INTERNATIONAL TELECOMMUNICATION UNION

ITU-T Q.814

TELECOMMUNICATION
STANDARDIZATION SECTOR
OF ITU

(02/2000)

SERIES Q: SWITCHING AND SIGNALLING

Specifications of Signalling System No. 7 – Q3 interface

Specification of an electronic data interchange interactive agent

ITU-T Recommendation Q.814

(Formerly CCITT Recommendation)

ITU-T Q-SERIES RECOMMENDATIONS

SWITCHING AND SIGNALLING

SIGNALLING IN THE INTERNATIONAL MANUAL SERVICE	Q.1–Q.3
INTERNATIONAL AUTOMATIC AND SEMI-AUTOMATIC WORKING	Q.4–Q.59
FUNCTIONS AND INFORMATION FLOWS FOR SERVICES IN THE ISDN	Q.60–Q.99
CLAUSES APPLICABLE TO ITU-T STANDARD SYSTEMS	Q.100–Q.119
SPECIFICATIONS OF SIGNALLING SYSTEMS No. 4 AND No. 5	Q.120–Q.249
SPECIFICATIONS OF SIGNALLING SYSTEM No. 6	Q.250–Q.309
SPECIFICATIONS OF SIGNALLING SYSTEM R1	Q.310–Q.399
SPECIFICATIONS OF SIGNALLING SYSTEM R2	Q.400–Q.499
DIGITAL EXCHANGES	Q.500–Q.599
INTERWORKING OF SIGNALLING SYSTEMS	Q.600–Q.699
SPECIFICATIONS OF SIGNALLING SYSTEM No. 7	Q.700–Q.849
General	Q.700
Message transfer part (MTP)	Q.701–Q.709
Signalling connection control part (SCCP)	Q.711–Q.719
Telephone user part (TUP)	Q.720–Q.729
ISDN supplementary services	Q.730–Q.739
Data user part	Q.740–Q.749
Signalling System No. 7 management	Q.750–Q.759
ISDN user part	Q.760–Q.769
Transaction capabilities application part	Q.770–Q.779
Test specification	Q.780–Q.799
Q3 interface	**Q.800–Q.849**
DIGITAL SUBSCRIBER SIGNALLING SYSTEM No. 1	Q.850–Q.999
PUBLIC LAND MOBILE NETWORK	Q.1000–Q.1099
INTERWORKING WITH SATELLITE MOBILE SYSTEMS	Q.1100–Q.1199
INTELLIGENT NETWORK	Q.1200–Q.1699
SIGNALLING REQUIREMENTS AND PROTOCOLS FOR IMT-2000	Q.1700–Q.1799
BROADBAND ISDN	Q.2000–Q.2999

For further details, please refer to the list of ITU-T Recommendations.

ITU-T Recommendation Q.814

Specification of an electronic data interchange interactive agent

Summary

This ITU-T Recommendation defines the technical specification of a session layer protocol module called Electronic Communications Interactive Agent. This may be used as an interface reference point in a TMN model for the asynchronous exchange of data between peer application entities. The Interactive Agent (IA) supports the exchange of near real time Electronic Data Interchange (EDIFACT or ASC X12 EDI) transactions. In addition, this ITU-T Recommendation defines the architecture, design, structure, and process-flow for both *normal*[1] and *high priority*[2] business functions utilizing Transport Layer Security (TLS).

Source

ITU-T Recommendation Q.814 was prepared by ITU-T Study Group 4 (1997-2000) and approved under the WTSC Resolution 1 procedure on 4 February 2000.

[1] *Normal Priority* – An example of a Normal Priority Business Function is an Order Request Transaction.

[2] *High Priority* – An example of a High Priority Business Function is an interactive inquiry transaction.

FOREWORD

The International Telecommunication Union (ITU) is the United Nations specialized agency in the field of telecommunications. The ITU Telecommunication Standardization Sector (ITU-T) is a permanent organ of ITU. ITU-T is responsible for studying technical, operating and tariff questions and issuing Recommendations on them with a view to standardizing telecommunications on a worldwide basis.

The World Telecommunication Standardization Conference (WTSC), which meets every four years, establishes the topics for study by the ITU-T study groups which, in turn, produce Recommendations on these topics.

The approval of ITU-T Recommendations is covered by the procedure laid down in WTSC Resolution 1.

In some areas of information technology which fall within ITU-T's purview, the necessary standards are prepared on a collaborative basis with ISO and IEC.

NOTE

In this Recommendation, the expression "Administration" is used for conciseness to indicate both a telecommunication administration and a recognized operating agency.

INTELLECTUAL PROPERTY RIGHTS

ITU draws attention to the possibility that the practice or implementation of this Recommendation may involve the use of a claimed Intellectual Property Right. ITU takes no position concerning the evidence, validity or applicability of claimed Intellectual Property Rights, whether asserted by ITU members or others outside of the Recommendation development process.

As of the date of approval of this Recommendation, ITU had not received notice of intellectual property, protected by patents, which may be required to implement this Recommendation. However, implementors are cautioned that this may not represent the latest information and are therefore strongly urged to consult the TSB patent database.

CONTENTS

		Page
1	Scope	1
2	References	1
2.1	Normatives references	1
2.2	Informative references	1
3	Definitions	2
4	Abbreviations	2
5	Conventions	2
6	Architecture and Service Characteristics	3
6.1	Architecture	3
6.2	Service characteristics	5
	6.2.1 Elements of Service	5
	6.2.2 Classifications of Elements of Service	6
7	Data Flow	6
8	IA Messages	7
8.1	Message Format Definitions	7
8.2	Message Syntax Definitions	7
	8.2.1 Basic Message	7
	8.2.2 IA Status/Control Message	8
	8.2.3 Enhanced Message	8
8.3	IA Status Message Detail Format	8
	8.3.1 First Octet	8
	8.3.2 Second Octet	8
	8.3.3 Third and Fourth Octets	8
	8.3.4 Special Test Message	9
	8.3.5 Invalid Message	9
9	Client Specifications	9
9.1	Determine IP Destination Address	9
9.2	Connect to Server	9
	9.2.1 Allocate TLS Data Structure and Memory	10
	9.2.2 Open Socket	10
	9.2.3 Send TLS Client Hello	10
	9.2.4 Send Client's Certificate to Server	10
	9.2.5 Client Key Exchange	10
	9.2.6 Send Client Certificate Verify	10
	9.2.7 Change Cipher Specs	11

Page

9.2.8 Send Client Finished 11
9.3 Send Application Data to Server 11
9.4 Transmission Logging 11
9.5 Client Disconnect 11
10 Server Specifications 12
10.1 Initialize Server 12
10.2 Accept Connection from Client 12
10.3 Message Read Setup 12
10.3.1 Allocate TLS Data Structure and Memory 13
10.3.2 Bind TLS Data Structure to the Socket 13
10.3.3 Send TLS Server Hello 13
10.3.4 Send Server's Certificate to Client 13
10.3.5 Server Key Exchange 13
10.3.6 Send Client Certificate Request 13
10.3.7 Send Server Hello Done 13
10.3.8 Execute Change Cipher Specs 13
10.3.9 Send Server Finished 13
10.4 TLS Read Processing 13
10.5 Server Disconnect 14
10.6 Parsing the Received Message 14
10.7 Transfer Data to Immediate User (Translator/Security Module) 14
10.8 Receipt Logging 15
11 Operational requirements 15
11.1 Security 15
11.2 Digital Certificates 15
11.3 Flow Control 16
12 Port Assignments 16
Annex A – ASN.1 Production Module 17
Annex B – Design Considerations 17
B.1 Multi-processing/Multi-threading 17
B.2 Non-Persistent Versus Persistent Connections 17
B.3 Resumable TLS Sessions 18
Annex C – Error Handling/Recovery 18
Appendix I – Non-normative references 18

Introduction

This ITU-T Recommendation defines the specifications for an Electronic Data Interchange Interactive Agent (IA). The IA supports the exchange of Electronic Data Interchange transactions between peer entities. It maps EDI transactions into the transport layer. More specifically, it interfaces with Transport Layer Security (TLS) to request the establishment and termination of secure (i.e. supporting peer entity authentication, integrity, and privacy) TCP sessions and secure transport of EDI messages. The IA further provides basic flow control functionality.

ITU-T Recommendation Q.814

Specification of an electronic data interchange interactive agent

1 Scope

This ITU-T Recommendation provides a specification for the Electronic Data Interchange Interactive Agent (IA). The IA supports the interchange of Electronic Data Interchange (EDIFACT/ASC X12 EDI) transactions over a Transmission Control Protocol/Internet Protocol (TCP/IP) network utilizing Transport Layer Security (TLS). This ITU-T Recommendation specifies the general architecture of the IA, the syntax of the message formats to be used, the encoding rules for the messages and the applicable security transformations.

2 References

2.1 Normatives references

The following ITU-T Recommendations and other references contain provisions which, through reference in this text, constitute provisions of this Recommendation. At the time of publication, the editions indicated were valid. All Recommendations and other references are subject to revision; all users of this Recommendation are therefore encouraged to investigate the possibility of applying the most recent edition of the Recommendations and other references listed below. A list of the currently valid ITU-T Recommendations is regularly published.

- ITU-T Recommendation Q.815 (2000), *Specification of a security module for whole message protection.*
- ITU-T Recommendation X.509 (1997) | ISO/IEC 9495-8:1998, *Information technology – Open Systems Interconnection – The Directory: Authentication framework.*
- ITU-T Recommendation X.680 (1997) | ISO/IEC 8824-1:1998, *Information technology – Abstract Syntax Notation One (ASN.1): Specification of basic notation.*
- ITU-T Recommendation X.681 (1997) | ISO/IEC 8824-2:1998, *Information technology – Abstract Syntax Notation One (ASN.1): Information object specification.*
- ITU-T Recommendation X.682 (1997) | ISO/IEC 8824-3:1998, *Information technology – Abstract Syntax Notation One (ASN.1): Constraint specification.*
- ITU-T Recommendation X.683 (1997) | ISO/IEC 8824-4:1998, *Information technology – Abstract Syntax Notation One (ASN.1): Parameterization of ASN.1 specifications.*
- ITU-T Recommendation X.690 (1997) | ISO/IEC 8825-1:1998, *Information technology – ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER).*

Internet Society/Internet Engineering Task Force:

- RFC 2246, *The TLS Protocol Version 1.0.*

2.2 Informative references

- Directory Implementors Guide (Version 12) (1999).

3 Definitions

This ITU-T Recommendation defines the following terms:

3.1 interactive agent transfer protocol (IATP): This protocol is utilized between peer interactive agents wishing to exchange electronic data interchange transactions/messages via transmission control protocol/Internet protocol utilizing transport layer security.

3.2 EDI translator: An EDI translator is typically a computer software module or program that translates private data formats and representations to/from standard formats and standard data representations such as those specified by ISO 9735 or ANSI ASC X.12.

3.3 interactive agent (IA): The IA supports the exchange of electronic data interchange (UN/EDIFACT or ASC X12 EDI) transactions between peer entities. The IA functions as an interface between its direct user (normally an EDIFACT/ASC X12 EDI translator or a security module) and the transport layer security. Various implementation approaches may be taken ranging from a simple API (Application Program Interface) through a stand-alone program. The IA is described in this ITU-T Recommendation and the Security Module is described in ITU-T Recommendation Q.815.

3.4 transport layer security (TLS): The TLS protocol optionally provides communications privacy. The protocol allows client/server applications to communicate in a way that is designed to prevent eavesdropping, tampering, and intrusion. The TLS protocol also provides strong peer authentication and data flow integrity.

4 Abbreviations

This ITU-T Recommendation uses the following abbreviations:

IA	Interactive Agent
IATP	Interactive Agent Transfer Protocol
MD	Message Digest
SHA-1	Secure Hashing Algorithm, Revision 1
SM	Security Module
TLS	Transport Layer Security
WAN	Wide Area Network

5 Conventions

The following conventions are used within this ITU-T Recommendation:

The term *EDI,* as used within this ITU-T Recommendation, refers to any or all of the following:

- UN/EDIFACT as defined by the UN/ECE Trade Division and adopted by ISO/TC 154
- EDIFACT as defined by ISO 9735

NOTE – This also includes EDI as defined by ANSI ASC X12.

Table 1 in 6.2.2 uses the following conventions:

M	Mandatory
O	Optional

All occurrences of *C Language* code appearing in this ITU-T Recommendation are for illustrative purposes only.

6 Architecture and Service Characteristics

6.1 Architecture

The IA functions as an interface between its direct user (normally an application such as EDIFACT or ASC X12 EDI translator) and the transport layer. (See Figure 1.) Basic security of EDI transactions is provided by TLS. Additional security capabilities (e.g. non-repudiation) may be provided by a separate security module that performs security transformations on whole EDI messages. Such security module can also be a direct user of the IA as illustrated in Figure 2.

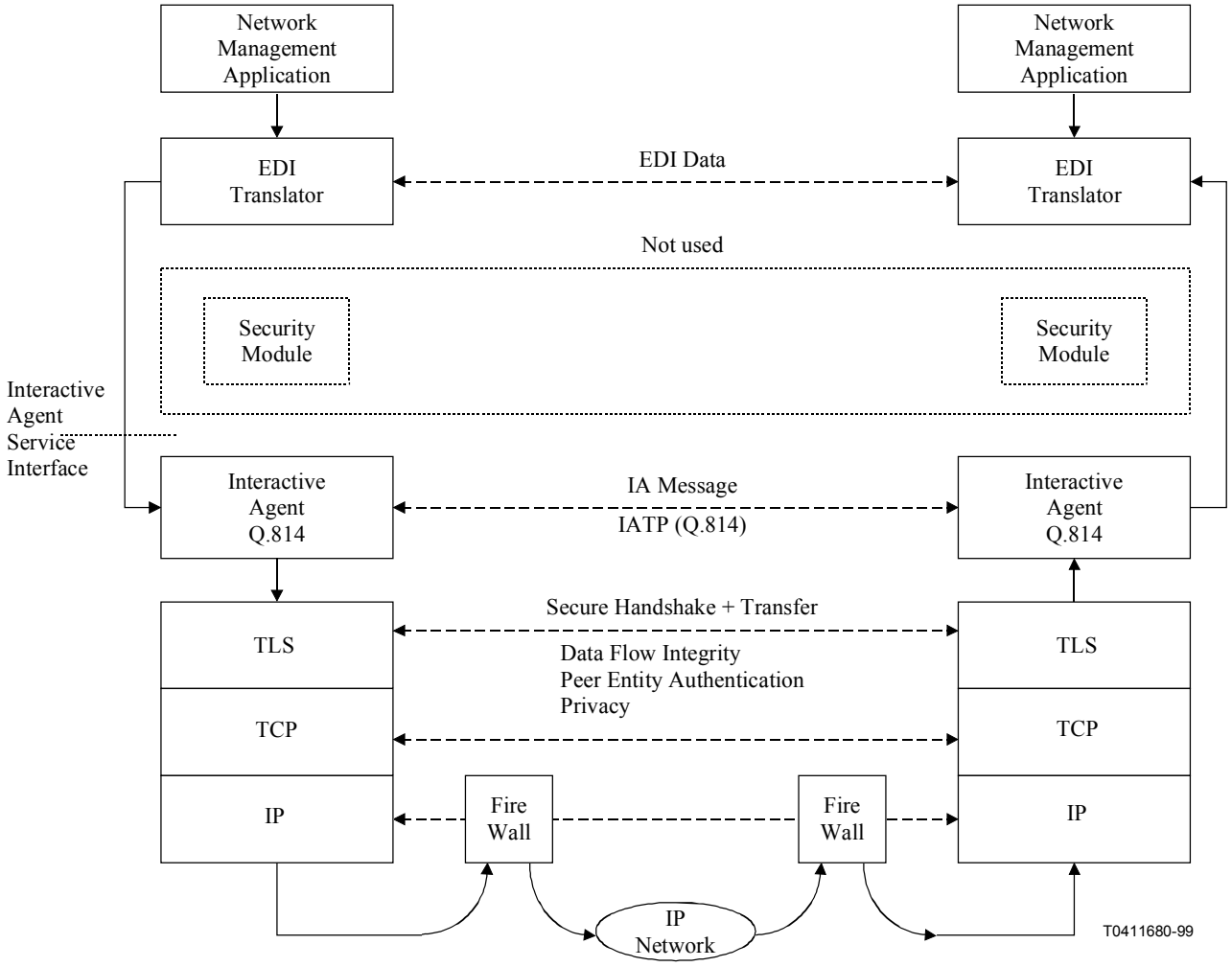

Figure 1/Q.814 – Message flow relationship (without Security Module)

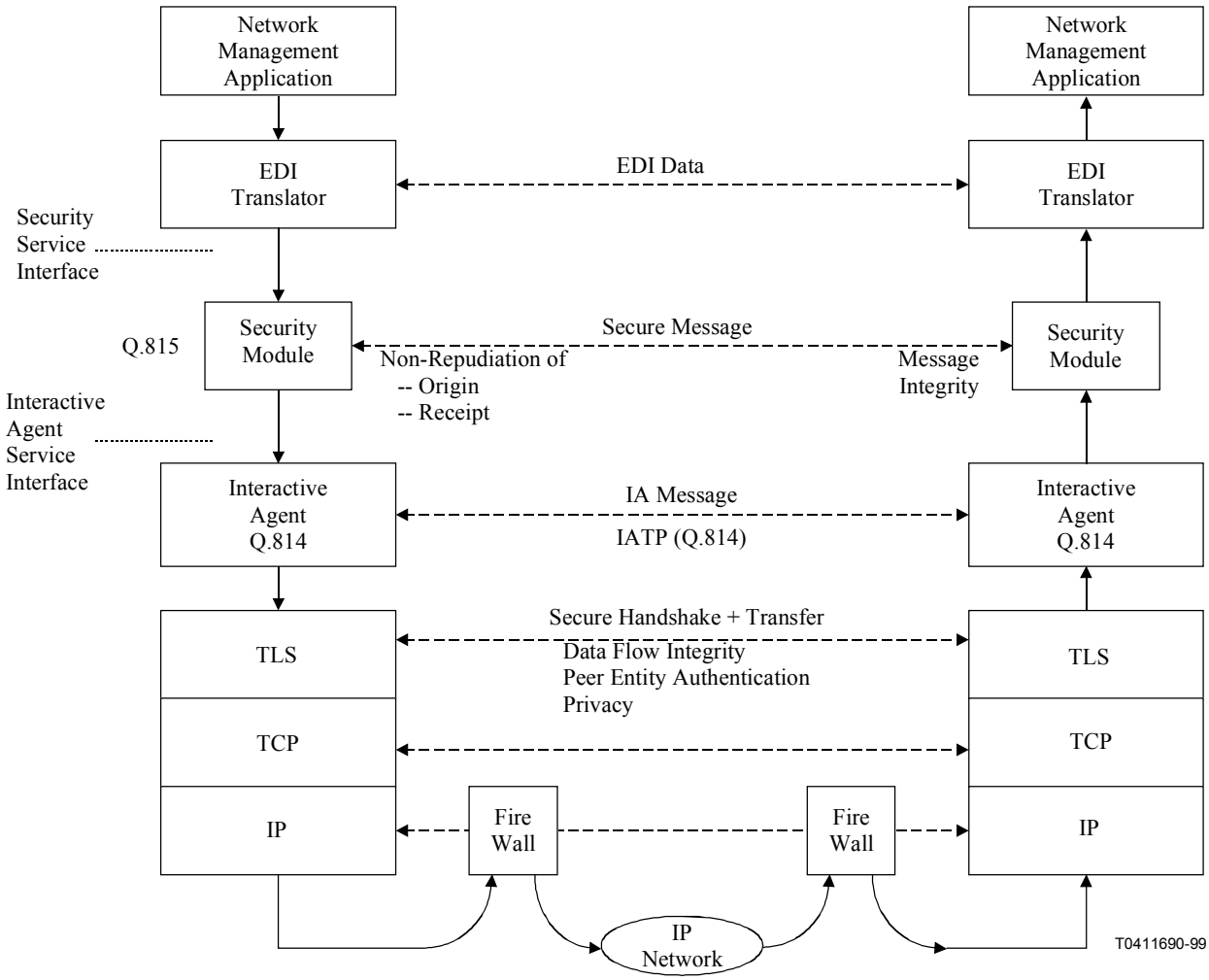

Figure 2/Q.814 – Message flow relationship (with Message Security Services)

The underlying structure of the IA is a symmetrical Client/Server configuration where both the Client and Server functions are required at each implementation. See Figure 3.

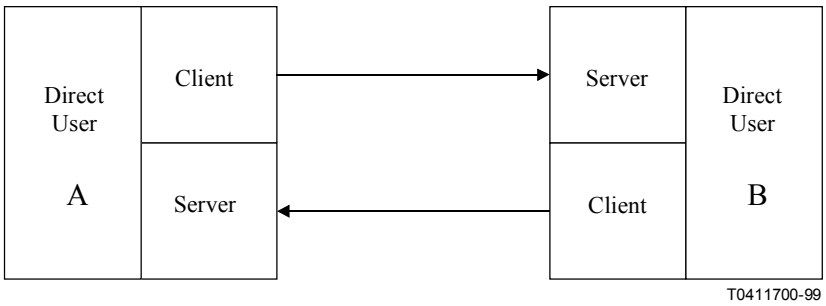

Figure 3/Q.814 – Interactive Agent Architecture

NOTE – The symmetrical client/server architecture may not be appropriate for certain types of *near real time* applications.

The IA performs session layer functions of the OSI, Seven-Layer Model. The session layer functions provided by the IA include the establishment, management and closing of communications sessions between peer entities. The IA may also perform the conversion of EDIFACT/ASC X12 EDI identities to network addresses and manage the transport layer's session. At the conclusion of a session, the IA will determine whether to close a session or to suspend it in a state whereby it can be *resumed*.

6.2 Service characteristics

The IA may expose to its direct user or peer IAs the following services:

- IA Recipient Name.
- IA Priority.
- IA Basic Message.
- IA Enhanced Message.
- IA Control.
- IA Logging.

6.2.1 Elements of Service

6.2.1.1 IA Recipient Name

The direct user supplies the IA with the identity of the recipient of the message being transferred (e.g. the trading partner identity in an ASC X12 based EDI message). The IA will convert this value to a transport layer address.

6.2.1.2 IA Priority

The direct user supplies the IA with an indicator of the priority of the message. The IA supports *normal* and *high* priority messages. The priority indicator is converted to a destination port address and combined with the transport layer address for use in establishing a session layer connection request.

6.2.1.3 IA Basic Message

The direct user supplies the content of an non-enhanced data-bearing message to the IA.

6.2.1.4 IA Control

The direct user requests the IA to transmit session control information to the peer entity.

6.2.1.5 IA Enhanced Message

The direct user supplies the content of a data-bearing message to the IA.

6.2.1.6 Logging (Local Matter)

The direct user may specify the mechanism, classifications and contents of data to be logged.

6.2.2 Classifications of Elements of Service

See Table 1.

Table 1/Q.814 – Classifications of Elements of Service

Class of Service	Origination	Reception
IA Recipient Name	M	M
IA Priority	O	M
IA Basic Message	O[a)]	M
IA Enhanced Message	O[a)]	M
IA Control	O	M
IA Logging (local matter)	O	O
[a)] At least one shall be chosen.		

7 Data Flow

This clause illustrates the establishment of a session between a client and a server. Figure 4 shows a full TLS session establishment. Figure 5 shows an abbreviated session establishment also known as a "resumed" session.

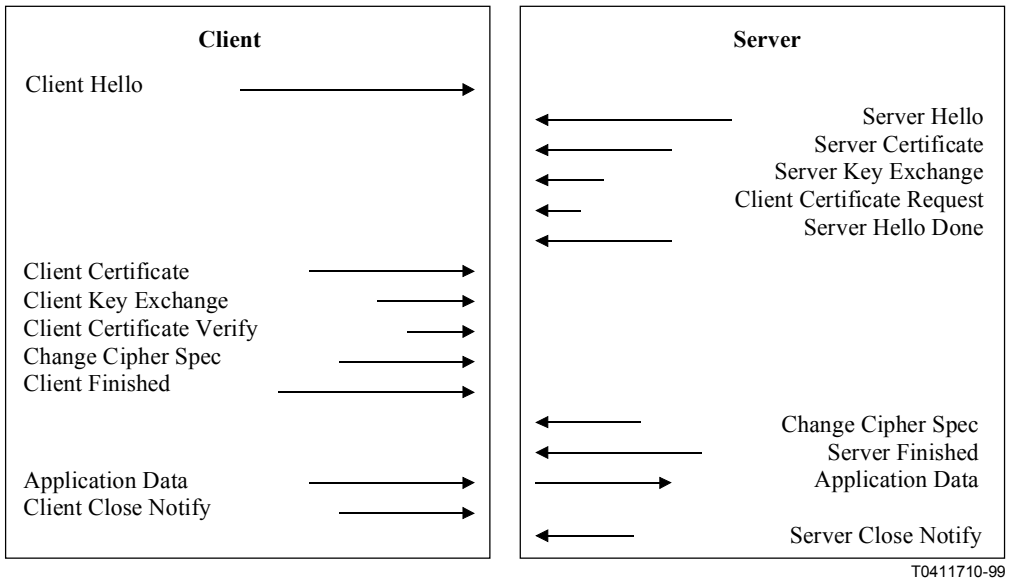

Figure 4/Q.814 – IA Message Flow – Full TLS Handshake

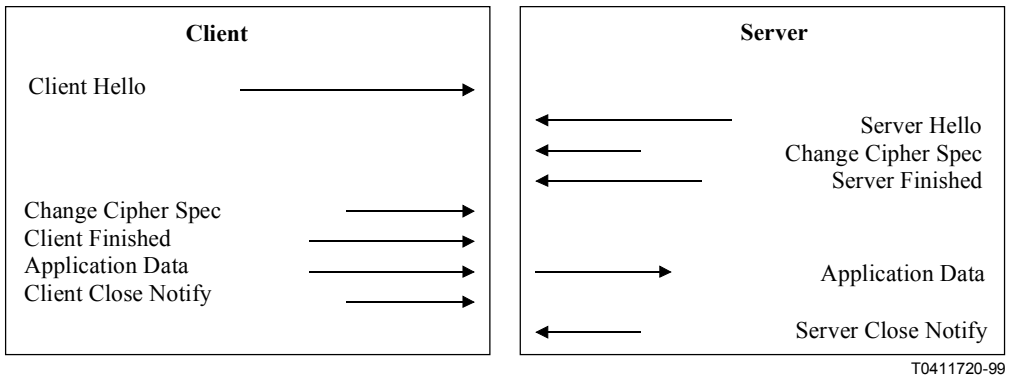

Figure 5/Q.814 – IA Message Flow – Abbreviated TLS Handshake

8 IA Messages

In order to meet both business and security needs of an entity, two types of messages have been defined. These are:

- Basic Message (data flow integrity and/or privacy encryption).
- Enhanced Message (whole message integrity or non-repudiation).

8.1 Message Format Definitions

Discrete message formatting requirements are associated with each of the two message formats defined above. In addition, a third format has been defined for communication of status/control between the Interactive Agents, namely the IA Status Message. Each of these is encoded in accordance with the corresponding Abstract Syntax Notation One (ASN.1) definition.

The Distinguished Encoding Rules (DER) for ASN.1 shall be used to encode IA messages. Only the definite-length method of encoding length octets shall be used.

8.2 Message Syntax Definitions

The Interactive Agent may be used to transport EDIFACT, ASC X12 EDI and/or any other form of character string data. The ASN.1 definitions for each of the supported message types can be found in this subclause.

The general syntax is as follows:

```
IaMessage ::= CHOICE {
        basicMessage        BasicMessage,        -- Basic Message
        iaStatusMessage     IaStatusMessage,     -- IA Status/Control Message
        enhancedMessage     EnhancedMessage      -- Security Module Enhanced Message
}
```

8.2.1 Basic Message

The Basic Message is chosen to convey the text of IA messages in either of 2 character repertoires.

```
BasicMessage ::= CHOICE {
basicMessage1     GeneralString,
basicMessage2      IA5String
}
```

8.2.2 IA Status/Control Message

The Status Message may be used to exchange flow control or error conditions between IAs (see 8.3).

IaStatusMessage ::= BIT STRING (SIZE (32))

8.2.3 Enhanced Message

The Enhanced Message syntax is chosen to transparently convey messages that have been encoded by a higher layer security function. Typically, this service will be provided by ITU-T Recommendation Q.815.

EnhancedMessage ::= OCTET STRING -- *Security Module Enhanced Message*

8.3 IA Status Message Detail Format

The IA Status Message consists of four octets containing status information. A summary of the values of the IA Status Message is found in Table 2. The four octets conveying the status are structured as follows:

8.3.1 First Octet

Utilized for status conditions pertaining to the wide area interface of the IA (i.e. TCP/IP and TLS).

Coding structure

This octet is to be interpreted as two hexadecimal values in the range 00 to FF.

Values in this octet are to be defined solely by this ITU-T Recommendation.

Currently defined values include:

00 = Ignored
08 = Request for peer to cease transmissions on inbound data stream due to problems with WAN interface

All other values are currently undefined.

8.3.2 Second Octet

Utilized for status conditions pertaining to the downstream systems on the local interface (i.e. the EDI translator or other related, receiving IA's direct user.)

Coding structure

This octet is to be interpreted as 2 hexadecimal values in the range 00 to FF.

Values in this octet are to be defined solely by this ITU-T Recommendation.

Currently defined values include:

00 = Ignored
08 = Request for peer to cease transmissions on inbound data stream due to problems with IA communicating with EDI translator or other downstream processing systems.

All other values are currently undefined.

8.3.3 Third and Fourth Octets

When octets one, two, three, and four are all zeroes it shall mean "Test Message".

When octets one and two are zeroes, octets three and four are available for peer entities to define. Peer entities may define the values in these octets in any manner they choose. It is recommended that the values be used as two hexadecimal values per octet.

When either octet one or two is non-zero, then the values of octets three and four are reserved for further standardization.

8.3.4 Special Test Message

A message containing four octets of hexadecimal 00 00 00 00 is defined as a NULL message that may be utilized as a test message. No actions shall be taken in response to receipt of such a message.

8.3.5 Invalid Message

The following message is invalid and shall be ignored if received:

08 08 xx xx

Table 2/Q.814 – IA Status Message Values

Byte 1	Byte 2	Byte 3	Byte 4	Usage
00	00	00	00	Test Message
08	00	00	00	Congestion Lower Layers
00	08	00	00	Congestion Upper Layers
08	00	xx	xx	Reserved Future Use
00	08	xx	xx	Reserved Future Use
00	00	yy	yy	User Defined
08	08	xx	xx	Invalid Message Ignored
yy	00	xx	xx	Reserved
00	yy	xx	xx	Reserved
NOTE 1 – xx = 00-FF. NOTE 2 – yy = 01-FF.				

9 Client Specifications

This clause defines the processing associated with the Interactive Agent Client Process.

Upon receipt of data from its direct user across the IA Service Interface (e.g. the EDIFACT/ASC X12 EDI translator or the Security Module), the client performs the following steps. However, before this process can be followed the IA determines if an outstanding ceased transmission request has been received, see 11.3.

9.1 Determine IP Destination Address

Based upon the peer entity's identity and other relevant information, the corresponding IP address and port number is determined for routing of the data to the appropriate server.

NOTE – The selection of the *port number* determines which of two levels of priority will be utilized. See clause 12 regarding port assignments.

9.2 Connect to Server

The following sequence of steps summarizes this operation. Details of the steps are provided following the summary:

- Allocate TLS Data Structure and Memory, as required.
- Open Socket.

- Send TLS Client Hello.
- Send Client's Certificate to Server.
- Client Key Exchange.
- Send Client Certificate Verify.
- Execute Change Cipher Specs.
- Send Client Finished.

The following details are provided for each of the steps identified above:

9.2.1 Allocate TLS Data Structure and Memory

A data structure is used to store the cryptographic data associated with an individual connection including certificates, references to callbacks, and various other data. An independent data structure must be allocated and maintained for each connection.

9.2.2 Open Socket

A client application creates a socket and then issues a connect to a service specified in a structure, (e.g. *sockaddr_in)*. The following is an example of an Open Socket operation in "C":

```
int     tcpopen(host,service)
char   *service, *host;
{  int    unit;
   struct  sockaddr_in    sin;
   struct  servent        *sp;
   struct  hostent        *hp;
   if ((sp=getservbyname(service,"tcp")) == NULL) then error…
   if ((hp=gethostbyname(host)) == NULL) then error...
   bzero((char *)&sin, sizeof(sin))
       etc...
   if ((unit=socket(AF_INET,SOCK_STREAM,0)) < 0) then error...
   if (connect(unit,&sin,sizeof(sin)) < 0) then error...
   return(unit);
}
```

The result returned is a *file descriptor* that is connected to a server process. This is a communications channel on which one can conduct an application specific protocol.

9.2.3 Send TLS Client Hello

The function sends the client's current date/time, session identifier, cipher suite list, compression algorithm list, a random data structure and a client version parameter. This parameter specifies what version(s) of the TLS protocol can be used for the connection. This should be set to a value of {3.1} for TLS. After sending a *Client Hello,* the client must wait until it receives a *Server Hello* in response.

9.2.4 Send Client's Certificate to Server

The client sends its digital certificate to the server.

9.2.5 Client Key Exchange

This message sets the 48 byte pre-master secret, encrypts it with the server's public key and sends the results in an encrypted pre-master secret message.

9.2.6 Send Client Certificate Verify

This message is used to provide explicit verification of the client's certificate. This message is only sent if a client certificate has been sent that has signing capability. It will immediately follow the *client key exchange* message.

9.2.7 Change Cipher Specs

This message is sent to notify the peer system that subsequent records will be protected under the newly negotiated *cipher spec* and keys.

9.2.8 Send Client Finished

The client will then send the *Client Finished* message immediately after a *Change Cipher Specs Message* to verify that the key exchange and authentication processes were successful. This message is the first message sent protected with the just negotiated algorithms, keys and secrets. The client is ready to commence sending application data.

9.3 Send Application Data to Server

After the connection to the server is established, the client will encode the message using DER and transmit it to the server as a data stream. This process is accomplished using the TLS write function, as specified by the TLS toolkit or library being utilized.

9.4 Transmission Logging

Log entries are created to record the transmission of data to the server. The minimal set of data to be logged consists of the following:

- Date/Time.
- A unique message identifier (e.g. the ISA Segment if ASC X12 EDI).
- Remote IP address and port number of the peer.
- Success/fail indicator with the interfaces.

9.5 Client Disconnect

The client and server must share knowledge that the connection is ending. The client will initiate the following sequence after completion of the writing of all data to the peer:

- Execute the TLS Close Notify:

  This function closes the TLS session with the peer. No further transmissions are to be expected. The client must wait for the server to respond prior to proceeding with the remaining steps in this sequence.
- Execute the Socket Close:

  The client closes the socket connection with the *close()* function, passing the socket descriptor as a parameter.
- Execute the TLS Data Structure Delete:

  This function typically releases the resources utilized by the TLS connection.

  This is only performed if this session is to be terminated and is not done for sessions that may later be resumed. Depending on the implementation, the memory used by the data structure may have to be de-allocated after use.

If the server *Close Notify* is not received, the client will recognize an error condition and perform the following steps:

- Execute the Socket Close.
- Execute the TLS Data Structure Delete and mark the session as non-resumable.
- Notify the user application of communication failure.

10 Server Specifications

This clause defines the processing associated with the Interactive Agent server process.

Upon startup of the server process, the server performs the following sequence of steps:

10.1 Initialize Server

To accept connections, a socket is created and is bound to a service port. A queue for incoming connections is specified and connections are accepted as shown in this example C-language code fragment:

```
struct servent          *sp;
struct sockaddr_in sin,from;
if ((sp=getservbyname(service,"tcp")) == NULL) then error...
sin.sin_family=etc...
if ((s=socket(AF_INET,SOCK_STREAM,0)) < 0) then error...
if (bind(s, &sin, sizeof(sin)) < 0) then error...
if (listen(s,QUELEN) < 0) then error...
for (;;) {
   if ((g=accept(f,&from,&len)) < 0) then error...
   if (!fork()) {
      child handles request...
         ...and exits
      exit(0);
   }
   close(g);   /* parent releases file */
}
```

When the process and the port are bound, the server will listen to the port for connection requests. After a connection request is received, a socket is created. For example, when using a multi-processing scheme, connections are made and the process forks a *child process* to handle that service request. The *parent process* continues to listen for and accept further service requests.

10.2 Accept Connection from Client

Based upon the peer entity identity and other relevant information (which could include the remote IP address and local port number), routing for the data will normally be determined to the appropriate EDIFACT/ASC X12 EDI translator or the Security Module.

10.3 Message Read Setup

The following sequence of steps summarizes this operation. Details of the steps are provided following the summary:

- Allocate TLS Data Structure and Memory.
- Bind TLS Data Structure to the socket.
- Send TLS Server Hello.
- Send Server's Certificate to Client.
- Server Key Exchange.
- Send Client Certificate Request.
- Send Server Hello Done.
- Execute Change Cipher Specs.
- Send Server Finished.

When an inbound connection request is received, the server *listening* process will pass the connection request to the connection *processing* software. This software will perform the following:

10.3.1 Allocate TLS Data Structure and Memory

A data structure is used to store the cryptographic data associated with an individual connection including certificates, references to callbacks, and various other data. An independent data structure must be allocated and maintained for each connection.

10.3.2 Bind TLS Data Structure to the Socket

This action logically connects the TLS data structure with the socket on which the inbound connection request is pending. The actual steps performed to accomplish this will vary based on the TLS toolkit and socket management software in use.

10.3.3 Send TLS Server Hello

This function sends the session identifier (either a new one or the value sent by the client in the case of a resumed session), a single selected cipher suite, a single selected compression algorithm, a random data structure (different from the client's) and a server version parameter. This parameter specifies what version of the TLS protocol will be used for the connection. This must be a value of {3.1} for TLS. This message is sent as a response to a *Client Hello*.

10.3.4 Send Server's Certificate to Client

The server sends the contents of its X.509 version 3 certificate to the client.

10.3.5 Server Key Exchange

This message must be sent immediately after the server certificate message, if required. The server key exchange message is sent only in cases where the server certificate message does not contain enough data to allow the client to exchange a pre-master secret. This will be applicable if an RSA_EXPORT key exchange method has been selected and the public key in the server certificate is longer than 512 bits.

10.3.6 Send Client Certificate Request

This function requests a certificate from the client. This message must immediately follow the Server Key Exchange Message (if it is sent), otherwise the Server Certificate message.

10.3.7 Send Server Hello Done

This function notifies the client that the server has completed sending the Server Hello and its associated messages. After sending this message the server will wait for a response from the client.

10.3.8 Execute Change Cipher Specs

This message instructs the peer system to activate the most recently received set of cryptographic parameters.

10.3.9 Send Server Finished

The server will then send the Server Finished message immediately after a Change Cipher Specs Message to verify that the key exchange and authentication processes were successful. This message is the first message sent protected with the just negotiated algorithms, keys and secrets. The server side of the connection is now ready to receive data.

10.4 TLS Read Processing

This process requires that an initial TLS *read* of two octets be performed. The first of these octets will contain the DER representation of an ASN.1 tag identifying the type of IA message as specified in clause 8. The second octet will either be a DER short form definite length or the first octet of a

DER long form definite length, enumerating the number of octets contained in the remainder of the length field.

If the second octet is a short form DER length, it represents the length of the remainder of the current IA message.

If the second octet is a long form DER length, a second *read* of the indicated number of octets is required to determine the remaining number of octets comprising the long form definite length field. These remaining octets enumerate the total length of the remainder of the IA message.

A final *read* should then be initiated to *read* all of the octets comprising the remainder of the IA message. This third read may be done either as a single *read* operation or as a series of partial *reads* whose sum equals the total value. All *reads* in this section are accomplished using the TLS *read* function, as specified by the TLS toolkit or library being utilized.

If the TLS *read* fails due to insufficient resources, the IA initiates flow control procedures as specified in 11.3.

10.5 Server Disconnect

The client and server must share knowledge that the connection is ending. The client will signal the server that it has completed the transmission of the current message by sending a Client Close Notify. The server must complete the pending TLS read processing and then execute the following procedures:

- Execute the TLS Close Notify:

  This function closes the TLS session with the peer. No further communications are expected. This function will normally be performed after the client has sent the Client Close Notify message and all incoming data has been read from the incoming data stream.

- Execute Socket Close:

  The server closes the socket connection with the *close()* function, passing the socket descriptor as a parameter.

- Execute the TLS Data Structure Delete:

  This function typically releases the resources utilized by the TLS Connection. This is only performed if this session is to be terminated and is not done for sessions that may later be resumed. Depending on implementation, the memory used by the data structure may have to be de-allocated after use.

10.6 Parsing the Received Message

Once an IA has received a message from a peer IA, the message is parsed. In parsing the message the initial tag is examined to determine which of the standard syntaxes shall be used to decode and interpret the remainder of the message.

10.7 Transfer Data to Immediate User (Translator/Security Module)

In the event that the message is of the "Basic" type, user data contained in the message should be passed to the IA's immediate user (generally an EDIFACT/ASC X12 EDI translator) for whatever further processing is required. The mechanism by which this is achieved in the IA is outside the scope of this ITU-T Recommendation.

If the message is of the "Enhanced" type, the contents should be passed to the relevant security module for further parsing and required security validations.

If the transfer of data to the immediate user of the Interactive Agent fails, the IA may initiate flow control procedures as specified in 11.3.

10.8 Receipt Logging

Log entries should be created as a consequence of the reception of data by the server. The minimal set of data to be logged should consist of the following:

- Date/Time received.
- Unique message identifier (e.g. ASC X12 EDI ISA segment).
- Remote IP address and local port number.
- Success/fail indicators of reception and transfer to the immediate user (normally an EDIFACT/ASC X12 EDI translator or a security module).

11 Operational requirements

11.1 Security

Two communicating peer entities shall bilaterally agree on the level of to be employed in the IA exchange, and on the method for supporting the agreed upon level of security.

The following guidelines apply to the user of Transport Layer Security (TLS) services:

- Strong peer entity authentication, based on public key encryption, shall be provided for all associations.
- Both IA clients and IA servers are expected to exchange digital certificates.
- The session secret key is encrypted with the receiver's public key.
- The use of message encryption is optional, but recommended.
- SHA-1 is the recommended digest algorithm for TLS transmission block integrity functions.
- If TLS privacy protection is selected, Data Encryption Standard in the Cipher Block Chaining (DES-CBC) mode is recommended for symmetric key encryption.
- Every entity using the IA is required to obtain a public key certificate from a CA acceptable to the peer entities.
- Certificates shall be compatible with X.509 version 3.

NOTE 1 – Resumable sessions do not present any additional security threat.

NOTE 2 – Refer to ITU-T Recommendation Q.815 for details of message level security specifications.

11.2 Digital Certificates

The IA architecture requires that both parties exchange digital certificates during the TLS handshake. Upon receiving a certificate from a peer, the certificate information should be passed from the TLS (transport layer) via the IA to the Security Module where it will be retained and used for enhanced security operations such as non-repudiation. The same key pair and certificate used for authentication of the peer system will also be utilized for any required digital signatures.

Peer entities should agree to mutually acceptable trusted Certificate Authorities. Only certificates issued by these trusted authorities should be exchanged or referred to in Digital Signatures.

In the event digital certificates or certificate lists use the ASN.1 data type **`UTCTime`**, the following procedure is to be followed:

Before a value of **`Time`** is used in any comparison operation, and if the syntax of **`Time`** has been chosen as the **`UTCTime`** type, the value of the two-digit year field shall be rationalized into a four-digit year value as follows:

- If the 2-digit value is 00 through 49 inclusive, the value shall have 2000 added to it
- If the 2-digit value is 50 through 99 inclusive, the value shall have 1900 added to it.

NOTE 1 – The use of **GeneralizedTime** may prevent interworking with implementations unaware of the possibility of choosing either **UTCTime** or **GeneralizedTime**. It is the responsibility of those specifying the domains in which certificates defined in this Directory Specification will be used, e.g. profiling groups, as to when the **GeneralizedTime** may be used. In no case shall **UTCTime** be used for representing dates beyond 2049.

NOTE 2 – The procedure exists to satisfy a Y2K issue created by ASN.1 defining **UTCTime** to comprise of only a 2-character year.

NOTE 3 – More information regarding **UTCTime** in X.509 certificates and digital signatures is specified for ITU-T Recommendation X.509 (1995) in ITU-T's Directory Implementors Guide (Version 11) and incorporated in ITU-T Recommendation X.509 (1997).

11.3 Flow Control

The transport mechanism specified by the Interactive Agent requires an individual TLS session to be set up for each message. The communication is essentially one way, from the client to the server. The IA status message is a mechanism that allows peer entities to exchange errors and other types of flow control information. Specific message codes may be defined by the peer entities outside the scope of this ITU-T Recommendation; see Table 2.

The server also has the option of refusing an offered TLS session setup if conditions on its side prevent prompt processing of an incoming message. In this situation the client is expected to retry the connection after an agreed upon delay.

When an IA server receives a Status Message with a value indicating either upper or lower level congestion, it indicates to its client side, by a means external to this ITU-T Recommendation, to cease transmission of subsequent IA messages destined to the originator of the Status Message until outside intervention causes processing to resume.

12 Port Assignments

TCP/IP port assignments are to be agreed upon by the peer entities. These ports need not be the same at each end of the connection. The ports are associated with the TCP protocol stack within the system configuration.

The Internet Assigned Number Authority (IANA) has registered the protocol described in this ITU-T Recommendation and assigned it protocol number 117. The protocol is registered as "Interactive Agent Transfer Protocol" and is abbreviated as ***iatp***.

The IANA has assigned port number 6999 to ***iatp-normalpri***. The use of this port is recommended for normal priority transactions using this protocol.

The IANA has assigned port number 6998 to ***iatp-highpri***. The use of this port is recommended for high priority transactions using this protocol.

ANNEX A

ASN.1 Production Module

InteractiveAgent {itu-t(0) recommendation(0) q(17) q814(814) ia(0) messages(0)} DEFINITIONS IMPLICIT TAGS ::= BEGIN

-- EXPORTS everything

```
IaMessage ::= CHOICE {
        basicMessage        BasicMessage,        -- Basic Message
        iaStatusMessage     IaStatusMessage,     -- IA Status/Control Message
        enhancedMessage     EnhancedMessage      -- Security Module Enhanced Message
}

BasicMessage ::= CHOICE {

basicMessage1     GeneralString,

basicMessage2     IA5String

}
IaStatusMessage ::= BIT STRING ( SIZE (32) )
EnhancedMessage ::= OCTET STRING          -- Security Module Enhanced Message
END
```

ANNEX B

Design Considerations

This annex identifies the design considerations/constraints that will make the Interactive Agent efficient and robust.

B.1 Multi-processing/Multi-threading

The nature and application usage of the IA will normally require multiple concurrent IA entities to be operating simultaneously. Multiple clients may be achieved by simply invoking multiple instances of IA client software or processes. Other schemes may be utilized, as well. IA server processes are normally structured to handle multiple simultaneous client requests through processes such as *child-process forking, multi-threading, multiplexing,* or other comparable technologies.

NOTE – Each IA Server should consider providing up to a maximum of 16 concurrent connections per peer entity.

B.2 Non-Persistent Versus Persistent Connections

Each TLS connection may support the transfer of either a single EDIFACT or ASC X12 EDI message or multiple such messages over a single session. The former case requires that a session should exist only for the duration of transmission of a single message while the latter allows a connection to be maintained for as long as mutually agreed to by the peer entities.

The decision to use a single message per session versus multiple messages per session is a design decision based on application requirements. In an application where the number of clients may be large, the single message per session paradigm may make the best use of resources. On the other hand, in an application that requires a large number of messages to be exchanged between a small number of clients, the multiple messages per session case will likely be the most efficient.

B.3 Resumable TLS Sessions

For performance reasons, resumable TLS sessions are recommended. Full TLS establishment is processor intensive; therefore, a mechanism needs to be provided that allows a session to be resumed using an optimal subset of the TLS setup processing.

The length of time that a session will be maintained as *resumable* is outside the scope of this ITU-T Recommendation. Implementations should provide for this parameter to be configurable. If sessions are to *age out* of a *resumable* session cache, this parameter should be configured to initiate these expirations after the indicated number of minutes has elapsed since the last use of the session ID. Recommended values are in the range of one to thirty minutes.

ANNEX C

Error Handling/Recovery

An Interactive Agent session may be abruptly terminated due to technical failures. Recovery in this case consists of establishing a new session and re-sending the transaction in progress. However, there exists a well defined point at which the client considers a transaction to be successfully sent. This point is when the *Close Notify* is received from the server. After this point recovery is at the end-to-end application level, which will be triggered by failure to receive an acknowledgment transaction (e.g. ASC X12 997 – Functional Acknowledgment) within the time-out period.

Application level recovery is indicated by the time-out expiring while awaiting the matching acknowledgment transaction (e.g. ASC X12 997 – Functional Acknowledgment). Separate time-outs may be specified for Normal and High Priority transaction types for each peer entity.

APPENDIX I

Non-normative references

- ISO 9735:1988, *Electronic data interchange for administration, commerce and transport (EDIFACT) – Application level syntax rules.*
- ANSI ASC X12: American National Standards Institute (ANSI) Accredited Standards Committee X12. The Committee was chartered by ANSI in 1979 to develop uniform standards for the electronic interchange of business documents.

SERIES OF ITU-T RECOMMENDATIONS

Series A	Organization of the work of ITU-T
Series B	Means of expression: definitions, symbols, classification
Series C	General telecommunication statistics
Series D	General tariff principles
Series E	Overall network operation, telephone service, service operation and human factors
Series F	Non-telephone telecommunication services
Series G	Transmission systems and media, digital systems and networks
Series H	Audiovisual and multimedia systems
Series I	Integrated services digital network
Series J	Transmission of television, sound programme and other multimedia signals
Series K	Protection against interference
Series L	Construction, installation and protection of cables and other elements of outside plant
Series M	TMN and network maintenance: international transmission systems, telephone circuits, telegraphy, facsimile and leased circuits
Series N	Maintenance: international sound programme and television transmission circuits
Series O	Specifications of measuring equipment
Series P	Telephone transmission quality, telephone installations, local line networks
Series Q	**Switching and signalling**
Series R	Telegraph transmission
Series S	Telegraph services terminal equipment
Series T	Terminals for telematic services
Series U	Telegraph switching
Series V	Data communication over the telephone network
Series X	Data networks and open system communications
Series Y	Global information infrastructure and Internet protocol aspects
Series Z	Languages and general software aspects for telecommunication systems

Printed in Switzerland
Geneva, 2001