to_tp	The IA EDI Message Type to send to this Trading Partner. May be any of the following: **BASIC_EDI –** Basic EDI message **INTEGRITY_EDI –** Integrity (with digest) EDI message **SIGNED_EDI –** Signed (with digital signature) EDI message
ia_receipt_from_tp	The IA Receipt Type supported from this trading partner. May be any of the following: **NO_RECEIPT -** No receipt required or supported **BASIC_RECEIPT –** Basic receipt required **INTEGRITY_RECEIPT –** Integrity receipt (with digest) required **SIGNED_RECEIPT –** Signed receipt (with digital signature) required Note: the **SIGNED_RECEIPT** receipt type is only supported for a Signed EDI message type.
ia_receipt_to_tp	The IA Receipt Type to send to this trading partner. May be any of the following: **NO_RECEIPT -** No receipt required or supported **BASIC_RECEIPT –** Basic receipt required **INTEGRITY_RECEIPT –** Integrity receipt (with digest) required **SIGNED_RECEIPT –** Signed receipt (with digital signature) required Note: the **SIGNED_RECEIPT** receipt type is only supported for a Signed EDI message type.
receipt_timeout	The time to wait (in seconds) for a receipt response before logging an error and removing pending receipt entry. If 0 then no timeout. This will only work if the ReceiptMode is set to internal.
server_status	The initial state of this trading partner's IA server. 0 = not operational 1 = active

Table 9. Trading Partner Configuration Fields (continued)

An example of a standard trading partner configuration file may be found in *Appendix C*.

CHAPTER 11 – TRADING PARTNER CONNECTIVITY

It is required that the IAgent machine has a permanent TCP/IP connection to each trading partner machine it will communicate with. The basic options for these connections are:

A Direct Connection (same LAN or MAN or WAN),

A Public Connection (e.g. the Internet, or other public networks) without a Firewall,

A Public Connection (e.g. the Internet, or other public networks) with Firewall,

A Private Connection (Frame Relay or ATM) without Firewall, and

A Private Connection (Frame Relay or ATM) with Firewall.

This list is organized in from a least desirable but easiest to implement to a most desirable and hardest to implement.

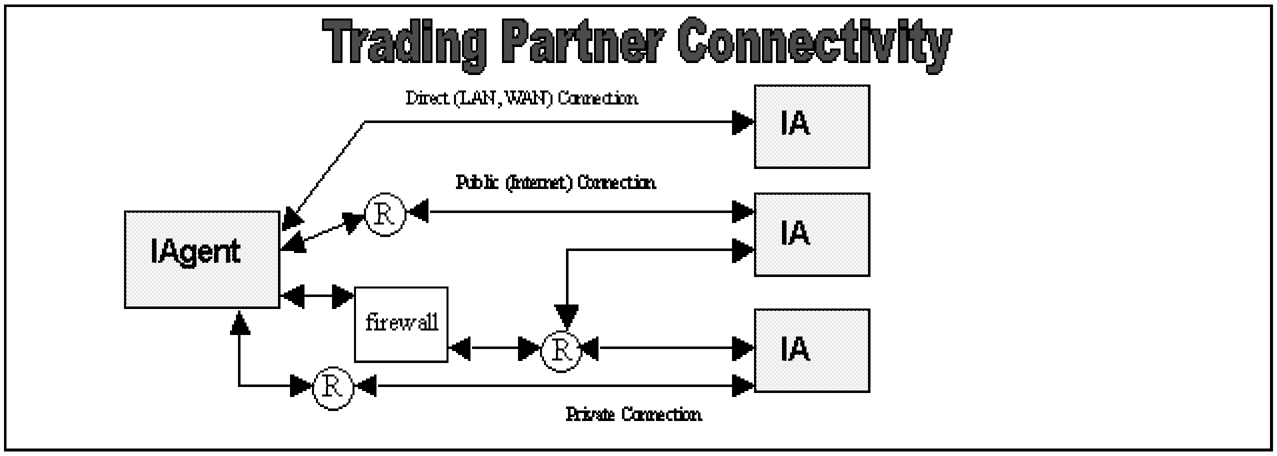

Trading Partner Connectivity
Direct (LAN, WAN) Connection
Public (Internet) Connection
IA
IA
IA
IAgent
R
firewall
R
R
Private Connection

Figure 5. Trading Partner Connectivity

Every element in the TCP/IP pipeline must be tested and direct connectivity (which can be tested with `telnet` (see below) must exist prior to any IA connectivity testing.

Simple Telnet Testing

To ensure bi-directional TCP/IP testing the `telnet` utility (supplied with all supported platform operating systems) can be used. The following must be performed by each trading partner (local and remote) to ensure TCP/IP connectivity and proper routing.

1.Find the remote trading partner's server IP address (from the Trading Partner Worksheet), for this example we will use 191.168.1.1.

2.Find the remote trading partner's standard priority port (also from the Trading Partner Worksheet), we will use the IA default of 6999.

3.Log in to your local machine as root.

4. telnet to the remote trading partner's machine with the following command:

```
telnet <IP address> <port>
```

so in our example the command line would look like this:

```
telnet 191.168.1.1 6999
```

5. At this point one of two things will happen. The connection will either be allowed or rejected. If rejected then an error message similar to this will be displayed:

```
telnet: Unable to connect to remote host: Connection refused
```

Otherwise a connection message will be displayed, which means that TCP/IP connectivity does exist. To exit telnet type **Ctrl-]** (the control key and the right square bracket key) and at the telnet prompt type **exit**.

6. Repeat Steps 3 through 5 using the High Priority port from the remote trading partner.

Simple TCP/IP connectivity should now be established.

Several reasons may be the cause of the `Connection refused` message. They include incorrect IP address and port numbers, invalid firewall rule sets, bad routing (which could be caused by a host of sources). It is usually a good idea to have a network administrator debug the problem from each "side" of the connection to repair the required connectivity.

See *Appendix D* for a sample Trading Partner connectivity test plan.

See *Chapter 29* for a sample Firewall rules set.

CHAPTER 12 – MULTIPLE TRADING PARTNER ISSUES

Multiple trading partners are both supported and encouraged in the IAgent system with the following caveats:

Remote Trading Partners may (but are not required to):

Have different port assignments,

Have different IP Addresses,

Have different message and receipt settings,

Have different Client Certificates,

Have different signing CA Certificates.

Each Remote Trading Partner must:

Have different trading partner identifiers (in both the ISA segment of EDI messages, and in the trading partner file),

Support the same single Client certificate and signing CA certificate from the local IAgent system.

Support the same trading partner identity used by the local IAgent system. **Note**: An IAgent system only supports a **single** local trading partner. Additional IAgent products (typically on other machines) will be required to support multiple local trading partners.

A **Trading Partner Worksheet** should be completed for the local machine. A copy of this worksheet should be provided to all remote trading partners, prior to connectivity testing.

These issues should be agreed to and captured by both trading partners in a Joint Implementation Agreement (JIA), a sample of which is available from TCIF through the ATIS web site (`www.atis.org`).

CHAPTER 13 – MESSAGE RECEIPT OVERVIEW

Message receipts allow trading partners to receive positive acknowledgements of receipt of a specific message.

Trading partners must agree both to support receipts (this is optional, and not required), and the version of receipt supported (basic, digest, or signed). This information should then be stored in the trading partner record in the trading partner configuration file.

The basic data flow of a message with receipt acknowledgement is as follows:

Trading Partner #1 sends a basic/digest/signed IA EDI message to Trading Partner #2

Trading Partner #2 receives the IA EDI message

Trading Partner #2 sends a basic/digest/signed IA Receipt message to Trading Partner #1

Trading Partner #1 receives the IA Receipt message

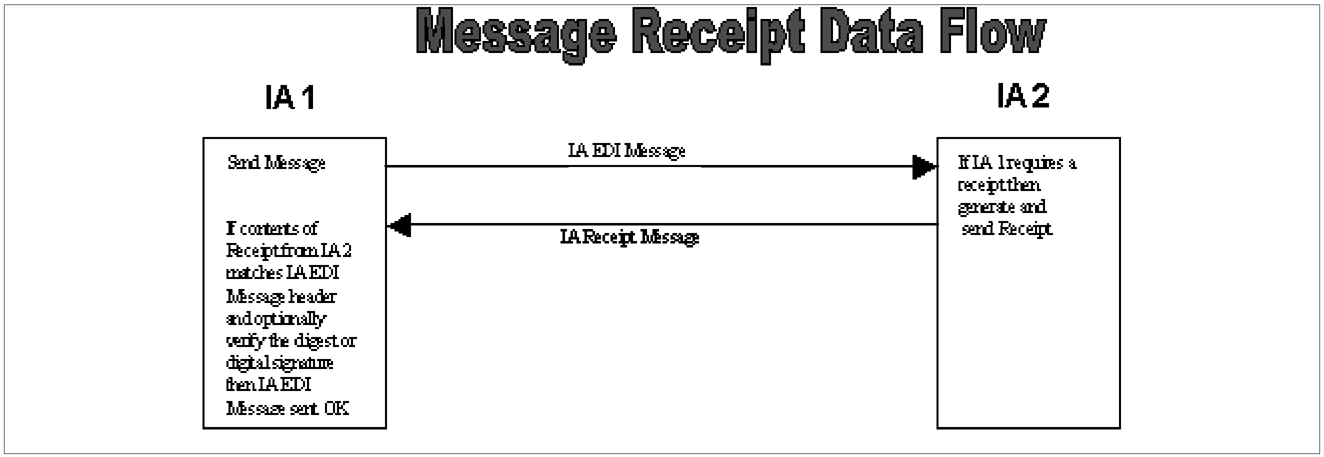

Message Receipt Data Flow
IA 1
IA 2
Send Message
IA EDI Message
If IA 1 requires a receipt then generate and send Receipt
IA Receipt Message
If contents of Receipt from IA 2 matches IA EDI Message header and optionally verify the digest or digital signature then IA EDI Message sent OK

Figure 6. Message Receipt Data Flow

There are currently four versions of IA receipts supported by the IAgent system. They are:

- **No receipt** – No receipt is sent to acknowledge an IA EDI message from a remote trading partner. No receipt is expected or allowed for IA EDI messages sent to the remote trading partner.
- **Basic Receipt** (SSL encryption only) – A basic receipt (which consists of the ISA segment of the EDI message and a timestamp) will be sent to acknowledge an IA EDI message from a remote trading partner. A basic receipt is expected and required for each IA EDI message sent to the remote trading partner.
- **Integrity Receipt** (SSL encryption and digest) - An integrity receipt (which consists of the ISA segment of the EDI message, a timestamp, and a message digest) will be sent to

acknowledge an IA EDI message from a remote trading partner. An integrity receipt is expected and required for each IA EDI message sent to the remote trading partner.

- **Signed Receipt** (includes SSL encryption and signed message integrity) - A signed receipt (which consists of the ISA segment of the EDI message, a timestamp, and a digital signature) will be sent to acknowledge an IA EDI message from a remote trading partner. A signed receipt is expected and required for each IA EDI message sent to the remote trading partner.

CHAPTER 14 – MESSAGE RECEIPT ISSUES

One of the requirements of receipt processing is the unique nature of EDI messages, and the ISA segment in particular. If for any reason (including re-send or duplicate sending) any message is transmitted more than once with the IAgent system, a non-fatal error of DUPLICATE PENDING RECORD FOUND will be reported. This is due to the fact that a pending receipt record already exists. The second message will not generate a pending receipt record.

IAgent allows a third-party or external process to send an external receipt message to the remote trading partner through the IAgent system (see the receipt message format in *Chapter 16*). These receipt messages will not create a pending receipt record and therefore will not be able to timeout or alert the user if no matching initial message exists.

Receipt Modes

IAgent supports two receipt modes, internal and external. Only a single mode will be used for all trading partner communication.

Internal Receipt Mode

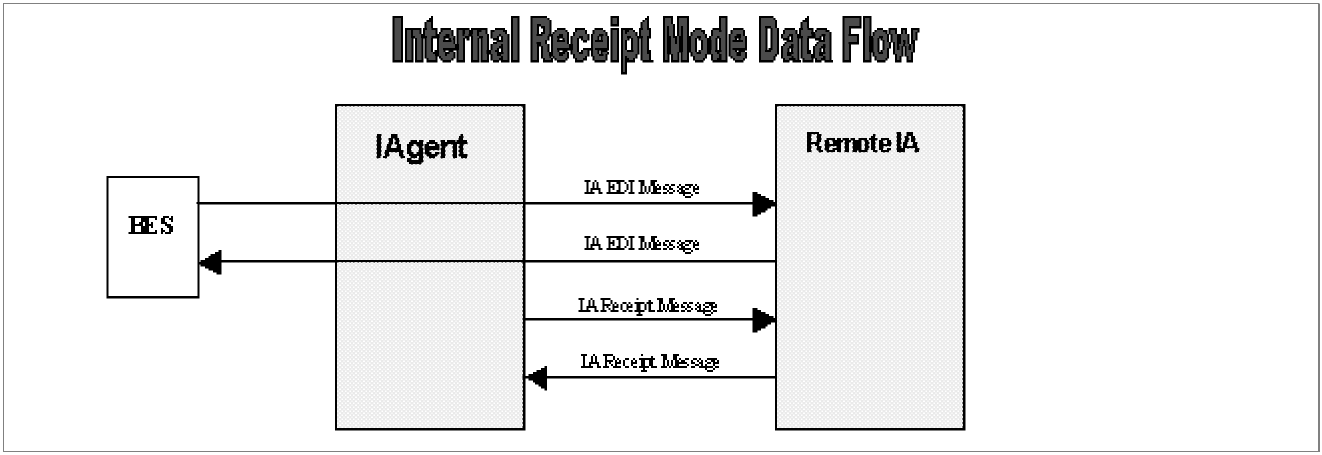


Figure 7. Internal Receipt Mode Data Flow

The Internal Receipt Mode is as described in the TCIF standard, and can be though of as the automatic mode. The IAgent system will generate the appropriate pending receipt for each outbound IA EDI message and will send the appropriate receipt back for each IA EDI message it receives. IA EDI messages sent which do not receive an acknowledging receipt within the trading partner specific receipt timeout period will be determined to be undeliverable and logged as such.

External Receipt Mode

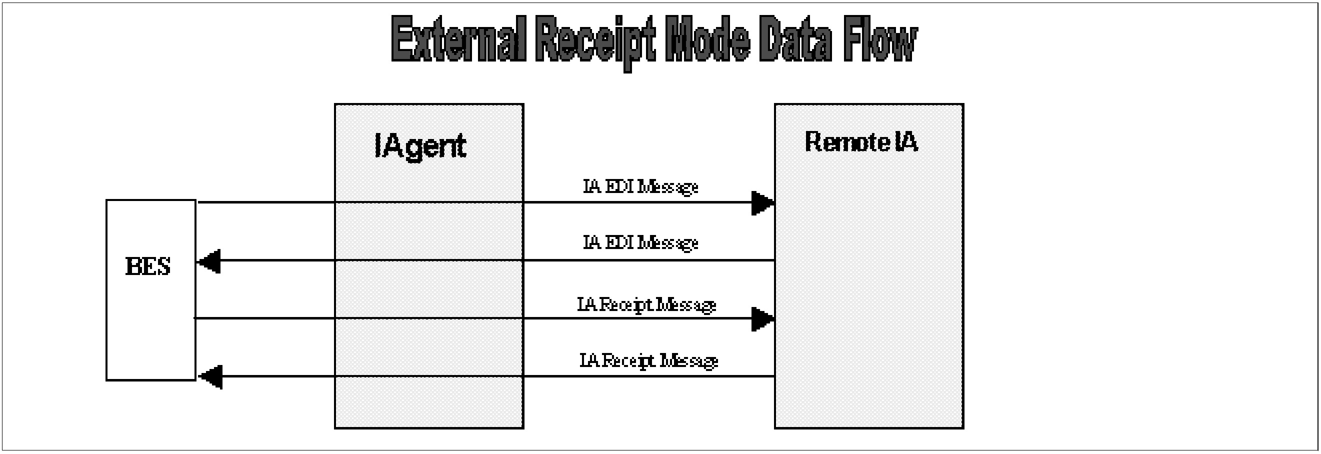


Figure 8. External Receipt Mode Data Flow

The External Receipt Mode allows an external program or process (not supplied) to generate the response receipt (in IAgent IAR format, see *Chapter 16*) and have the IAgent system deliver it to the correct trading partner. This could be thought of as manual mode. The IAgent system will send the input receipt to the correct trading partner. The external program or process can find any response receipts (from remote trading partners) in the output/sever API location (using the same interface as the standard server). No receipt generation or receipt timeout support is provided in this mode.

Part 4 – Integration API Reference

Part 4 of this manual contains an overview of the integration APIs and then specific details for each API's use and message format. This section also includes sample code fragments and a outline for testing integration with "back-end" systems.

CHAPTER 15 – INTEGRATION API OVERVIEW

The IAgent system supports several different possible interface modes (via the Integration APIs), for both input and output. The four supported integration APIs (for internal message input and output transfer) include: the File/Directory interface, the Named Pipe interface, the IP Socket interface and the Tibco^{tm} TIB/Rendezvous message bus interface.

These interfaces are read from and written to by a third-party program or system, which we call the Back-end system (BES). The BES can be anything from a standard EDI translator, to a Gateway product, to a full Operational Support System (OSS). The IAgent just treats all data from the BES as message to send to trading partners and all data from trading partners as messages to send to the BES.

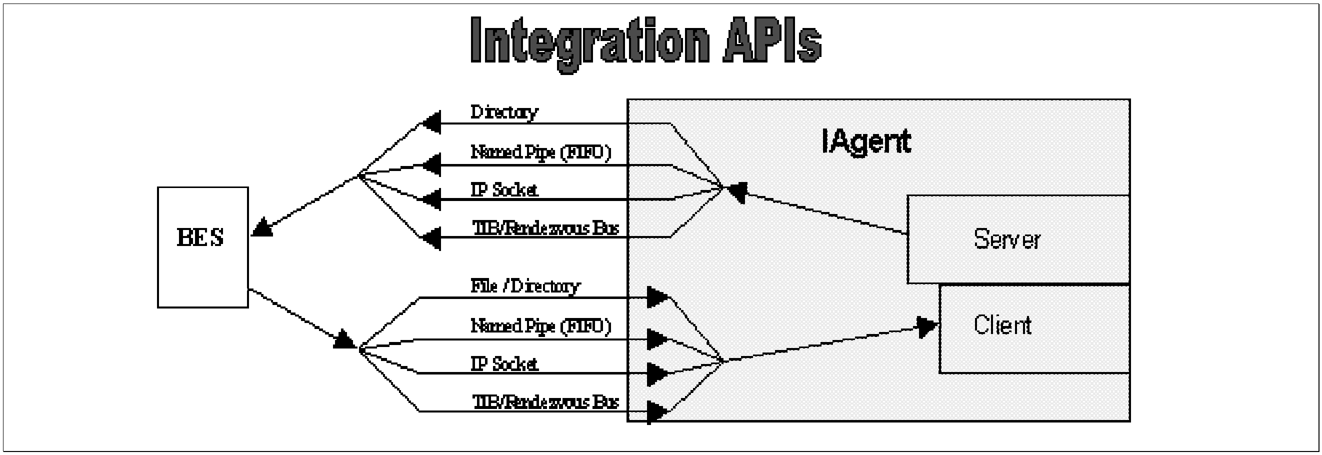


Figure 9. Backend System to Integration API Interfaces

- The File/Directory interface atomically reads and writes ASCII text files (EDI messages in standard ASC X.12 formats).
- The Named Pipe interface reads from and writes to specific user defined named pipes (or FIFOs).
- The IP Socket interface reads from a single user defined input socket and writes to a single output socket. The BES processes reading and writing to this interface do not have to reside on the same platform as the IAgent system.
- The (optional) Tibco^{tm} TIB/Rendezvous message bus interface reads input Tibco messages with a single subscriber and writes output Tibco messages with multiple publishers, supporting multiple platforms.

Mode Restrictions

Note that not all Integration API interfaces are supported by each of the three system modes (Combined or full, Standalone Client or Standalone Server). The following table lists the restrictions.

Mode	Interfaces Supported
Standalone Client	File, Directory, Named Pipe (FIFO), Socket, Tibco
Standalone Server	Directory, Named Pipe (FIFO), Socket, Tibco
Combined or Full	Directory, Named Pipe (FIFO), Socket, Tibco

Table 10 Interfaces supported by mode

Back End System (BES) Locations by Interface

The location of the back end system (BES) may be limited by the selected interface. The following table lists the limitations.

Interface	BES Locations Supported
File, Directory, Named Pipe (FIFO)	Same machine as IAgent product Note: network or network mounded drives are not recommended for working directories or pipes.
Socket	Any IP connected machine.
Tibco	Any IP connected machine on same subnet Note: the use of Tibco's `rvrd` is not recommended.

Table 11. BES Locations supported by Interface

CHAPTER 16 – OUTBOUND (CLIENT) FORMATS

The IAgent standard client process expects three possible input message formats (regardless of input mode). They are: EDI message, Receipt message, and IA Status message formats. A description and example of each is included below.

Input EDI message format

The IAgent client process expects to find an ANSI X.12 ISA segment as the first 105 bytes of the message (actually the remainder of the message need not be in X.12 EDI format). The client process will aggressively check for the ISA segment identifier, and the sender and receiver ID (up to 15 characters). Other fields are not checked or verified. However, both X.12 and the IAgent expect the ISA segment (and all its fields) to be mandatory (even if empty) and of fixed length. We suggest using a newline (ASCII 0x0a) as a segment terminator (makes viewing data easier for the rest of us). The IAgent client will use the appropriate Trading Partner setting to determine the IA EDI destination message format (Basic, Integrity, or Signed).and convert the input EDI message to the correct format.

A typical input EDI message is as follows (lines NOT wrapped)

```
ISA^00^IAGENT    ^00^TEST      ^ZZ^TP_1           ^ZZ^TP_TWO
^990629^1248^U^00303^000000001^0^T^>
GS^PO^CLECAPP^LWC^990629^1248^000001^X^003030
ST^864^000000001
BMG^28
DTM^097^990629^124849^21^19
MIT^000000001
MSG^This is a test message from TP_ONE to TP_TWO over IAgent
MSG^This is a test message line 2
MSG^This is a test message line 3
MSG^This is a test message line 4
MSG^This is a test message line 5
MSG^This is a test message line 6
MSG^This is a test message line 7
MSG^This is a test message line 8
MSG^This is a test message line 9
MSG^This is a test message line 10
MSG^This is a test message line 11
MSG^This is a test message line 12
MSG^This is a test message line 13
MSG^This is a test message line 14
MSG^This is a test message line 15
MSG^This is a test message line 16
MSG^This is a test message line 17
MSG^This is a test message line 18
MSG^This is a test message line 19
CTT^0
SE^25^000000001
GE^1^000001
IEA^1^000000001
```

Example 1. An example of an EDI Input message

A typical ISA segment is as follows (line NOT wrapped)
Note: the ISA segment consists of fixed size fields with a sum of 105 bytes.

```
ISA^00^IAGENT    ^00^TEST      ^ZZ^LWC7           ^ZZ^LWC_TEST
^990629^1248^U^00307^000002317^0^T^>
```

Example 2. An example of a typical ISA EDI Segment

Receipt input message format

The IAgent client process expects to find an "IAR:" as the first 4 bytes of the receipt message, followed by the entire ISA segment from the original EDI message (105 bytes), then a single field separator ":" (a colon) followed by a fixed length timestamp (15 bytes, in yyyymmddhhmmssz format). If the receipt is of type Integrity, or Signed then another field separator ":" (a colon) will follow the timestamp, followed by a field identifier of "I" for integrity (or message digest) or "S" for signed (or digital signature) receipts. A field separator must follow the field identifier ":" then either the digest or the digital signature (in hexadecimal format). The sender ID in the ISA segment determines the Destination Trading Partner. The IAgent client will use the appropriate Trading Partner setting to determine the IA Receipt message destination format to convert the input message into (Basic, Integrity, or Signed).

A typical Basic Receipt input message is as follows (line NOT wrapped)

```
IAR:ISA^00^IAGENT    ^00^TEST      ^ZZ^LWC_TEST       ^ZZ^LWC_7
^990629^1621^U^00307^000002466^0^T^>:19990630120000Z
```

Example 3. An example of a Basic Receipt input message

A typical Integrity Receipt input message is as follows (line NOT wrapped)

```
IAR:ISA^00^IAGENT    ^00^TEST      ^ZZ^LWC_TEST       ^ZZ^LWC_7
^990629^1621^U^00307^000002466^0^T^>:19990630120000Z:I:542ac34623f098790
ed987698b0a62f1436c91a7
```

Example 4. An example of an Integrity Receipt input message

A typical Signed Receipt input message is as follows (line NOT wrapped and incomplete)

```
IAR:ISA^00^IAGENT    ^00^TEST      ^ZZ^LWC_TEST       ^ZZ^LWC_7
^990629^1621^U^00307^000002466^0^T^>:19990630120000Z:S:542ac34623f098790
ed987698b0f743c6a06591ca410027ed7f7f . . .
```

Example 5. An example of a Signed Receipt input message

IA Status input message format

The IAgent client process identifies a message starting with "IAS:" as an IA Status message. There are two types of IA Status input messages: Defined, and Custom. Defined status messages are as defined in the IA Specification (test, stop-WAN and stop-OSS is how we name them). Custom status messages are not specifically defined and will actually be treated as a pass-through by the receiving IAgent server with the bit-string being ignored. It is expected that some other back end process will handle them.

The Defined status input message format is:

IAS: *trading partner ID***:** *command*

with *command* being any one of the following:

TEST	Special Test Message (will be logged, but otherwise ignored)
STOP_WAN	Problem with inbound WAN, peer should stop transmission.
STOP_OSS	Problem with downstream OSS, peer should stop transmission
PING_REQUEST	Request "ping" from the remote trading partner

A typical Defined Status input message is as follows:

```
     IAS:LWC_TEST       :STOP_WAN
```

Example 6. An example of a Defined status input message

The Custom status input message format is:

IAS:*trading partner ID***:***custom bit string (32 bits)*

A typical Custom status input message is as follows:

```
     IAS:LWC_7          :00000000110011000000000000000000
```

Example 7. An example of a Custom status input message

CHAPTER 17 – INBOUND (SERVER) FORMATS

The IAgent server processes generate three possible output message formats (regardless of output mode). They are EDI message, Receipt message, and IA Status message formats. A description and example of each is included below.

EDI Output message formats

The IAgent server processes generate a raw ASCII text EDI (ANSI X.12) message with a valid ISA segment as the first 105 bytes of the message (actually the remainder of the message need not be in X.12 EDI format). The message will use a newline (ASCII 0x0a) as a segment terminator.

A typical ISA segment is as follows (line NOT wrapped)

```
ISA^00^IAGENT    ^00^TEST      ^ZZ^LWC7           ^ZZ^LWC_TEST
^990629^1248^U^00307^000002317^0^T^>
```

Example 8. A typical ISA EDI segment

A typical output EDI message is as follows (lines NOT wrapped)

```
ISA^00^IAGENT    ^00^TEST      ^ZZ^TP_1           ^ZZ^TP_TWO
^990629^1248^U^00303^000000001^0^T^>
GS^PO^CLECAPP^LWC^990629^1248^000001^X^003030
ST^864^000000001
BMG^28
DTM^097^990629^124849^21^19
MIT^000000001
MSG^This is a test message from TP_ONE to TP_TWO over IAgent
MSG^This is a test message line 2
MSG^This is a test message line 3
MSG^This is a test message line 4
MSG^This is a test message line 5
MSG^This is a test message line 6
MSG^This is a test message line 7
MSG^This is a test message line 8
MSG^This is a test message line 9
MSG^This is a test message line 10
MSG^This is a test message line 11
MSG^This is a test message line 12
MSG^This is a test message line 13
MSG^This is a test message line 14
MSG^This is a test message line 15
MSG^This is a test message line 16
MSG^This is a test message line 17
MSG^This is a test message line 18
MSG^This is a test message line 19
MSG^This is a test message line 20
CTT^0
SE^26^000000001
GE^1^000001
IEA^1^000000001
```

Example 9. An example of an EDI output message

Receipt Output message formats

The receipt output message format is identical to the receipt input message format.
The IAgent server processes generate a raw ASCII text message with the following colon-delimited fields:

> an "IAR" header as the first 4 bytes of the receipt message,
> followed by the entire ISA segment from the original EDI message (105 bytes),
> followed by a fixed length timestamp (15 bytes, in yyyymmddhhmmssz format).

If the receipt is of type Integrity, or Signed then another field separator ":" (a colon) will follow the timestamp, followed by a field identifier of “I” for integrity (or message digest) or “S” for signed (or digital signature) receipts. A field separator “:” will follow the field identifier then either the digest or the digital signature (in hexadecimal format).

A typical Basic Receipt output message is as follows (line NOT wrapped)

```
IAR:ISA^00^IAGENT    ^00^TEST      ^ZZ^LWC_TEST       ^ZZ^LWC_7
^990629^1621^U^00307^000002466^0^T^>:19990630120000Z
```

Example 10. An example of a Basic Receipt output message

A typical Integrity Receipt output message is as follows (line NOT wrapped)

```
IAR:ISA^00^IAGENT    ^00^TEST      ^ZZ^LWC_TEST       ^ZZ^LWC_7
^990629^1621^U^00307^000002466^0^T^>:19990630120000Z:I:542ac34623f098790
ed987698b0
```

Example 11. An example of an Integrity Receipt output message

A typical Signed Receipt output message is as follows (line NOT wrapped)

```
IAR:ISA^00^IAGENT    ^00^TEST      ^ZZ^LWC_TEST       ^ZZ^LWC_7
^990629^1621^U^00307^000002466^0^T^>:19990630120000Z:S:542ac34623f098790
ed987698b0f743c6a06591ca410027ed7f7f
```

Example 12. An example of a Signed Receipt output message

IA Status Output message formats

The IAgent server processes generate a message starting with "IAS:" as an IA Status message. There are two types of IA Status output messages: Defined, and Custom. Defined status messages are as defined in the IA Specification (test, stop-WAN and stop-OSS is how we name them). The server will log and process the message but will not pass it on to the back-end system (BES). Custom status messages are not specifically defined and will actually be treated as a pass-through by the receiving IAgent server with the bit-string being ignored. It is expected that the back-end system will handle them.

The Defined status output message format is:

IAS: *trading partner ID***:** *command*

with *command* being any one of the following:

TEST	Special Test Message (will be logged, but otherwise ignored)
STOP_WAN	Problem with peer's inbound WAN, local IAgent should stop transmission to peer Trading Partner.
STOP_OSS	Problem with peer's downstream OSS, local IAgent should stop transmission to peer Trading Partner.
PING_RESPONSE	Response to a previous Request "ping" from a remote trading partner.

.

A typical Defined Status output message is as follows:

```
      IAS:LWC_TEST       :STOP_WAN
```

Example 13. An example of a Defined status output message

The Custom status output message format is:

IAS : *trading partner ID* **:** *custom bit string (32 bits)*

A typical Custom status output message is as follows:

```
      IAS:LWC_7          :00000000110011000000000000000000
```

Example 14. An example of a Custom status output message

CHAPTER 18 – FILE AND DIRECTORY API INTEGRATION

File Interface Mode

The file interface mode (`--interface=file`) is only supported by the Standalone Client (`--mode=C`) and assumes that the oneshot flag has also been set. The Client will require the input file argument to be supplied (`--file FILENAME`) and expects the file to be an ASCII text file, readable by the IAgent process, and containing any one of the Input Message Formats. The input file is not modified or moved. EDI messages (starting with “ISA” will be sent with standard priority.

Directory Interface Mode

The directory interface mode reads and writes all messages as ASCII text files, either from or to a specified directory. The (input or output) directory must exist or the specific process (client or server) will exit with a **Fatal Error**.

On input the client process reads only files with an `.edi` extension (for standard priority), or a `.hpri` extension (for hi-priority). Processed files will have their extensions changed to either:

`.sent`, if successfully sent, or to
`.notsent`, if handshake or socket errors are encountered.

On output (for any server process) all files will be generated for processing by a downstream EDI translator. All files are written with either an `.edi` extension (for standard priority) or a `.hpri` extension (for hi-priority) , in ASCII text mode and are saved atomically.

To assure atomic creation of input files it is recommended that the files be built with a temporary extension (not `.edi` or `.hpri`) and after the full message has been created, rename the file with an `.edi` or `.hpri` extension. The Move (`mv`) user command is known to be atomic on all IAgent supported environments.

The input (outbound) directory may be set with the `–a PATH` or `--indir PATH` options. The output (inbound) directory may be set with the `–D PATH` or `--outdir PATH` options.

All trading partner messages will exist and be processed in the same above directories.

Sample code to support the Directory Interface is supplied in *Chapter 23*.

CHAPTER 19 – NAMED PIPE API INTEGRATION

Named Pipe Interface Mode

The named pipe interface mode reads and writes all messages as a single atomic bytestream with a 2 byte header consisting of the entire message length, and a 2 byte priority flag (with 0 for standard delivery priority and 1 for hi delivery priority). This data is either written to or read from a named pipe.

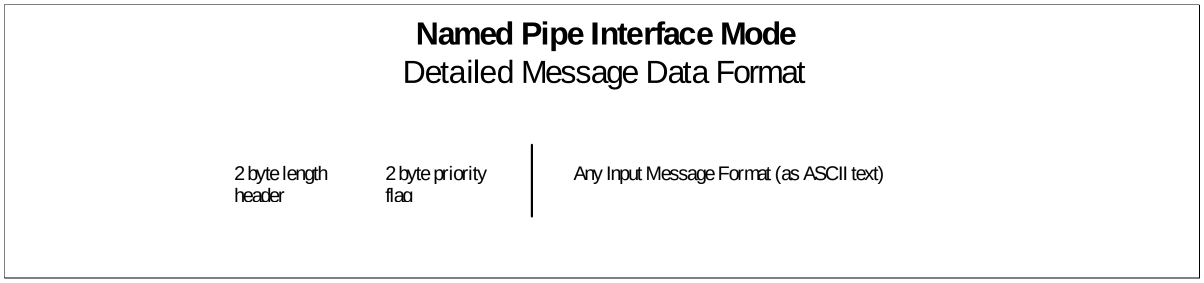

Named Pipe Interface Mode
Detailed Message Data Format
2 byte length header
2 byte priority flag
Any Input Message Format (as ASCII text)

Figure 10. The detailed message data format for the named pipe interface mode

If the message is shorter or longer than the length value in the header, than the message will not be correctly processed. It is important to note that processes writing to the named pipe interface must write the entire message (header and message body) in a single write to insure an atomic message.

All processes using the named pipe interface will fail if the specified pipe does not exist or has such file permissions as not to allow reading or writing.

The input (outbound) pipe name may be set with the `–b PIPENAME` or `--inpipe PIPENAME` options. The output (inbound) pipe name may be set with the `–p PIPENAME` or `--outpipe PIPENAME` options.

All trading partner messages will be fed into the same input pipe and will exit the same output pipe.

Sample code to support the Named Pipe Interface is supplied in *Chapter 23*.

CHAPTER 20 – SOCKET API INTEGRATION

Socket Interface Mode

The socket interface mode reads and writes all messages as a byte-stream with a 4 byte header consisting of the entire message length and a 2 byte priority flag (with 0 for standard delivery priority and 1 for hi delivery priority) . All data is either written to or read from a BSD-style socket (the source identified with a port and the destination identified as a fully qualified IP address, in the form of AAA.BBB.CCC.DDD:port).

On input the standard client will listen on the input port of the local IP address of the host machine. Connections will be accepted in a serial fashion, with each connection used only for a single message.

If the message is shorter or longer than the length value in the header, than the message will not be correctly processed. It is important to note that processes writing to the socket interface must write the entire message (header and message body) in a single write to insure an atomic message.

Socket messages consist of a four byte binary length header (containing the entire length of the message), a two byte binary flag (containing the delivery priority) and the message body (containing the raw EDI message).

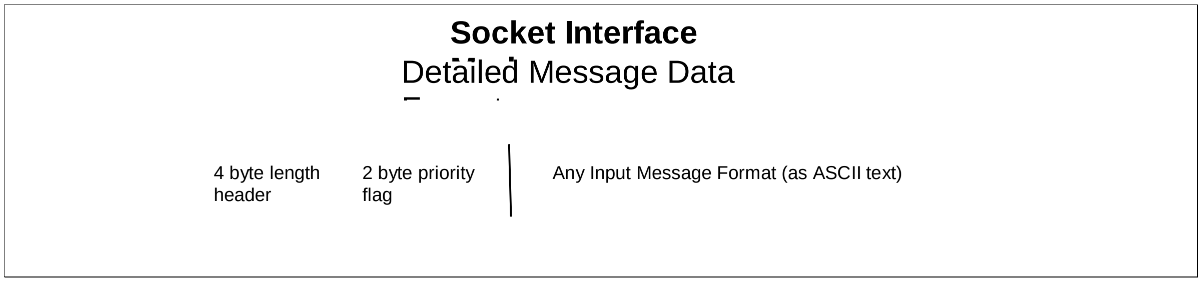


Figure 11. The detailed message data format for the socket interface mode

On output the server(s) will attempt to connect to the specified socket IP address (including port) to deliver the message. If the connect fails then the message will be stored as a file (see Directory Interface Mode, *Chapter18* for file details) in the $TEMP directory (`/tmp` by default).

Use of port numbers greater than 1024 is recommended so as not to run into root permission and port-sharing problems.

The input (outbound) socket port may be set with the `–g PORT` or `--insocket PORT` options. The output (inbound) socket address and port may be set with the `–S ADDRESS` or `--outsocket ADDRESS` options. Be sure to include a port in the address.

All trading partner messages will be fed into the same input socket and will exit to the same output socket.

Sample code to support the Socket Interface is supplied in *Chapter 23*.

CHAPTER 21 – TIBCO^{tm} MESSAGE API INTEGRATION

Tibco Interface Mode

TIBCO Software Inc, of Palo Alto, California, USA sells a number of middle-ware, message bus type products, which allow distributed processes in a heterogeneous environment to communicate using a publisher subscriber model. Their TIB/Rendezvous product allows multiple publishers to send atomic RV messages to multiple subscribers, identified with a unique RV message subject name.

The tibco interface mode reads and writes an IAgent specific RV message, containing a length and an arbitrary ASCII string (containing the Input or Output Message Format). The RV message is defined as:

```
iaRVMessageType
{
     RV_int         length;
     RV_int         priority;
     RV_string      data;
}
```

The priority element has only two valid values 0 for standard priority and 1 for high priority delivery.

On input the standard client will subscribe to the IAgent RV message subject name (`iagent.tibco.client.message.version1.1`) and process valid IAgent RV messages from any publisher. If a message is found to contain the correct subject name but does not conform to the IAgent RV message layout, then the message will be dropped and the message data will be lost.

On output the server(s) will attempt to publish valid IAgent RV messages to any subscribers to the IAgent RV message subject name (`iagent.tibco.server.message.version1.1`). If any error is encountered during publication, then the message will be stored as a file (see Directory Interface Mode for file details) in the $TEMP directory (`/tmp` by default).

Warning: If there are no active subscribers present when the message is published then the message will be lost.

The input (outbound) RV message subject name may be set with the `–j SUBJECTNAME` or `--intibco SUBJECTNAME` options. The output (inbound) RV message subject name may be set with the `–T SUBJECTNAME` or `--outtibco SUBJECTNAME` options.

All trading partner messages will be fed to the same input tibco subject name (currently a single subscriber) and will be published to the same output tibco subject name (which may be one or more subscribers).

Sample code to support the Tibco Interface is supplied in *Chapter 23*. Additional tibco rv examples are provided with the TIB/Rendezvous product.

Tibco Installation Issues

After following all Tibco supplied installation directions, the following local installation tasks may be required:

1. Acquire and install the `rvd.tix` license file.
2. Set the IAgent user's PATH environmental variable to include the `rvd` executable path.
3. Set the IAgent user's LD_LIBRARY_PATH to include the `librvd.so` location.

Note: This IAgent implementation does not support either the certified messaging or fault tolerant options available in some versions of Rendezvous. This implementation supports Rendezvous version 5.3 and 6.6.

For more information on TIB/Rendezvous concepts, installation instructions, and the TIB/Rendezvous APIs and products available, please reference the TIB/Rendezvous listed in *Appendix H*. Information on TIB/ Rendezvous availability and licensing may be found at **www.rv.tibco.com**.

CHAPTER 22 – TESTING INTEGRATION API CONFIGURATIONS

Because of the IAgent input and output messages symmetric nature, any BES should be able to test support for any of the Integration APIs by feeding its output data back to its own input interface as shown below.

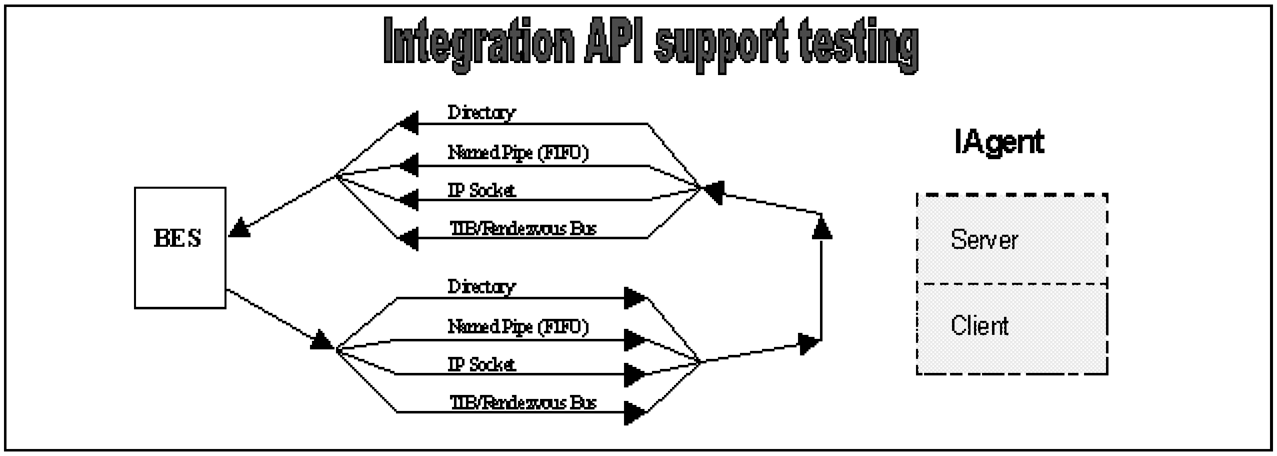


Figure 12. Integration API Support Testing

It is expected that any BES output function or routine should be able to feed the matching BES input function or routine successfully, using a range of expected types of input or output data, and covering specific data cases known to exist within the BES.

Integration API Specific testing issues

Directory Interface Issues

The directory interface expects the input files to be written atomically and to have IAgent process read and write permissions (since the extensions will be changed after processing). It is required that both the input and output directories exist and have IAgent process read, write and execute permissions.

Named Pipe Interface

The named pipe interface expects the input pipe messages to be written atomically and to have the length portion of the message accurately reflect the size of the data portion of the message. Invalid length values will cause unpredictable results and may cause problems with all further input messages. It is required that both the input and output pipes exist and have IAgent process read and write permissions set.

On some operating systems the maximum length of a FIFO may limit the size of IA messages supported. If this is the case and IA message can exceed this system limit than use of another Integration API is recommended.

Socket Interface

The socket interface expects the input messages to be written atomically and to have the length portion of the message accurately reflect the size of the data portion of the message. Invalid length values will cause unpredictable results and may cause problems with all further input messages.

Tibco Interface

The tibco interface expects the input messages to be published as an RVMSG_RVMSG message type, and will ignore all other message types. Only the three defined fields (in *Chapter 21*) are required, optional fields can be included but will be ignored by the IAgent client processes.

It is required that the length portion of the message accurately reflects the size of the data portion of the message. Invalid length values will cause the message to be dropped and not processed.

It is important to remember that if a tibco subscriber is not up and listening for the specific subject name published to, that the data will be lost.

See *Appendix D* for a sample simple BES Integration API test plan.

CHAPTER 23 – EXAMPLE INTEGRATION API CODE SAMPLES

The following code samples demonstrate how to interface to each of the supported Integration APIs. All examples are in Perl 5.6.0 and are only relevant fragments of working programs. More information about Perl can be found at: http://www.perl.org

Directory API Code Samples

Directory API Reader Example

```
$DIR = '/opt/iagent/dir_out';

while(1)  # forever
{
  # start directory read loop
  foreach $file (glob("$DIR/*"))
  {
       open (FILE, "$file") || die "can't read $file: $!";

       # read entire IA message
       read(FILE, $IAmessage, 128*1024);
  }
  sleep 1;  # dir loop read delay
}
```

Example 15. Directory API Reader Example

Directory API Writer Example

```
$DIR = '/opt/iagent/dir_in/';

       my $file = $DIR . "fn" . $Counter . ".edi";
       my $file2 = $file . ".tmp";
       open (FILE, "> $file2") || die "can't create/write to $file2: $!";

# write entire IA message to file2 in single print
       print FILE $IAmessage;
       close FILE;
       sleep 2;    # to avoid dup signals
       rename($file2, $file);  # Atomic creation of new valid IA input file
```

Example 16. Directory API Writer Example

Named Pipe API Code Samples

Named Pipe API Reader Example

```
$FIFO = '/opt/iagent/iapipe';

        # next line blocks until there's a writer
        open (FIFO, "< $FIFO") || die "can't read $FIFO: $!";

        # read length
        my $binary = '  ';
        read(FIFO, $binary, 2);
        my $IAlength = unpack("N", $binary);

        # read priority
        my $binary = '  ';
        read(FIFO, $binary, 2);
        my $IApriority = unpack("N", $binary);

        # read data
        read(FIFO, $IAdata, $IAlength);
```

Example 17. Named Pipe API Reader Example

Named Pipe API Writer Example

```
$FIFO = '/opt/iagent/iapipe';
    while (1) {
        # next line blocks until there's a reader
        open (FIFO, "> $FIFO") || die "can't write $FIFO: $!";

        # store length
        my $binary1 = pack("N", $IAlength);

        # store priority
        my $binary2 = pack("N", $IApriority);

        # write it all
        print FIFO $IAlength $IApriority $IAdata;
        close FIFO;
        sleep 2;    # to avoid dup signals
```

Example 18. Named Pipe API Writer Example

Socket API Code Samples

Socket API Reader Example

```
my $iaddr = gethostbyname('frodo');
my $proto = getprotobyname('tcp');
my $port = 6669;
my $paddr = sockaddr_in(0, $iaddr);

socket(SOCKET, PF_INET, SOCK_STREAM, $proto)   || die "socket: $!";
connect(SOCKET, $paddr)          || die "bind: $!";

# read length
my $binary = '  ';
read(SOCKET, $binary, 4);
my $IAlength = unpack("N", $binary);

# read priority
my $binary = '  ';
read(SOCKET, $binary, 2);
my $IApriority = unpack("N", $binary);

# read data
read(SOCKET, $IAdata, $IAlength);
```

Example 19. Socket API Reader Example

Socket API Writer Example

```
my $proto = getprotobyname('tcp');
   my $port = 6669;

   socket(SOCKET, PF_INET, SOCK_STREAM, $proto)   || die "socket: $!";
   setsockopt(SOCKET, SOL_SOCKET, SO_REUSEADDR,
      pack("l", 1))   || die "setsockopt: $!";
   bind(SOCKET, sockaddr_in($port, INADDR_ANY))   || die "bind: $!";
   listen(SOCKET,SOMAXCONN)                        || die "listen: $!";

   my $paddr = accept(CLIENT,SOCKET);

   # store length
   my $binary1 = pack("N", $IAlength);

   # store priority
   my $binary2 = pack("N", $IApriority);

   # write it all
   print CLIENT $IAlength $IApriority $IAdata;

 close CLIENT;
```

Example 20. Socket API Writer Example

Tibco API Code Samples

Additional example code is provided with the Tibco product. These examples use the Tibco RV 5.x interface

Tibco API Reader Example

```
use Rv;
$subjectname = "iagent.tibco.server.message.version1.1";
my $IAmessage;
my $IAlength;
my $IApriority;

# Callback sub to print the messages
sub my_callback
{
   my($session, $name, $replyName, $msgType, $msgSize, $msg, $arg) = @_;

    if ($msgType == RVMSG_RVMSG)
   {
      # then look for our three fields
      # IA message length
      $ret = rvmsg_Get($session, $msg, "length", $msgType, $msgSize, $IAlength);
      die "rvmsg_Get $ret length\n" unless $ret == RVMSG_OK;

      # IA message priority
      $ret = rvmsg_Get($session, $msg, "priority", $msgType, $msgSize, $IApriority);
      die "rvmsg_Get $ret priority\n" unless $ret == RVMSG_OK;

      # IA message data
      $ret = rvmsg_Get($session, $msg, "data", $msgType, $msgSize, $IAmessage);
      die "rvmsg_Get $ret data\n" unless $ret == RVMSG_OK;
   }
}

# Initialize the rv session
$ret = rv_Init($session);
die "rv_Init $ret\n" unless $ret == RV_OK;

# Listen for IA subject name
$ret = rv_ListenSubject($session, $sub_id, $subjectname, \&my_callback, 0);
die "rv_ListenSubject\n" unless $ret == RV_OK;

# Enter the event loop
rv_MainLoop($session);
```

Example 21. Tibco API Reader Example

Tibco API Writer Example

```
use Rv;
$subjectname = "iagent.tibco.client.message.version1.1";
$IAmessage = spaces(1024);  # for test only
$IAlength = strlen($IAmessage);
$IApriority = 1;

$rvMessage = spaces($IAlength+4);

# Initialize the rv session
$ret = rv_InitSync($session);
die "rv_InitSync $ret\n" unless $ret == RV_OK;

# load up the rvMessage
$ret = rvmsg_Init($session,$rvMessage,strlen($rvMessage));
die "rvmsg_Init $ret\n" unless $ret == RVMSG_OK;

# Build the message to be sent using rvmsg_Append()
## Add IA message length as RVMSG_INT
$ret = rvmsg_Append($session, $rvMessage, strlen($rvMessage),
        "length", RVMSG_INT, 2, $IAlength);
die "rvmsg_Append $ret\n" unless $ret == RVMSG_OK;

## Add IA message priority as RVMSG_INT
$ret = rvmsg_Append($session, $rvMessage, strlen($rvMessage),
        "priority", RVMSG_INT, 2, $IApriority);
die "rvmsg_Append $ret\n" unless $ret == RVMSG_OK;

## Add IA message data as RVMSG_STRING
$ret = rvmsg_Append($session, $rvMessage, strlen($rvMessage),
        "data", RVMSG_STRING, $IAlength, $IAmessage);
die "rvmsg_Append $ret\n" unless $ret == RVMSG_OK;

# now send it
$ret = rv_Send($session, $subjectname, RVMSG_RVMSG, 0, $rvMessage);
die "rv_Send $ret\n" unless $ret == RV_OK;

rv_Term($session);
```

Example 22. Tibco API Writer Example

CHAPTER 24 – TESTING INTEGRATION WITH BACKEND SYSTEMS

Back end systems (BESs) are assumed to be even driven and capable of producing valid IAgent input message formats and processing valid IAgent output message formats for the specific Integration API selected.

It is suggested that the Directory Integration API first be supported so input message formats may be examined, prior to back end system integration.

A simple back end system (BES) integration test must:

1. Test each input message format, independent of Integration API,
2. Test each output message format, independent of Integration API,
3. Test the specific input Integration API functionality. This may be tested with a standalone IAgent client (see *Chapter 25* on how to start a standalone IAgent client),
4. Test the specific output Integration API functionality. This may be tested with a standalone IAgent server (see *Chapter 25* on how to start a standalone IAgent server).
5. Do an end-to-end test with the specific Integration API. This may be tested with a standalone IAgent client and a standalone IAgent server.
6. Do a second end-to-end test with the specific Integration API. This should be tested with a full IAgent system.

See *Appendix D* for a sample Backend Integration API test plan.

Part 5 – IAgent Administration

Part 5 of this manual contains IAgent administration information, including operations tasks, maintenance issues and troubleshooting guidelines. This section also includes security recommendations.

CHAPTER 25 – IAGENT OPERATIONS ISSUES

Product Startup Issues

The IAgent system can be run in one of two modes: Combined, or Standalone. Combined mode allows a single binary to spawn the required four major processes to support all IA processing. The four processes are: a standard client, a receipt client, a standard server, and a high priority server. Each process will be started from the original process and will share data via several shared memory databases and message queues.

STARTING THE IAGENT SYSTEM

You must choose to start IAgent either as a system daemon (at boot) or from the command line. **Do not do both!** Either you can put in `/etc/rc3.d` and have it started on system boot, or you can start it as from the command line. See *Chapter 9* for details on the command line options.

It is suggested that you start IAgent from the command line to test your configuration and configuration files. When this has been successfully completed, run the following to start IAgent as a system daemon with the following command:

```
su root -c "/etc/rc3.d/S99iagent start”
```

Example 23. Starting IAgent as a system daemon

Sun Solaris Only
It is assumed that the PRNG daemon was already started prior to the above command. This daemon is also automatically started on boot.

The Standalone mode allows a single process (client or server) to be started independently. At this time processes running in standalone mode do not have access to the shared memory database, and therefore have limited functionality. It may be useful to start a client or server in standalone mode for testing purposes.

Because specific functions may require shared memory database support (pending receipt list processing, for example), they will not be available in standalone mode.

Starting the system in Combined Mode

The IAgent system (clients and servers) may be started from the command line as follows (note this is a single line):

```
./bin/iagent –-mode=F --interface=dir
--indir=in_dir --outdir=out_dir
```

Example 24. Starting IAgent form the command line in Combined mode

This example will start the IAgent system up in Combined (or Full) mode (`--mode=F`) with the directory interface supported (`--interface=dir`) in both Clients and Servers. Input messages will be read from the Client directory (`--indir=in_dir`) and output messages will be written to the Servers directory (`--outdir=out_dir`).

Starting the system in Standalone Mode

The IAgent process (client or server) may be started from the command line as follows:

```
./bin/iagent –-debug=1 --interface=file --mode=C --
file=tpone2lwctst.edi
```

Example 25. Starting the IAgent system in standalone mode

This example will start the IAgent process up in Client mode (`--mode=C`) with the file interface supported (`--interface=file`). Basic debug messages will be displayed (`--debug=1`) and a single file will be processed, then the process will exit.

```
./bin/iagent -d 1 -m S -i dir –D test_dir
```

Example 26. Another startup example

This example will start the IAgent process up in Server mode (`-m S`) with the directory interface supported (`-i dir`). Basic debug messages will be displayed (`-d 1`) and all output transactions will be written to the output directory (`-D test_dir`). The server process will continue to process transactions until killed with a TERM or KILL signal, then the process will exit.

At this point the IAgent system may be monitored by watching all of the following log and status files:

iagent.log,
iagent.err,
iagent.alert,
client_trans.log,
receipt_trans.log,
server_trans.log, and
hipri_trans.log

IAgent Debug Startup

A debug startup script is provided (in `/opt/iagent/bin`) which may be used to cause all debug messages to be displayed, with messages going to either `iagent.log` or `iagent.err`. This may be useful to use during connectivity testing with a remote IA or during testing of BES connections to the IAgent system.

To run the IAgent system in debug mode, the following steps need to be done:

1. login to the IA machine as `root`
2. change the working directory to `/opt/iagent`
3. stop the IAgent system (if it was running)
4. start the IAgent system with the new debug startup script as detailed below:

The `root` user will then be able to start/stop the IAgent system in debug mode manually by typing:

```
   ./bin/debugia start
```
or
```
   ./bin/debugia stop
```

Log and Output Files

The IAgent system and processes generate several different types of log and output files. The log files consist of:

- A system-wide output log file (`iagent.log`) which contains almost everything that is written to the console (stdout), including verbose messages.

- A system-wide error log file (`iagent.err`) which contains all error messages written to the console (stderr), including warnings, debug statements, and fatal error messages.

- A system-wide alert log file (`iagent.alert`) which contains all major error alerts, as single line alert entries. This log may be automatically inspected to alert support personnel of IAgent system distress that should be immediately attended to.

- Process specific transaction logs (one per major process) which log the final status of a transaction (either inbound, to the server(s) or outbound, from the client(s). These logs are called: `client_trans.log, receipt_trans.log, server_trans.log, and hipri_trans.log`. The log file contents are described in *Chapter 26*.

- The IAgent system will create a process ID file (PID file, `iagent.pid`) for the process group of all IAgent processes created. It is suggested that all signals sent to the IAgent processes use this file.

- All received IA Status messages are saved to text files, with a specific file name:

  *Transaction_Partner_ID***.** *Status*

  With *Status* being either: **test** (for status TEST messages), **stop** (for WAN or OSS stop messages) **ping** (for PING response messages), or **status** for custom status messages. A custom status message file will only contain a single bit string (up to 32 bits).

- All failed transactions are saved in (very ugly) temporary file names to the failed message directory (`/opt/iagent/failed_messages`). If possible they are saved in their ASN1 DER binary format (so don't try to cat them!). They may be inspected with the included **asn1dump** utility (see *Appendix F*).

- All invalid messages and receipts will be saved in (very ugly) temporary file names to the invalid message directory (`/opt/iagent/invalid_messages`). If possible they are also saved in their ASN1 DER (binary) format. They may also be inspected with the included **asn1dump** utility.

- All inbound receipts can be saved in Receipt Input Message Format to temporary file names in the receipt log directory (`/opt/iagent/receipts`) for manual examination and archiving, if the savereceipts flag is set..

CHAPTER 26 – IAGENT MAINTENANCE ISSUES

Log Rolling

The output log files (`iagent.log`, `iagent.err`, and `iagent.alert`) and transaction log files (*_trans.log) can be safely "rolled" by performing the following actions:

1. Move all the log files to a new location/name with the `mv` command,
2. Send a SIGHUP to the IAgent process or process group identified by the `iagent.pid` file.

This last step will signal all IAgent processes to close then re-open all the log files and will in effect, complete the prior move command. It is recommended that a `cron` job performing the above steps be scheduled on a daily basis to insure minimal log size and allow log archiving.

Log File Details

The IAgent system generates a single transaction log file per major process. These process specific transaction logs (one per major process) log the final status of a transaction (either inbound, to the server(s) or outbound, from the client(s). These logs are called: `client_trans.log, receipt_trans.log, server_trans.log, and hipri_trans.log`. The log files consist of a transaction per line in the following format:

date time: *message type*: *ISA segment of the message*: *source or destination IP address and port*: *message priority*: and *numeric transaction status*

The *date time* is the time the entry was logged, in local time.

The *message type* may be any of the following:

Message Type	Type Number
UNSUPPORTED	0
BASIC_EDI	1
INTEGRITY_EDI	2
SIGNED_EDI	3
BASIC_RECEIPT	4
INTEGRITY_RECEIPT	5
SIGNED_RECEIPT	6
IA_STATUS	9

Table 12 Transaction Log Message Type Numbers

The *ISA segment of the message* should be the first 105 bytes of a raw EDI or Receipt message. For IA Status messages this field is populated with the status bit string in four hexadecimal numbers.

The *source or destination IP address and port* will reflect the other side of the exchange. In the server(s) it should be the transaction's source IP address. In clients(s) it should reflect the destination IP address of the transaction.

The *message type* is the delivery priority of the inbound or outbound message and may be either of the following:

STANDARD PRIORITY	= 0
HI PRIORITY	= 1

The *numeric transaction status* is the final disposition of the transaction. Zero (0) means successful transaction, with negative numbers reflecting an internal error condition and positive numbers an external error condition. The following table contains all available transaction error/status codes for the current release.

Transaction Error / Status Code Description	**Value**
IA_ERRM_CANNOT_SAVE_IASTATUS_TO_PIPE	-28
IA_ERRM_CANNOT_SAVE_IASTATUS_OUT_FILE	-27
IA_ERRM_CANNOT_REMOVE_PENDING_RECEIPT	-26
IA_ERRM_CANNOT_GENERATE_RECEIPT	-25
IA_ERRM_CANNOT_SAVE_RECEIPT_TO_PIPE	-24
IA_ERRM_CANNOT_WRITE_TIBCO_EDI_MSG	-23
IA_ERRM_CANNOT_WRITE_EDI_MSG_TO_SOCKET	-22
IA_ERRM_CANNOT_WRITE_EDI_MSG_TO_PIPE	-21
IA_ERRM_NO_PENDING_REC_FOUND	-20
IA_ERRM_TP_ID_MISSING_FROM_INPUT_IA_STATUS_MSG	-19
IA_ERRM_ISA_SEG_MISSING_FROM_INPUT_RECEIPT_MSG	-18
IA_ERRM_CANNOT_GENERATE_IA_STATUS_MSG	-17
IA_ERRM_CLIENTS_CA_CERTS_MISSING	-16
IA_ERRM_CLIENTS_OWN_CERT_MISSING	-15
IA_ERRM_CANNOT_SAVE_RECEIPT_OUT_FILE	-14
IA_ERRM_CANNOT_SAVE_EDI_MSG_FILE	-13
IA_ERRM_CANNOT_OPEN_OUTPUT_FILE	-12
IA_ERRM_CANNOT_WRITE_OUTPUT_FILE	-11
IA_ERRM_UNSUPPORTED_MESSAGE_TYPE	-10
IA_ERRM_CANNOT_GENERATE_EDI_MSG_FILE	-9
IA_ERRM_UNKNOWN_MSG_TYPE_IN_TPFILE	-8
IA_ERRM_UNKNOWN_RECEIPT_MSG_TYPE_IN_TPFILE	-7
IA_ERRM_UNKNOWN_TYPE_OF_MSG_FOR_TP	-6
IA_ERRM_CANNOT_ALLOCATE_CTX	-5
IA_ERRM_CANNOT_ALLOCATE_SSLCTX	-4
IA_ERRM_CANNOT_ALLOCATE_CLIENT_SOCKET	-3
IA_ERRM_CANNOT_READ_OWN_CERT,	-2
IA_ERRM_CANNOT_READ_OWN_KEY,	-1
IA_SUCCESSFUL_TRANSMISSION	0

Table 13. IAgent Error Numbers

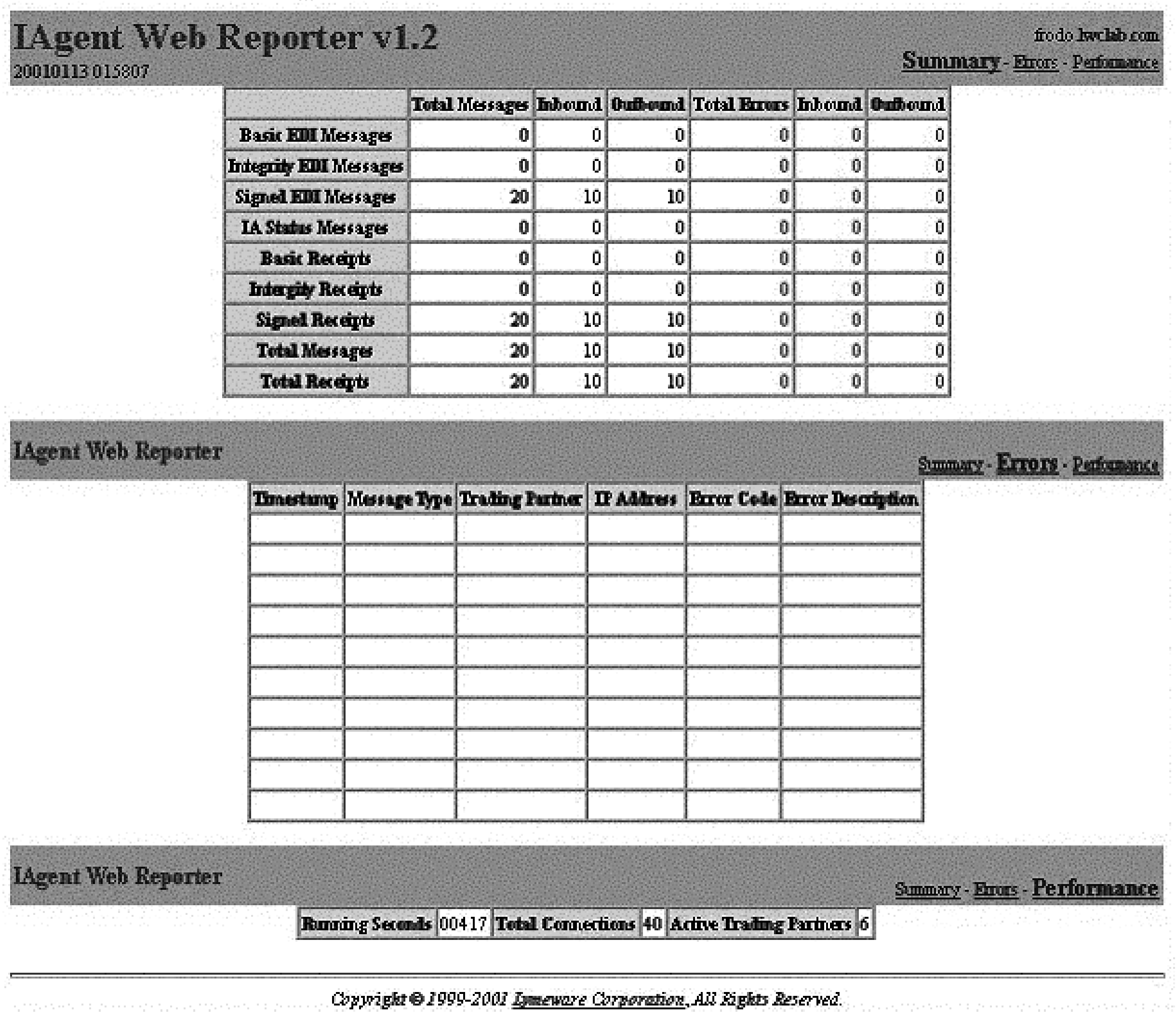
IA_ERRM_UNSUPPORTED_INPUT_MSG_FOUND	1
IA_ERRM_UNKNOWN_ISAID_FOUND	2
IA_ERRM_UNKNOWN_ISAID_FOUND_IN_CERT	3
IA_ERRM_NO_ISAID_FOUND_IN_TPFILE	4
IA_ERRM_CANNOT_CONNECT_TO_REMOTE_SERVER	5
IA_ERRM_CANNOT_START_CLIENT_HANDSHAKE	6
IA_ERRM_CANNOT_WRITE_SSL_CLIENT	7
IA_ERRM_CANNOT_READ_SSL_SERVER	8
IA_ERRM_CANNOT_FIND_SUBJECTNAME_IN_CERT	9
IA_ERRM_CANNOT_FIND_ISSUERNAME_IN_CERT	10
IA_ERRM_CANNOT_ACCEPT_CLIENT_HANDSHAKE	11
IA_ERRM_CANNOT_VERIFY_CLIENT_CERT	12
IA_ERRM_NO_CLIENT_CERT_SENT	13
IA_ERRM_UNKNOWN_IPADDR_SENT_FROM_CLIENT	14
IA_ERRM_CANNOT_PARSE_ASN1	15
IA_ERRM_MESSAGE_INTEGRITY_BROKEN	16
IA_ERRM_MESSAGE_BAD_SIGNITURE_FOUND	17
IA_ERRM_RECEIPT_INTEGRITY_BROKEN	18
IA_ERRM_RECEIPT_BAD_SIGNITURE_FOUND	19
IA_ERRM_CNAME_MISMATCH	20
IA_ERRM_INCORRECT_MESSAGE_TYPE_SENT_FROM_TP	21
IA_ERRM_RECEIVED_MESSAGE_FROM_INACTIVE_TP	22

Table 14. IAgent Error Numbers (continued)

More details of specific errors can be found in *Chapter 28.*

The Web Reporter

This release of the IAgent system supports dynamic monitoring via the IAgent Web Reporter. An example screen is displayed in the following figure.



IAgent Web Reporter v1.2
20010113 015807

frodo.lwclab.com
Summary - Errors - Performance

	Total Messages	Inbound	Outbound	Total Errors	Inbound	Outbound
Basic EDI Messages	0	0	0	0	0	0
Integrity EDI Messages	0	0	0	0	0	0
Signed EDI Messages	20	10	10	0	0	0
IA Status Messages	0	0	0	0	0	0
Basic Receipts	0	0	0	0	0	0
Intergity Receipts	0	0	0	0	0	0
Signed Receipts	20	10	10	0	0	0
Total Messages	20	10	10	0	0	0
Total Receipts	20	10	10	0	0	0

IAgent Web Reporter

Summary - **Errors** - Performance

Timestamp	Message Type	Trading Partner	IP Address	Error Code	Error Description

IAgent Web Reporter

Summary - Errors - **Performance**

Running Seconds	00417	Total Connections	40	Active Trading Partners	6

Copyright © 1999-2001 Lymeware Corporation, All Rights Reserved.

Figure 13. Example Web Reporter display

The Web Reporter allows remote monitoring of traffic, errors, and performance data. Specifically it displays all inbound and outbound message traffic, in message detail and with message type summaries. The web reporter also displays the 10 most current errors encountered (inbound or outbound) and provides limited performance statistics.

Any HTML 3.2 or newer compatible web browser may be used for display. The reporter function updates the page (really the reporter file) every minute and the resulting HTML also auto-refreshes itself (using the HTML **meta** tag of `http-equiv=refresh`) every minute.

The reporter file (`iastatus.html`) may be locally browsed (with a URL of something like: file://opt/iagent/iastatus.html) without the need for a web server.

The reporter file may be served as a single web page with any HTTP 1.0 / HTML 3.2 or newer web server, if remote access is desired.

CHAPTER 27 – TROUBLESHOOTING PROBLEMS AND ERRORS

Troubleshooting errors and problems with the IAgent system may seem like a daunting task, given the distributed processing and collaborative nature of the IAgent system and its peer systems (both all other IA systems and the BES). The IAgent system can best be diagnosed by envisioning the entire system as a series of pipes and checking the status and data at each pipe connection.

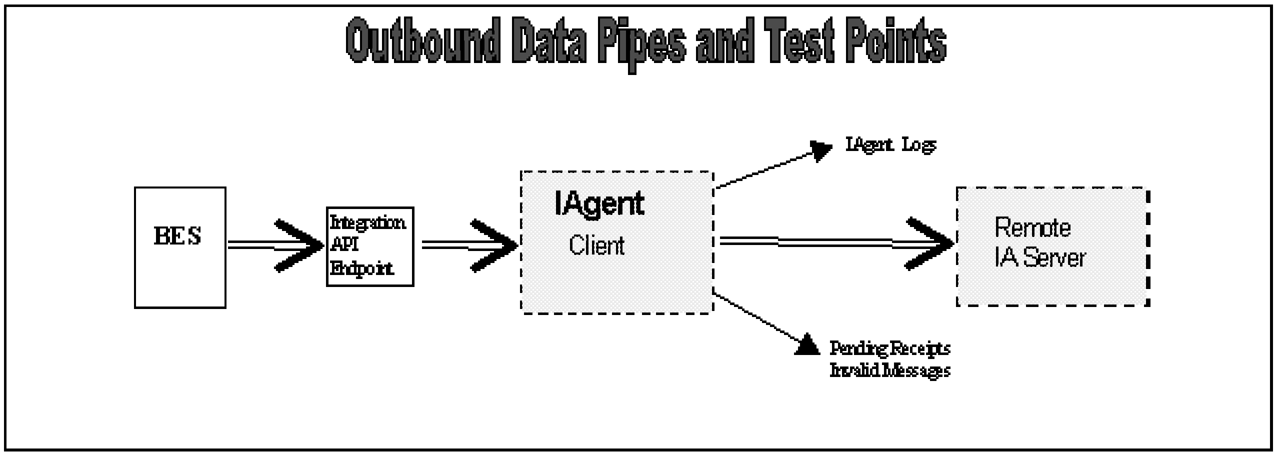

Outbound Data Pipes and Test Points
BES
Integration API Endpoint
IAgent Client
IAgent Logs
Remote IA Server
Pending Receipts Invalid Messages

Figure 14. IAgent Processing as a Series of Pipes (outbound)

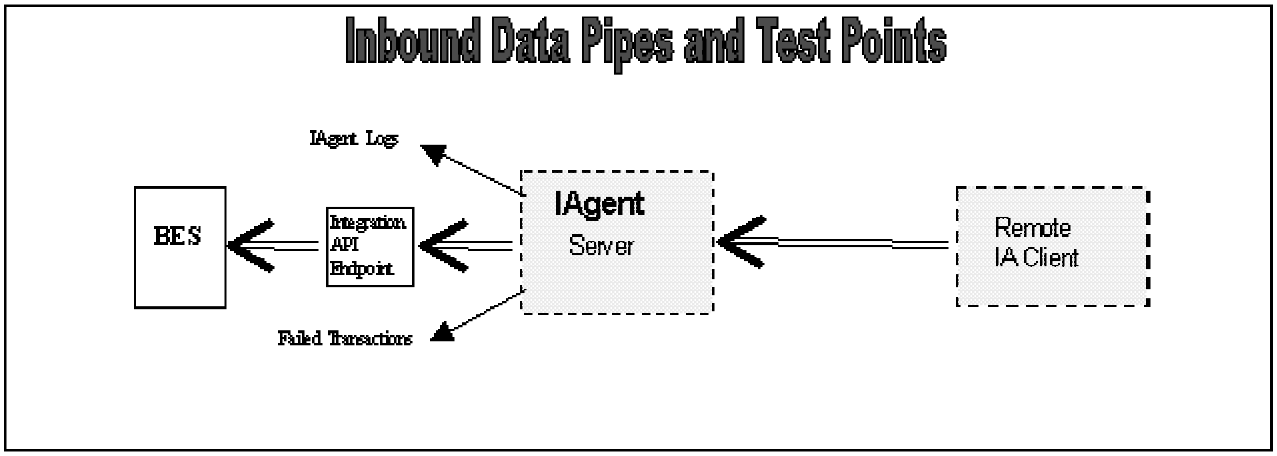


Figure 15. IAgent Processing as a Series of Pipes (inbound)

It is suggested that running the IAgent product, in a standalone mode, from a command line and with the verbose and debug display flag set is the first and usually best step. Inspect the `iagent.log`, `iagent.err` and `iagent.alert` logs after sending a single message to the IAgent system from the BES.

Failed Transactions

Failed Transactions are local input messages that, for some reason, could not be sent to the remote trading partner IA system. Currently all failed transactions will be saved in (very ugly) temporary file names to the failed message directory (`/opt/iagent/failed_messages`). If possible they will be saved in their ASN1 DER binary format (so don't try to cat them!).

Invalid Messages

Invalid Messages are remote input messages which, for some reason, could not be properly received from the remote trading partner IA system. All invalid messages and receipts will be saved in (very ugly) temporary file names to the invalid message directory (`/opt/iagent/invalid_messages`). If possible they will be saved in their ASN1 DER (binary) format.

IAgent Alert Messages

All operational alerts (events and errors) the IAgent processes wish to report to the operator are logged to the `iagent.alert` log file. This log should be inspected on a regular basis for any entries.

Debug Messages

The `debug_level` option (see *Chapter 9*) allows multiple levels of display messages depending on the value provided. The `debug_level` value is processed as a series of masks added together. For example, to display parsed ASN1 messages (on input and output) and display the Basic Debug messages the value would be 32 + 1 = 33. It is suggested that the smallest number of mask values be used to minimize the size and complexity of the output.

Note: Use of debug messages will adversely impact IAgent system performance and is not recommended for standard production use. All defined Debug level masks are listed in the following table:

Debug Level Mask	**Description**
1	Display Basic Debug messages
2	Display Debug SSL Handshake State messages
4	Display Debug Transaction Data messages
8	Display Debug Certificate messages
16	Display Debug ASN1 messages
32	Display Parsed ASN1 messages
64	Display Debug RSA Key messages
128	Display Debug Receipt messages
256	Display IA Timer messages
512	Display Internal Debug messages

Table 15. Debug Level Masks

Examples of all of the `debug_level` logging display screens follow below.

```
20000417 035739:iagent(5348):=== Trading Partner settings =====================
20000417 035739:iagent(5348): isaid = LWC_TEST, cn = LWC_TEST
20000417 035739:iagent(5348): ip = 191.168.0.10
20000417 035739:iagent(5348): stdport = 1111, hipriport = 2222
20000417 035739:iagent(5348): session timeout = 0
20000417 035739:iagent(5348): EDI message type to TP = EDI with Non-repudiation
(signature)
20000417 035739:iagent(5348): EDI message type from TP = EDI with Non-
repudiation (signature)
20000417 035739:iagent(5348): Receipt message type from TP = Receipt with
Message Integrity (digest) required
20000417 035739:iagent(5348): Receipt message type to TP = Receipt with Message
Integrity (digest) required
20000417 035739:iagent(5348): Trading Partner's Active server flag = 1
20000417 035739:iagent(5348): IA Standard "Flavor" default = ECIC 12/98
20000417 035739:iagent(5348):=== Trading Partner settings =====================
20000417 035739:iagent(5348): isaid = TP_1, cn = TP_1
20000417 035739:iagent(5348): ip = 127.0.0.1
20000417 035739:iagent(5348): stdport = 9001, hipriport = 9011
20000417 035739:iagent(5348): session timeout = 30
20000417 035739:iagent(5348): EDI message type to TP = EDI with Non-repudiation
(signature)
20000417 035739:iagent(5348): EDI message type from TP = EDI with Non-
repudiation (signature)
20000417 035739:iagent(5348): Receipt message type from TP = No Receipt required
20000417 035739:iagent(5348): Receipt message type to TP = No Receipt required
20000417 035739:iagent(5348): Trading Partner's Active server flag = 1
20000417 035739:iagent(5348): IA Standard "Flavor" default = ECIC 12/98
20000417 035740:iagent(5348):Generating temp (512 bit) RSA key...
20000417 035852:iagent(5348):i= 3 partner[i].cname = TP_1 strp = TP_1
   1 items in the session cache
   0 client connects (SSL_connect())
   0 client connects that finished
   1 server connects (SSL_accept())
   1 server connects that finished
   0 session cache hits
   0 session cache misses
   0 session cache timeouts
20000417 035852:iagent(5348):ASN1_parse returned a 1
20000417 035852:iagent(5348):mk_ASN1_parse() FOUND object of signedEDImessage
type
20000417 035852:iagent(5348):RSA digital signature verified
20000417 035852:iagent(5348):i= 3 partner[i].isaid = TP_1 strp = TP_1
20000417 035852:iagent(5348):No receipt required for TP_1

20000417 035850:iagent(5355):i= 4 partner[i].isaid = TP_TWO strp = TP_TWO
20000417 035850:iagent(5355):AE2ENR:i2d_IA_EDI_MSG_signed() returned a 1422
```

Example 27. An example of Basic Debug Messages

```
20000417 035850:iagent(5355):CTX is NULL - loading CTL fill stuff
20000417 035850:iagent(5355):only loaded CTX - return 0
20000417 035850:iagent(5355):CACHED SESSION lookup ===========================
20000417 035850:iagent(5355):GENERATING A NEW SESSION [0]
20000417 035850:iagent(5355):CTX already loaded - skipping CTL fill stuff
20000417 035850:iagent(5355):SSL is NULL - loading SSL with SSL_new
20000417 035850:iagent(5355):SSL_connect:before/connect initialization
20000417 035850:iagent(5355):SSL_connect:SSLv3 write client hello A
20000417 035851:iagent(5355):SSL_connect:SSLv3 read server hello A
20000417 035851:iagent(5355):PKI_verify_cert
20000417 035851:iagent(5355):SSL_connect:SSLv3 read server certificate A
20000417 035851:iagent(5355):SSL_connect:SSLv3 read server certificate request A
20000417 035851:iagent(5355):SSL_connect:SSLv3 read server done A
20000417 035851:iagent(5355):SSL_connect:SSLv3 write client certificate A
20000417 035851:iagent(5355):SSL_connect:SSLv3 write client key exchange A
20000417 035851:iagent(5355):SSL_connect:SSLv3 write certificate verify A
20000417 035851:iagent(5355):SSL_connect:SSLv3 write change cipher spec A
20000417 035851:iagent(5355):SSL_connect:SSLv3 write finished A
20000417 035851:iagent(5355):SSL_connect:SSLv3 flush data
20000417 035852:iagent(5355):SSL_connect:SSLv3 read finished A
20000417 035852:iagent(5355):SSL connection using DES-CBC3-SHA
20000417 035852:iagent(5355):SSL Session:
20000417 035852:iagent(5355):SSL Handshake DONE
```

Example 28. An example of Debug SSL Handshake State Messages (Client side)

```
20000417 035850:iagent(5348):calling SSL_accept
20000417 035850:iagent(5348):Starting SSL handshake (with SSL_accept)
20000417 035850:iagent(5348):SSL_accept:before/accept initialization
20000417 035850:iagent(5348):SSL_accept:SSLv3 read client hello A
20000417 035850:iagent(5348):SSL_accept:SSLv3 write server hello A
20000417 035850:iagent(5348):SSL_accept:SSLv3 write certificate A
20000417 035850:iagent(5348):SSL_accept:SSLv3 write certificate request A
20000417 035851:iagent(5348):SSL_accept:SSLv3 flush data
20000417 035851:iagent(5348):PKI_verify_cert
20000417 035851:iagent(5348):SSL_accept:SSLv3 read client certificate A
20000417 035852:iagent(5348):SSL_accept:SSLv3 read client key exchange A
20000417 035852:iagent(5348):SSL_accept:SSLv3 read certificate verify A
20000417 035852:iagent(5348):SSL_accept:SSLv3 read finished A
20000417 035852:iagent(5348):SSL_accept:SSLv3 write change cipher spec A
20000417 035852:iagent(5348):SSL_accept:SSLv3 write finished A
20000417 035852:iagent(5348):SSL_accept:SSLv3 flush data
20000417 035852:iagent(5348):DEBUG: Getting Client's certificate
20000417 035852:iagent(5348):SSL connection using DES-CBC3-SHA
20000417 035852:iagent(5348):SSL Handshake DONE
```

Example 29. An example of Debug SSL Handshake State Messages (Server side)

```
20000417 035850:iagent(5355):RAW Input Data ++++++++++++++++++++++++++++++++++++
20000417 035850:iagent(5355):ISA^00^IAGENT    ^00^TEST      ^ZZ^TP_1
^ZZ^TP_TWO         ^990629^1248^U^00303^000000001^0^T^>
GS^PO^CLECAPP^LWC^990629^1248^000001^X^003030
ST^864^000000001
BMG^28
DTM^097^990629^124849^21^19
MIT^000000001
MSG^This is a test message from TP_ONE to TP_TWO over IAgent
MSG^This is a test message line 2
MSG^This is a test message line 3
MSG^This is a test message line 4
MSG^This is a test message line 5
MSG^This is a test message line 6
MSG^This is a test message line 7
MSG^This is a test message line 8
MSG^This is a test message line 9
MSG^This is a test message line 10
MSG^This is a test message line 11
MSG^This is a test message line 12
MSG^This is a test message line 13
MSG^This is a test message line 14
MSG^This is a test message line 15
MSG^This is a test message line 16
MSG^This is a test message line 17
MSG^This is a test message line 18
MSG^This is a test message line 19
MSG^This is a test message line 20
CTT^0
SE^26^000000001
GE^1^000001
IEA^1^000000001

20000417 035850:iagent(5355):++++++ End of RAW Input Data +++++++++++++++++++++
```

Example 30. An example of Debug Transaction Data Messages

```
20000417 035851:iagent(5355):Certificate Verification: depth: 1, subject:
/C=US/ST=Connecticut/L=Old Lyme/O=Lymeware Corporation/OU=IAgent
Development/CN=LymewareDemoCA/Email=security@lymeware.com,issuer:
/C=US/ST=Connecticut/L=Old Lyme/O=Lymeware Corporation/OU=IAgent
Development/CN=LymewareDemoCA/Email=security@lymeware.com
20000417 035851:iagent(5355):PKI_verify_cert
20000417 035851:iagent(5355):Certificate Verification: depth: 0, subject:
/C=US/ST=Connecticut/O=Lymeware Corporation/OU=Iagent
development/CN=TP_2/Email=none@lymeware.com,issuer: /C=US/ST=Connecticut/L=Old
Lyme/O=Lymeware Corporation/OU=IAgent
Development/CN=LymewareDemoCA/Email=security@lymeware.com
```

Example 31. An example of Debug Certificate Messages (Client side)

```
20000417 035851:iagent(5348):Certificate Verification: depth: 0, subject:
/C=US/ST=Connecticut/O=Lymeware Corporation/OU=Iagent
development/CN=TP_1/Email=none@lymeware.com,issuer: /C=US/ST=Connecticut/L=Old
Lyme/O=Lymeware Corporation/OU=IAgent
Development/CN=LymewareDemoCA/Email=security@lymeware.com
20000417 035851:iagent(5348):SSL_accept:SSLv3 read client certificate A
20000417 035852:iagent(5348):DEBUG: Getting Client's certificate
Client certificate:
-----BEGIN CERTIFICATE-----
```

```
MIID+DCCA2GgAwIBAgIBDDANBgkqhkiG9w0BAQQFADCBsTELMAkGA1UEBhMCVVMx
FDASBgNVBAgTC0Nvbm5lY3RpY3V0MREwDwYDVQQHEwhPbGQgTHltZTEdMBsGA1UE
ChMUTHltZXdhcmUgQ29ycG9yYXRpb24xGzAZBgNVBAsTEklBZ2VudCBEZXZlbG9w
bWVudDEXMBUGA1UEAxMOTHltZXdhcmVEZW1vQ0ExJDAiBgkqhkiG9w0BCQEWFXNl
Y3VyaXR5QGx5bWV3YXJlLmNvbTAeFw05OTEyMTYwMjA5NTBaFw0wMDEyMTUwMjA5
NTBaMIGQMQswCQYDVQQGEwJVUzEUMBIGA1UECBMLQ29ubmVjdGljdXQxHTAbBgNV
BAoTFEx5bWV3YXJlIENvcnBvcmF0aW9uMRswGQYDVQQLExJJYWdlbnQgZGV2ZWxv
cG1lbnQxDTALBgNVBAMUBFRQXzExIDAeBgkqhkiG9w0BCQEWEW5vbmVAbHltZXdh
cmUuY29tMIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQDiNrXXQSCPQ3LqmCg0
6ASZbAkaWMordDqsNxIc7RJcdam5G2dkD71Bb7EDMoiKReEUK/sxRVAKNBS13KIi
cp7p137nZ7db/IidLH1GFdY5nttbXYE+G/kObywoSYXu7ImBvobkhkD8XjcM1HyU
dMnZTj3+Qc9hFSlozsaTN6QK3QIDAQABo4IBPTCCATkwCQYDVR0TBAIwADAsBglg
hkgBhvhCAQ0EHxYdT3BlblNTTCBHZW5lcmF0ZWQgQ2VydGlmaWNhdGUwHQYDVR0O
BBYEFN54YgJJsAq801pCwIHm50F7Zpf9MIHeBgNVHSMEgdYwgdOAFC7/bnL1D0B+
O4ABurW9u2VQfNjwoYG3pIG0MIGxMQswCQYDVQQGEwJVUzEUMBIGA1UECBMLQ29u
bmVjdGljdXQxETAPBgNVBAcTCE9sZCBMeW1lMR0wGwYDVQQKExRMeW1ld2FyZSBD
b3Jwb3JhdGlvbjEbMBkGA1UECxMSSUFnZW50IERldmVsb3BtZW50MRcwFQYDVQQD
Ew5MeW1ld2FyZURlbW9DQTEkMCIGCSqGSIb3DQEJARYVc2VjdXJpdHlAbHltZXdh
cmUuY29tggEAMA0GCSqGSIb3DQEBBAUAA4GBAG0Xs0LSHrPwhn+dajBZlstUQxn+
4rZ5FCoT2TIX2yeijgqGCBN7vBWiad3gRb3KD6huY1V/1rs9v7V0yeRsIp9iJRZm
w05vcB6v9vvIasHTkLqhw53JK4Jv7RfaVkGfMyowVA1rFH22VsrVfewriAD+cFmg
6j7E9l+GPKCy0dPg
```

```
-----END CERTIFICATE-----
```

Example 32. An example of Debug Certificate Messages (Server side)

```
       subject: /C=US/ST=Connecticut/O=Lymeware Corporation/OU=Iagent
development/CN=TP_1/Email=none@lymeware.com
       issuer: /C=US/ST=Connecticut/L=Old Lyme/O=Lymeware Corporation/OU=IAgent
Development/CN=LymewareDemoCA/Email=security@lymeware.com

Certificate:
    Data:
Version: 3 (0x2)
        Serial Number: 12 (0xc)
        Signature Algorithm: md5WithRSAEncryption
        Issuer: C=US, ST=Connecticut, L=Old Lyme, O=Lymeware Corporation,
OU=IAgent Development, CN=LymewareDemoCA/Email=security@lymeware.com
        Validity
            Not Before: Dec 16 02:09:50 1999 GMT
            Not After : Dec 15 02:09:50 2000 GMT
        Subject: C=US, ST=Connecticut, O=Lymeware Corporation, OU=Iagent
development, CN=TP_1/Email=none@lymeware.com
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
            RSA Public Key: (1024 bit)
                Modulus (1024 bit):
                    00:e2:36:b5:d7:41:20:8f:43:72:ea:98:28:34:e8:
                    04:99:6c:09:1a:58:ca:2b:74:3a:ac:37:12:1c:ed:
                    12:5c:75:a9:b9:1b:67:64:0f:bd:41:6f:b1:03:32:
                    a2:22:72:9e:e9:d7:7e:e7:67:b7:5b:fc:88:9d:2c:
                    7d:46:15:d6:39:9e:db:5b:5d:81:3e:1b:f9:0e:6f:
                    2c:28:49:85:ee:ec:89:81:be:86:e4:86:40:fc:5e:
                    37:0c:d4:7c:94:74:c9:d9:4e:3d:fe:41:cf:61:15:
                    29:68:ce:c6:93:37:a4:0a:dd
                Exponent: 65537 (0x10001)
        X509v3 extensions:
            X509v3 Basic Constraints:
                CA:FALSE
            Netscape Comment:
                OpenSSL Generated Certificate
            X509v3 Subject Key Identifier:
                DE:78:62:02:49:B0:0A:BC:D3:5A:42:C0:81:E6:E7:41:7B:66:97:FD
            X509v3 Authority Key Identifier:

keyid:2E:FF:6E:72:F5:0F:40:7E:3B:80:01:BA:B5:BD:BB:65:50:7C:D8:F0
                DirName:/C=US/ST=Connecticut/L=Old Lyme/O=Lymeware
Corporation/OU=IAgent Development/CN=LymewareDemoCA/Email=security@lymeware.com
                serial:00

    Signature Algorithm: md5WithRSAEncryption
        6d:17:b3:42:d2:1e:b3:f0:86:7f:9d:6a:30:59:96:cb:54:43:
        19:fe:e2:b6:79:14:2a:13:d9:32:17:db:27:a2:8e:0a:86:08:
        13:7b:bc:15:a2:69:dd:e0:45:bd:ca:0f:a8:6e:63:55:7f:d6:
        1e:af:f6:fb:c8:6a:c1:d3:90:ba:a1:c3:9d:c9:2b:82:6f:ed:
        17:da:56:41:9f:33:2a:30:54:0d:6b:14:7d:b6:56:ca:d5:7d:
        ec:2b:88:00:fe:70:59:a0:ea:3e:c4:f6:5f:86:3c:a0:b2:d1:
        d3:e0
```

Example 33. An example of Debug Certificate Messages (Server side) (continued)

```
20000417 035850:iagent(5355):Adding IA specific ASN.1 Object Identifiers ++++++
20000417 035850:iagent(5355):
1.3.6.1.4.1.3576.7          IA_X12       eciaAscX12Edi
1.3.6.1.4.1.3576.8          IA_EDIFACT   eciaEdifact
1.3.6.1.4.1.3576.9          IA_NON_EDI   eciaNonEdi
1.3.6.1.4.1.3576.7.1        IA_EDI_1     plainEDImessage
1.3.6.1.4.1.3576.7.2        IA_EDI_2     signedEDImessage
1.3.6.1.4.1.3576.7.5        IA_EDI_3     integrityEDImessage
1.3.6.1.4.1.3576.7.65       IA_R_1       iaReceiptMessage
1.3.6.1.4.1.3576.7.97       IA_S_1       iaStatusMessage
20000417 035850:iagent(5355):++++++ End of IA ASN.1 OIDs +++++++++++++++++++++
20000417 035850:iagent(5355):Number of new ASN.1 Objects added = 8
```

Example 34. An example of Debug ASN1 Messages

```
20000417 035850:iagent(5355):ASN1_parse returned a 1
    0:d=0  hl=4 l=1418 cons: SEQUENCE
    4:d=1  hl=2 l=   9 prim:  OBJECT            :signedEDImessage
   15:d=1  hl=4 l=1403 cons:  SEQUENCE
   19:d=2  hl=2 l=   1 prim:   INTEGER           :00
   22:d=2  hl=2 l=   7 cons:   SET
   24:d=3  hl=2 l=   5 prim:    OBJECT            :SHA1digestAlgorthm
   31:d=2  hl=4 l=1176 cons:   SEQUENCE
   35:d=3  hl=2 l=   9 prim:    OBJECT            :plainEDImessage
   46:d=3  hl=4 l=1161 prim:    OCTET STRING      :ISA^00^IAGENT    ^00^TEST
^ZZ^TP_1           ^ZZ^TP_TWO         ^990629^1248^U^00303^000000001^0^T^>
GS^PO^CLECAPP^LWC^990629^1248^000001^X^003030
ST^864^000000001
BMG^28
DTM^097^990629^124849^21^19
MIT^000000001
MSG^This is a test message from TP_ONE to TP_TWO over IAgent
MSG^This is a test message line 2
MSG^This is a test message line 3
MSG^This is a test message line 4
MSG^This is a test message line 5
MSG^This is a test message line 6
MSG^This is a test message line 7
MSG^This is a test message line 8
MSG^This is a test message line 9
MSG^This is a test message line 10
MSG^This is a test message line 11
MSG^This is a test message line 12
MSG^This is a test message line 13
CTT^0
SE^23^000000001
GE^1^000001
IEA^1^000000001
```

Example 35. An example of Debug Parsed ASN1 Messages

```
1211:d=2  hl=3 l= 208 cons:   SET
1214:d=3  hl=3 l= 205 cons:    SEQUENCE
1217:d=4  hl=2 l=   1 prim:     INTEGER           :00
1220:d=4  hl=2 l=  51 cons:     SEQUENCE
1222:d=5  hl=2 l=  13 cons:      SEQUENCE
1224:d=6  hl=2 l=  11 cons:       SET
1226:d=7  hl=2 l=   9 cons:        SEQUENCE
1228:d=8  hl=2 l=   3 prim:         OBJECT            :countryName
1233:d=8  hl=2 l=   2 prim:         PRINTABLESTRING   :US
1237:d=5  hl=2 l=  31 cons:      SEQUENCE
1239:d=6  hl=2 l=  29 cons:       SET
1241:d=7  hl=2 l=  27 cons:        SEQUENCE
1243:d=8  hl=2 l=   3 prim:         OBJECT            :organizationName
1248:d=8  hl=2 l=  20 prim:         PRINTABLESTRING   :Lymeware Corporation
1270:d=5  hl=2 l=   1 prim:      INTEGER           :0C
1273:d=4  hl=2 l=   5 prim:     OBJECT            :SHA1digestAlgorthm
1280:d=4  hl=2 l=   9 prim:     OBJECT            :rsaEncryption
1291:d=4  hl=3 l= 128 prim:     OCTET STRING
```

Example 36. An example of Debug Parsed ASN1 Messages (continued)

```
20000417 035852:iagent(5348):== Peer RSA Public Key ===========================
      Modulus (1024 bit):
          00:e2:36:b5:d7:41:20:8f:43:72:ea:98:28:34:e8:
          04:99:6c:09:1a:58:ca:2b:74:3a:ac:37:12:1c:ed:
          12:5c:75:a9:b9:1b:67:64:0f:bd:41:6f:b1:03:32:
          88:8a:45:e1:14:2b:fb:31:45:50:0a:34:14:b5:dc:
          a2:22:72:9e:e9:d7:7e:e7:67:b7:5b:fc:88:9d:2c:
          7d:46:15:d6:39:9e:db:5b:5d:81:3e:1b:f9:0e:6f:
          2c:28:49:85:ee:ec:89:81:be:86:e4:86:40:fc:5e:
          37:0c:d4:7c:94:74:c9:d9:4e:3d:fe:41:cf:61:15:
          29:68:ce:c6:93:37:a4:0a:dd
      Exponent: 65537 (0x10001)
20000417 035852:iagent(5348):== END of Peer RSA Public Key ====================
```

Example 37. An example of Debug RSA Key Messages

```
20000417 035852:iagent(5348):No receipt required for TP_1
20000417 035852:iagent(5348):Basic receipt required for TP_1
20000417 035852:iagent(5348):Digest receipt required for TP_1
20000417 035852:iagent(5348):Signed receipt required for TP_1

20000417 035852:iagent(5348):Digest verified
20000417 035852:iagent(5348):RSA digital signature verified
```

Example 38. An example of Debug Receipt Messages

```
20000417 035852:iagent(5348):SSL Handshake start time = 035852
20000417 035854:iagent(5348):SSL Handshake end time = 035854
20000417 035855:iagent(5348):Total Handshake time = 02.5 Seconds

20000417 040012:iagent(5348): Message Transport start time = 040012
20000417 040013:iagent(5348): Message Transport end time = 040013
20000417 040014:iagent(5348):Total Message Transport time = 01.1 Seconds
```

Example 39. An example of IA Timer Messages

CHAPTER 28 – PRODUCT ERROR MESSAGES

Descriptions and possible solutions for specific IAgent **Transaction Errors:**

CANNOT_ACCEPT_CLIENT_HANDSHAKE

This message is usually caused by SSL version or cipher mismatch from IA Client or Remote IA.

CANNOT_ALLOCATE_CLIENT_SOCKET

This message is caused by system resource problems.

CANNOT_ALLOCATE_CTX

This message is caused by system resource problems.

CANNOT_ALLOCATE_SSLCTX

This message is caused by system resource problems.

CANNOT_CONNECT_TO_REMOTE_SERVER

This message can be caused by several issues, including connectivity, invalid trading partner configuration settings, and invalid EDI data.

<u>Connectivity Issues:</u>
The IAgent client processes need to be able to connect to the remote IA server, using the ports found in the Trading Partner configuration file. Connections may have problems if any of the following is true:

- The remote IA is not reachable via TCP/IP. For successful operation all remote IA servers must be "reachable" via the local IA client machine. Dialup or intermittent connections are not supported. Possible problems:
  - the remote IA server is down or not connected to the IP network,
  - the local machine has lost it's own connection to the network or the specific network connection used to reach the remote IA server.
- The remote IA may be behind a firewall, and the firewall may not allow access on the required ports.
- The local IA machine may be behind a firewall, and the firewall may be doing any of the following, which could cause problems:
  - block ports or addresses,
  - translate source IP addresses (NAT) which may "confuse" remote firewalls,
  - or proxy requests through a proxy server.

<u>Invalid Trading Partner configuration setting Issues:</u>
IAgent uses the information in the Trading Partner configuration file to determine the destination of outbound IA messages. Incorrect information in any of the following fields can cause this error:

- ip_address
- standard_ip_port
- hipri_ip_port

Invalid EDI data:

IAgent reads the ISA header of the input EDI message to determine the remote Trading Partner and connection information for that partner's IA server. The field of interest is ISA08, The Trading Partner Receiver ID (**TP_TWO** in the example below).

```
ISA^00^IAGENT    ^00^TEST     ^ZZ^TP_1         ^ZZ^TP_TWO       ^990629^1248^U^
00303^000000001^0^T^>
```

Example 40. Example EDI ISA Segment (wrapped)

CANNOT_FIND_ISSUERNAME_IN_CERT
This message is caused by error parsing/reading the supplied Remote IA certificate.

CANNOT_FIND_SUBJECTNAME_IN_CERT
This message is caused by error parsing/reading the supplied Remote IA certificate.

CANNOT_GENERATE_EDI_MSG_FILE
This message can be caused by system resource problems, file permission errors or an existing duplicate file.

CANNOT_GENERATE_IA_STATUS_MSG
This message can be caused by system resource problems, file permission errors or an existing duplicate file.

CANNOT_GENERATE_RECEIPT
This message can be caused by system resource problems, file permission errors or an existing duplicate file.

CANNOT_OPEN_OUTPUT_FILE
This message can be caused by system resource problems, file permission errors or an existing duplicate file.

CANNOT_PARSE_ASN1
This message can be caused by invalid or incomplete IA messages from remote trading partners. There are known problems with specific versions of specific ILEC IA systems. This error can also be caused by an IA version/issue mis-match (e.g. IA Issue 2 is not compatible with IA Issue 3).

CANNOT_READ_OWN_CERT
This message can be caused by system resource problems, file permission errors or a missing or misnamed file.

CANNOT_READ_OWN_KEY
This message can be caused by system resource problems, file permission errors or a missing or misnamed file.

CANNOT_READ_SSL_SERVER

CANNOT_REMOVE_PENDING_RECEIPT
This message can be caused by system resource problems, file permission errors or a missing or misnamed receipt file. This may also be caused by a malformed pending receipt file.

CANNOT_SAVE_EDI_MSG_FILE
This message can be caused by system resource problems, file permission errors or an existing duplicate file.

CANNOT_SAVE_IASTATUS_OUT_FILE
This message can be caused by system resource problems, file permission errors or an existing duplicate file.

CANNOT_SAVE_IASTATUS_TO_PIPE
This message can be caused by system resource problems, file/pipe permission errors or a halted read process on the pipe.

CANNOT_SAVE_RECEIPT_OUT_FILE
This message can be caused by system resource problems, file permission errors or an existing duplicate file.

CANNOT_SAVE_RECEIPT_TO_PIPE
This message can be caused by system resource problems, file/pipe permission errors or a halted read process on the pipe.

CANNOT_START_CLIENT_HANDSHAKE

CANNOT_VERIFY_CLIENT_CERT

CANNOT_WRITE_EDI_MSG_TO_PIPE
This message can be caused by system resource problems, file/pipe permission errors or a halted read process on the pipe.

CANNOT_WRITE_EDI_MSG_TO_SOCKET

CANNOT_WRITE_OUTPUT_FILE
This message can be caused by system resource problems, file permission errors or an existing duplicate file.

CANNOT_WRITE_SSL_CLIENT

CANNOT_WRITE_TIBCO_EDI_MSG

CLIENTS_CA_CERTS_MISSING

CLIENTS_OWN_CERT_MISSING

CNAME_MISMATCH

INCORRECT_MESSAGE_TYPE_SENT_FROM_TP

ISA_SEG_MISSING_FROM_INPUT_RECEIPT_MSG

MESSAGE_BAD_SIGNITURE_FOUND

MESSAGE_INTEGRITY_BROKEN

NO_CLIENT_CERT_SENT

NO_ISAID_FOUND_IN_TPFILE

NO_PENDING_REC_FOUND

RECEIPT_BAD_SIGNITURE_FOUND

RECEIPT_INTEGRITY_BROKEN

RECEIVED_MESSAGE_FROM_INACTIVE_TP

TP_ID_MISSING_FROM_INPUT_IA_STATUS_MSG

UNKNOWN_IPADDR_SENT_FROM_CLIENT

UNKNOWN_ISAID_FOUND

UNKNOWN_ISAID_FOUND_IN_CERT

UNKNOWN_MSG_TYPE_IN_TPFILE

UNKNOWN_RECEIPT_MSG_TYPE_IN_TPFILE

UNKNOWN_TYPE_OF_MSG_FOR_TP

UNSUPPORTED_INPUT_MSG_FOUND

UNSUPPORTED_MESSAGE_TYPE

CHAPTER 29 – SECURITY ISSUES

Post-Installation Security Tasks

Re-Run tripwire and store the tripwire database to an external device (floppy/tape).

Make a final full system backup, and reinstall from the backup to test it.

Run Satan/Saint or another security configuration verification tool and fix what insecure network configurations are discovered.

General Security Suggestions

Limit the application software installed on the IAgent machine.

Disable all unnecessary port services, as found in /etc/services.

Disable all BSD r* utilities (several can be replaced with more secure alternate open source versions).

Replace RPC services with RPC-Bind.

Insure that both bind and sendmail are running the latest versions available for the current operating system. The newest versions have significantly more security than past versions.

Do not allow SNMP, SMTP, HTTP, IMAP or POP access from outside of the local LAN.

Do add a firewall between the IAgent machine and the outside connections. Configure the firewall using the suggestions below.

Suggested Security Tools

The following tools are strongly suggested for inclusion into a security site maintenance plan, and should be used on a periodic basis.

Tool Name	**Tool Descriptions**
Satan	Detects insecure system and network configurations
Saint	Detects insecure system and network configurations
Tripwire	Intrusion Detection System (IDS)
sudo	Monitored root Access
SSH	Secure Shell
Nmap	IP Port Scanner

Table 16. Suggested Security Tools

Certificate and Key Issues

Always save your private keys and certificates to an external device (floppy/tape) for archiving. Additional Off-site archiving is suggested.

It is suggested to use a minimum of a 768 bit RSA key pair. 1024 bits is even more secure and should be used.

The use of a major commercial third party Certificate Authority is suggested.

The use of a externally stored domain name service (DNS) hostname and Common Name in the certificate request is recommended.

Operating System Issues

IAgent does use shared memory in its operation and does require the shared memory and semaphores be enabled to operate properly. The minimum shared memory segment size should support at least 128K bytes.

Other Access Issues

It is suggested that all forms of direct network access (including telnet, rlogin, ftp, finger, rwho, etc.) be minimized to only those specifically required for operation of the IAgent machine.

FTP should be replaced with SCP, and telnet should be replaced with SSH.

It is recommended that all indirect network access, particularly mounted or shared network file systems (e.g. SMB, NFS, AFS, DFS) should not be used for both performance and security reasons.

Suggested Firewall rules-set

The firewall rules should only allow the specific trading partners IP addresses access to only the local IAgent via specific standard and high priority ports. All other access should be blocked form these addresses.

ICMP (ping) packets should probably not be allowed, due to the amount of information that can be determined from ICMP fragment pings. More information on this interesting but hazardous loophole can be found at www.insecure.com.

The use of Network Address Translation (NAT) should be used if available. This would allow outside elements (including remote IA trading partners) to access virtual IP addresses which translate to specific real internal (to the local LAN) IP addresses.

The use of a Demilitarized Zone (DMZ) external network is suggested.

CHAPTER 30 – HOW TO GET HELP

This chapter explains how to contact Lymeware Product Support if you need assistance with your IAgent product.

Scope of Support Services

Lymeware Product Support can provide assistance and information for the following:

Installing the IAgent product

IAgent product questions

Software revisions and upgrades

Implementing a specific feature

How to use the IAgent product

The status of your support call

Requests for product enhancement

Unfortunately, we cannot assist you with problems involving the following, but we may be able to suggest a next step or another vendor to call:

Your hardware

Your operating system or other system software

Your application or user-written programs

Software not developed by Lymeware Corporation

Scripts written by Lymeware consultants, service partners, or other third parties.

Try this first

Before you call Lymeware Product Support, use your software manuals (including this manual) to locate the section that documents the program or feature where you are having problems. The documentation may explain the software's behavior or give you insight to help you solve the problem.

Contact Lymeware Product Support

Two e-mail addresses are available for IAgent product support or to report a potential bug in the software or documentation. Please use the following addresses:

Support@lymeware.com for all technical inquires and problem reports, including documentation issues from customers with support contracts. Customers should include relevant contact details, including company name and phone number, in initial message to speed processing. Messages that are continuations of an existing problem report should include the problem report ID in the subject line. Customers without support contracts with Lymeware Corporation should not use this address, but should contact their distributor directly.

Bugs@lymeware.com for bug reports and documentation problems.

Bug reports on software releases are always welcome. These may be sent by any means, but e-mail to the bug reporting address listed above is preferred. Please send proposed fixes and successful workarounds with the report if possible. Additional useful information would include IAgent software version, hardware description, operating system version and patches, screen dumps, relevant sections of logs and configuration files, and failed messages files. Any reports will be acknowledged, but further action is not guaranteed. Any changes resulting from bug reports may be included in future releases.

Appendixes

The final part of this manual contains appendixes with additional information on IAgent administration, including worksheets and other maintenance issues.

APPENDIX A – CONFIGURATION WORKSHEETS AND FORMS

This appendix contains worksheets that should be used to complete specific tasks during the installation, configuration and maintenance of your IAgent system. The following table describes each worksheet.

IAgent Pre-Installation Worksheet	This worksheet is used to identify all the information and resources required for installing the IAgent product.
IAgent Connectivity Worksheet	This worksheet is used to capture all networking information required to complete a network diagram for the local IAgent system and its connection to remote trading partners.
IAgent Configuration Worksheet	This worksheet is used to identify all required and optional configuration settings and to assist the user in configuration file management.
Trading Partner Worksheet	This worksheet is used to capture all required remote trading partner information to support trading partner management.
Certificate Request Worksheet	This worksheet is used to capture all the usual information required for requesting a certificate from a commercial Certificate Authority.
License Request Form	This form is required for issuing of an evaluation or permanent license (required for IAgent operation)
Problem Reporting Form	This form should be used for any problems encountered which can be reported back to Lymeware

Table 17. IAgent Product Worksheets

These worksheets may be copied for use in maintaining your IAgent system.

It is expected that completed worksheets will be used during product instillation and configuration and should be saved with other machine documentation for future use. Digital version of these worksheets are available at the Lymeware web site (http://www.lymeware.com/products.html).

IAgent Pre-Installation Worksheet **version 1.1**
For IAgent Machine information collection to be used before product installation
Machine specific information
Hostname:
FQDN:
IP address:
Host id (if SPARC platform): Physical Location:
Machine Hardware information
CPU Type:
Number of CPUs:
Model:
Amount of System Memory:
Disk space:
CD-ROM Device:
Machine Software information
Operating system version:
Operating system patches installed:
Optional software installed: Perl?: Tibco?: Web Browser?: Other:
System administration Contact information
Primary Name:
Primary Telephone Number:
Primary Pager Number:
Primary E-mail address:
Second Name:
Second Telephone Number:
Second Pager Number:
Second E-mail address:
Primary Root availability: (Y/N) Second Root availability: (Y/N)

IAgent Connectivity Worksheet **version 1.2**	
For IAgent Machine information collection to be used during connectivity testing	
Machine specific information	
Hostname:	
FQDN:	
IP address:	
Standard IA port:	High Priority IA port:
Net mask: Gateway IP address:	
Firewall Machine information	
IP Address:	
Enabled ports:	
Is ICMP (echo/ping packets) allowed to pass through?: (Y/N)	
Network Address Translation (NAT) supported?: (Y/N)	
Network Address Translation (NAT) information	
External IP address for IAgent machine:	
External Standard IA port:	
External High Priority IA port:	
Router information	
IP Address:	
Subnet Limit?:	
Port filtering supported?:	
Port filtering for IA ports disabled?:	

IAgent Configuration Worksheet **version 1.1**
For IAgent configuration information collection. To be used during before building the configuration file.
Machine specific information
Hostname:
FQDN:
IP address: Standard IA port: High Priority IA port:
Certificate specific information
RSA Private Key file(Must be in PEM format):
CSR file: Format (PEM/DER):
Server Certificate file (Must be in PEM format):
CA Certificate file (Must be in PEM format):
Interface specific options
Interface to support:
Directory Interface specific options
Input Directory: Output Directory: Directory Polling Delay (in seconds):
Named Pipe Interface specific options
Input Pipe Name: Output Pipe Name:
Socket Interface specific options
Input Socket Port: Output Socket IP Address and Port:
Tibco Interface specific options
Input Subject Name: Output Subject Name:
Advanced options
Receipt Mode (internal or external):
Resumable Session Cache Size:
Source of entropy (filename):
Trading Partner Configuration File:
Will allow any type of IA message from any Trading Partner? (Sloppy Allow):
Will allow messages from inactive Trading Partner? (Sloppy Inactive):
Will allow duplicate receipts (and duplicate BES messages)? (Ignore Duplicate Receipts):
Send IA Status messages to Remote trading Partners on error? (Send Status):

IAgent Trading Partner Worksheet **version 1.3**		
This worksheet is to be used for each local and remote trading partner to be supported by the IAgent system. Fill this out before generating/modifying the trading partner configuration file. Valid field values will be found in *Chapter 10* of the ***IAgent's User's Guide***.		
Trading Partner Company specific information		
Full Company Name:		
Street Address:		
City:	State:	Zip:
Contact Person:		
Contact Phone Number:		
Contact FAX Number:		
Contact E-mail Address:		
Trading Partner specific information		
Local Trading Partner ID (as CLEC in ISA segment):		
Remote Trading Partner ID (as ILEC in ISA segment):		
Trading Partner Common Name (CN):		
Trading Partner's IA Server IP Address:		
Trading Partner's Standard Priority Port:		
Trading Partner's High Priority Port:		
Trading Partner's SSL Session Timeout (in seconds):		
Trading Partner's sending IA message type:		
Trading Partner's receiving IA message type:		
Trading Partner's sending IA receipt type:		
Trading Partner's receiving IA receipt type:		
Trading Partner's IA Server status:		
Maintenance information		
Date added to IAgent configuration:		
Added by who:		
This information should be saved after being added to the IAgent trading partner configuration file.		

Certificate Request Worksheet **version 1.2**		
For requesting a valid commercial server certificate		
Customer specific information		
Full Company Name:		
Street Address:		
City:	State:	Zip:
Contact Person:		
Contact Phone Number:		
Contact FAX Number:		
Contact E-mail Address:		
Certificate specific information		
Target Machine Hostname:		
Common Name (CN):		
Organization Name (O):		
Organizational Unit Name (OU):		
Locality (city):		
State or Province Name:		
Country Name (C):		
Certificate Authority specific information		
CA Contact information:		
This Certificate Request Worksheet should be used to complete the specific forms provided by your selected Certificate Authority. Please contact your CA for any additional information that may be required.		

License Request Form

A specific license data file will be required to run the IAgent system. Lymeware or your distributor will supply this license file if the following information is supplied:

IAgent License Request Form **version 1.2**
For requesting a valid commercial or evaluation IAgent product license
Customer specific information
Customer (Company) Name:
Lymeware Product Name:
Lymeware Product Version:
Lymeware Product Options:
Target Machine IP Address:
Target Machine Host ID:
Target Machine Make and Model:
Target Machine Operating System and Version:
Contact Person:
Contact Phone Number:
Contact E-mail Address:
This License Request Form may be faxed to Lymeware Corporation at (801) 383-9021 or the same information may be e-mailed to support@lymeware.com.

The license file will be delivered to the Contact E-Mail Address. The license file must be installed at `/opt/iagent` as `license.dat` and should be owned by root.

IAgent Problem Report Form   **version 1.1**
For reporting IAgent product problems
Customer specific information
Your Name:
Your Company Name:
Your Telephone Number:
Your E-mail Address:
Your IAgent product version:
Your IAgent platform:
Any software add-ons to your IAgent system:
A detailed description of the problem:
The sequence of steps that led to the problem:
Actions you have taken to diagnose or resolve the problem:
This Problem Report Form may be faxed to Lymeware Corporation at (801) 383-9021   or the same information may be e-mailed to support@lymeware.com.

APPENDIX B – CERTIFICATES AND THE GEN_CSR TUTORIAL

X.509 Certificates and Certificate Signing Request (CSR) Creation

X.509 Certificates

Interactive Agent (IA) trading partners recognize and verify the identity of peer IA trading partners with X.509 certificates. But what is a certificate and what does X.509 mean? Simply, a certificate associates a public key (half of the public/private key pair) with the real identity of an individual, trading partner, server, company, or other entity. The X.509 refers to the standard (Recommendation X.509: The Directory - Authentication Framework. 1988 CCITT) which defined the framework of the certificate elements (also known as certificate fields). Netscape adopted the X.509 certificate structure for its Secured Socket Layer (SSL) protocol design. IA message systems also use X.509 certificates to verify the identity and public key of all validated IA trading partners.

If each trading partner has a certificate which validates the other's identity, confirms the public key, and is signed by a trusted agency, then they both will be assured that they are communicating with whom they think they are. Such a trusted agency is called a *Certificate Authority*, and certificates are used for authentication.

Contents of Certificates

A certificate associates a public key with the real identity of an individual, or some other entity, known as the subject. As shown in Table 1, information about the subject includes identifying information (the distinguished name), and the public key. It also includes the identification and signature of the Certificate Authority that issued the certificate (Issuer information), and the period of time during which the certificate is valid. It may have additional information (or extensions) as well as administrative information for the Certificate Authority's use, such as a serial number, Dun and Bradstreet (DUNs) company number or other internal information.

Subject	Distinguished Name, Public Key
Issuer	Distinguished Name, Signature
Period of Validity	Not Before Date, Not After Date
Administrative Information	Version, Serial Number
Extended Information	Optional Requestor's Extensions Optional Issuer's Extensions Optional Application Extensions

Table 18. Certificate Information Elements

A distinguished name is used to provide an identity in a specific context -- for instance, an individual might have a personal certificate as well as one for their identity as an employee. Distinguished names are defined by the X.509 standard that defines the fields, field names, and abbreviations used to refer to the fields.

Field	Abbreviation	Description	Example
Common Name	CN	Name being certified. Typically the fully qualified domain name (FQDN) (e.g. host.company.com)	CN=www. lymeware.com
Organization or Company	O	Name is associated with this organization	O=Lymeware Corporation
Organizational Unit	OU	Name is associated with this organization unit, such as a department. This field is often blank.	OU=Research Lab
City / Locality	L	Name is located in this City	L=Old Lyme
State / Province	SP	Name is located in this State or Province	SP=Connecticut
Country	C	Name is located in this Country (ISO code)	C=US

Table 19. Distinguished Name Information

A Certificate Authority may define a policy specifying which distinguished field names are optional, and which are required. It may also place requirements upon the field contents, as may users of certificates. As an example, a Netscape browser requires that the Common Name for a certificate representing a server have a name which matches a regular expression for the domain name of that server, such as *www.lymeware.com*.

The binary format of a certificate is defined using the ASN.1 notation. This notation defines how to specify the contents, and encoding rules define how this information is translated into binary form. The binary encoding of the certificate is defined using Distinguished Encoding Rules (DER), which are based on the more general Basic Encoding Rules (BER). For those transmissions that cannot handle binary, the binary form may be translated into an ASCII form by using base-64 encoding. This encoded version is also called PEM (from Privacy Enhanced Mail) encoded, when placed between the following lines:

```
-----BEGIN CERTIFICATE-----
-----END CERTIFICATE-----
```

Certificate Authorities

By first verifying the information in a certificate request before granting the certificate, the Certificate Authority assures the identity of the private key owner of a key-pair. For instance, if Alice requests a personal certificate, the Certificate Authority must first make sure that Alice really is the person the certificate request claims.

But what is a Root Level Certificate Authority?

As noted earlier, each certificate requires an issuer to assert the validity of the identity of the certificate subject, up to the top-level Certificate Authority. This presents a problem: Since this is who vouches for the certificate of the top-level authority, which has no issuer? In this unique case, the certificate is "self-signed", so the issuer of the certificate is the same as the subject. As a result, one must exercise extra care in trusting a self-signed certificate. The wide publication of a public key by the root authority reduces the risk in trusting this key -- it would be obvious if someone else publicized a key claiming to be the authority.

A number of companies, such as VeriSign, and EnTrust have established themselves as certificate authorities. These companies provide the following services:

- Verifying certificate requests
- Processing certificate requests
- Issuing and managing certificates

To use the services of one of these companies, one must be able to generate a valid certificate request file (containing identification information and the RSA public key) in the format required. And that is exactly what is described in the next section.

Creating a Client or Machine Certificate Request

A *client certificate* is used to authenticate a client to a server. Creating and installing a client certificate is difficult because the client must generate a key-pair, keep the private key to itself, and send the public key to the certificate authority to be incorporated into a certificate request. Our utility, `gen_csr`, does just that! Once a signed certificate has been created using the Certificate Authority, this client certificate must be installed in the client so that the client may present it when needed. In IAgent operation the "client" certificate is often also called the machine certificate.

The general steps for creating a client certificate are as follows:

1. User generates a key pair (public and private key)
2. Private key is stored locally
3. User enters identification information and public key into a certificate signing request (CSR)
4. Submission of the request to a Certificate Authority:
   - 4.1. Certificate Authority verifies identity of user
   - 4.2. Creates a valid certificate (good for only a specific period of time and for a specific unique CN)
   - 4.3. And returns the new certificate back to the user

The Gen_csr Certificate Environment and Configuration File

`Gen_csr` (Generate Certificate Signing Request) is the supplied utility which builds RSA (PKCS#1) key pairs and (PKCS#10) certificate signing requests in either PEM or DER format. The following examples assume you:

- have installed the IAgent system,
- are working in the */opt/iagent* directory,
- And have available the Common Name (CN) of your local trading partner.

`Gen_csr` uses a configuration file (*request.cnf*) which supports multiple sections, so one configuration file may be used for several purposes. The **req** section of the configuration file is used when creating certificate requests with `gen_csr`, and supplies defaults and length limits for the various distinguished name fields. In our examples it has the configuration as shown in Example 40.

```
#
# Gen_csr example configuration file.
# This is being used for generation of certificate requests.
#

RANDFILE                = $ENV::HOME/.rnd

[ req ]
default_bits            = 768
default_keyfile         = privkey.pem
distinguished_name      = req_distinguished_name
attributes              = req_attributes

[ req_distinguished_name ]
countryName                     = Country Name (2 letter code)
countryName_default             = US
countryName_min                 = 2
countryName_max                 = 2

stateOrProvinceName             = State or Province Name (full name)
stateOrProvinceName_default     = Connecticut

localityName                    = Locality Name (e.g., city)

0.organizationName              = Organization Name (e.g., company)
0.organizationName_default      = Lymeware Corporation

organizationalUnitName          = Organizational Unit Name
#organizationalUnitName_default =

commonName                      = Common Name (e.g., YOUR FQDN or CN)
commonName_max                  = 15

emailAddress                    = Email Address
emailAddress_max                = 40

[ req_attributes ]
challengePassword               = A challenge password
challengePassword_min           = 4
challengePassword_max           = 20

#EOF
```

Example 41. gen_csr Configuration File: Request Section

Creating a Certificate Signing Request

The `gen_csr` command is used to create a PKCS#10 (Public-Key Cryptography Standards) certificate signing request. It also generates an RSA key pair (PKCS#1), and you may indicate the length of the RSA keys (in bits) with the **-rsakey** switch. The available command line options for gen_req are shown in Example 41.

```
./gen_csr -help
gen_csr v1.4
 - generates X.509v3 certificate signing requests (CSRs) and RSA key pairs
  Copyright (c) 2001 Lymeware Corporation, All rights reserved.
        using OpenSSL 0.9.6b

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

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


unknown option -help
./gen_csr [options]  >outfile
where options  are
 -reqform arg   request format - one of DER TXT PEM, with PEM as default
 -reqout arg    request output file, stdout as default
 -keyform arg   key file format - one of DER TXT PEM, with PEM as default
 -keyout arg    private key output file
 -rsakey bits   generate a new RSA key of 'bits' in size
 -config file   request template file.
 -text          text form of request
 -nodes         don't encrypt the output key
 -asn1-kludge   Output the 'request' in a format that is wrong but some CA's
                have been reported as requiring
                [ It is now always turned on but can be turned off with -no-
asn1-kludge ]
```

Example 42. Command line arguments for gen_csr

A PEM (text / base-64) certificate signing request may be created as shown in Example 42. Or a DER (binary ASN.1) certificate signing request may be created as shown in Example 43.

```
./tools/gen_csr -keyout newkey.pem -reqout newreq.pem -nodes

gen_csr v1.4
 - generates X.509v3 certificate signing requests (CSRs)and RSA key pairs
  Copyright (c) 2001 Lymeware Corporation, All rights reserved.
    using OpenSSL 0.9.6b

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

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


Using configuration from /opt/iagent/request.cnf
Generating a 768 bit RSA private key
...+++++
....+++++
writing new private key to 'newkey.pem'
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [US]:US
State or Province Name (full name) [Connecticut]:
Locality Name (eg, city) []:Old Lyme
Organization Name (eg, company) [Lymeware Corporation]:
Organizational Unit Name (eg, section) []:
Common Name (eg, YOUR FQDN or CN) []:www.lymeware.com
Email Address []:iagent@lymeware.com

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:none

Writing RSA private Key and X509 certificate request
```

Example 43. Using gen_csr to create a PEM Certificate Signing Request

```
./tools/gen_csr -keyout newkey.pem -reqout newreq.pem
        -reqform DER -nodes

gen_csr v1.4
 - generates X.509v3 certificate signing requests (CSRs)and RSA key pairs
  Copyright (c) 2001 Lymeware Corporation, All rights reserved.
    using OpenSSL 0.9.6b

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

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


Using configuration from /opt/iagent/request.cnf
Generating a 768 bit RSA private key
...+++++
....+++++
writing new private key to 'newkey.pem'
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [US]:US
State or Province Name (full name) [Connecticut]:
Locality Name (eg, city) []:Old Lyme
Organization Name (eg, company) [Lymeware Corporation]:
Organizational Unit Name (eg, section) []:
Common Name (eg, YOUR FQDN or CN) []:www.lymeware.com
Email Address []:iagent@lymeware.com

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:none

Writing RSA private Key and X509 certificate request
```

Example 44. Using gen_csr to create a DER Certificate Signing Request

Example 44 shows the results of a certificate signing request being created in *newreq.pem* (in this document certificates and keys are truncated). This is the text file (in PEM format) or binary file (in DER format) that you send to your selected Certificate Authority for certificate creation.

```
-----BEGIN CERTIFICATE REQUEST-----
MIGyMIGnAgEAMEIxCzAJBgNVBAYTAlVTMRQwEgYDVQQIEwtDb25uZWN0aWN1dDEd
MBsGA1UEChMUTHltZXdhcmUgQ29ycG9yYXRpb24wXDANBgkqhkiG9w0BAQEFAANL
ADBIAkEAy9g5SaqvZ2Bv82MftHF3Ns80xq452QV+vxzieQPOj8gjkS7SeShFTSoU
dEg0pWOX6R58HMFBRJ9KC8Rid03FNwIDAQABoAAwAwYBAAMBAA==
-----END CERTIFICATE REQUEST-----
```

Example 45. Sample PEM Certificate Signing Request

Example 45 shows a private key for the certificate being created in *newkey.pem*. This is the key to keep locally. **IMPORTANT**: Do not lose or let an intruder copy your local key or someone may be able to successfully impersonate you.

```
-----BEGIN RSA PRIVATE KEY-----
jY0TuymeQBjZFYfz1Lt0zlnPj8giZ2DZ84TByqe8LevjudN96sg3HLiWIBzelL+E
LYiqFCcW6RFO57qU5rkn6jAvrD7DGeUdNmfKoie3RisXSPC/hmoWE7P9xCVPvGkx
gXwOv5FtC3IPBjKiNFam1Hanu44yJi92yFR5BuA+jJKTLK5OY12aJHuEqTLC1J4D
ch4qsNTqeLSSyxOE2gX9+w2QDhnIuKKTjcSIuK8jNKfDArVaHed1IGM1pI5JCwoD
N552yNI6qjEwBgfbgWRsu8DYbMiGsSra37Jd+oJpZjQKr0bOTpghHTGlwC1ltX7u
IVfEU1iBfd7m2BEUc7e7lftfHgzAjvWStCFgkY1y3kg7Zh6jPALu0/DFYzmPtMZ+
VTft/d+49NoX5R+ZEM+eKJXSIoCUk7ZUriFPfkRsdDdcz7AfW99rHA==
-----END RSA PRIVATE KEY-----
```

Example 46. Sample Private Key

The CSR file should then be sent to the appropriate Certificate Authority, approved by both trading partners (the local and remote) in their joint implementation agreement. The Certificate Authority will then provide a signed certificate file with themselves as the issuers.

APPENDIX C – SAMPLE CONFIGURATION FILES

The following files may be used as samples but should not be used without modification. A digital (soft copy) of these files may be found under `/opt/iagent/examples` after installation.

System (or IAgent) configuration file

A simple example of the IAgent configuration file.

```
# iagent.conf - IAgent local process configuration file
# Copyright (C)1999 Lymeware Corporation, all rights reserved
# $Header: /export/home/kobar/iagent/RCS/iagent.conf,v 1.2 2000/01/24 05:47:15
kobar Exp $
#
# IAgent configuration file.  All entries (except configfile) override all
# default settings.  Any arguments set on the command line will overwrite
# anything in this file.
#

# default section ###########################################################

# this machine is the main test machine (frodo)
trading_partner               = LWC_TEST

iagent_home             = /opt/iagent/

# Define sections for each process to use (REQUIRED)
standard_client               = ${trading_partner}_std_client
receipt_client                = ${trading_partner}_rec_client
standard_server               = ${trading_partner}_std_server
highpri_server                = ${trading_partner}_hipri_server

# system-wide settings --------------------------------------------
#
#######################################################################
[ LWC_TEST_std_client ]
# Own private key file (in PEM format)
key  = ./client_keys/LWCTkey.pem
#
# Own certificate file (in PEM format)
cert = ./client_certs/LWCTcert.pem
#
# Path to CA certificates (in PEM format)
CApath = ./ca_certs
#
# Certificate authority (CA) file (in PEM format)
CAfile = CAcert.pem
#
```

```
####################################################################
[ LWC_TEST_rec_client ]
#
key  = ./client_keys/LWCTkey.pem
#
# Own certificate file (in PEM format)
cert = ./client_certs/LWCTcert.pem
#
# Path to CA certificates (in PEM format)
CApath = ./ca_certs
#
# Certificate authority (CA) file (in PEM format)
CAfile = CAcert.pem
#
####################################################################
[ LWC_TEST_std_server ]
#
# Standard server port
stdport = 1111
#
# Own private key file (in PEM format)
key  = ./client_keys/LWCTkey.pem
#
# Own certificate file (in PEM format)
cert = ./client_certs/LWCTcert.pem
#
# Path to CA certificates (in PEM format)
CApath = ./ca_certs
#
# Certificate authority (CA) file (in PEM format)
CAfile = CAcert.pem
#
####################################################################
[ LWC_TEST_hipri_server ]
#
# High priority server port
hipriport = 2222
#
key  = ./client_keys/LWCTkey.pem
#
# Own certificate file (in PEM format)
cert = ./client_certs/LWCTcert.pem
#
# Path to CA certificates (in PEM format)
CApath = ./ca_certs
#
# Certificate authority (CA) file (in PEM format)
CAfile = CAcert.pem
#
# EOF iagent.conf
```

A second example of the IAgent configuration file, demonstrating the Directory Interface.

```
# iagent.conf.eg1 - example IAgent process configuration file
# Copyright (C)1999-2001 Lymeware Corporation, all rights reserved
# $Header: $
#
# IAgent configuration file.  All entries (except configfile) override all
# default settings.  Any arguments set on the command line will overwrite
# anything in this file.
#
# This is an example of a Directory Interface configuration file.

# default section ###########################################################

trading_partner                 = TP1
iagent_home             = /opt/iagent

# define sections for each process to use (REQUIRED)
standard_client                 = ${trading_partner}_std_client
receipt_client                  = ${trading_partner}_rec_client
standard_server                 = ${trading_partner}_std_server
highpri_server                  = ${trading_partner}_hipri_server

# system-wide settings --------------------------------------------
interface = dir
indir = /opt/iagent/indir
outdir = /opt/iagent/outdir

# flag to force A LOT of logging
#verbose = yes
#

####################################################################
[ TP1_std_client ]

# own private key file (in pem format)
key  = ./client_keys/privkey.pem
#
# own certificate file (in pem format)
cert = ./client_certs/cert.cer
#
# path to CA certificates (in pem format)
CApath = ./ca_certs
#
# certificate authority (CA) file (in pem format)
CAfile = /opt/iagent/ca_certs/ca_cert.txt
#
```

```
#####################################################################
[ TP1_rec_client ]
#
# own private key file (in pem format)
key = $TP1_std_client::key
#
# own certificate file (in pem format)
cert = $TP1_std_client::cert
#
# path to CA certificates (in pem format)
CApath = $TP1_std_client::CApath
#
# certificate authority (CA) file (in pem format)
CAfile = $TP1_std_client::CAfile
#
#####################################################################
[ TP1_std_server ]
#
# own private key file (in pem format)
key = $TP1_std_client::key
#
# standard server port
stdport = 9001
#
# own certificate file (in pem format)
cert = $TP1_std_client::cert
#
# path to CA certificates (in pem format)
CApath = $TP1_std_client::CApath
#
# certificate authority (CA) file
CAfile  = $TP1_std_client::CAfile
#
#####################################################################
[ TP1_hipri_server ]
#
# hi priority server port
hipriport = 9011
#
# own private key file (in pem format)
key = $TP1_std_client::key
#
# own certificate file (in pem format)
cert = $TP1_std_client::cert
#
# path to CA certificates (in pem format)
CApath = $TP1_std_client::CApath
#
# certificate authority (CA) file (in pem format)
CAfile = $TP1_std_client::CAfile
#
#####################################################################
# EOF iagent.conf.eg1
```

Trading Partner configuration file

This is an example of a typical trading partner configuration file.

```
# IAgent tpartner.conf - Trading Partner configuration file
# $Header: /home/iagent/tp_one/RCS/tpartner.conf,v 1.2 2000/07/03 21:25:32 $
#
# Iagent example Trading Partner configuration file.
# This is mostly being used for client-side lookups during sending of
# EDI requests over SSL.
#
# default section ##########################################################

trading_partners = LWC_TEST, TP_1, TP_2
iagent_dir              = /opt/iagent/

# Each of the trading partners above should have its own section
# below.  Each trading partner section should contain the following
# list of fields:
#
#  partner_id - as found in the EDI ISA segment, 15 chars max as per X.12
#  certificate_common_name - as found in the client certificate, 64 chars max
#  ip_address - as string (AAA.BBB.CCC.DDD)
#  standard_ip_port - partner's standard IA server IP port address
#                   (same IP address as above)
#  hipri_ip_port - partner's hi priority IA server IP port address
#                   (same IP address as above)
#  session_timeout - session cache timeout in seconds.
#                     if 0 then no resumable (cached) sessions.
#  ia_message_to_tp - IA EDI Message Type supported as input by Trading Partner.
#                    May be any of the following:
#                      BASIC_EDI
#                      INTEGRITY_EDI
#                      SIGNED_EDI
#  ia_message_from_tp - IA EDI Message Type accepted from this Trading Partner.
#                    May be any of the following:
#                      BASIC_EDI
#                      INTEGRITY_EDI
#                      SIGNED_EDI
#  ia_receipt_to_tp - IA Receipt Type required as input by this Trading Partner.
#                    May be any of the following:
#                      NO_RECEIPT - no receipt required
#                      BASIC_RECEIPT
#                      INTEGRITY_RECEIPT
#                      SIGNED_RECEIPT
#  ia_receipt_from_tp - IA Receipt Type accepted from this Trading Partner.
#                    May be any of the following:
#                      NO_RECEIPT - no receipt required
#                      BASIC_RECEIPT
#                      INTEGRITY_RECEIPT
#                      SIGNED_RECEIPT
#  server_status - initial state of this partner's IA server.
#                   (0 = not operational, 1 = active).

####################################################################
[ TP_1 ]
# volume testing trading partner #1
partner_id          = TP_1
certificate_common_name = TP_1
ip_address          = 127.0.0.1
standard_ip_port    = 9001
hipri_ip_port       = 9011
session_timeout     = 30    # 30 min resumable sessions
ia_message_to_tp    = SIGNED_EDI
ia_message_from_tp  = SIGNED_EDI
ia_receipt_to_tp = INTEGRITY_RECEIPT
ia_receipt_from_tp = INTEGRITY_RECEIPT
server_status       = 1

####################################################################
[ TP_2 ]
# volume testing trading partner #2
partner_id          = TP_TWO
certificate_common_name = TP_TWO
ip_address          = 127.0.0.1
standard_ip_port    = 9002
hipri_ip_port       = 9012
session_timeout     = 30    # 30 min resumable sessions
ia_message_to_tp    = SIGNED_EDI
ia_message_from_tp  = SIGNED_EDI
ia_receipt_to_tp = INTEGRITY_RECEIPT
ia_receipt_from_tp = INTEGRITY_RECEIPT
server_status       = 1

####################################################################
# EOF - tpartner.conf
```

gen_csr Request.conf configuration file

This is the `Request.conf` configuration file used by `gen_csr`. It may be modified to reflect customer defaults but should be backed up first.

```
#
# Gen_csr example configuration file (version 1.4).
# This is mostly being used for generation of certificate requests.
#
# Copyright (C) 2000-2001 Lymeware Corporation
#
# $Header: /export/home/kobar/iagent/src/RCS/request.cnf,v 1.3 2001/01/13
18:44:05 kobar Exp $
#
####################################################################
[ req ]
#RANDFILE               = $ENV::HOME/.rnd
RANDFILE                = RAND_egd("/tmp/entropy")
default_bits            = 768
default_keyfile   = privkey.pem
distinguished_name      = req_distinguished_name
attributes              = req_attributes
#default_md             = md5            # which message digest to use.

[ req_distinguished_name ]
countryName             = Country Name (2 letter code)
countryName_default           = US
countryName_min               = 2
countryName_max               = 2

stateOrProvinceName           = State or Province Name (full name)
stateOrProvinceName_default   = Connecticut

localityName                  = Locality Name (eg, city)

0.organizationName            = Organization Name (eg, company)
0.organizationName_default    = Lymeware Corporation

organizationalUnitName        = Organizational Unit Name (eg, section)
#organizationalUnitName_default     =

#commonName             = Common Name (eg, YOUR ISA ID)
commonName              = Common Name (eg, Your Fully qualified domain name)
#commonName_max               = 15
commonName_max                = 64

emailAddress                  = Email Address
emailAddress_max        = 40

[ req_attributes ]
challengePassword       = A challenge password
challengePassword_min         = 4
challengePassword_max         = 20

#EOF
```

APPENDIX D – CONNECTIVITY TESTING

Simple Trading Partner Connectivity Test Plan

A simple trading partner connectivity test plan may be found on out web site at (http://www.lymeware.com/download.html).

Simple Backend Integration API Test Plan

A simple backend integration API test plan may be found on out web site at (http://www.lymeware.com/download.html).

APPENDIX E – EXTENDED FEATURES

The following features are available as Lymeware Extensions to the TCIF 98-006 v2 Interactive Agent specification. Some of these Extensions are proposed future standards and some were added just for our convenience. These features will continue to be supported in future releases (as much as is possible) for backward compatibility,

Lymeware Specific Extensions

Defined status input message

See *Chapter 16* for more information on the defined status input message format.

Proposed Future Standard Extensions

IA Ping Status Messages

This feature has been added to the current proposed next version of the TCIF IA standard.(Issue 3)

See *Chapter 16* for more information on IA Ping status messages.

APPENDIX F – PRODUCT UTILITIES

The following utilities are packaged with the IAgent product and may be used during installation, configuration or testing/monitoring of the system.

asn1dump

This is a utility, from Peter Gutmann (http://www.cs.auckland.ac.nz/~pgut001/), allows ASN.1 DER binary files to be displayed (dumped) in text and tag friendly format. More information on the utility can be found by typing:

```
asn1dump -x
```

iaappck

This is a shell script, which may be used to allow non-root users to start and stop the IAgent system via touch scripts. This file must be installed as a **`root`** `cron` job (see the script for installation details). Once installed the user need only type:

```
touch /opt/iagent/iastart
```

To start the IAgent system, or

```
touch /opt/iagent/iastop
```

To stop the IAgent system.

listen

This is a shell script which can detect and display IAgent processes listening on the IA ports.

monitor

This is a shell script which displays all IAgent processes currently running. Is may be started in a second window and used to monitor the IAgent processes.

prngd_test.pl

This Perl script is used to test the installation of the PRNG. The test will display a success message if the PRNG is installed and running correctly. Note: this script does expect that the PRNG daemon was started with the `/etc/rc3.d/90prng` startup script.

pm

This shell script is used to test the installation of the PRNG. The test will display a success message if the PRNG is installed and running correctly. Note: this script does expect that the PRNG daemon was started with the `/etc/rc3.d/90prng` startup script.

pmloop

This shell script is used to test the installation of the PRNG. The test will display a success message if the PRNG is installed and running correctly. Note: this script does expect that the PRNG daemon was started with the `/etc/rc3.d/90prng` startup script.

debugia

This shell script is used to test the installation of the PRNG. The test will display a success message if the PRNG is installed and running correctly. Note: this script does expect that the PRNG daemon was started with the `/etc/rc3.d/90prng` startup script.

APPENDIX G – GLOSSARY AND ACRONYMS

ANSI - American National Standards Institute.

ASN.1 - Abstract Syntax Notation One, as defined in X.208.

ATIS - Alliance for Telecommunications Industry Solutions.

authentication - The action of verifying information such as identity, ownership or authorization.

Base-64 - see PEM.

BER - Basic Encoding Rules for ASN.1 as defined in X.209.

certificate - In cryptography, an electronic document binding some pieces of information together, such as a user's identity and public key. Certificate Authorities (CAs) provide certificates.

Certificate Authority (CA) - A person or organization that creates certificates and verifies the identity of the certificate owner.

Certificate Revocation List (CRL) - A list of certificates that have been revoked before their expiration date.

cipher - An encryption - decryption algorithm.

ciphertext - Encrypted data.

CLEC – Competitive Local Exchange Carrier.

Data Encryption Standard (DES) - a block cipher developed by IBM and the U.S. government in the 1970's as an official standard.

decryption - The inverse (reverse) of encryption.

DER - Distinguished Encoding Rules for ASN.1 as defined in X.509, Section 8.7.

digest - Commonly used to refer to the output of a hash function, e.g. message digest refers to the hash of a message.

digital signature - The encryption of a message digest with a private key.

ECIC – The Electronic Communications Implementation Committee of TCIF.

Electronic Data Interchange (EDI) – an international business to business data interchange format as specified by ANSI X.12 standards.

encryption - The transformation of plaintext into an apparently less readable form (called ciphertext) through a mathematical process. The ciphertext may be read by anyone who has the key that decrypts (undoes the encryption) the ciphertext.

expiration date - Certificates and keys may have a limited lifetime; expiration dates are used to monitor this.

handshake - A protocol two computers or processes use to initiate a communication session.

HTML – The HyperText Markup Language. The "language" used to build and describe web pages.

ILEC – Incumbent Local Exchange Carrier.

International Standards Organization (ISO) - creates international standards, including cryptography standards.

Internet Engineering Task Force (IETF) – creates Internet standards, including security standards.

ITU-T - International Telecommunications Union - Telecommunications standardization sector.

key - A string of bits used widely in cryptography, allowing people to encrypt and decrypt data. Given a cipher, a key determines the mapping of the plaintext to the ciphertext. See also private key, public key.

key exchange - A process used by two more parties to exchange keys in a cryptosystem.

key pair - The full key information in a public-key cryptosystem, consisting of the public key and private key.

message digest - The result of applying a hash function to a message.

non-repudiation - A property of a cryptosystem. Non-repudiation cryptosystems are those in which the users cannot deny actions they performed.

OSS – Operational Support System. In the Telecommunications Industry the OSS is the sum of all in-house provisioning and billing systems and databases.

PEM – Internet Privacy Enhanced Mail as defined in RFC 1421 and RFC 1422. Also may refer to the Base-64 certificate format defined in RFC 1422.

plaintext - The data to be encrypted.

private key - In public-key cryptography, this key is the secret key. It is primarily used for decryption but is also used for encryption with digital signatures.

public key - In public-key cryptography this key is made public to all, it is primarily used for encryption but can be used for verifying signatures.

public-key cryptography - Cryptography based on methods involving a public key and a private key.

Public-Key Cryptography Standards (PKCS) - A series of cryptographic standards dealing with public-key issues, published by RSA Laboratories.

Public-Key Infrastructure Standards (PKIX) - A series of cryptographic standards dealing with public-key infrastructure and X.509 certificate issues, published by the IETF.

RBOC – Regional Bell (system) Operating Company. The pieces of AT&T created to provide Local telephone service. Often referred to as the "Baby Bells".

RSA algorithm - A public-key cryptosystem developed by RSA Data Security, Inc.

secret key - In secret-key cryptography, this is the key used both for encryption and decryption.

secure channel - A communication medium safe from the threat of eavesdroppers.

Secure Socket Layer (SSL) - A protocol (developed by Netscape) used for secure Internet communications.

session key - A key for symmetric-key cryptosystems which is used for the duration of one message or communication session

TCIF - Telecommunications Industry Forum.

Transport Layer Security (TLS) - A protocol (originally based on SSL and developed by the IETF) used for secure Internet communications. RFC-2246.

URL – Universal Resource Locator, typically a web browser address or location value.

verification - The act of recognizing that a person or entity is who or what it claims to be.

X.509 Certificate – A certificate as defined in X.509.

APPENDIX H – REFERENCES

Standards Identification	Standards Body	Standards Title and Publication Date
TCIF-98-006 Issue 2	ECIC	Electronic Communications Interactive Agent Functional Specification. TCIF-98-006, Issue 2. December 16, 1998
SSLv3	Netscape Communications	The SSL Protocol Version 3. November 18, 1996
RFC2246	IETF	The TLS Protocol Version 1.0. January, 1999
X.200	CCITT	Recommendation X.200: Reference Model of Open Systems Interconnection for CCITT Applications. 1984.
X.208	CCITT	Recommendation X.208: Specification of Abstract Syntax Notation One (ASN.1). 1988.
X.209	CCITT	Recommendation X.209: Specification of Basic Encoding Rules for Abstract Syntax Notation One (ASN.1). 1988.
X.501	CCITT	Recommendation X.509: The Directory – Models. 1988.
X.509	CCITT	Recommendation X.509: The Directory – Authentication Framework. 1988.
X.520	CCITT	Recommendation X.520: The Directory – Selected Attribute Types. 1988.
RFC1421	IETF	Privacy Enhancement for Internet Electronic Mail: Part I: Message Encryption and Authentication Procedures. February, 1993.
RFC1422	S. Kent	Privacy Enhancement for Internet Electronic Mail: Part II: Certificate-Based Key Management. February, 1993.
PKCS #1	RSA Laboratories	PKCS #1: RSA Cryptography Standard, version 2.0. September, 1998.
PKCS #6	RSA Laboratories	PKCS #6: Extended-Certificate Syntax Standard, version 1.5. November, 1993.
PKCS #7	RSA Laboratories	PKCS #7: Cryptographic Message Syntax Standard, version 1.5. November, 1993.
PKCS #8	RSA Laboratories	PKCS #8: Private-Key Information Syntax Standard, version 1.5. November, 1993.
PKCS #9	RSA Laboratories	PKCS #9: Selected Attribute Types, version 1.1. November, 1993.
PKCS #10	RSA Laboratories	PKCS #10: Certification Request Syntax Standard, version 1.0. November, 1993.
X.12.5	ANSI	ANSI X.12.5-1977: Interchange Control Structure, November 24, 1997.

Table 20. National, International, Internet, and Industry Standards used by this product

X.12.6	ANSI	ANSI X.12.6-1977: Application Control Structures, November 24, 1997.
X.12.22	ANSI	ANSI X.12.22-1977: Segment Directory, January 21, 1998.
X.12.3	ANSI	ANSI X.12.3-1977: Data Element Dictionary, January 21, 1998.

Table 21. National, International, Internet, and Industry Standards used by this product (concluded)

TIBCO Software Inc	TIB / Rendezvous Installation Guide, Release 5.0, September 1998.
TIBCO Software Inc	TIB / Rendezvous Concepts, Release 5.0, September 1998.
TIBCO Software Inc	TIB / Rendezvous Administrator’s Guide, Release 5.0, September 1998.
TIBCO Software Inc	TIB / Rendezvous C Programmer’s Guide, Release 5.0, September 1998.

Table 22. Commercial or third party documentation used by this product or manual

INDEX