Electronic Trading-Partner Agreement for E-Commerce

(tpaML)

Pre-submission Draft, Version: 1.0.3

Technical Contacts:

Martin Sachs
IBM T. J. Watson Research Center
POB 704
Yorktown Hts., NY 10598
USA
email: mwsachs@us.ibm.com

John Ibbotson
MQSeries Technical Strategy & Planning,
IBM UK Ltd,
Hursley Park,
Winchester, SO21 2JN
United Kingdom
email: john_ibbotson@uk.ibm.com

Copyright IBM Corporation 2000

Preface

The Internet stretches traditional strict transaction processing concepts in several directions. First, transactions spanning multiple independent organizations may need to address enforcement of pairwise trading-partner agreements (TPAs). Second, a new transaction processing paradigm is required that supports different views of a unit of business for all participants, i.e., service providers as well as end consumers. This paradigm provides loose coupling of autonomous organizations rather than global data consistency. Global data consistency is not feasible in this environment, both because of long network delays and because it is not appropriate for autonomous organizations to lock each other's resources. There may be a unit of business consisting of several related interactions between any two interacting parties dispersed in time, creating a ***long-running conversation****. Hence, persistent records of business actions need to be kept. Additionally, since each organization's internal processes are independent of those of the other organizations, rollback of the effects of these interactions cannot be done. Instead the results of an interaction may be cancelable (however, this may not mean that all effects are undone, e.g., nonrefundable payments). The greater variability in response time for network computing creates a need for asynchronous and event-driven processing, in which correct handling of reissued and canceled requests is critical. The rules of interaction specified explicitly in the TPA permit interactions across autonomous organizations. This document presents a specification of electronic TPAs.*

Table of Contents

Preface 2
Table of Contents 3
1 Objective 6
1.1 Normative References 7
1.2 Document organization 7
2 TPA Definition 9
2.1 Overview 9
2.2 Formats of Character Strings 9
2.3 Tag Names and Attribute Names 9
2.3.1 Attribute Values 10
2.3.2 Party Names and Reference Attributes 10
2.3.3 Alphanumeric Strings 11
2.3.4 Numeric Strings 11
2.3.5 Addresses (geographic, postal) 11
2.3.6 Telephone Numbers 11
2.3.7 Time and Date Quantities 11
2.3.7.1 Time Intervals 11
2.3.7.2 Dates 12
2.3.7.3 Time of Day 12
2.3.8 URLs 12
2.4 TPA Structure 12
2.5 TPA Preamble 13
2.5.1 Identification 13
2.5.2 TPA Type 14
2.5.3 Role Definitions 14
2.5.4 Participants 16
2.5.4.1 Arbitration of TPA Disputes 18
2.5.4.2 Syntax of Participants Section 19
2.5.5 TPA Lifetime 20
2.5.6 Invocation Limit 21
2.5.7 Concurrent Conversations 21
2.5.8 Conversation Instantiation, Closure, and Lifetime 21
2.5.9 Preamble Syntax 22
2.6 Transport 22
2.6.1 Communication Protocol and Version 23
2.6.2 Communication Addressing 23
2.6.3 Network Delay 23
2.6.4 Communication Syntax 23
2.6.4.1 HTTP 24
2.6.4.2 MQSeries 24
2.6.4.3 SMTP 27
2.6.4.4 VAN-EDI 27
2.6.5 Transport Encoding 28
2.6.6 Transport Security 28
2.6.6.1 Specifics for HTTP 30
2.6.6.2 Transport Security Syntax 30
2.7 Document Exchange 31
2.7.1 Document-Exchange Protocol 31
2.7.2 Idempotency Test 32
2.7.3 Message Security 32
2.7.3.1 Message Security Syntax 34
2.8 Business Protocol 35

2.9 Service Interface 35
2.9.1 Service Provider 36
2.9.2 Client 36
2.9.3 Task Name 36
2.9.4 Actions 36
2.9.5 Server Service Time 36
2.9.6 Sequencing Rules 37
2.9.7 Conversation Termination 38
2.9.8 Service Security 38
2.9.9 Service Interface Syntax 38
2.9.10 Action Menu 39
2.9.11 Action Definition 39
2.9.12 Attributes of the Action Tag 39
2.9.13 Action ID 40
2.9.14 Action Type 40
2.9.14.1 Action Type Concurrent 40
2.9.15 Invocation Attribute 40
2.9.16 Request 41
2.9.16.1 Request Name 41
2.9.16.2 Input Information to Request 41
2.9.17 Response to Request 42
2.9.17.1 Response 42
2.9.17.2 Query 43
2.9.17.3 Application Result Information 44
2.9.18 Timing of Action Results 44
2.9.19 Presumed Result Following Timeout 45
2.9.20 Error Handling 45
2.9.21 Sequencing 46
2.9.22 Action Security 46
2.9.23 Complete Action Definition 46
2.10 Comment Text 47
Appendix 1 Complete TPA Definition 48
Appendix 2 DTD Corresponding to Complete TPA Definition 55
Appendix 3 XML Schema Document Corresponding to Complete TPA Definition 59
Appendix 4 Example: OBI 69
Appendix 4.1 TPA for Open Buying on the Internet (OBI) 70
Appendix 5 Examples of Reusable and Fully Qualified TPAs 76
Appendix 5.1 Reusable TPA 76
Appendix 5.2 Fully Qualified TPA 80
Appendix 6. Mapping of TPA Constructs to Message Header 86

1 Objective

This specification defines the language for creating electronic trading-partner agreements (TPAs) between two business partners (parties to the TPA). Like the trading-partner agreements used in Electronic Data Interchange (EDI), these TPAs define the "information technology terms and conditions" that enable business documents to be electronically interchanged between partners. However, these TPAs are not paper documents. Rather, they are electronic documents, written in XML, which can be processed by computers at the partners' sites in order to set up and then execute the desired business information exchanges.

In general, the parties to a TPA can have both client and server characteristics. A client requests services and a server provides services to the party requesting services. In some applications, one party only requests services and one party only provides services. These applications have some resemblance to traditional client-server applications. In other applications, each party may request services of the other. As will be explained later, tags under the `<ServiceInterface>` tag define the client and server roles of the different parties.

A TPA describes all the valid visible, and hence enforceable, interactions between the parties and is independent of the internal business processes of each party. Each party builds its own internal business process to satisfy these external TPAs and interface them to the rest of its business processes. However, the internal business processes are in general not visible to other parties (unless desired by the service providers themselves). The intent is to provide a high-level specification that can be easily comprehended by humans and yet is precise enough for enforcement by computers.

> NOTE: It is possible to provide a graphic TPA-authoring tool that understands both the semantics of the TPA definition and the XML syntax. Equally important, the definitions in this specification make it feasible to automatically generate, at each party's site, the code needed to execute the TPA, enforce its rules, and invoke the application-specific programs.

A typical TPA consists of various kinds of information that are listed below. This list is not intended to show the structure of the TPA; the structure will be described in "TPA Definition".

> NOTE: Many fields in the TPA are the same in most TPAs; the authoring tools and TPA templates can supply these fields. Where appropriate, the tool can supply default values for many fields; these only need to be stated when other than the default values are desired.

Information in the TPA includes:

1. *Identification* to identify uniquely the TPA document, and the parties,
2. *Communication* to specify the transport protocol(s), electronic addresses of the parties,
3. *Security*, to define the certificates used for authentication, nonrepudiation, and digital envelope, and other security parameters,

4. *Invocation-Independent Properties* to specify overall properties of the TPA, e.g., the valid duration of the TPA.
5. *Data Definition* for describing formats of the data being passed around,
6. *Role Definition* that describes each of the roles specified in the TPA that can be filled by specific parties,
7. *Action list* to describe the requests each party can issue to the other. These actions are the independent units of work. The action definitions define the associated message flows between the invoker and the service provider, responsiveness, failure handling, and other attributes,
8. *Sequencing rules* to describe valid action invocation sequences in each party,
9. *Global properties* to describe default properties of various fields in the TPA, e.g., responsiveness.
10. *Comments* to describe handling of disputes, termination of the TPA as a whole, and other exceptional conditions.

NOTE: Conceptually, the TPA may be considered to be implemented by a business to business (B2B) server at each party's site. The B2B server provides the code for the services needed to support the TPA including the middleware which supports communication with the other party, execution of the functions specified in the TPA, interfacing to each party's back-end processes, and logging the interactions between the parties for purposes such as audit and recovery. The middleware might support the concept of a long-running conversation as the embodiment of a single unit of business between the parties. This specification uses the term "registration" to denote the process of setting up the TPA for use at each party's site, i.e. recording the static information in a local database and generating the necessary code to support the TPA.

1.1 Normative References

The following standards contain provisions that, through reference in this standard, constitute provisions of this standard.

1. Message Header Specification

   NOTE: The precise title and document identification will be added when known.

   NOTE: This specification is normative for new applications but may be disregarded for legacy applications.

2. NOTE: Other standards referenced by this standard will be added to this list.

1.2 Document organization

This specification is organized as follows. Section 2, contains the definition of the TPA, identifying the structure and all necessary fields. The appendices include the complete XML TPA definition, a DTD, an XML Schema document equivalent to the DTD, a sample application, and various samples of TPAs.

In this specification, the term "framework" denotes the software infrastructure that supports the TPA.

2 TPA Definition

2.1 Overview

This section describes in detail each of the TPA sections, and the syntax for specification of these sections.

The TPA may exist at several different levels of abstraction:

- An abstract description, as in this standards specification, from which any TPA among any parties for any purpose could be derived.
- A prototype TPA (template) between two or more parties for a specific purpose. This template can be tailored as needed, resulting in a particular TPA. In the template, one or more parties can be represented by role parameters, such as *name*, to be replaced by specific parties in a particular TPA.
- An actual TPA among a specific set of parties.
- Instantiation of the TPA among a specific set of parties for a particular long-running conversation.

  NOTE: Following is an abstract description of how support of the TPA might be implemented. This description is not part of the TPA specification. Although object-oriented terminology is used here, there is no implication that TPA support must be in this fashion.

  The TPA is instantiated as a conversation by the framework, triggered by application code outside the scope of the TPA. Instantiation of the TPA as a conversation results in the creation of objects at each party that represent the TPA instance. They are known as server TPA objects and client TPA objects.

The TPA is formulated as an XML document. The tags are formally defined in a DTD file which is shown in appendix "2 DTD Corresponding to Complete TPA Definition". The corresponding XML Schema document is shown in appendix "3 XML Schema Document Corresponding to Complete TPA Definition".

2.2 Formats of Character Strings

In XML, the lowest level tags in any subtree have values which are character strings (#PCDATA in XML terminology). Tag names, attribute names, and attribute values are also character strings. This section defines format rules for these strings. Exceptions to these rules, and other details, are stated in the description of each tag.

2.3 Tag Names and Attribute Names

Tag names and attribute names are case sensitive (must match DTD and XML Schema document).

2.3.1 Attribute Values

If the DTD or XML Schema document specifies the permissible values for a particular attribute, the attribute stated in the TPA must match the value of one of the attributes defined in the DTD or XML Schema document including case. In this specification, case sensitivity is implied whenever a specific list of values is shown for an attribute.

Attribute values for which the DTD or XML Schema document does not specify a set of choices are also case sensitive.

Several tags, such as `<Action>`, have attributes (e.g. `ActionId`) which define identifiers that must be unique throughout the TPA.

> NOTE: An authoring tool could ensure uniqueness of these identifiers.

2.3.2 Party Names and Reference Attributes

Party names are referred to in various places in the TPA. In order to ensure that party names are used consistently throughout the TPA, the party name is defined in the `<PartyName>` tag under `<Participants>` as an ID-type attribute named `Partyname.` The value of `<PartyName>` is the name of the party. It is case sensitive. For example:

```
<PartyName Partyname="_ComputerCo">Computer Co</PartyName>
```

The `<OrgName>` tag is ued in those places that have to refer to the party name. It contains an IDREF-type attribute named `Partyname`. The value of this attribute must match (case sensitive) the value of the attribute in the matching `<PartyName>` tag. For example:

```
<OrgName Partyname="_ComputerCo"/>
```

It is recommended, for readability, that the value of the `Partyname` attribute be the name of the party, preceded by an underscore, and with any spaces omitted. For example, for Pens Are We, use "_PensAreWe". The attribute value must be written the same way wherever it is used.

> NOTE: The reason for the leading underscore is to accommodate party names that begin with numeric characters, such as 3COM and 3M. The underscore is needed to make the attribute value conform to the XML rules for valid ID-type attribute values.

For a TPA written in terms of roles (see "Role Definitions"), it is recommended, for readability, that the value of the `Partyname` attribute be the role name as given in `<RoleName>`, preceded by an underscore. In role substitution, the value of `<PartyName>` is replaced by the actual party name but the `Partyname` attribute value is not changed. For example:

```
<PartyName Partyname="_obiseller">@obiseller</PartyName>
<OrgName Partyname="_obiseller"/>
```

2.3.3 Alphanumeric Strings

Alphanumeric strings not further defined in this section follow these rules unless otherwise stated in the description of an individual tag:

- In tags under <ServiceInterface> (including those under <Action>), they are case sensitive unless otherwise stated.
- In tags elsewhere in the TPA, they are case insensitive unless otherwise stated.

> NOTE: The reason for the difference in the two above points is that the majority of the tags under <ServiceInterface> have case-sensitive values for specific reasons. It was therefore decided to simplify the rules by defining all tags under <ServiceInterface> as case sensitive.

- Strings that are copied into a database at TPA registration time for use by other programs or people and are not otherwise processed by the framework have no format requirements from a TPA viewpoint. Other programs that process this information may have requirements. In some cases, this specification recommends particular formats.
- Strings that are mapped into method calls or class names are case sensitive.
- Strings which represent file or directory names are case sensitive to ensure that they are acceptable to both UNIX and Windows systems.

2.3.4 Numeric Strings

A numeric string is a signed or unsigned decimal integer in the range imposed by a 32-bit binary number, i.e. -2,147,483,648 to +2,417,483,647. Negative numbers may or may not be permitted in particular tags; for example, a time interval is always positive.

2.3.5 Addresses (geographic, postal)

Values of tags which comprise a geographic or postal address (e.g. <City>, <State>, <Zip>) are in USA format.

> NOTE: An authoring tool can accept input of other national forms of address and convert them to US format. Local mailing software can convert and format the values of these tags as appropriate (e.g. to the national destination of a letter). The value of <Zip> should allow enough digits to contain the largest known postal code.

2.3.6 Telephone Numbers

A telephone number or fax number consists of a numeric string containing the area code followed by the phone number with no punctuation marks.

2.3.7 Time and Date Quantities

2.3.7.1 Time Intervals

A time interval is expressed as a numeric character string whose value is a positive integer number of seconds.

NOTE: An authoring tool could allow the author to enter time intervals in any convenient units, perhaps selected from a pull-down list, and convert the input to seconds.

2.3.7.2 Dates

A date is expressed in United States format, mm/dd/yyyy, i.e. a three numeric integer strings separated by slashes, a 2-digit month, a 2-digit day, and a 4-digit year.

NOTE: An authoring tool could accept other date formats and convert them to the US format.

2.3.7.3 Time of Day

A time of day is expressed in the form hh:mm:ss (hours, minutes, seconds) using a 24-hour clock, i.e. the value of hh is in the range 00-24. Times are expressed as Coordinated Universal Time.

2.3.8 URLs

A URL conforms to the standard URL format. Fields to the left of the first slash (i.e. protocol and domain name) are case insensitive. Fields to the right of the first slash are case sensitive since most such fields are directory or file names, which must be case sensitive to conform to UNIX rules.

2.4 TPA Structure

In this section we describe the overall structure of the TPA. Unless otherwise noted, TPA fields must be in the order shown here. Subsequent sections describe each of the functions provided by the TPA.

Following is the overall structure of the TPA, expressed in XML.

```
<TPA>
<TPAInfo>  <!-- TPA preamble -->
     ...  <!--TPA name, participants, etc.-->
     </TPAInfo>
<Transport>
     ... <!--communication information-->
     </Transport>
<DocExchange>
     ... <!--document exchange information-->
     </DocExchange>
<BusinessProtocol>
     ... <!--Business protocol information-->
     </BusinessProtocol>
<Comment>
     <!--any reference text-->
     </Comment>
</TPA>
```

The <BusinessProtocol>, <DocExchange>, and <Transport> sections describe the processing of a unit of business (conversation). These sections form a layered structure somewhat analogous to a layered communication model.

Business-Protocol layer - The Business Protocol layer defines the heart of the business agreement between the trading partners: the services (actions) which parties to the TPA can request of each other and sequencing rules that determine the order of requests. The Business-Protocol layer is the interface between the TPA-defined actions and the business-application functions that actually perform the actions.

Document-Exchange layer - The Document Exchange layer accepts a business document from the Business Protocol layer, optionally encrypts it, optionally adds a digital signature for nonrepudiation, and passes it to the transport layer for transmission to the other party. The options selected for the Document Exchange layer depend on those selected for the Transport layer. For example, if message security is desired and the selected transport protocol does not provide message encryption then it must be specified at the Document-Exchange layer.

Transport layer - The transport layer is responsible for message delivery using the selected transport protocol. The selected protocol affects the choices selected for the Document-Exchange layer. For example, some transport layer protocols may provide encryption and authentication while others have no such facility. Optional functions selected for the Document-Exchange layer must take into account the nature of the transport layer.

It should be understood that the functional layers encompassed by the TPA have no understanding of the contents of the payload of the business documents.

The TPA preamble (<TPAInfo>) contains those definitions that are independent of instantiation, such as TPA name, role definitions, party names, and TPA duration.The optional comment section (<Comment>) may contain any reference text that must be included in the TPA.

2.5 TPA Preamble

The preamble identifies the TPA and its participants and defines some invocation-independent properties such as the valid duration. The preamble is defined by the information under the <TPAInfo> tag.

2.5.1 Identification

The TPA is given a unique identification string, such as US14442869, using the <TPAName> tag.

2.5.2 TPA Type

The <TPAType> tag provides information about the general type of TPA where the TPA is based on a defined standard business-level protocol such as OBI. It is optional. It includes the following information: business protocol, protocol version, and protocol type. The syntax is

```
<TPAType>
     <Protocol>protocol</Protocol>
     <Version>version</Version>
     <Type>type</Type>
     </TPAType>
```

The <Protocol> tag identifies the business-level protocol. The <Version> tag identifies the version of the protocol being used
The <Type> tag provides additional information about the business protocol. Specific values depend on the particular protocol and its optional features.

> NOTE: An example of a protocol is OBI. For OBI, an example of the value of <Version> is 1.0, and the values of <Type> are SS (server to server), and SBS (server-browser-server), which define whether or not one or more of the message flows is via the client's browser.

The values of all tags under <TPAType> are case-insensitive alphanumeric strings.

2.5.3 Role Definitions

A TPA may be between specific parties or between some combination of specific parties and arbitrary parties (roles). The <Role> tag defines the relationships between roles and actual parties. It is optional and should be omitted if roles are not used. A TPA containing one or more roles is referred to as a prototype TPA or a reusable TPA. The resulting TPA, with the roles replaced by specific parties, is called a fully qualified TPA.

The syntax of the role definition section is:

```
<Role>
     <RoleDefn>     <!--one or more-->
          <RoleName>role_name</RoleName>
               <!--example:  hotel, airline -->
          <RolePlayer>player_name</RolePlayer>
               <!--example:  Hotelco, Airlineco-->
          </RoleDefn>
     </Role>
```

In a prototype TPA, the value of <RolePlayer> may be null, blank, or an arbitrary character string. This must be replaced by an actual party name when a party is assigned to a role.

An arbitrary party is referred to as a role and is denoted by a role parameter. The <RoleName> tag defines role parameters. Any alphanumeric string may be used as a role parameter. This

string is case sensitive and must be written the same way wherever it appears in the TPA. The actual party name is given in the associated <RolePlayer> tag. It is case sensitive.

A specific party must be assigned to play each unfilled role during the instantiation of a TPA from a prototype TPA. To identify a specific party to play a role, the role parameter must be replaced by the actual party's organization name.

Role parameters are used in the <PartyName> tags. In these tags, the role parameter must be preceded by @. For example, where the role parameter is airline:

```
<PartyName Partyname="_airline">@airline</PartyName>
```

NOTE: The @ has been chosen as the designator since it is one of the few special characters that have no significance to XML. The leading underscore on the value of PartyName is to satisfy an XML requirement that ID attributes must not begin with numeric characters. Adding the underscore allows for party names such as 3COM and 3M.

The <OrgName> tag identifies the parties in other places in the TPA. The identification is via the Partyname attribute. See "Party Names and Reference Attributes".

In addition, party-dependent tags must be given the appropriate values. In general, these tags are in a subtree below the tag of which <OrgName> is a subelement. For example, when the value of <PartyName> is converted from a role parameter to an actual party name, most tags under <Member> must be given party-specific values. The <Member> tag also has party-dependent attributes (IdCodeType and MemberId) which must be filled in when replacing a role by a specific party.

NOTE: A prototype TPA written in terms of roles might contain some party-dependent tags that have values. It is suggested that an authoring tool present these tags to the TPA author and ask that the author either indicate acceptance of these values or supply replacement values.

Each place where party-specific tags exist will be pointed out below in the discussion these areas.

Note that each party to a TPA has some unique properties (at least unique contact name and address and communication addresses). Therefore, the number of role parameters is equal to the number of parties to the TPA that are not specified in a prototype by actual names.

When a role parameter is used for a party name, the associated party-dependent tags throughout the TPA should be included. Their values can be null strings, blanks, or arbitrary strings. These strings must be replaced by actual values when the role parameter is replaced by an actual party name. Any party-dependent attributes should be given values of null strings ("") or blank strings (" ").

See the appendix "5 Examples of Reusable and Fully Qualified TPAs" for an example of a reusable TPA and the resulting fully qualified TPA.

It should be noted that in principle, the whole TPA might have to be negotiated when specific parties are assigned to roles. Roles will be most useful when it can be expected that most parties who fill the roles can be expected to agree to most or all details of the TPA.

> NOTE: Using this method of associating role names with role players permits use of a tool for instantiating an actual TPA from a TPA template by using the associations under `<Role>` to automatically replace each role name by the player name. In general, such a tool would be an interactive tool which would request values for each of the party-specific tags associated with a role name everywhere in the TPA. An authoring tool could provide this capability.
>
> In addition to having a TPA author manually fill in the party-specific information, this process could be automated by providing a database containing the necessary information about all parties likely to participate in copies of a TPA generated from a particular template. Such a database could be used under the following circumstances:
>
> 1. An authoring tool could use it to automatically fill in the party-specific information throughout the TPA once the author has identified the parties that replace the role parameters. In principle, the party-dependent tags need not be provided in the template at all since the authoring tool could provide them when producing a TPA for a specific set of parties from the TPA. However, including the party-specific tags (even though empty) aids readability of the XML document and also enables the XML document to be used to generate future specific TPAs.
> 2. The party-specific information could be filled in from the database during registration of the TPA at each party's site. Note that in this case, a TPA document containing the complete party-specific definitions would never exist; the party-specific information would simply be transferred from the party database to the registration database for the particular TPA. This method would preclude a specific set of parties from negotiating a TPA that contains the party-specific information and would thus be useful only when the parties can agree based on a template written in terms of roles.
> 3. The party specific information could be filled in at the time each conversation is instantiated from the TPA. This would permit a single TPA to be the basis for multiple concurrent conversations among different sets of parties. As above, a TPA document containing the complete party-specific definitions would never exist; see the caveats above.

2.5.4 Participants

The `<Participants>` section of the TPA describes all of the parameters needed for person-to-person communication between parties to the TPA (e.g. mailing address, email address). It should be noted that the email address is only for person to person communications. Specific forms of address are defined in the communication section for each of the access protocols. There is a `<Member>` tag for each party to the TPA and also an optional `<Arbitrator>` tag (see "Arbitration of TPA Disputes").

NOTE: Typically, the values of all tags under <Participants> might be copied into a database for use by the application, other programs, and people and are not further processed by the framework. There are thus no format requirements imposed by the TPA except as noted below.

The complete syntax is in "Syntax of Participants Section". Most of the tag names are self-explanatory and will not be discussed here. If no specific rules are given for the value of a tag, any alphanumeric string is acceptable from an XML viewpoint.

The <Member> tag provides the details about each party. It has two attributes, which are required. The value of the MemberId attribute is an alphanumeric string that provides an identifier for the member of the type defined by the IdCodeType attribute.

The value of the IdCodeType attribute defines the kind of member identifier given by the MemberId attribute. IdCodeType="01" denotes a Dun and Bradstreet D-U-N-S number. IdCodeType="ZZ" denotes any mutually agreed identifier. In other words, for IdCodeType="ZZ", the identifiers are defined by the parties for the purpose of the TPA but do not come from any industry-standard registry.

NOTE: Additional ID code types will be defined as needed for future applications.

The <PartyName> tag under the <Member> tag identifies each party to the TPA by its usual organization name. For example, in an application in which a travel agent provides air reservations for customers, the <PartyName> tag identifies the travel agent and airline. In other places in the TPA which refer to parties, the <OrgName> tag is used. It contains an ID reference attribute pointing to the appropriate <PartyName> tag. See "Party Names and Reference Attributes".

The value of the <PartyName> tag in a prototype TPA may be a role parameter. When the role parameter is replaced by an actual name, all of associated tags (in the tree below <Member>) must be given actual values. In addition, the attributes of <Member> must be given appropriate values.

The address may be one of several types, denoted by <AddressType>. A value of billing denotes a billing address. A value of shipping denotes a shipping address. A value of location denotes an address that is not one of the foregoing types.

The <Country> tag is required.

NOTE: It is the responsibility of each party's mail addressing software to determine whether to include the country in the address in a given piece of mail.

NOTE: <Contact> should be used to define the people who are responsible for administering the TPA, not necessarily people who are involved with specific transactions such as particular purchase orders. The latter names could change from

transaction to transaction and are matters for the business application. Those names will generally be exchanged in the messages for each transaction.

NOTE: It should be understood that information in `<Participants>` could change within the lifetime of the TPA. Such changes would not necessarily be propagated to the original TPA.

The attribute `Type` is used with `<Contact>`, `<ContactTelephone>`, and `<Email>`. `Type="primary"` denotes the primary contact or primary telephone or email address for this contact. `Type="secondary"` denotes the alternate contact or the secondary telephone or email address for this contact. If `Type` is omitted, `"primary"` is assumed.

NOTE: If a party considers any information under `<Participants>` as sensitive, both parties should provide adequate protection for the TPA document.

2.5.4.1 Arbitration of TPA Disputes

The TPA provides for arbitration of TPA disputes by an outside party. Examples of issues that might require arbitration are timeouts, action sequence errors, and rejection of an action invocation for reasons other than normal application errors. An "unable to comply" response might require arbitration whereas a "no room at the hotel" response does not require arbitration.

When a condition that requires arbitration arises, the parties communicate with the arbitrator and provide information that the arbitrator can use to determine fault and assess penalties if appropriate.

NOTE: The specific interactions between each party and the arbitrator are outside the scope of the TPA between the parties. Those interactions could be formalized by a separate TPA between each party and the arbitrator.

NOTE: Among the information that might be provided to the arbitrator is copies of each party's logs for the conversations involved in the dispute.

NOTE: A universal arbitrator interface can be provided in the framework, to be used by all arbitrators in all TPAs. The interface definition should be included in the programming documentation and does not appear in the TPA.

Each arbitrator is responsible for being able to communicate with each party and for its own availability regarding both arbitrator server failures and communication failures.

Designation of an arbitrator is optional. In order to use the services of an arbitrator, the arbitrator definition must be included in the TPA. If an arbitrator is specified, all parties must agree on a single arbitrator. This arbitrator is named by the `<Arbitrator>` tag under `<Participants>`. The arbitrator must also be included in the `<Communication>`, `<TransportSecurity>`, and `<MessageSecurity>` sections of the TPA.

The value of <PartyName> for the arbitrator in a prototype TPA may be a role parameter. When the role parameter is replaced by an actual name, all of associated the tags (in the tree below <Contact>) must be given actual values.

The arbitrator syntax is:

```
<Arbitrator MemberId=identifer IdCodeType=code >
          <!--The contents of this tag are defined the same
          as for <Member> (see "Participants")-->
     </Arbitrator>
```

2.5.4.2 Syntax of Participants Section

The syntax is:

```
<Participants>
     <Member MemberId=identifer IdCodeType=code> <!--1 or more-->
          <PartyName Partyname=name>name
               </PartyName>  <!--name of organization-->
               <!--The value of the Partyname attribute is the ID
                    by which the OrgName tag references this
                    party.-->
          <CompanyTelephone>telephone_number</CompanyTelephone>
          <Address>
               <AddressType>type</AddressType>
                    <!--AddressType is shipping, billing, or
                         location-->
               <AddressLine>address_line1</AddressLine>
               <AddressLine>address_line2</AddressLine>
                    <!--as many address lines as needed-->
               <City>city_name</City>
               <State>state_name</State>
                    <!--or province or similar division-->
               <Zip>zipcode</Zip>
                    <!--or similar postal code-->
               <Country>country_name</Country>
               </Address>
          <Contact Type=type>  <!--one or more-->
               <!--Type is "primary" or "secondary",
                    default is "primary"-->
               <LastName>name</LastName>
               <FirstName>name</FirstName>
               <MiddleName>name</MiddleName>
               <Title>title</Title>
               <ContactTelephone Type=type> <!--1 or more-->
                    phone_no</ContactTelephone>
                    <!--Type values same as for <Contact>-->
               <Email Type=type>email_address</EMail>
                    <!--1 or more-->
                    <!--Type values same as for <Contact>-->
               <Fax>fax_number</Fax>   <!--optional-->
               </Contact>
```

```
     </Member>
     <!--Additional Member definitions-->
<Arbitrator MemberId=identifer IdCodeType=code >
     <!--The contents of this tag are defined the same
         as for <Member> -->
     </Arbitrator>
</Participants
```

2.5.5 TPA Lifetime

The <Duration> tag under <TPAInfo> specifies the valid duration of the TPA. <Duration> consists of <Start>, to specify the starting date and time of the TPA, and <End>, to specify the last date and time on which the TPA is valid. If <Start> is omitted, the TPA is valid immediately. If <End> is omitted, there is no limit to the valid duration. <Duration> can be omitted if the defaults for both <Start> and <End> are acceptable. <Start> and <End> each contain <Date> and <Time> to specify the date and the time on that day. See "Dates" and "Time of Day".

When the end date and time are reached, any actions that are still in progress will be allowed to complete and no new action requests will be accepted. When all in-progress actions on each conversation are completed, the conversation will be terminated whether or not it was completed.

> NOTE: It should be understood that if an application recognizes a "unit of business" consisting of multiple action requests, such units of business may be terminated with no error indication when the end date and time are reached. If the framework also recognizes this unit of business as a conversation, it could provide an error indication to the application. If the framework does not recognize this unit of business as a conversation, it is the responsibility of the application to recognize the situation and recover.
>
> NOTE: It should be understood that it may not be feasible to wait for outstanding conversations to terminate before ending the TPA since there is no limit on how long a conversation may last.

Following is the syntax of <Duration>.

```
<Duration>
     <Start>  <!--Optional-->
          <Date>date</Date>
          <Time>time</Time>
          </Start>
     <End>
          <Date>date</Date>
          <Time>time</Time>
          </End>
     </Duration>
```

2.5.6 Invocation Limit

The <InvocationLimit> tag specifies the total number of times the TPA can be instantiated (conversations created) before having to renegotiate it. If <InvocationLimit> is not stated, there is no limit to the number of invocations. The value of this tag is a positive numeric string.

2.5.7 Concurrent Conversations

The <ConcurrentConversations> tag defines the maximum number of conversations under this TPA that may be open at any time. If this tag is omitted, there is no defined limit though a given server may have an implementation-based limit and may reject a new conversation if this limit would be exceeded. The value of this tag is a positive numeric string.

2.5.8 Conversation Instantiation, Closure, and Lifetime

A conversation represents a unit of business defined by the business application. Therefore, the business application determines when the conversation is instantiated and closed. From the viewpoint of a TPA between party1 and party2, instantiation takes place at party1 when party1 sends the first action request to party2. At party2, the conversation is instantiated when it receives the first request of the unit of business from party1.

Closure of a conversation takes place when the parties have completed the unit of business. The application may use the service-interface tag <TerminateConversation> to indicate one or more actions whose completion terminates the conversation. See "Conversation Termination".

> NOTE: An implementation should provide means for the application to signal that a conversation is ended in addition to supporting <TerminateConversation>. This signal could be a user exit or explicit API call.
>
> NOTE: It is permissible to define an indefinite-length conversation that encompasses multiple units of business as understood by the business application. Doing so prevents the framework from providing services to help manage the units of business. In general, this technique should be limited to support of legacy applications that cannot indicate the beginning and end of a unit of business to the framework.
>
> NOTE: The framework may provide an interface by which the business application requests instantiation and closure of conversations.

To avoid tying up system resources indefinitely, a conversation should have a maximum lifetime defined in the TPA. The maximum lifetime of a conversation is defined by the <ConversationLife> tag under <TPAInfo>. The purpose of this tag is to detect "hung" conversations and perform error processing. Normally a conversation would end well before this time expires. The framework will detect expiration of this time and inform the application. The error processing must be performed by the application, applying suitable business rules depending on the state of the conversation. For example, money may have already changed

hands. If the <ConversationLife> tag is omitted, there is no defined maximum lifetime. The value of <ConversationLife> is an integer number of seconds.

Note that this section deals specifically with a single instantiation of the TPA (i.e. a single long-running conversation). Premature termination of the whole TPA is a legal matter and is discussed in "Comment Text".

2.5.9 Preamble Syntax

```
<TPAInfo>
     <TPAName>TPA name</TPAName>
          <!--name of TPA-->
     <TPAType>
          ...
          </TPAType>
     <Role>
          ...
          </Role>
     <Participants>
          ...
          </Participants>
     <Duration>     <!--valid duration for TPA-->
          ...
          </Duration>
     <InvocationLimit>number</InvocationLimit>
          <!--number of instantiations permitted before
               renegotiation is required-->
     <ConcurrentConversations>number</ConcurrentConversations>
          <!--Maximum number of concurrent conversations
               permitted-->
     <ConversationLife>time</ConversationLife>
          <!--maximum lifetime of a single conversation-->
     </TPAInfo>
```

2.6 Transport

The transport section of the TPA defines communication protocol, encoding, and transport security information. Specific forms of address are defined for each of the access protocols.

The overall structure of the transport section is:

```
     <Transport>
          <Communication>
               …
               </Communication>
          <TransportEncoding>encode</TransportEncoding>
          <TransportSecurity>
               …
               </TransportSecurity>
```

```
</Transport>
```

2.6.1 Communication Protocol and Version

Permissible communication protocols are HTTP, MQSeries, SMTP, and VAN-EDI. Only one protocol can be specified in a given TPA. The optional <Version> tag identifies a specific version of the protocol. The value of this tag is a character string.

2.6.2 Communication Addressing

The <...Node> tag identifies each party and its communication addressing information. The name of the node tag is specific to each protocol, e.g. <HTTPNode>. There is one <...Node> tag for each party. It identifies the party name using the <OrgName> tag that contains an ID reference attribute pointing to the corresponding <PartyName> tag under <Participants>. All parties must be represented in <Communication> including the <Arbitrator>. Under the <...Node> tag is also the unique electronic address of the party (typically an Internet address or URL). This address is contacted for creating an information-delivery channel. The form of address can be different for each communication protocol and is shown in the communication syntax below.

As shown in the syntax, there is a unique communication address tag for each protocol.

> NOTE: An authoring tool can choose the correct communication address tag for each protocol.

For each party which is represented by a role parameter in a prototype TPA, the communication addressing information must be supplied when the role parameter is replaced by an actual party name. This information is the values of the associated tags (under the same <...Node> tag under which the corresponding <OrgName> tag is located).

2.6.3 Network Delay

The <NetworkDelay> tag defines the expected worst-case round trip network delay for any message and its response. This delay excludes service time at the recipient of a message prior to its sending a response. Other tags in other parts of the TPA define the service time. The value of the <NetworkDelay> tag is a single parameter that must be agreed to by all parties. As explained elsewhere, timeouts for receipt of responses must include both the network delay and the appropriate service time value. If <NetworkDelay> is omitted, there is no defined network delay and the choice of network delay is a local option.

2.6.4 Communication Syntax

Following is the syntax for the communication section of the TPA along with discussion specific to each protocol.

2.6.4.1 HTTP

For HTTP, the address is a URL. Depending on the application, there may be one or more URLs, whose use is determined by the application. Each `<URL>` tag provides one URL. The `<URL>` tag has a required attribute, `URLName`, whose value identifies the purpose of the URL.

Following are the values of the `URLName` attribute:

`"logOnURL"` identifies the URL used for the initial connection between client and server.
`"requestURL"` identifies the URL used for receiving requests.
`"responseURL"` identifies the URL used for receiving response messages.

The default for `URLName` is `"requestURL".`

For example:

```
<HTTPAddress>
     <URL URLName="logOnURL">https://www.a.xxx.com</URL>
     <URL URLName="requestURL">https://www.b.xxx.com</URL>
     <URL URLName="responseURL">https://www.c.xxx.com</URL>
     </HTTPAddress>
```

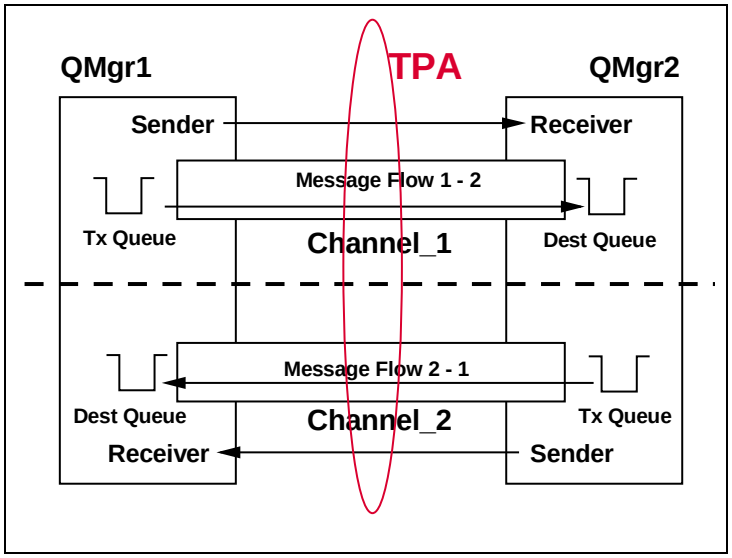
Following is the syntax for HTTP.

```
<Communication>
   <HTTP>
     <Version>version</Version>
     <HTTPNode> <!--One for each party-->
          <OrgName Partyname=name/>
          <HTTPAddress>
               <URL URLName=type>url</URL>
               <!--additional URL tags as needed>
               </HTTPAddress>
          </HTTPNode>
     <NetworkDelay>time</NetworkDelay>  <!--Optional-->
     </HTTP>
   </Communication>
```

2.6.4.2 MQSeries

This section explains the tags that define interconnecting two servers using IBM MQSeries queues.

In IBM MQSeries, each trading partner must be running a queue manager. In the following diagram, these are named QMgr1 and QMgr2. The connection between two remote queue managers is known as a channel. It can use either TCP/IP or some other transport. TCP/IP is used for TPAs. TPA-defined channels are known as Sender/Receiver channels.



A sender sends messages from its transmission queue, via the channel, to a receiver. The receiver puts the messages on its destination queue. Each partner must have a transmission queue defined but the name of that queue is not exposed in the TPA. It is simply part of the internal MQSeries configuration at each partner. The destination queue is either a request queue, for receiving request messages from the partner, or a reply queue, for receiving reply messages from the partner. The destination queues are defined in the TPA.

As shown in the above diagram, two channels are needed for bi-directional communication between the trading partners.

1. **Channel_1** handles communication initiated by **QMgr1** for **QMgr2**. This includes requests sent from **QMgr1** to **QMgr2** and responses from **QMgr1** returned to **QMgr2**. For this channel, **QMgr1** is the sender, **QMgr2** the receiver.

2. **Channel_2** handles communication initiated by **QMgr2** for **QMgr1**. This includes requests sent from **QMgr2** to **QMgr1** and responses from **QMgr2** returned to **QMgr1**. For this channel, **QMgr2** is the sender, **QMgr1** the receiver.

Following are explanations of the individual tags.

`<QMgrName>`: The partners' queue manager names (**QMgr1** and **QMgr2** in the above description). The name is case sensitive.

`<QmgrAddress>` and `<QmgrPort>`: The TCP/IP address (IP address or domain name) of each partner's queue manager machine together with an optional port address.

`<Channel>`: The interconnecting channel names (**Channel_1** and **Channel_2** in the above description). These must be unique within a partner's MQSeries installation. A sender channel is defined for each party. The remote end, at the other party, is the receiver channel. The name is case sensitive.

<RequestQ> and <ReplyQ>: The name of each partner's request and reply queues - unique within a partner. The names are case sensitive.

In addition, there are several parameters associated with MQSeries channels. If there is a mismatch in the parameters between the two ends of the channel, then there can be serious implications in the MQSeries performance. The following tags define channel parameters that may be different for each channel but each set must be agreed to by the two partners.

<BatchSize>: An integer numeric string that defines the maximum number of messages that can be sent through a channel before taking a checkpoint.

<DiscInt>: The disconnection interval, an integer numeric string that defines the minimum time in seconds for which the channel waits for a message to arrive on the transmission queue, after a batch ends, before terminating the channel. A value of zero causes the message channel agent to wait indefinitely. The value must lie in the range 0< DiscInt< 999999.

It should be understood that <BatchSize> and <DiscInt> are MQSeries performance parameters that do not affect the flow of messages at the business protocol (action) level. The stream of messages for a conversation may cross batch boundaries and <DiscInt> is independent of the network delay and service times defined elsewhere in the TPA. If the channel is terminated, it will be restarted with the next message.

<MaxMessageLength>: An integer numeric string that defines the maximum message length that can be transmitted on the channel. A value of zero means the maximum message length for the queue managers.

<SeqWrap>: Sequence Wrap, an integer numeric string that, when reached, causes the message sequence numbers to start again at 1. The value must lie in the range 100<SeqWrap<999999999. These are transport-level sequence numbers and are independent of the message identifiers used at the higher levels.

Following is the syntax for MQSeries.

```
<Communication>
     <MQSeries>
          <MQSNode> <!--One for each party-->
               <OrgName Partyname=name/>
               <MQSAddress>
                    <QMgrName>mgr_name</QMgrName>
                         <!-- Queue manager name -->
                         <!-- case sensitive-->
                    <QMgrAddress>address</QMgrAddress>
                         <!-- Queue manager TCP/IP address -->
                         <!--IP address or domain name-->
                    <QMgrPort>port</QMgrPort> <!--optional-->
                         <!-- TCP port number-->
                    <Channel>channel_name</Channel>
                         <!-- This party is a SENDER  -->
                         <!-- for this channel -->
                         <!-- case sensitive-->
```

```
                    <RequestQ>my_request_q</RequestQ>
                         <!-- Where this party expects to  -->
                         <!-- receive requests   -->
                         <!--case sensitive-->
                    <ReplyQ>my_reply_q</ReplyQ>
                         <!-- Where this party expects to  -->
                         <!-- receive replies  -->
                         <!-- case sensitive -->
                    </MQSAddress>
               <ChannelInfo>
                    <BatchSize>size</BatchSize>
                    <DiscInt>interval</DiscInt>
                    <MaxMessageLength>length</MaxMessageLength>
                    <SeqWrap>seq_no</SeqWrap>
                    </ChannelInfo>
               </MQSNode>
               <NetworkDelay>time</NetworkDelay>  <!--Optional-->
          </MQSeries>
     </Communication>
```

2.6.4.3 SMTP
SMTP is Simple Mail Transfer Protocol.

The specific tags are to be determined. Also to be determined is whether the definition should include MIME and S/MIME.

2.6.4.4 VAN-EDI
The name VAN-EDI refers to EDI as transported by a value-added network.

For VAN-EDI, the communication address (`<VANEDIAddress>`) defines the names of two mailboxes, one, `<InBox>`, for incoming and one, `<OutBox>`, for outgoing messages. The values of these two tags are strings whose contents represent the paths to the data in whatever form is appropriate for each party. The names are case sensitive.

Following is the syntax for VAN-EDI.

```
<Communication>
   <VANEDI>
     <Version>version</Version>
     <VANEDINode> <!--One for each party-->
          <OrgName Partyname=name/>
          <VANEDIAddress>
               <InBox>name</InBox> <!--mailbox name-->
               <OutBox>name</OutBox> <!--mailbox name-->
               </VANEDIAddress>
          </VANEDINode>
     <NetworkDelay>time</NetworkDelay>  <!--Optional-->
     </VANEDI>
   </Communication>
```

2.6.5 Transport Encoding

<TransportEncoding> specifies how the transport level encodes messages for transmission. Currently, BASE64 is the only choice.

Either <TransportEncoding> or <MessageEncoding> can be specified, but not both. <TransportEncoding> can be omitted if the encoding specification is dictated by a standard transport protocol and there are no options. Otherwise, if <TransportEncoding> is omitted, the default is no encoding (i.e. plain ASCII text).

The syntax is

```
<TransportEncoding>encoding</TransportEncoding>
```

2.6.6 Transport Security

The <TransportSecurity> tag provides the security specifications for the transport layer of the TPA. It may be omitted if transport security will not be used for this TPA. Unless otherwise specified below, transport security applies to messages in both directions.

For each party which is represented by a role parameter in a prototype TPA, the corresponding tags under <Party> and <LogonParty> must be given values when the role parameter is replaced by an actual name.

Transport security may be either encryption or authentication. Both encryption and authentication may be used in the same TPA, especially if authentication is by userid and password. Authentication verifies the sender of the message. Encryption prevents the message from being read by unauthorized parties but by itself performs no authentication or nonrepudiation

For encryption, the protocol and protocol version (optional) must be supplied by the appropriate tags. Examples of encryption protocols are RC2, RC5, and RC6. Only certificate-based encryption is defined.

For authentication, the authentication type must be specified with the <PasswordAuthen> or <CertificateAuthen> tag. The protocol (e.g. SSL), protocol version (optional), and certificate parameters or optional starting userid and password must be specified. A public key certificate must be supplied for each party (except as specified below).

Authentication is bi-directional if each party has to authenticate to the other during a given message exchange. The alternative is that the client authenticates to the server but not *vice-versa.* In most cases, the choice is determined by other factors. If SSL Ver. 3 is specified for encryption, authentication is always bi-directional. If the authentication type is PASSWORD, the client authenticates to the server. If the authentication type is CERTIFICATE, the authentication could be bi-directional or not. For the purpose of this TPA, it is defined as bi-directional unless otherwise specified below.

It should be understood that in a TPA in which each party can act as either a server or a client for different actions, the security definitions must enable each party to authenticate to the other, though not necessarily in the same message exchange.

The <Certificate> tag provides the parameters needed to define the certificates. The <CertType> tag identifies the type of certificate. Examples are X.509V1, X.509V2, and X.509V3. The <KeyLength> tag gives the key length in bits (e.g. 512). All parties must agree on the certificate type and key length. However different type and length may be specified for authentication, nonrepudiation, and digital envelope.

To support certificate authentication, the TPA must reference the public-key certificate for each party including the arbitrator. The <Party> tag defines the party-specific parameters. There must be one <Party> subtree for each party. <OrgName> contains an ID reference attribute which points to the corresponding <PartyName> tag.

For certificate authentication, the <IssuerOrgName> tag under <Party> identifies the certificate issuer, a recognized certificate authority. The value of <IssuerOrgName> must exactly match what is on the certificate, including case. The <IssuerCertSource> tag (under the <Party> tag) provides a URL from which the certificate issuer's certificate can be obtained during initialization of communication between parties. This URL is associated with the organization named in the <IssuerOrgName> tag. There is no requirement that all parties use the same certification authority. The corresponding private key for each party, of course, is private to each party and does not appear in the TPA.

> NOTE: Because the public-key infrastructure is still evolving, support for <IssuerOrgName> and <IssuerCertSource> is to some extent still to be determined. Parties to a TPA may have their own ways to obtain each other's certificates that are outside the scope of the TPA. A party may in any case use <IssuerOrgName> to validate the issuer of a certificate.

Password authentication includes the option of specifying a userid and password for use the first time each party invokes an action against the other. The <LogonInfo> tag is used for this purpose. The initial values specified for each party are those that the other party uses to log on.

> NOTE: Specifying initial values for userid and password in a TPA introduces a degree of security risk. If initial values are specified, the parties should take precautions. Both parties should keep the TPA document under security controls. Each party should change the password at the initial contact with the other party. Each TPA should specify a unique set of initial values (i.e. not reuse previously-used values). These precautions keep to a minimum the ability of an unauthorized party to make use of these values.

The security mode used for the communication that initializes a conversation depends on which security modes are defined in the <TransportSecurity> section. If authentication is defined, authentication is used. Otherwise, encryption is used if defined. If neither authentication nor encryption is defined, no security is applied to the initial communication.

2.6.6.1 Specifics for HTTP

For encryption with HTTP, the protocol is SSL (Secure Socket Layer) Version 2.0 or 3.0, which uses public-key encryption. The encryption tags include the certificate parameters.

Authentication may be either by password or by certificate. Certificate authentication is SSL version 3.0. For certificate authentication, the certificate is only the client certificate. However if each party may act as a server at times and a client at other times, the TPA must specify certificates for both parties.

2.6.6.2 Transport Security Syntax

```
<TransportSecurity>
     <Encryption>  <!--Optional-->
          <Protocol>protocol</Protocol>
          <Version>version</Version>
               <!--string denoting version of protocol-->
          <Certificate>
               <CertType>type</CertType>
                         <!--Example:  X.509V1, X.509V2-->
                    <KeyLength>length_in_bits</KeyLength>
                    <Party>
                         <!--one for each party that can act
                              as a server-->
                         <OrgName Partyname=name/>
                              <!--name of party-->
                         <IssuerOrgName>name</IssuerOrgName>
                              <!--Example:  VeriSign, Inc.-->
                         <IssuerCertSource>url</IssuerCertSource>
                              <!--URL of certificate-->
                         </Party>
               </Certificate>
          </Encryption>
     <Authentication>
          <PasswordAuthen>
               <Protocol>protocol</Protocol>
                    <!--Example: SSL-->
               <Version>version</Version> <!--optional-->
                    <!--version of protocol-->
               <LogonInfo>  <!--optional-->
                    <LogonParty>
                         <!--one for each party that can act as a
                            server-->
                         <OrgNamePartyname=name/>
                         <UserId>value</UserId>
                         <Password>value</Password>
                              </OrgName>
                         </LogonParty>
                    </LogonInfo>
               </PasswordAuthen>
          <CertificateAuthen>  <!--optional-->
               <Protocol>protocol</Protocol>
               <Version>version</Version>
               <Certificate>
```

```
                <CertType>type</CertType>
                     <!--Example:  X.509V1, X.509V2-->
                <KeyLength>length_in_bits</KeyLength>
                <Party>   <!--one for each party>
                     <OrgName Partyname=name/>
                          <!--name of party-->
                     <IssuerOrgName>name</IssuerOrgName>
                          <!--Example: VeriSign, Inc.-->
                     <IssuerCertSource>url</IssuerCertSource>
                          <!--URL of certificate-->
                     </Party>
                </Certificate>
           </CertificateAuthen-->
      </Authentication>
</TransportSecurity>
```

2.7 Document Exchange

The document-exchange section provides information that the parties must agree on regarding exchange of documents between them. This information includes the message security definition. The syntax is:

```
<DocExchange>
     <DocExchangeProtocol>protocol</DocExchangeProtocol>
     <MessageEncoding>encoding</MessageEncoding>
          <!--currently must be BASE64 or omitted-->
     <MessageIdempotency>choice</MessageIdempotency>
          <!--choice is yes or no-->
     <MessageSecurity>
          …
          </MessageSecurity>
     </DocExchange>
```

2.7.1 Document-Exchange Protocol

`<DocExchangeProtocol>` specifies the business protocol that defines the detailed message formats. Examples are `OBI` and `VAN-EDI`.

> NOTE: Some implementations may use this tag to identify the correct parser or mapper to process the messages.Message Encoding

`<MessageEncoding>` specifies how the messages are encoded by the document-exchange level for transmission. Currently, BASE64 is the only choice. Either `<MessageEncoding>` or `<TransportEncoding>` can be specified, but not both. `<MessageEncoding>` can be omitted if the encoding specification is dictated by a standard transport or document-exchange protocol and there are no options. Otherwise, if the `<MessageEncoding>` tag is omitted, the default is no document-exchange encoding (i.e. plain ASCII text).

Other definitions concerning message formats are provided within the action definition.

2.7.2 Idempotency Test

<MessageIdempotency> specifies whether all messages sent under the TPA are subject to an idempotency test (detection and discard of duplicate messages) in the document exchange layer. If the value of the tag is yes, all messages are subject to the test. If the value is no, messages are not subject to an idempotency test in the document-exchange layer. Testing for duplicates is based on the message identifier that is carried in the message header. See "Message Header Specification" in "Normative References".

> NOTE: Additional testing for duplicates may take place in the business application based on application information in the messages (e.g. purchase order number).

The idempotency test checks whether a message duplicates a prior message between the same client and server. If the idempotency test is requested, a duplicate message is passed to the recipient with a "duplicate" indication. A "duplicate" indication is also returned to the sender. One of the main purposes of this test is to aid in retry following timeouts and in recovery following node failures. In these cases, a request is reissued when it is not known whether the original request was received. If the original request was received, the duplicate is discarded by the server and the server returns the original results to the requester. A client which uses a query action to obtain results would re-send the query after receiving a "duplicate" indication.

If a communication protocol always checks for duplicate messages (e.g. MQSeries), the check in the communication protocol overrides any idempotency specifications in the TPA.

2.7.3 Message Security

The <MessageSecurity> tag provides the security specifications for the document exchange function. It may be omitted if message security will not be used for this TPA. Message security may be either nonrepudiation or digital envelope or both.

Message security applies to all messages in both directions for actions for which message security is enabled. See the discussion of <ServiceSecurity> and <ActionSecurity> below.

For each party which is represented by a role parameter in a prototype TPA, the corresponding tags under <Party> must be given values when the role parameter is replaced by an actual name. This must be done under <NonRepudiation> and <DigitalEnvelope>.

Nonrepudiation both proves who sent a message and prevents later repudiation of the contents of the message. Digital envelope is an encryption procedure in which the message is encrypted by a secret key and the secret key is sent to the message recipient encrypted with the recipient's public key.

The <NonRepudiation> and <DigitalEnvelope> tags are optional. Each must be supplied only if the particular mode will be used.

For nonrepudiation, the protocol (e.g. DigitalSignature), protocol version (optional), hash function (e.g. SHA1, MD5) encryption algorithm (see above), signature algorithm (e.g. DSA), and certificate type (see above) must be specified. A public key certificate must be supplied for each party (see below).

The <Certificate> tag provides the parameters needed to define the certificates. The <CertType> tag identifies the type of certificate. Examples are X.509V1, X.509V2, and X.509V3. The <KeyLength> tag gives the key length in bits (e.g. 512). All parties must agree on the certificate type and key length. However different type and length may be specified for nonrepudiation and digital envelope.

To support nonrepudiation and digital envelope, the TPA must reference the public-key certificate for each party including the arbitrator. Separate certificates can be defined for nonrepudiation and digital envelope. The <Party> tag defines the party-specific parameters. There must be one <Party> subtree for each party. <OrgName> contains an ID reference attribute that points to the corresponding <PartyName> tag. <OrgCertSource> is optional. If present, it provides the URL of a certificate issued by the party itself. <OrgCertSource> can be used only for digital envelope. The <IssuerOrgName> tag under <Party> identifies the certificate issuer, a recognized certificate authority. The value of <IssuerOrgName> must exactly match what is on the certificate, including case. The <IssuerCertSource> tag (under the <Party> tag) provides a URL from which the certificate issuer's certificate can be obtained during initialization of communication between parties. This URL is associated with the organization named in the <IssuerOrgName> tag. There is no requirement that all parties use the same certification authority. The corresponding private key for each party, of course, is private to each party and does not appear in the TPA.

Because the public-key infrastructure is still evolving, support for <OrgCertSource>, <IssuerOrgName>, and <IssuerCertSource> is to some extent still to be determined. Parties to a TPA may have their own ways to obtain each other's certificates that are outside the scope of the TPA. A party may in any case use <IssuerOrgName> to validate the issuer of a certificate.

The use of message security (i.e. nonrepudiation or digital envelope) for exchanges on a conversation is controlled by the <ServiceSecurity> tag in the <ServiceInterface> section and the <ActionSecurity> tag in the action definition. These tags control whether message security is applied to all, some, or no messages. The value of the <ServiceSecurity> and <ActionSecurity> tags is either yes or no. If the value of the <ServiceSecurity> or <ActionSecurity> tag is yes in any service interface or action definition, either or both of the message security protocols must be defined in the <MessageSecurity> section.

If the value of a <ServiceSecurity> tag is yes, then message security is applied to all messages to and from this server unless overridden by an <ActionSecurity> tag in an action definition with a value of no in the same service interface. If the <ServiceSecurity> tag has a value of no or is omitted, message security is not to applied to any messages unless specified in an <ActionSecurity> tag in an action definition in the same service interface.

In an action definition, an <ActionSecurity> tag with a value of yes means that message security will be applied to all messages for this action. A value of no means that message security will not be applied to messages for this action. If the <ActionSecurity> tag is not present in an action definition, the value of the <ServiceSecurity> tag in the same <ServiceInterface> section determines whether or not message security is applied to this action. If message security is not applicable to an action, encryption is used if defined in the <TransportSecurity> section. Otherwise no security is applied.

<DigitalEnvelope> consists of <Protocol>, <Version>, <EncryptionAlgorithm>, and <Certificate>, as previously described.

2.7.3.1 Message Security Syntax

```
<MessageSecurity>
     <NonRepudiation>
          <Protocol>protocol</Protocol>
               <!--example: DigitalSignature-->
          <Version>version</Version>
               <!--version of protocol-->
          <HashFunction>function</HashFunction>
               <!--example:  MD5, SHA1-->
          <EncryptionAlgorithm>algorithm
               </EncryptionAlgorithm>
               <!--Example:  RC2 -->
          <SignatureAlgorithm>algorithm</SignatureAlgorithm>
               <!--Example:  DSA-->
          <Certificate>
               <CertType>type</CertType>
                    <!--Example:  X.509V1, X.509V2-->
               <KeyLength>length_in_bits</KeyLength>
               <Party>   <!--one for each party>
                    <OrgName Partyname=name/>
                         <!--name of party-->
                    <IssuerOrgName>name</IssuerOrgName>
                         <!--Example:  VeriSign, Inc.-->
                    <IssuerCertSource>url</IssuerCertSource>
                         <!--URL of  certificate-->
                    </Party>
               </Certificate>
          </NonRepudiation>
     <DigitalEnvelope>
          <Protocol>DigitalEnvelope</Protocol>
          <Version>version</Version>
          <EncryptionAlgorithm>algor</EncryptionAlgorithm>
               <!--example:  RC4-->
          <Certificate>
               <CertType>type</CertType>
                    <!--Example:  X.509V3-->
               <KeyLength>length_in_bits</KeyLength>
                    <!example:  512-->
               <Party>   <!--one for each party>
```

```
                    <OrgName Partyname=name/>
                         <!--name of party-->
                    <OrgCertSource>url</OrgCertSource>
                         <!--URL of certificate-->
                    <IssuerOrgName>name</IssuerOrgName>
                         <!--Example:  VeriSign, Inc.-->
                    <IssuerCertSource>url</IssuerCertSource>
                         <!--URL of  certificate-->
                    </Party>
               </Certificate>
          </DigitalEnvelope>
     </MessageSecurity>
```

2.8 Business Protocol

The <BusinessProtocol> tag defines the section of the TPA which contains all the business-protocol definitions that support the business application. Under <BusinessProtocol> is the service interface definition for each party that can act as a server. The syntax is

```
<BusinessProtocol>
     <ServiceInterface>  <!--one or more-->
          ...
          </ServiceInterface>
     </BusinessProtocol>
```

2.9 Service Interface

The <ServiceInterface> tag defines the services provided by a party that acts as a server and the characteristics of the party that is a client. The services include the list of actions provided by that server, their characteristics, and the actions permitted initially. The TPA contains a separate service interface definition for each party that acts as a server.

> NOTE: For some applications, each party may have both server and client characteristics, i.e. each party may issue requests to the other party.

The <ServiceInterface> tag has one attribute, InterfaceId. The value of this attribute is an alphanumeric string that contains an identifier for this service interface. Each service interface must have an ID that is unique within the TPA. An example is InterfaceId="SERVER01".

For all tags under <ServiceInterface>, alphanumeric values are case sensitive unless otherwise stated. Any additional restrictions are described in the descriptions of the individual tags.

2.9.1 Service Provider

The <OrgName> tag identifies the party whose service interface is defined. This provides an ID reference attribute that points to the corresponding <PartyName> tag. If the service provider is represented by a role parameter, any party-specific tags under <Client> must be given values when the role parameter is replaced by an actual party.

2.9.2 Client

The <Client> tag identifies the party which can be a client to the party (acting as a server) defined by the particular service interface. It should also be noted that a given party can be both a server and a client, i.e. it may both receive requests and issue requests. The tags under <Client> define properties of that party with respect to the party which acts as a server.

The tag under <Client> is <OrgName>. This provides an ID reference attribute that points to the corresponding <PartyName> tag. See "Tag Names and Attribute Names".

The syntax of <Client> is

```
<Client>
     <OrgName Partyname=name/>
          <!--The value of Partyname is an ID reference to
          the corresponding PartyName tag.-->
     </Client>
```

2.9.3 Task Name

The <TaskName> tag provides a label to identify each service interface. The value is a case-insensitive aplhanumeric string. <TaskName> is optional.

2.9.4 Actions

The action menu is discussed following the other elements of the service interface.

2.9.5 Server Service Time

The <ServerServiceTime> tag provides information about the server's expected time (as seen at the server) to complete an action request and send the final reply. This tag gives a default value to be used with action definitions for this server, which do not include the <ResponseServiceTime> tag.

The <ServiceTime> tag gives the expected service time. The value is a character string containing a number of seconds.

<ServerServiceTime> is optional. If it is not present, there are no defined defaults for any of the quantities under <ServerServiceTime>. If this tag is not present, for any action definition that does not include <ResponseServiceTime>, the client's choice of a timeout value is outside the scope of the TPA.

See "Timing of Action Results" for additional discussion. See "Presumed Result Following Timeout" for a discussion of the <Presume> tag.

Following is the complete syntax of <ServerServiceTime>.

```
<ServerServiceTime>
     <ServiceTime>time</ServiceTime>
     <Presume>result</Presume>
          <!--Result is success or fail-->
     </ServerServiceTime>
```

2.9.6 Sequencing Rules

In general, within a single conversation, actions on a given service interface are performed sequentially. One action may not be invoked until the prior action is completed. An exception is actions with Type="concurrent". See "Action Type Concurrent".

Sequencing rules define which actions are allowed and which are not allowed by a given server after a given action on a given conversation. At any point in the conversation, some actions are allowed and others are not allowed. For example, a reservation cannot be canceled before it is requested. In the service interface definition for each provider, but outside the scope of the action menu, the <StartEnabled> tag identifies which actions are permitted as the initial action on that server. If <StartEnabled> is omitted, all actions defined by that server are permitted as the initial action.

In each action definition, sequencing rules cause actions to be added to or deleted from the current list of allowed actions. The <Sequencing> tag identifies which actions are allowed and which are not allowed following that action. The <Disable> tag identifies which actions are not permitted after the defined action. For example, a specific TPA may require that a new reservation request cannot be made after a successful reservation request. The <Enable> tag identifies which actions are permitted after the defined action. For example, a reservation can be confirmed, modified, or canceled after a successful reservation request. If the action definition does not include a <Sequencing> tag, the list of permitted actions is not changed. If the <enable> tag is omitted, no new actions are enabled. If the <disable> tag is omitted, no additional actions are disabled.

The rules in <Sequencing> are effective only if the action succeeded. If the action did not succeed, the lists of actions allowed and not allowed are unchanged.

An application might define a unit of business consisting of multiple actions and perform multiple sequential units of business in the same conversation. If so, the sequencing rules for the final action of a unit of business should restore the permitted and disallowed action lists to the state appropriate for the beginning of a new unit of business.

See "Action Type" and its subsections for any special considerations for the extended actions.

2.9.7 Conversation Termination

<TerminateConversation> may be used to indicate which action or actions cause the current conversation to be terminated. Termination occurs when the party that issued the action request receives the final reply message.

> NOTE: If this tag is not used, the application can signal conversation termination to the framework using an exit or API call provided by the framework.

The syntax is

```
<TerminateConversation>  <!--optional>
     <RequestName>action_name</RequestName>
          <!--one for each action defined as ending a
               conversation-->
     </TerminateConversation>
```

2.9.8 Service Security

See "Message Security" for a discussion of the <ServiceSecurity> tag.

2.9.9 Service Interface Syntax

```
<ServiceInterface InterfaceId=id>
     <!--Example:  InterfaceId="SERVER01">
     <OrgName Partyname=name/>
          <!--The party that acts as a server-->
     <Client>
          <OrgName Partyname=party/>
               <!--Party that acts as a client-->
          </Client>
     <TaskName>name</TaskName>  <!--optional-->
     <ActionMenu>   <!--see "Action Menu"-->
          <Action>  <!--one for each action-->
               ...
               </Action>
          </ActionMenu>
     <ServerServiceTime>  <!--see "Server Service Time"-->
          ...
          </ServerServiceTime>
     <StartEnabled>
          <RequestName>action_name</RequestName>
               <!--one for each action permitted as the initial
                   action-->
          </StartEnabled>
     <TerminateConversation>  <!--optional>
          <RequestName>action_name</RequestName>
               <!--one for each action defined as ending a
                    conversation-->
          </TerminateConversation>
     <ServiceSecurity>option</ServiceSecurity>
          <!-- option is yes or no-->
     </ServiceInterface>
```

2.9.10 Action Menu

The following subsections discuss the elements of the action menu. These discussions are followed by the complete syntax of the action definition.

2.9.11 Action Definition

This section gives a general description of the action. Subsequent sections provide the details of the action definition.

An action is a unit of work defined in the TPA. It consists of a request for service and associated information to be discussed below. An action may be associated with multiple message flows. The maximum expected response time must be specified for each action (see "Timing of Action Results"). Some actions can be revoked by corresponding cancel actions defined in the TPA. Note that the implementation of each cancel function does not necessarily imply complete undo or rollback; rather the intended use is to compensate for the effects of the associated, previously performed action.

Each action is invoked via a request message. During the long execution period, the service provider can send one or more informative messages, followed by a final reply message. A completed action can be canceled if allowed by the specification. An action description also specifies handling of failures.

> NOTE: In reality, an action defined in the TPA is a placeholder for what really goes on at the service provider. The TPA states an action name or equivalent but at the TPA level, it is not known what the action really does. For example, cancel might cancel the reservation, do nothing, or even make a new reservation. Of course there must be some agreement between the parties as to the nature of each action but the details are hidden in the local implementation.

There is a separate set of action definitions within the service interface definition for each party.

2.9.12 Attributes of the Action Tag

The `<Action>` tag has the following attributes. These are discussed in the following subsections.

`ActionId` - The value is a string that is unique for each action in the TPA.

`Type` - Permissible values are:

```
"basic"
"concurrent"
```

The default value is `"basic"`

`Invocation` - Permissible values are:

```
"syncOnly"
```

"asyncOnly"
"either"

The default is "asyncOnly".

2.9.13 Action ID

The ActionId attribute is required. Its value is an alphanumeric string which is unique among all actions defined in the TPA. For example, ActionId="action03".

2.9.14 Action Type

The Type attribute identifies the action as either Type="basic" (performs the function specified for it) or one of several kinds of extended actions in which the framework supplies function in addition to the basic action. The extended action is Type="concurrent".

> NOTE: Additional action types will be defined once the request patterns for a range of applications are understood. One possibility is an action type in which the service provider does not commit the results of a successful action until the client confirms.

If the Type attribute is omitted, the default is "basic", which means that no extended service is defined for this action.

Following are the services that may be applied through extended actions:

2.9.14.1 Action Type Concurrent

Type="concurrent" means that this action may be invoked concurrently with other actions in the same conversation, provided that it is enabled by the sequencing rules.

> NOTE: This attribute concerns only actions in the same conversation. In addition, the framework may support multiple concurrent conversations.

2.9.15 Invocation Attribute

The result of an action can be received by the client in an asynchronous or synchronous fashion. The distinction is primarily whether the client blocks until the results are received (synchronous) or does not block and receives the results at a later time. A given action may be specified as always synchronous, always asynchronous, or either synchronous or asynchronous. "Either synchronous or asynchronous" means that the client may choose either synchronous or asynchronous for a given invocation of the action. The synchronous/asynchronous properties are specified as part of the definition of each action and must be agreed to between the parties at the time the TPA is written.

The synchronous/asynchronous properties of an action as seen by the client are specified using the Invocation attribute of the <Action> tag. The following values can be specified:

Invocation="syncOnly" - The action can only be invoked as synchronous.

`Invocation="asyncOnly"` - The action can only be invoked as asynchronous.
`Invocation="either"` - The action can be invoked as either synchronous or asynchronous.

If the Invocation attribute is not provided, the default is `"asyncOnly"`.

2.9.16 Request

The <`Request`> tag within the action definition defines the name of an action, the input information and the result information.

2.9.16.1 Request Name

`<RequestName>` defines the name of the action. Each action under a given service interface must have a name which is unique within that service interface. The name consists of a case-sensitive string.

NOTE: An implementation may place a restriction on the length of the string.

2.9.16.2 Input Information to Request

The `<RequestMessage>` tag defines the input information for the request. Examples are `EDI840` (EDI message 840) and `OBIPOR` (OBI Purchase Order Request).

The value of `<RequestMessage>` is an alphanumeric string that references information that defines the format of the message. An example of information that might be referenced is a DTD or XML Schema document. The value may be either a keyword or a URL whose destination is accessible to both the server and the client.

NOTE: It is essential that both parties agree on the format of the message and record this agreement in a way that ensures that both will be using the same information. One way to ensure this is to use the URL of a DTD or other schema description as the value of `<RequestMessage>`. This ensures that both parties are using exactly the same document format description. The same thing can be done for document formats other than XML as long as there is a document that defines the format and can be referred to by `<RequestMessage>.`

NOTE: One possible implementation is that the value of the tag is a keyword that is an entry in a dictionary local to each party. The dictionary entry contains a local file path or URL from which the format definition or the name of a suitable document generator or parser (for a response message) is obtained. The keyword is resolved to the file path or URL when the TPA is registered at each party's site. If the document format is defined by local information at each party, the two parties must reach agreement on the document format and ensure that both maintain the same local information.

NOTE: If the tag value is a reference to a non-local repository, it is suggested that the item be obtained from the repository at TPA registration time or conversation startup time and be stored locally for subsequent use. Whether it is obtained at registration or conversation startup depends on how frequently it is expected to change.

2.9.17 Response to Request

The response information discussed here is returned for both the final result of an action and any intermediate responses. The final result and any intermediate response information are transferred by means of response messages or query requests. It should be understood that, depending on the function performed by an action, it may or may not return any result information.

The response to a request is normally by means of one or more asynchronous reply messages defined by <Response>. If <Response> is not provided, the client uses an application-defined query action to obtain the results. See "Query".

An action that results in multiple result sets must be asynchronous. For an action that results in multiple result sets, a separate <Response> tag must be provided for each expected response. The tags must be in the order in which the responses will be received. The intermediate result sets do not affect the sequencing rules. The final result set is transmitted when the action is complete and causes the sequencing rules to take effect.

It is not required that a client or server support asynchronous reply messages. The means used to convey results is a matter of negotiation when the contract is written.

See "Timing of Action Results" for a discussion of specifying the service time for each of these responses. See "Error Handling" for more information.

2.9.17.1 Response

<Response> is specified when the results information is in the form of an asynchronous reply message. <Response> consists of <ResponseName>, <ResponseMessage>, and <ResponseServiceTime>.

The value of <ResponseName> is the name of the function (e.g. a method name) at the client that is to process the response message. The value of <ResponseName> is an alphanumeric string.

NOTE: An implementation might place a restriction on the length of the string.

<ResponseMessage> identifies the specific message, such as EDI997. The value of <ResponseMessage> is an alphanumeric string that references information that defines the format of the message. An example of information that might be referenced is a DTD. The rules for using this tag are similar to those for <RequestMessage>. See also "Input Information to Request".

<ResponseServiceTime> is discussed in "Timing of Action Results" and "Presumed Result Following Timeout".

NOTE: Some transport protocols, such as SMTP, do not provide reliable delivery. For such protocols, it is recommended that when no response to a request is received within

the defined timeout (<ServerServiceTime> or <ActionServiceTime>, the request be retried at least once before declaring a TPA violation. The retry should use the same identifiers as the original request so that a duplicate message will be recognized if the server had received the original request.

The syntax of <Response> is:

```
<Response>
     <ResponseName>name</ResponseName>
          <!--example:  rsrvRsp-->
     <ResponseMessage>message</ResponseMessage>
     <ResponseServiceTime>
          <ServiceTime>time>/ServiceTime>
          <Presume>result</Presume>
               <!--result is success or fail-->
          </ResponseServiceTime>
     </Response>
```

2.9.17.2 Query

If the client does not support asynchronous reply messages, the client must issue a query at a later time to obtain the results. If the action produces intermediate results, each query returns one result set and the client must keep issuing queries until it obtains all the results. Successive queries return the result sets in the same order in which they were produced.

The query is an application-defined action that the client application invokes on the server on which it invoked the original action. The query request may be dependent on the specifics of the action request that it supports.

A query request to be used to obtain action results is synchronous.

NOTE: If the client could accept asynchronous responses, it would not need to use query.

NOTE: The message header for the query request must include the ID of the past request for which results are being requested. The application specifies this ID when invoking the query action.

NOTE: The response to a query may be lost because of, for example, a network error. The framework should provide a retry capability. For example, for a retry, software could provide the sequence number of the desired result set.

NOTE: To enable retry, result sets should be retained by the application at the server until the end of the conversation. One possible implementation for a query is that the application at the server obtains the result sets from the conversation log.

Following is an example of a query action definition:

```
<Action ActionId="action01" Invocation="syncOnly">
     <Request>
          <RequestName>GetResults</RequestName>
          <RequestMessage>QUERYMSG</RequestMessage>
          </Request>
     <Response>
          <ResponseMessage>message</ResponseMessage>
          </Response>
     </Action>
```

2.9.17.3 Application Result Information

The details of the application-level result information are determined and interpreted by the application. These details are included in the payload of the response message. The framework-level information specifies only success or failure. In addition, a field in the message header, which is processed by the framework, indicates success or failure.

2.9.18 Timing of Action Results

For an action invocation, the time to receive results may be considerably longer than the time to acknowledge the request since the time to the results may include processing at the server, actions invoked by the server on subcontractors, and similar functions, in addition to the network delays. The time to perform a given action may be quite long in some cases, possibly on the order of hours or longer.

The `<ServerServiceTime>` tag and the `<ResponseServiceTime>` tag (in the action definition) define the maximum service time until the server completes processing the action request. The `<ServiceTime>` tag gives the maximum service time. It is a character string containing an integer number of seconds.

It must be understood that the server service time (`<ServerServiceTime>` and `<ResponseServiceTime>`) and the network delay (`<NetworkDelay>`) are separately stated in the TPA and must be added together to determine the timeout value to be used in waiting for the response. If the recovery from the timeout fails, the timeout is logged as a permanent error condition.

The `<ResponseServiceTime>` tag contains the value of the service time for a particular action. It is optional in the action definition. If omitted, the value of `<ServerServiceTime>` for the same service interface is used. If stated in the action definition, it overrides the `<ServerServiceTime>` value. If neither tag is present, the client must assume a value outside the scope of the TPA.

For the first or only result set, the specified service time is the time since the request message. For an action that produces multiple result sets, the specified service time for each result set after the first is the time since the previous result set.

> NOTE: It should be understood that recoverable errors, such as short-duration network failures, are detected by the framework on time scales much shorter than the TPA-defined

timeout for the results. An example is failure to establish a TCP/IP connection to transmit the action-invocation message. Since failure to receive an acknowledgment to a message is usually caused by a network error or failure, the lack of an acknowledgment should not be considered a TPA violation. The message should be retried. In general, the framework should retry these error conditions. Once an acknowledgment is received, a results timeout indicates a serious problem at the server that prevented it from completing the requested action, or a TPA violation by the server.

2.9.19 Presumed Result Following Timeout

If a timeout occurs, it may be possible to presume that the action nonetheless succeeded. Whether or not this is possible depends on the nature of the action and the kind of information that would be returned in the reply. If success can be presumed, then some other rule must define what happens if the presumption is wrong.

One example of an action in which success might be presumed is an action which merely confirms the result of a prior action. If the result of the confirmation request is not received, it can be retried at the application level without logging an error.

The <`Presume`> tag in the under `<ResponseServiceTime>` indicates whether the action is presumed to have succeeded or failed after a timeout. The value of `<Presume>` is `success` or `fail`. If the `<Presume>` tag is provided under `<ServerServiceTime>`, it states the presumed result for all actions under that service interface unless overridden by the `<Presume>` tag in a specific action definition. If the <`Presume`> tag is not provided, the presumption is `fail`.

It should be understood that if the action invocation reaches the point where the timeout condition is relevant, it can be assumed that the server received and acknowledged the invocation message and has assumed responsibility for completed the requested action. See "Timing of Action Results".

2.9.20 Error Handling

Errors of concern to the TPA are grouped in the following categories:

- Connectivity errors
- Response errors
- Sequencing errors
- Business-specific errors

Connectivity errors are detected as failure to receive the acknowledgment to the transmission of a message, such as the message that sends an action request to a server.

Response errors are detected as failure to receive the results of an action. This subject is discussed in "Timing of Action Results".

Sequencing errors are violations of the order of actions specified in the sequencing rules. See "Sequencing Rules". A sequencing error could be committed by a client (action invoker) or

could result from a state error in the server. Therefore a sequencing error may require arbitration by the arbitrator specified in <TPAInfo>. See "Arbitration of TPA Disputes".

> NOTE: Sequencing errors can be handled by the framework and error-handling rules do not appear in the TPA. As part of processing sequencing errors, the framework can obtain the recent history of the conversation and send it to the arbitrator.

Business-specific errors are included in payload of the message indicated by the <ResponseMessage> tag.

> NOTE: Other error conditions that may be indicated in the return code are system-related and can be trapped by the framework and handled by it. These conditions are not shown in the TPA and are defined in the application-programming documentation.

2.9.21 Sequencing

See "Sequencing Rules".

2.9.22 Action Security

See "Message Security" for a discussion of the <ActionSecurity> tag.

2.9.23 Complete Action Definition

Following is the syntax of the action definition.

```
<Action ActionId=identifier
          Type=type
          Invocation=invoc>
          <!--see "Attributes of the Action Tag" for discussion
               of purpose of each attribute and permissible
               values to right of equal signs-->
          <!--attribute value must be enclosed in quotes-->
     <Request>
          <RequestName>name_of_action</RequestName>
               <!--example:  rsrvAir-->
          <RequestMessage>message</RequestMessage>
          </Request>
     <Response> <!--There can be more than one response tag-->
          <ResponseName>name</ResponseName>
               <!--example: rsrvResp-->
          <ResponseMessage>message</ResponseMessage>
          <ResponseServiceTime>
               <ServiceTime>time_value</ServiceTime>
               <Presume>result</Presume>
                    <!--result is success or fail-->
               </ResponseServiceTime>
          </Response>
     <Sequencing>   <!--optional-->
          <Enable>   <!--actions permitted after this one-->
```

```
            <RequestName>name_of_action</RequestName>
                 ...
            </Enable>
        <Disable>  <!--actions not permitted after this one-->
            <RequestName>name_of_action</RequestName>
            <RequestName>name_of_action</RequestName>
                 ...
            </Disable>
        </Sequencing>
    <ActionSecurity>option</ActionSecurity>
        <!-- option is yes or no-->
    </Action>
```

2.10 Comment Text

In general, a TPA between businesses can be expected to be accompanied by a traditional legal contract document. In addition, it may be desirable to include within the TPA textual statements explaining various points or referring to the traditional legal contract. The <Comment> tag is used for this purpose. The value of <Comment> is a text string that can contain alphanumeric characters and punctuation but no tags or angle brackets (XML restriction). <Comment> is optional. Multiple <Comment> tags are permitted to provide a very simple formatting capability.

Examples of such text are shown below.

```
<Comment>
     If it can be proved via the following third parties, that we
     are not reachable via the network, we will honor late
     cancellation.
     </Comment>
<Comment>
     If Hotelco is not reachable for more than three consecutive
     days, Hotelco will offer the customer free two nights stay.
     </Comment>
```

Another use of <Comment> might be to identify a traditional paper contract that accompanies the TPA.

> NOTE: Information that is copied from other sources, e.g. an accompanying legal contract, generally should not be included in <Comment>. Such information could change during the lifetime of the electronic TPA and the changes would not necessarily be reflected back into the original TPA.

Appendix 1 Complete TPA Definition

Following is the full XML TPA definition. Note that this is really a TPA template since it does not contain specific quantities for most of the tag values.

```
<?xml version="1.0"?>
<!DOCTYPE TPA SYSTEM "TPA.dtd">
<!--The DOCTYPE line above is for use only with a DTD parser.
     It must be removed for use with an XML Schema parser.-->
<TPA xmlns="tpa.xsd">
<!--The xmlns attribute is for use with an XML Schema parser.
     The definition in the DTD causes it to be ignored by
     a DTD parser.-->
   <TPAInfo>  <!-- TPA preamble -->
     <TPAName>TPA name</TPAName>
          <!--name of TPA-->
     <TPAType>
          <Protocol>protocol</Protocol>
          <Version>version</Version>
          <Type>type</Type>
          </TPAType>
     <Role>
          <!--defines the role parameters representing parties to
               the TPA-->
          <RoleDefn>     <!--one or more-->
               <RoleName>role_name</RoleName>
                    <!--example:  hotel, airline -->
               <RolePlayer>player_name</RolePlayer>
                    <!--example:  Hotelco, Airlineco-->
               </RoleDefn>
          </Role>
     <Participants>
          <Member MemberId=identifer IdCodeType=code>
               <PartyName Partyname=name>name
                    </PartyName>
                    <!--name of organization-->
               <CompanyTelephone>
                    telephone_number</CompanyTelephone>
               <Address>
                    <AddressType>type</AddressType>
                         <!--AddressType is shipping, billing, or
                              location-->
                    <AddressLine>address_line1</AddressLine>
                    <AddressLine>address_line2</AddressLine>
                         <!--as many address lines as needed-->
                    <City>city_name</City>
                    <State>state_name</State>
                         <!--or province or similar division-->
                    <Zip>zipcode</Zip>
                         <!--or similar postal code-->
                    <Country>country_name</Country>
                    </Address>
```

```
               <Contact Type=type>  <!--one or more-->
                    <!--Type is "primary" or "secondary",
                         default is "primary"-->
                    <LastName>name</LastName>
                    <FirstName>name</FirstName>
                    <MiddleName>name</MiddleName>
                    <Title>title</Title>
                    <ContactTelephone Type=type> <!--1 or more-->
                         phone_no</ContactTelephone>
                         <!--Type values same as for <Contact>-->
                    <Email Type=type>email_address</EMail>
                         <!--1 or more-->
                         <!--Type values same as for <Contact>-->
                    <Fax>fax_number</Fax>   <!--optional-->
                    </Contact>
               </Member>
               <!--Additional Member definitions-->
          <Arbitrator MemberId=identifer IdCodeType=code >
               <!--The contents of this tag are defined same
                    as for <Member> -->
               </Arbitrator>
          </Participants
     <Duration>     <!--valid duration for TPA-->
          <Start>  <!--Optional-->
               <Date>date</Date>
               <Time>time</Time>
               </Start>
          <End>     <!--Optional-->
               <Date>date</Date>
               <Time>time</Time>
               </End>
          </Duration>
     <InvocationLimit>number</InvocationLimit>
          <!--number of instantiations permitted before
               renegotiation is required-->
     <ConcurrentConversations>number</ConcurrentConversations>
          <!--Maximum number of concurrent conversations
               permitted-->
     <ConversationLife>time</ConversationLife>
          <!--maximum lifetime of a single conversation-->
     </TPAInfo>
  <Transport>
  <!--Following are separate Communication tag sets for the
          different protocols.  Only one of these can be used
          in a given TPA.-->
  <!------------------------------------------------------>
   <Communication>
     <HTTP>
          <Version>version</Version>
          <HTTPNode> <!--One for each party-->
               <OrgName Partyname=name/>
               <HTTPAddress>
                    <URL URLName=type>url</URL>
                    <!--additional URL tags as needed>
```

```
                    </HTTPAddress>
               </HTTPNode>
          <NetworkDelay>time</NetworkDelay>  <!--Optional-->
          </HTTP>
     </Communication>
  <!------------------------------------------------------>
   <Communication>
      <MQSeries>
         <MQSNode> <!--One for each party-->
              <OrgName Partyname=name/>
              <MQSAddress>
                   <QMgrName>mgr_name</QMgrName>
                        <!-- Queue manager name -->
                        <!-- case sensitive-->
                   <QMgrAddress>address</QMgrAddress>
                        <!-- Queue manager TCP/IP address -->
                        <!--IP address or domain name-->
                   <QMgrPort>port</QMgrPort> <!--optional-->
                        <!-- TCP port number-->
                   <Channel>channel_name</Channel>
                        <!-- This party is a SENDER  -->
                        <!-- for this channel -->
                        <!-- case sensitive-->
                   <RequestQ>my_request_q</RequestQ>
                        <!-- Where this party expects to  -->
                        <!-- receive requests   -->
                        <!--case sensitive-->
                   <ReplyQ>my_reply_q</ReplyQ>
                        <!-- Where this party expects to  -->
                        <!-- receive replies  -->
                        <!-- case sensitive -->
                   </MQSAddress>
              <ChannelInfo>
                   <BatchSize>size</BatchSize>
                   <DiscInt>interval</DiscInt>
                   <MaxMessageLength>length</MaxMessageLength>
                   <SeqWrap>seq_no</SeqWrap>
                   </ChannelInfo>
              </MQSNode>
          <NetworkDelay>time</NetworkDelay>  <!--Optional-->
          </MQSeries>
     </Communication>
<!------------------------------------------------------>
 <Communication>
     <SMTP>
       <SMTPNode>
          <!--to be determined-->
          </SMTPNode>
       <NetworkDelay>time</NetworkDelay> <!--optional-->
       </SMTP>
     </Communication>
<!------------------------------------------------------>
 <Communication>
     <VANEDI>
```

```
       <Version>version</Version>
       <VANEDINode>
          <!--One for each party-->
          <OrgName Partyname=name/>
          <VANEDIAddress>
               <InBox>name</InBox> <!--mailbox name-->
               <OutBox>name</OutBox> <!--mailbox name-->
               </VANEDIAddress>
          </VANEDINode>
        <NetworkDelay>time</NetworkDelay>  <!--optional-->
       </VANEDI>
     </Communication>
<!-------------------------------------------------------->
   <TransportSecurity>
     <Encryption>  <!--Optional-->
          <Protocol>protocol</Protocol>
          <Version>version</Version>
               <!--string denoting version of protocol-->
          <Certificate>
               <CertType>type</CertType>
                         <!--Example:  X.509V1, X.509V2-->
                    <KeyLength>length_in_bits</KeyLength>
                    <Party>
                         <!--one for each party that can act
                              as a server-->
                         <OrgName Partyname=name/>
                              <!--name of party-->
                         <IssuerOrgName>name</IssuerOrgName>
                              <!--Example:  VeriSign, Inc.-->
                         <IssuerCertSource>url</IssuerCertSource>
                              <!--URL of certificate-->
                         </Party>
               </Certificate>
          </Encryption>
     <Authentication>
          <PasswordAuthen>
               <Protocol>protocol</Protocol>
                    <!--Example: SSL-->
               <Version>version</Version> <!--optional-->
                    <!--version of protocol-->
               <LogonInfo>  <!--optional-->
                    <LogonParty>
                         <!--one for each party that can act as a
                            server-->
                         <OrgName Partyname=name/>
                         <UserId>value</UserId>
                         <Password>value</Password>
                         </LogonParty>
                    </LogonInfo>
               </PasswordAuthen>
          <CertificateAuthen>  <!--optional-->
               <Protocol>protocol</Protocol>
               <Version>version</Version>
               <Certificate>
```

```
                    <CertType>type</CertType>
                         <!--Example:  X.509V1, X.509V2-->
                    <KeyLength>length_in_bits</KeyLength>
                    <Party>   <!--one for each party>
                         <OrgName Partyname=name/>
                              <!--name of party-->
                         <IssuerOrgName>name</IssuerOrgName>
                              <!--Example: VeriSign, Inc.-->
                         <IssuerCertSource>url</IssuerCertSource>
                              <!--URL of certificate-->
                         </Party>
                    </Certificate>
               </CertificateAuthen-->
          </Authentication>
       </TransportSecurity>
     </Transport>
<DocExchange>
     <DocExchangeProtocol>protocol</DocExchangeProtocol>
     <MessageEncoding>encoding</MessageEncoding>
          <!--currently must be BASE64 or omitted-->
     <MessageIdempotency>choice</MessageIdempotency>
          <!--choice is yes or no-->
     <MessageSecurity>
          <NonRepudiation>
               <Protocol>protocol</Protocol>
                    <!--example: DigitalSignature-->
               <Version>version</Version>
                    <!--version of protocol-->
               <HashFunction>function</HashFunction>
                    <!--example:  MD5, SHA1-->
               <EncryptionAlgorithm>algorithm
                    </EncryptionAlgorithm>
                    <!--Example:   RC2 -->
               <SignatureAlgorithm>algorithm</SignatureAlgorithm>
                    <!--Example:  DSA-->
               <Certificate>
                    <CertType>type</CertType>
                         <!--Example:  X.509V1, X.509V2-->
                    <KeyLength>length_in_bits</KeyLength>
                    <Party>   <!--one for each party>
                         <OrgName Partyname=name/>
                              <!--name of party-->
                         <IssuerOrgName>name</IssuerOrgName>
                              <!--Example:  VeriSign, Inc.-->
                         <IssuerCertSource>url</IssuerCertSource>
                              <!--URL of  certificate-->
                         </Party>
                    </Certificate>
               </NonRepudiation>
          <DigitalEnvelope>
               <Protocol>DigitalEnvelope</Protocol>
               <Version>version</Version>
               <EncryptionAlgorithm>algor</EncryptionAlgorithm>
                    <!--example:  RC4-->
```

```
            <Certificate>
                 <CertType>type</CertType>
                      <!--Example:  X.509V3-->
                 <KeyLength>length_in_bits</KeyLength>
                      <!--example:  512-->
                 <Party>   <!--one for each party>
                      <OrgName Partyname=name/>
                           <!--name of party-->
                      <OrgCertSource>url</OrgCertSource>
                           <!--URL of certificate-->
                      <IssuerOrgName>name</IssuerOrgName>
                           <!--Example:  VeriSign, Inc.-->
                      <IssuerCertSource>url</IssuerCertSource>
                           <!--URL of  certificate-->
                      </Party>
                 </Certificate>
            </DigitalEnvelope>
       </MessageSecurity>
  </DocExchange>
<BusinessProtocol>
 <ServiceInterface InterfaceId=id>
  <!-- one for each service provider-->
  <!--Example:  InterfaceId="SERVER01">
  <OrgName Partyname=name/>
       <!--the party which acts as a server-->
  <Client>       <!--one or more-->
       <OrgName Partyname=name/>
            <!--The party which acts as a client-->
       </Client>
  <TaskName>name</TaskName>  <!--optional-->
  <ActionMenu>
   <!--one Action tag for each action-->
   <Action
       ActionId=identifier
       Type=type
       Invocation=invoc>
            <!--attribute value must be enclosed in quotes-->
       <Request>
            <RequestName>name_of_action</RequestName>
                 <!--example:  rsrvAir-->
            <RequestMessage>message</RequestMessage>
            </Request>
       <Response>
            <!--There can be more than one Response tag.-->
            <ResponseName>name</ResponseName>
                 <!--example: rsrvResp-->
            <ResponseMessage>message</ResponseMessage>
            <ResponseServiceTime>
                 <ServiceTime>time_value</ServiceTime>
                 <Presume>result</Presume>
                      <!--result is success or fail-->
                 </ResponseServiceTime>
            </Response>
       <Sequencing>   <!--optional-->
```

```
            <Enable>   <!--actions permitted after this one-->
                 <RequestName>name_of_action</RequestName>
                      ...
                 </Enable>
            <Disable>
                 <!--actions not permitted after this one-->
                 <RequestName>name_of_action</RequestName>
                 <RequestName>name_of_action</RequestName>
                      ...
                 </Disable>
            </Sequencing>
       <ActionSecurity>option</ActionSecurity>
            <!-- option is yes or no>
       </Action>
      </ActionMenu>
  <ServerServiceTime>
       <ServiceTime>time</ServiceTime>
       <Presume>result</Presume>
            <!--result is success or fail-->
       </ServerServiceTime>
  <StartEnabled>
       <RequestName>action_name</RequestName>
            <!--one for each action permitted as the initial
            action-->
       </StartEnabled>
  <TerminateConversation>  <!--optional>
       <RequestName>action_name</RequestName>
            <!--one for each action defined as ending a
                 conversation-->
       </TerminateConversation>
  <ServiceSecurity>option</ServiceSecurity>
       <!-- option is yes or no-->
  </ServiceInterface>
 </BusinessProtocol>
<Comment>   <!--One or more-->
  <!--Explanatory text-->
  </Comment>
</TPA>
```

Appendix 2 DTD Corresponding to Complete TPA Definition

Following is the DTD that defines the contents of the TPA.

```
<!--*************************************************************************-->
<!--                TPA DTD                                                  -->
<!-- File name: TPA.dtd                                                      -->
<!--*************************************************************************-->
<!--*************************************************************************-->
<!ELEMENT TPA (TPAInfo, Transport,
  DocExchange, BusinessProtocol, Comment*)>
<!ATTLIST TPA xmlns CDATA #IMPLIED>
<!--The xmlns attribute identifies the XML Schema document when a TPA is  -->
<!--parsed by an XML Schema parser. It is ignored by a DTD parser.        -->
<!--*************************************************************************-->
<!-- General information                                                    -->
<!--*************************************************************************-->
<!ELEMENT TPAInfo (TPAName, TPAType?, Role?, Participants,
  Duration, InvocationLimit?, ConcurrentConversations?, ConversationLife?)>
<!ELEMENT TPAName (#PCDATA)>
<!ELEMENT TPAType (Protocol, Version, Type)>
<!ELEMENT Protocol (#PCDATA)>
<!ELEMENT Version (#PCDATA)>
<!ELEMENT Type (#PCDATA)>
<!--*************************************************************************-->
<!-- Specification of Roles and Participants                                -->
<!--*************************************************************************-->
<!ELEMENT Role (RoleDefn+)>
<!ELEMENT RoleDefn (RoleName, RolePlayer)>
<!ELEMENT RoleName (#PCDATA)>
<!ELEMENT RolePlayer (#PCDATA)>
<!ELEMENT Participants (Member+, Arbitrator?)>
<!ELEMENT Member (PartyName, CompanyTelephone, Address+, Contact+)>
<!ATTLIST Member
  MemberId CDATA #REQUIRED
  IdCodeType CDATA #REQUIRED >
<!ELEMENT PartyName (#PCDATA)>
<!ATTLIST PartyName Partyname ID #REQUIRED>
<!ELEMENT CompanyTelephone (#PCDATA)>
<!ELEMENT Address (AddressType, AddressLine+, City, State, Zip, Country)>
<!ELEMENT AddressType (#PCDATA)>
<!ELEMENT AddressLine (#PCDATA)>
<!ELEMENT City (#PCDATA)>
<!ELEMENT State (#PCDATA)>
<!ELEMENT Zip (#PCDATA)>
<!ELEMENT Country (#PCDATA)>
<!ELEMENT Contact (LastName, FirstName, MiddleName, Title, ContactTelephone+,
 EMail+, Fax?)>
<!ATTLIST Contact
  Type (primary|secondary) "primary" >
<!ELEMENT LastName (#PCDATA)>
<!ELEMENT FirstName (#PCDATA)>
<!ELEMENT MiddleName (#PCDATA)>
<!ELEMENT Title (#PCDATA)>
<!ELEMENT ContactTelephone (#PCDATA)>
<!ATTLIST ContactTelephone
  Type (primary|secondary) "primary" >
<!ELEMENT EMail (#PCDATA)>
<!ATTLIST EMail
  Type (primary|secondary) "primary" >
<!ELEMENT Fax (#PCDATA)>
```

```
<!ELEMENT Arbitrator (PartyName, CompanyTelephone, Address+, Contact+)>
<!ATTLIST Arbitrator
  MemberId CDATA #REQUIRED
  IdCodeType CDATA #REQUIRED >
<!ELEMENT Duration  (Start?, End?)>
<!ELEMENT Start (Date, Time)>
<!ELEMENT End (Date, Time)>
<!ELEMENT Date (#PCDATA)>
<!ELEMENT Time (#PCDATA)>
<!ELEMENT InvocationLimit (#PCDATA)>
<!ELEMENT ConcurrentConversations (#PCDATA)>
<!ELEMENT ConversationLife (#PCDATA)>
<!--************************************************************************-->
<!-- Specification of Transport Protocol                                    -->
<!--************************************************************************-->
<!ELEMENT Transport (Communication, TransportEncoding?, TransportSecurity?)>
<!ELEMENT Communication (HTTP|MQSeries|SMTP|VANEDI)>
<!ELEMENT HTTP (Version?, HTTPNode+, NetworkDelay?)>
<!ELEMENT MQSeries (MQSNode+, ChannelInfo, NetworkDelay?)>
<!ELEMENT SMTP (Version?, SMTPNode+, NetworkDelay?)>
<!ELEMENT VANEDI (Version?, VANEDINode+, NetworkDelay?)>
<!ELEMENT HTTPNode (OrgName, HTTPAddress)>
<!ELEMENT MQSNode (OrgName, MQSAddress)>
<!ELEMENT SMTPNode (OrgName, SMTPAddress)>
<!ELEMENT VANEDINode (OrgName, VANEDIAddress)>
<!ELEMENT OrgName EMPTY>
<!ATTLIST OrgName Partyname IDREF #REQUIRED>
<!ELEMENT HTTPAddress (URL+)>
<!ELEMENT URL (#PCDATA)>
<!ATTLIST URL
  URLName (logOnURL|requestURL|responseURL) "requestURL" >
<!ELEMENT MQSAddress (QMgrName, QMgrAddress, QMgrPort?, Channel, RequestQ,
  ReplyQ)>
<!ELEMENT QMgrName (#PCDATA)>
<!ELEMENT QMgrAddress (#PCDATA)>
<!ELEMENT QMgrPort (#PCDATA)>
<!ELEMENT Channel (#PCDATA)>
<!ELEMENT RequestQ (#PCDATA)>
<!ELEMENT ReplyQ (#PCDATA)>
<!ELEMENT VANEDIAddress (InBox, OutBox)>
<!ELEMENT InBox (#PCDATA)>
<!ELEMENT OutBox (#PCDATA)>
<!ELEMENT SMTPAddress (#PCDATA)>
<!ELEMENT ChannelInfo (BatchSize, DiscInt, MaxMessageLength, SeqWrap)>
<!ELEMENT BatchSize (#PCDATA)>
<!ELEMENT DiscInt (#PCDATA)>
<!ELEMENT MaxMessageLength (#PCDATA)>
<!ELEMENT SeqWrap (#PCDATA)>
<!ELEMENT TransportEncoding (#PCDATA)>
<!ELEMENT NetworkDelay (#PCDATA)>
<!--************************************************************************-->
<!-- Specification of Transport Security Protocol                           -->
<!--************************************************************************-->
<!ELEMENT TransportSecurity (Encryption?, Authentication?)>
<!ELEMENT Encryption (Protocol, Version?, Certificate)>
<!ELEMENT Authentication (PasswordAuthen?, CertificateAuthen?)>
<!ELEMENT PasswordAuthen (Protocol, Version?, LogonInfo?)>
<!ELEMENT LogonInfo (LogonParty+)>
<!ELEMENT LogonParty (OrgName, UserId, Password)>
<!ELEMENT UserId (#PCDATA)>
<!ELEMENT Password (#PCDATA)>
<!ELEMENT CertificateAuthen (Protocol, Version?, Certificate)>
<!ELEMENT Certificate (CertType, KeyLength, Party+)>
<!ELEMENT CertType (#PCDATA)>
```

```
<!ELEMENT KeyLength (#PCDATA)>
<!ELEMENT Party (OrgName, OrgCertSource?, IssuerOrgName, IssuerCertSource)>
<!ELEMENT OrgCertSource (#PCDATA)>
<!ELEMENT IssuerOrgName (#PCDATA)>
<!ELEMENT IssuerCertSource (#PCDATA)>
<!--************************************************************************-->
<!-- Specification of DocExchange Protocol                                  -->
<!--************************************************************************-->
<!ELEMENT DocExchange (DocExchangeProtocol, MessageEncoding?,
   MessageIdempotency,
   MessageSecurity?)>
<!ELEMENT DocExchangeProtocol (#PCDATA)>
<!ELEMENT MessageEncoding (#PCDATA)>
<!ELEMENT MessageIdempotency (#PCDATA)>
<!--************************************************************************-->
<!-- Specification of Message Security Protocol                             -->
<!--************************************************************************-->
<!ELEMENT MessageSecurity (NonRepudiation?, DigitalEnvelope?)>
<!ELEMENT NonRepudiation (Protocol, Version?, HashFunction,
  EncryptionAlgorithm,
  SignatureAlgorithm, Certificate)>
<!ELEMENT HashFunction (#PCDATA)>
<!ELEMENT EncryptionAlgorithm (#PCDATA)>
<!ELEMENT SignatureAlgorithm (#PCDATA)>
<!ELEMENT DigitalEnvelope (Protocol?, Version?, EncryptionAlgorithm,
  Certificate?)>
<!--************************************************************************-->
<!-- Specification of Business Protocol                                     -->
<!--************************************************************************-->
<!ELEMENT BusinessProtocol (ServiceInterface+)>
<!ELEMENT ServiceInterface (OrgName, Client+, TaskName?, ActionMenu,
  ServerServiceTime?,
  StartEnabled?, TerminateConversation?, ServiceSecurity?)>
<!ATTLIST ServiceInterface
  InterfaceId CDATA #REQUIRED >
<!ELEMENT Client (OrgName)>
<!ELEMENT TaskName (#PCDATA)>
<!ELEMENT ActionMenu (Action+)>
<!ELEMENT Action (Request, Response*,
  Sequencing?, ActionSecurity?)>
<!ATTLIST Action
  ActionId CDATA #REQUIRED
  Type (basic|concurrent) "basic"
  Invocation (syncOnly | asyncOnly | either ) "asyncOnly" >
<!--************************************************************************-->
<!-- Each Request is identified by a request name.                          -->
<!--************************************************************************-->
<!ELEMENT Request (RequestName, RequestMessage)>
<!ELEMENT RequestName (#PCDATA)>
<!ELEMENT RequestMessage (#PCDATA)>
<!--************************************************************************-->
<!-- Response is an asynchronous reply from the sever following a prior   -->
<!-- request invocation. There can be zero, one or multiple responses for -->
<!-- a given request, depending on the application.                         -->
<!--************************************************************************-->
<!ELEMENT Response (ResponseName, ResponseMessage, ResponseServiceTime?)>
<!ELEMENT ResponseName (#PCDATA)>
<!ELEMENT ResponseMessage (#PCDATA)>
<!ELEMENT ResponseServiceTime (ServiceTime, Presume?)>
<!ELEMENT ServiceTime (#PCDATA)>
<!ELEMENT Presume (#PCDATA)>
<!--************************************************************************-->
<!-- Specify none or more sequencing rules to be applied to this action   -->
<!--************************************************************************-->
```

```
<!ELEMENT Sequencing (Enable?, Disable?)>
<!ELEMENT Enable (RequestName+)>
<!ELEMENT Disable (RequestName+)>
<!--*************************************************************-->
<!-- Specify ActionSecurity yes for this action to overide the message    -->
<!-- security default defined in ServiceSecurity.                         -->
<!-- No Need to specify this tag if default is no                         -->
<!--*************************************************************-->
<!ELEMENT ActionSecurity (#PCDATA)>
<!--*************************************************************-->
<!-- Service time-out if specified is the default time-out for actions    -->
<!-- that do not have time-out defined.                                   -->
<!--*************************************************************-->
<!ELEMENT ServerServiceTime (ServiceTime, Presume?)>
<!--*************************************************************-->
<!ELEMENT StartEnabled (RequestName+)>
<!ELEMENT TerminateConversation (RequestName+)>
<!--*************************************************************-->
<!-- Specify default message security for all actions in the server.      -->
<!-- Value is yes. No Need to specify this tag if message security not    -->
<!-- wanted since default is no.                                          -->
<!--*************************************************************-->
<!ELEMENT ServiceSecurity (#PCDATA)>
<!--*************************************************************-->
<!ELEMENT Comment (#PCDATA)>
```

Appendix 3 XML Schema Document Corresponding to Complete TPA Definition

Following is the XML Schema document that defines the contents of the TPA.

```
<?xml version="1.0"?>
<!DOCTYPE schema PUBLIC "-//W3C/DTD XML Schema Version 1.0//EN"
  "http:\\www.w3.org\TR/1999/WD-xmlschema-1-19991217/structures.dtd">
<schema targetNamespace="\authoring\tpa.xsd">
<type name="TPA" content="elementOnly">
    <group order="seq">
      <element name="TPAInfo"/>
      <element name="Transport"/>
      <element name="DocExchange"/>
      <element name="BusinessProtocol"/>
      <group maxOccurs="*" minOccurs="0">
        <element name="Comment"/>
      </group>
</group>
  </type>
<type name="TPAInfo" content="elementOnly">
    <group order="seq">
      <element name="TPAName"/>
      <group minOccurs="0" maxOccurs="1">
        <element name="TPAType"/>
      </group>
<group minOccurs="0" maxOccurs="1">
        <element name="Role"/>
      </group>
<element name="Participants"/>
      <element name="Duration"/>
      <group minOccurs="0" maxOccurs="1">
        <element name="InvocationLimit"/>
      </group>
<group minOccurs="0" maxOccurs="1">
        <element name="ConcurrentConversations"/>
      </group>
<group minOccurs="0" maxOccurs="1">
        <element name="ConversationLife"/>
      </group>
</group>
  </type>
<type name="TPAName" content="textOnly"/>
<type name="TPAType" content="elementOnly">
    <group order="seq">
      <element name="Protocol"/>
      <element name="Version"/>
      <element name="Type"/>
    </group>
  </type>
<type name="Protocol" content="textOnly"/>
<type name="Version" content="textOnly"/>
<type name="Type" content="textOnly"/>
<type name="Role" content="elementOnly">
    <group maxOccurs="*">
      <element name="RoleDefn"/>
    </group>
</type>
<type name="RoleDefn" content="elementOnly">
    <group order="seq">
```

```
      <element name="RoleName"/>
      <element name="RolePlayer"/>
    </group>
  </type>
<type name="RoleName" content="textOnly"/>
<type name="RolePlayer" content="textOnly"/>
<type name="Participants" content="elementOnly">
    <group order="seq">
      <group maxOccurs="*">
        <element name="Member"/>
      </group>
<group minOccurs="0" maxOccurs="1">
        <element name="Arbitrator"/>
      </group>
</group>
  </type>
<type name="Member" content="elementOnly">
    <group order="seq">
      <element name="PartyName"/>
      <element name="CompanyTelephone"/>
      <group maxOccurs="*">
        <element name="Address"/>
      </group>
<group maxOccurs="*">
        <element name="Contact"/>
      </group>
</group>
      <attribute name="IdCodeType" type="string" minOccurs="1"/>

      <attribute name="MemberId" type="string" minOccurs="1"/>
</type>
<type name="PartyName" content="textOnly">
      <attribute name="Partyname" type="ID" minOccurs="1"/>
</type>
<type name="CompanyTelephone" content="textOnly"/>
<type name="Address" content="elementOnly">
    <group order="seq">
      <element name="AddressType"/>
      <group maxOccurs="*">
        <element name="AddressLine"/>
      </group>
<element name="City"/>
      <element name="State"/>
      <element name="Zip"/>
      <element name="Country"/>
    </group>
  </type>
<type name="AddressType" content="textOnly"/>
<type name="AddressLine" content="textOnly"/>
<type name="City" content="textOnly"/>
<type name="State" content="textOnly"/>
<type name="Zip" content="textOnly"/>
<type name="Country" content="textOnly"/>
<type name="Contact" content="elementOnly">
    <group order="seq">
      <element name="LastName"/>
      <element name="FirstName"/>
      <element name="MiddleName"/>
      <element name="Title"/>
      <group maxOccurs="*">
        <element name="ContactTelephone"/>
      </group>
<group maxOccurs="*">
        <element name="EMail"/>
```

```
      </group>
<group minOccurs="0" maxOccurs="1">
        <element name="Fax"/>
      </group>
</group>
      <attribute name="Type" type="NMTOKEN" minOccurs="0" default="primary">
      <datatype source="string">
        <enumeration value="primary|secondary"/>
      </datatype>
    </attribute>
  </type>
<type name="LastName" content="textOnly"/>
<type name="FirstName" content="textOnly"/>
<type name="MiddleName" content="textOnly"/>
<type name="Title" content="textOnly"/>
<type name="ContactTelephone" content="textOnly">
      <attribute name="Type" type="NMTOKEN" minOccurs="0" default="primary">
      <datatype source="string">
        <enumeration value="primary|secondary"/>
      </datatype>
    </attribute>
  </type>
<type name="EMail" content="textOnly">
      <attribute name="Type" type="NMTOKEN" minOccurs="0" default="primary">
      <datatype source="string">
        <enumeration value="primary|secondary"/>
      </datatype>
    </attribute>
  </type>
<type name="Fax" content="textOnly"/>
<type name="Arbitrator" content="elementOnly">
    <group order="seq">
      <element name="PartyName"/>
      <element name="CompanyTelephone"/>
      <group maxOccurs="*">
        <element name="Address"/>
      </group>
<group maxOccurs="*">
        <element name="Contact"/>
      </group>
</group>
      <attribute name="IdCodeType" type="string" minOccurs="1"/>

      <attribute name="MemberId" type="string" minOccurs="1"/>
</type>
<type name="Duration" content="elementOnly">
    <group order="seq">
      <group minOccurs="0" maxOccurs="1">
        <element name="Start"/>
      </group>
<group minOccurs="0" maxOccurs="1">
        <element name="End"/>
      </group>
</group>
  </type>
<type name="Start" content="elementOnly">
    <group order="seq">
      <element name="Date"/>
      <element name="Time"/>
    </group>
  </type>
<type name="End" content="elementOnly">
    <group order="seq">
      <element name="Date"/>
```

```
      <element name="Time"/>
    </group>
  </type>
<type name="Date" content="textOnly"/>
<type name="Time" content="textOnly"/>
<type name="InvocationLimit" content="textOnly"/>
<type name="ConcurrentConversations" content="textOnly"/>
<type name="ConversationLife" content="textOnly"/>
<type name="Transport" content="elementOnly">
    <group order="seq">
      <element name="Communication"/>
      <group minOccurs="0" maxOccurs="1">
        <element name="TransportEncoding"/>
      </group>
<group minOccurs="0" maxOccurs="1">
        <element name="TransportSecurity"/>
      </group>
</group>
  </type>
<type name="Communication" content="elementOnly">
    <group order="choice">
      <element name="HTTP"/>
      <element name="MQSeries"/>
      <element name="SMTP"/>
      <element name="VANEDI"/>
    </group>
  </type>
<type name="HTTP" content="elementOnly">
    <group order="seq">
      <group minOccurs="0" maxOccurs="1">
        <element name="Version"/>
      </group>
<group maxOccurs="*">
        <element name="HTTPNode"/>
      </group>
<group minOccurs="0" maxOccurs="1">
        <element name="NetworkDelay"/>
      </group>
</group>
  </type>
<type name="MQSeries" content="elementOnly">
    <group order="seq">
      <group maxOccurs="*">
        <element name="MQSNode"/>
      </group>
<element name="ChannelInfo"/>
      <group minOccurs="0" maxOccurs="1">
        <element name="NetworkDelay"/>
      </group>
</group>
  </type>
<type name="SMTP" content="elementOnly">
    <group order="seq">
      <group minOccurs="0" maxOccurs="1">
        <element name="Version"/>
      </group>
<group maxOccurs="*">
        <element name="SMTPNode"/>
      </group>
<group minOccurs="0" maxOccurs="1">
        <element name="NetworkDelay"/>
      </group>
</group>
  </type>
```

```
<type name="VANEDI" content="elementOnly">
    <group order="seq">
      <group minOccurs="0" maxOccurs="1">
        <element name="Version"/>
      </group>
<group maxOccurs="*">
        <element name="VANEDINode"/>
      </group>
<group minOccurs="0" maxOccurs="1">
        <element name="NetworkDelay"/>
      </group>
</group>
  </type>
<type name="HTTPNode" content="elementOnly">
    <group order="seq">
      <element name="OrgName"/>
      <element name="HTTPAddress"/>
    </group>
  </type>
<type name="MQSNode" content="elementOnly">
    <group order="seq">
      <element name="OrgName"/>
      <element name="MQSAddress"/>
    </group>
  </type>
<type name="SMTPNode" content="elementOnly">
    <group order="seq">
      <element name="OrgName"/>
      <element name="SMTPAddress"/>
    </group>
  </type>
<type name="VANEDINode" content="elementOnly">
    <group order="seq">
      <element name="OrgName"/>
      <element name="VANEDIAddress"/>
    </group>
  </type>
<type name="OrgName" content="empty">
      <attribute name="Partyname" type="IDREF" minOccurs="1"/>
</type>
<type name="HTTPAddress" content="elementOnly">
    <group maxOccurs="*">
      <element name="URL"/>
    </group>
</type>
<type name="URL" content="textOnly">
      <attribute name="URLName" type="NMTOKEN" minOccurs="0"
default="requestURL">
      <datatype source="string">
        <enumeration value="logOnURL|requestURL|responseURL"/>
      </datatype>
    </attribute>
  </type>
<type name="MQSAddress" content="elementOnly">
    <group order="seq">
      <element name="QMgrName"/>
      <element name="QMgrAddress"/>
      <group minOccurs="0" maxOccurs="1">
        <element name="QMgrPort"/>
      </group>
<element name="Channel"/>
      <element name="RequestQ"/>
      <element name="ReplyQ"/>
    </group>
```

```
  </type>
<type name="QMgrName" content="textOnly"/>
<type name="QMgrAddress" content="textOnly"/>
<type name="QMgrPort" content="textOnly"/>
<type name="Channel" content="textOnly"/>
<type name="RequestQ" content="textOnly"/>
<type name="ReplyQ" content="textOnly"/>
<type name="VANEDIAddress" content="elementOnly">
    <group order="seq">
      <element name="InBox"/>
      <element name="OutBox"/>
    </group>
  </type>
<type name="InBox" content="textOnly"/>
<type name="OutBox" content="textOnly"/>
<type name="SMTPAddress" content="textOnly"/>
<type name="ChannelInfo" content="elementOnly">
    <group order="seq">
      <element name="BatchSize"/>
      <element name="DiscInt"/>
      <element name="MaxMessageLength"/>
      <element name="SeqWrap"/>
    </group>
  </type>
<type name="BatchSize" content="textOnly"/>
<type name="DiscInt" content="textOnly"/>
<type name="MaxMessageLength" content="textOnly"/>
<type name="SeqWrap" content="textOnly"/>
<type name="TransportEncoding" content="textOnly"/>
<type name="NetworkDelay" content="textOnly"/>
<type name="TransportSecurity" content="elementOnly">
    <group order="seq">
      <group minOccurs="0" maxOccurs="1">
        <element name="Encryption"/>
      </group>
<group minOccurs="0" maxOccurs="1">
        <element name="Authentication"/>
      </group>
</group>
  </type>
<type name="Encryption" content="elementOnly">
    <group order="seq">
      <element name="Protocol"/>
      <group minOccurs="0" maxOccurs="1">
        <element name="Version"/>
      </group>
<element name="Certificate"/>
    </group>
  </type>
<type name="Authentication" content="elementOnly">
    <group order="seq">
      <group minOccurs="0" maxOccurs="1">
        <element name="PasswordAuthen"/>
      </group>
<group minOccurs="0" maxOccurs="1">
        <element name="CertificateAuthen"/>
      </group>
</group>
  </type>
<type name="PasswordAuthen" content="elementOnly">
    <group order="seq">
      <element name="Protocol"/>
      <group minOccurs="0" maxOccurs="1">
        <element name="Version"/>
```

```
      </group>
<group minOccurs="0" maxOccurs="1">
        <element name="LogonInfo"/>
      </group>
</group>
  </type>
<type name="LogonInfo" content="elementOnly">
    <group maxOccurs="*">
      <element name="LogonParty"/>
    </group>
</type>
<type name="LogonParty" content="elementOnly">
    <group order="seq">
      <element name="OrgName"/>
      <element name="UserId"/>
      <element name="Password"/>
    </group>
  </type>
<type name="UserId" content="textOnly"/>
<type name="Password" content="textOnly"/>
<type name="CertificateAuthen" content="elementOnly">
    <group order="seq">
      <element name="Protocol"/>
      <group minOccurs="0" maxOccurs="1">
        <element name="Version"/>
      </group>
<element name="Certificate"/>
    </group>
  </type>
<type name="Certificate" content="elementOnly">
    <group order="seq">
      <element name="CertType"/>
      <element name="KeyLength"/>
      <group maxOccurs="*">
        <element name="Party"/>
      </group>
</group>
  </type>
<type name="CertType" content="textOnly"/>
<type name="KeyLength" content="textOnly"/>
<type name="Party" content="elementOnly">
    <group order="seq">
      <element name="OrgName"/>
      <group minOccurs="0" maxOccurs="1">
        <element name="OrgCertSource"/>
      </group>
<element name="IssuerOrgName"/>
      <element name="IssuerCertSource"/>
    </group>
  </type>
<type name="OrgCertSource" content="textOnly"/>
<type name="IssuerOrgName" content="textOnly"/>
<type name="IssuerCertSource" content="textOnly"/>
<type name="DocExchange" content="elementOnly">
    <group order="seq">
      <element name="DocExchangeProtocol"/>
      <group minOccurs="0" maxOccurs="1">
        <element name="MessageEncoding"/>
      </group>
<element name="MessageIdempotency"/>
      <group minOccurs="0" maxOccurs="1">
        <element name="MessageSecurity"/>
      </group>
</group>
```

```
  </type>
<type name="DocExchangeProtocol" content="textOnly"/>
<type name="MessageEncoding" content="textOnly"/>
<type name="MessageIdempotency" content="textOnly"/>
<type name="MessageSecurity" content="elementOnly">
    <group order="seq">
      <group minOccurs="0" maxOccurs="1">
        <element name="NonRepudiation"/>
      </group>
<group minOccurs="0" maxOccurs="1">
        <element name="DigitalEnvelope"/>
      </group>
</group>
  </type>
<type name="NonRepudiation" content="elementOnly">
    <group order="seq">
      <element name="Protocol"/>
      <group minOccurs="0" maxOccurs="1">
        <element name="Version"/>
      </group>
<element name="HashFunction"/>
      <element name="EncryptionAlgorithm"/>
      <element name="SignatureAlgorithm"/>
      <element name="Certificate"/>
    </group>
  </type>
<type name="HashFunction" content="textOnly"/>
<type name="EncryptionAlgorithm" content="textOnly"/>
<type name="SignatureAlgorithm" content="textOnly"/>
<type name="DigitalEnvelope" content="elementOnly">
    <group order="seq">
      <group minOccurs="0" maxOccurs="1">
        <element name="Protocol"/>
      </group>
<group minOccurs="0" maxOccurs="1">
        <element name="Version"/>
      </group>
<element name="EncryptionAlgorithm"/>
      <group minOccurs="0" maxOccurs="1">
        <element name="Certificate"/>
      </group>
</group>
  </type>
<type name="BusinessProtocol" content="elementOnly">
    <group maxOccurs="*">
      <element name="ServiceInterface"/>
    </group>
</type>
<type name="ServiceInterface" content="elementOnly">
    <group order="seq">
      <element name="OrgName"/>
      <group maxOccurs="*">
        <element name="Client"/>
      </group>
<group minOccurs="0" maxOccurs="1">
        <element name="TaskName"/>
      </group>
<element name="ActionMenu"/>
      <group minOccurs="0" maxOccurs="1">
        <element name="ServerServiceTime"/>
      </group>
<group minOccurs="0" maxOccurs="1">
        <element name="StartEnabled"/>
      </group>
```

```
<group minOccurs="0" maxOccurs="1">
        <element name="TerminateConversation"/>
      </group>
<group minOccurs="0" maxOccurs="1">
        <element name="ServiceSecurity"/>
      </group>
</group>
      <attribute name="InterfaceId" type="string" minOccurs="1"/>
</type>
<type name="Client" content="elementOnly">
    <element name="OrgName"/>
  </type>
<type name="TaskName" content="textOnly"/>
<type name="ActionMenu" content="elementOnly">
    <group maxOccurs="*">
      <element name="Action"/>
    </group>
</type>
<type name="Action" content="elementOnly">
    <group order="seq">
      <element name="Request"/>
      <group maxOccurs="*" minOccurs="0">
        <element name="Response"/>
      </group>
<group minOccurs="0" maxOccurs="1">
        <element name="Sequencing"/>
      </group>
<group minOccurs="0" maxOccurs="1">
        <element name="ActionSecurity"/>
      </group>
</group>
      <attribute name="Type" type="NMTOKEN" minOccurs="0" default="basic">
      <datatype source="string">
        <enumeration value="basic|concurrent"/>
      </datatype>
    </attribute>
      <attribute name="ActionId" type="string" minOccurs="1"/>

      <attribute name="Invocation" type="NMTOKEN" minOccurs="0"
default="asyncOnly">
      <datatype source="string">
        <enumeration value="syncOnly|asyncOnly|either"/>
      </datatype>
    </attribute>
  </type>
<type name="Request" content="elementOnly">
    <group order="seq">
      <element name="RequestName"/>
      <element name="RequestMessage"/>
    </group>
  </type>
<type name="RequestName" content="textOnly"/>
<type name="RequestMessage" content="textOnly"/>
<type name="Response" content="elementOnly">
    <group order="seq">
      <element name="ResponseName"/>
      <element name="ResponseMessage"/>
      <group minOccurs="0" maxOccurs="1">
        <element name="ResponseServiceTime"/>
      </group>
</group>
  </type>
<type name="ResponseName" content="textOnly"/>
<type name="ResponseMessage" content="textOnly"/>
```

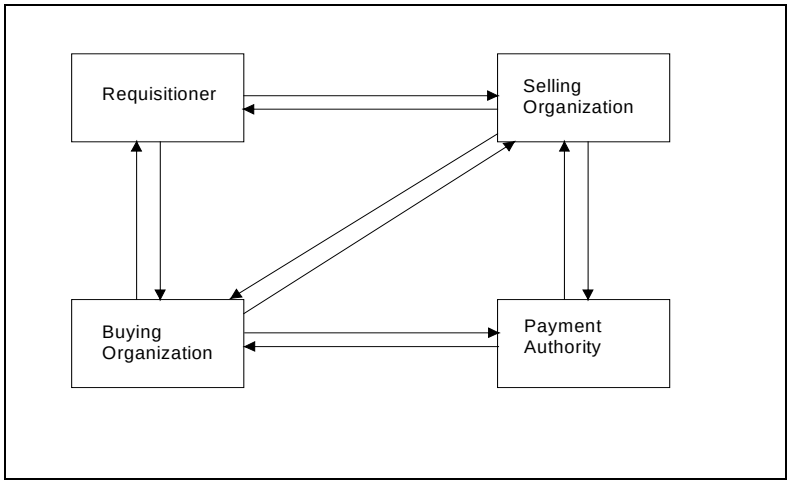
```
<type name="ResponseServiceTime" content="elementOnly">
    <group order="seq">
      <element name="ServiceTime"/>
      <group minOccurs="0" maxOccurs="1">
        <element name="Presume"/>
      </group>
</group>
  </type>
<type name="ServiceTime" content="textOnly"/>
<type name="Presume" content="textOnly"/>
<type name="Sequencing" content="elementOnly">
    <group order="seq">
      <group minOccurs="0" maxOccurs="1">
        <element name="Enable"/>
      </group>
<group minOccurs="0" maxOccurs="1">
        <element name="Disable"/>
      </group>
</group>
  </type>
<type name="Enable" content="elementOnly">
    <group maxOccurs="*">
      <element name="RequestName"/>
    </group>
</type>
<type name="Disable" content="elementOnly">
    <group maxOccurs="*">
      <element name="RequestName"/>
    </group>
</type>
<type name="ActionSecurity" content="textOnly"/>
<type name="ServerServiceTime" content="elementOnly">
    <group order="seq">
      <element name="ServiceTime"/>
      <group minOccurs="0" maxOccurs="1">
        <element name="Presume"/>
      </group>
</group>
  </type>
<type name="StartEnabled" content="elementOnly">
    <group maxOccurs="*">
      <element name="RequestName"/>
    </group>
</type>
<type name="TerminateConversation" content="elementOnly">
    <group maxOccurs="*">
      <element name="RequestName"/>
    </group>
</type>
<type name="ServiceSecurity" content="textOnly"/>
<type name="Comment" content="textOnly"/>
</schema>
```

Appendix 4 Example: OBI

In this section, we describe an OBI scenario. A sample TPA for this example is in Appendix "4.1 TPA for Open Buying on the Internet (OBI)".

Open Buying on the Internet (OBI) is a protocol for business-to-business Internet commerce. It was designed by the Internet Purchasing Roundtable and is supported by the OBI consortium (http://www.openbuy.org). OBI defines the procedures for the high-volume, low-dollar purchasing transactions that make up most of an organization's purchasing activity.

The following figure illustrates the participants in an OBI transaction and the basic information flows.



The requisitioner is a member of the buying organization (e.g. an employee of a company) and is permitted to place orders directly with the selling organization's merchant server. The requisitioner can browse a catalog and place an order with the selling organization using a browser. The catalog may be tailored to the requisitioner's organization. When the requisitioner has placed an order, the selling organization's server sends a partial purchase order (purchase order request) to the buying organization's server. The buying organization validates the purchase order request and transforms it into a complete purchase order that it returns to the selling organization. The selling organization then prepares an invoice or otherwise arranges for payment and ships the ordered merchandise. The payment authority is an optional part of the system. Its purpose is to handle electronic payments. Using the browser, the requisitioner can also view and update various information at the buying organization server such as the requisitioner's profile, outstanding requests, etc. The requisitioner can also check the status of an order at the selling organization.

An additional possibility is that the buying organization can send an "unsolicited" purchase order to the selling organization without a prior request and partial purchase order initiated by a requisitioner.

There is a TPA between the buying organization and the selling organization, each of which has a B2B server. The payment authority, if present, is outside the scope of the 2-party TPA between the buying organization and the selling organization. It may interact with the buying organization and the selling organization in a variety of ways. The interaction may be through separate 2-party TPAs between the payment authority and the buyer and seller organizations. It may also be simply through application programs without involving the B2B server.

Appendix 4.1 TPA for Open Buying on the Internet (OBI)

Following is the TPA for the OBI example.

NOTE: To use this example with an XML Schema parser, remove the DOCTYPE line.

```
<?xml version="1.0"?>
<!DOCTYPE TPA SYSTEM "TPA.dtd" >
<!--*************************************************************************-->
<!--              OBI TPA      between Large Co (buying company)             -->
<!--                     and Pens Are We (selling company)                    -->
<!--*************************************************************************-->
<TPA xmlns="tpa.xsd">
<!--*************************************************************************-->
<!-- The xmlns attribute is for use with an XML Schema parser.  Its          -->
<!-- definition in the DTD causes it to be ignored by a DTD parser.          -->
<!--*************************************************************************-->
<!-- General information                                                     -->
<!--*************************************************************************-->
  <TPAInfo>
    <TPAName>OBIStandard</TPAName>
  <TPAType>
    <Protocol>OBI</Protocol>
    <Version>1.0</Version>
    <Type>SS</Type>
   </TPAType>
<!--*************************************************************************-->
    <Participants>
<!--*************************************************************************-->
<!-- Specification of Buyer                                                  -->
<!--*************************************************************************-->
      <Member IdCodeType="ZZ" MemberId="777777777777777">
        <PartyName Partyname="_LargeCo">Large Co</PartyName>
        <CompanyTelephone>914-945-3000</CompanyTelephone>
        <Address>
          <AddressType>location</AddressType>
          <AddressLine>Large Co</AddressLine>
          <AddressLine>HQ Building</AddressLine>
          <AddressLine>1 Main Street</AddressLine>
          <City>SmallTown</City>
          <State>NY</State>
          <Zip>10000</Zip>
          <Country>USA</Country>
        </Address>
        <Address>
          <AddressType>billing</AddressType>
          <AddressLine>Large Co</AddressLine>
          <AddressLine>Accounting Department</AddressLine>
          <AddressLine>100 Bean Counters Road</AddressLine>
          <City>Any City</City>
          <State>CT</State>
          <Zip>06000</Zip>
```

```
          <Country>USA</Country>
        </Address>
        <Address>
          <AddressType>shipping</AddressType>
          <AddressLine>Large Co</AddressLine>
          <AddressLine>Procurement Department</AddressLine>
          <AddressLine>99 Purchase Road</AddressLine>
          <City>Buy City</City>
          <State>NY</State>
          <Zip>10001</Zip>
          <Country>USA</Country>
        </Address>
        <Contact Type = "primary">
          <LastName>Smith</LastName>
          <FirstName>John</FirstName>
          <MiddleName>L.</MiddleName>
          <Title>Senior Buyer</Title>
          <ContactTelephone Type = "primary">914-111-6789
            </ContactTelephone>
          <ContactTelephone Type = "secondary">914-111-6790
            </ContactTelephone>
          <EMail Type = "primary">jjsmith@largeco.com</EMail>
          <EMail Type = "secondary">
            http://www.largeco.com/procurement/jsmith.html
            </EMail>
          <Fax>914-111-6780</Fax>
        </Contact>
        <Contact Type = "secondary">
          <LastName>Blow</LastName>
          <FirstName>Joe</FirstName>
          <MiddleName>J.</MiddleName>
          <Title>Buyer</Title>
          <ContactTelephone Type = "primary">914-111-6722
            </ContactTelephone>
          <ContactTelephone Type = "secondary">914-111-6725
            </ContactTelephone>
          <EMail Type = "primary">jblow@largeco.com</EMail>
          <Fax>914-111-6780</Fax>
        </Contact>
      </Member>
<!--*********************************************************************-->
<!-- Specification of Seller                                             -->
<!--*********************************************************************-->
      <Member IdCodeType="ZZ" MemberId="888000009000000">
        <PartyName Partyname="_PensAreWe">Pens Are We
            </PartyName>
        <CompanyTelephone>945-123-1000</CompanyTelephone>
        <Address>
          <AddressType>location</AddressType>
          <AddressLine>Pens Are We</AddressLine>
          <AddressLine>Building 001</AddressLine>
          <AddressLine>123 High Street</AddressLine>
          <City>EarthQuake City</City>
          <State>CA</State>
          <Zip>94567</Zip>
          <Country>USA</Country>
        </Address>
        <Contact Type = "primary">
          <LastName>Doe</LastName>
          <FirstName>Jane</FirstName>
          <MiddleName>E.</MiddleName>
          <Title>Vice President of Internet Sales</Title>
          <ContactTelephone Type = "primary">945-123-4567
            </ContactTelephone>
```

```
          <ContactTelephone Type = "secondary">945-123-4570
            </ContactTelephone>
          <EMail Type = "primary">janedoe@pensarewe.com</EMail>
          <EMail Type = "secondary">
            http://www.pensarewe.com/sales/jdoe.html
            </EMail>
          <Fax>945-123-9999</Fax>
        </Contact>
      </Member>
<!--*************************************************************************-->
<!-- Specification of Arbitrator                                             -->
<!--*************************************************************************-->
      <Arbitrator IdCodeType="01" MemberId="888000009000001">
        <PartyName Partyname="_XYZArbitrator">XYZArbitrator</PartyName>
        <CompanyTelephone>780-333-1111</CompanyTelephone>
        <Address>
          <AddressType>location</AddressType>
          <AddressLine>XYZArbitrator</AddressLine>
          <AddressLine>Suite 3</AddressLine>
          <AddressLine>77 Lawyers Blvd</AddressLine>
          <City>ABC City</City>
          <State>MA</State>
          <Zip>01234</Zip>
          <Country>USA</Country>
        </Address>
        <Contact Type = "primary">
          <LastName>Black</LastName>
          <FirstName>Joe</FirstName>
          <MiddleName>K.</MiddleName>
          <Title>Mr.</Title>
          <ContactTelephone Type = "primary">780-333-4040
            </ContactTelephone>
          <ContactTelephone Type = "secondary">780-333-4045
            </ContactTelephone>
          <EMail Type = "primary">jblack@xyzarbitrator.com</EMail>
          <EMail Type = "secondary">
            http://www.xyzarbitrator.com/jblack.html</EMail>
          <Fax>780-333-5000</Fax>
        </Contact>
      </Arbitrator>
    </Participants>
    <Duration>
      <Start>
        <Date>01/01/1999</Date>
        <Time>00:00:00</Time>
      </Start>
      <End>
        <Date>01/01/2001</Date>
        <Time>00:00:00</Time>
      </End>
    </Duration>
    <InvocationLimit>100000</InvocationLimit>
    <ConcurrentConversations>1</ConcurrentConversations>
    <ConversationLife>86400</ConversationLife>
  </TPAInfo>
<!--*************************************************************************-->
<!-- Specification of Transport Protocol #01                                 -->
<!--*************************************************************************-->
  <Transport>
   <Communication>
    <HTTP>
     <HTTPNode>
      <OrgName Partyname="_LargeCo"/>
      <HTTPAddress>
```

```
        <URL URLName="requestURL">
            https://www.largeco.com/jackal/servlet/OBIBuy</URL>
       </HTTPAddress>
      </HTTPNode>
      <HTTPNode>
       <OrgName Partyname="_PensAreWe"/>
       <HTTPAddress>
         <URL URLName="logOnURL">
             https://www.pensarewe.com/coyote/servlet/OBILogon</URL>
         <URL URLName="requestURL">
             https://www.pensarewe.com/coyote/servlet/OBIsell</URL>
         <URL URLName="responseURL">
             https://www.pensarewe.com/coyote/servlet/OBIsell</URL>
       </HTTPAddress>
      </HTTPNode>
     <NetworkDelay>300</NetworkDelay>
     </HTTP>
    </Communication>
<!--*************************************************************************-->
<!-- Specification of Transport Security Protocol                            -->
<!--*************************************************************************-->
    <TransportSecurity>
       <Encryption>
         <Protocol>SSL</Protocol>
         <Version>3.0</Version>
         <Certificate>
           <CertType>X509.V3</CertType>
           <KeyLength>1024</KeyLength>
           <Party>
             <OrgName Partyname="_LargeCo"/>
             <IssuerOrgName>VeriSign, Inc.</IssuerOrgName>
             <IssuerCertSource>http://www.verisign.com/certs
                   </IssuerCertSource>
           </Party>
           <Party>
             <OrgName Partyname="_PensAreWe"/>
             <IssuerOrgName>GTE, Inc.</IssuerOrgName>
             <IssuerCertSource>http://www.gte.com/certs
                   </IssuerCertSource>
           </Party>
         </Certificate>
       </Encryption>
       <Authentication>
         <CertificateAuthen>
         <Protocol>SSL</Protocol>
         <Version>3.0</Version>
         <Certificate>
           <CertType>X509.V3</CertType>
           <KeyLength>1024</KeyLength>
           <Party>
             <OrgName Partyname="_LargeCo"/>
             <IssuerOrgName>VeriSign, Inc.</IssuerOrgName>
             <IssuerCertSource>http://www.verisign.com/certs
                   </IssuerCertSource>
           </Party>
           <Party>
             <OrgName Partyname="_PensAreWe"/>
             <IssuerOrgName>GTE, Inc.</IssuerOrgName>
             <IssuerCertSource>http://www.gte.com/certs
                   </IssuerCertSource>
           </Party>
         </Certificate>
        </CertificateAuthen>
       </Authentication>
```

```
     </TransportSecurity>
    </Transport>
<!--*******************************************************************-->
<!-- Specification of DocExchange Protocol                             -->
<!--*******************************************************************-->
  <DocExchange>
    <DocExchangeProtocol>OBI</DocExchangeProtocol>
    <MessageEncoding>BASE64</MessageEncoding>
    <MessageIdempotency>yes</MessageIdempotency>
<!--*******************************************************************-->
<!-- Specification of Message Security                                 -->
<!--*******************************************************************-->
    <MessageSecurity>
      <NonRepudiation>
        <Protocol>DigitalSignature</Protocol>
        <HashFunction>MD5</HashFunction>
        <EncryptionAlgorithm>RSA</EncryptionAlgorithm>
        <SignatureAlgorithm>DSA</SignatureAlgorithm>
        <Certificate>
          <CertType>X509.V3</CertType>
          <KeyLength>1024</KeyLength>
          <Party>
            <OrgName Partyname="_LargeCo"/>
            <IssuerOrgName>Verisign Inc.</IssuerOrgName>
            <IssuerCertSource>http://www.verisign.com/certs
                  </IssuerCertSource>
          </Party>
          <Party>
            <OrgName Partyname="_PensAreWe"/>
            <IssuerOrgName>GTE Inc.</IssuerOrgName>
            <IssuerCertSource>http://www.gte.com/certs</IssuerCertSource>
          </Party>
        </Certificate>
      </NonRepudiation>
    </MessageSecurity>
  </DocExchange>
  <BusinessProtocol>
<!--*******************************************************************-->
<!-- Specification of Service Interface 01                             -->
<!--*******************************************************************-->
   <ServiceInterface InterfaceId="interface01">
    <OrgName Partyname="_LargeCo"/>
    <Client>
      <OrgName Partyname="_PensAreWe"/>
    </Client>
    <ActionMenu>
      <Action ActionId="action01" Type="basic">
        <Request>
          <RequestName>putOPOR</RequestName>
          <RequestMessage>OBIPOR</RequestMessage>
        </Request>
        <Response>
          <ResponseName>getOPO</ResponseName>
          <ResponseMessage>OBIPO</ResponseMessage>
          <ResponseServiceTime>
            <ServiceTime>3600</ServiceTime>
            <Presume>fail</Presume>
          </ResponseServiceTime>
        </Response>
      </Action>
    </ActionMenu>
    <ServerServiceTime>
        <ServiceTime>3660</ServiceTime>
        <Presume>fail</Presume>
```

```
    </ServerServiceTime>
    <StartEnabled>
      <RequestName>putOPOR</RequestName>
    </StartEnabled>
   </ServiceInterface>
<!--*********************************************************************-->
<!-- Specification of Service Interface 02                               -->
<!-- This interface below is for UnSolicited OBIPO from                  -->
<!-- buying organization to selling organization                         -->
<!--*********************************************************************-->
  <ServiceInterface InterfaceId="interface02">
    <OrgName Partyname="_PensAreWe"/>
    <Client>
      <OrgName Partyname="_LargeCo"/>
    </Client>
    <ActionMenu>
      <Action ActionId="action03" Type="basic">
        <Request>
          <RequestName>shop</RequestName>
           <!--Initiates shopping at merchant server-->
          <RequestMessage>shopMessage</RequestMessage>
        </Request>
      </Action>
      <Action ActionId="action02" Type="basic">
        <Request>
          <RequestName>putOPO</RequestName>
          <RequestMessage>OBIPO</RequestMessage>
        </Request>
      </Action>
    </ActionMenu>
    <ServerServiceTime>
        <ServiceTime>3660</ServiceTime>
        <Presume>fail</Presume>
    </ServerServiceTime>
   </ServiceInterface>
  </BusinessProtocol>
</TPA>
```

Appendix 5 Examples of Reusable and Fully Qualified TPAs

Following is an example of a reusable TPA using roles. First is part of the OBI TPA example written with roles. This is followed by the same example in which the roles have been evaluated into specific parties and the party-specific information filled in making a fully qualified TPA.

NOTE: In this section, we use the term TPA since that is the term generally associated with the OBI standard.

Appendix 5.1 Reusable TPA

In the reusable TPA, role names appear, preceded by @, wherever party names would appear in a completed TPA. The values of the associated party-specific tags are blank.

```
<?xml version="1.0"?>
<!DOCTYPE TPA SYSTEM "TPA.dtd" >
<!--*************************************************************************-->
<!--              OBI TPA between Large Co (buying company)                  -->
<!--                    and Pens Are We (selling company)                    -->
<!--*************************************************************************-->
<TPA xmlns="tpa.xsd">
<!--*************************************************************************-->
<!-- General TPA information                                                 -->
<!--*************************************************************************-->
  <TPAInfo>
    <TPAName>OBIStandard</TPAName>
    <TPAType>
      <Protocol>OBI</Protocol>
      <Version>1.0</Version>
      <Type>SS</Type>
    </TPAType>
<!--*************************************************************************-->
<!--Role parameters                                                          -->
<!--*************************************************************************-->
    <Role>
      <RoleDefn>
        <RoleName>OBIBuyer</RoleName>
        <RolePlayer> </RolePlayer>
       </RoleDefn>
      <RoleDefn>
        <RoleName>OBISeller</RoleName>
        <RolePlayer> </RolePlayer>
       </RoleDefn)
      </Role>
    <Participants>
<!--*************************************************************************-->
<!-- Specification of Buyer                                                  -->
<!--*************************************************************************-->
      <Member IdCodeType=" " MemberId=" ">
        <PartyName Partyname="_obibuyer">@OBIBuyer</PartyName>
        <CompanyTelephone> </CompanyTelephone>
        <Address>
          <AddressType>location</AddressType>
          <AddressLine> </AddressLine>
          <AddressLine> </AddressLine>
          <AddressLine> </AddressLine>
          <City> </City>
          <State> </State>
```

```
        <Zip> </Zip>
        <Country </Country>
      </Address>
      <Address>
        <AddressType>billing</AddressType>
        <AddressLine> </AddressLine>
        <AddressLine> </AddressLine>
        <AddressLine> </AddressLine>
        <City> </City>
        <State> </State>
        <Zip> </Zip>
        <Country> </Country>
      </Address>
      <Address>
        <AddressType>shipping</AddressType>
        <AddressLine> </AddressLine>
        <AddressLine> </AddressLine>
        <AddressLine> </AddressLine>
        <City> </City>
        <State> </State>
        <Zip> </Zip>
        <Country> </Country>
      </Address>
      <Contact Type = "primary">
        <LastName> </LastName>
        <FirstName> < MiddleName>
        <Title> </Title>
        <ContactTelephone Type = "primary"> </ContactTelephone>
        <ContactTelephone Type = "secondary"> </ContactTelephone>
        <EMail Type = "primary"> </EMail>
        <EMail Type = "secondary"> </EMail>
        <Fax> </Fax>
      </Contact>
      <Contact Type = "secondary">
        <LastName> </LastName>
        <FirstName> </FirstName>
        <MiddleName> </MiddleName>
        <Title> </Title>
        <ContactTelephone Type = "primary"> </ContactTelephone>
        <ContactTelephone Type = "secondary"> </ContactTelephone>
        <EMail Type = "primary"> </EMail>
        <Fax> </Fax>
      </Contact>
    </Member>
<!--*****************************************************************-->
<!-- Specification of Seller                                         -->
<!--*****************************************************************-->
    <Member IdCodeType=" " MemberId=" ">
      <PartyName Partyname="_obiseller">@OBISeller/PartyName>
      <CompanyTelephone> </CompanyTelephone>
      <Address>
        <AddressType> </AddressType>
        <AddressLine> </AddressLine>
        <AddressLine> </AddressLine>
        <AddressLine> </AddressLine>
        <City> </City>
        <State> </State>
        <Zip> </Zip>
        <Country> </Country>
      </Address>
      <Contact Type = "primary">
        <LastName> </LastName>
        <FirstName> </FirstName>
        <MiddleName> </MiddleName>
```

```
          <Title> </Title>
          <ContactTelephone Type = "primary"> </ContactTelephone>
          <ContactTelephone Type = "secondary"> </ContactTelephone>
          <EMail Type = "primary"> </EMail>
          <EMail Type = "secondary"> </EMail>
          <Fax> </Fax>
        </Contact>
      </Member>
        </Participants>
  </TPAInfo>

...

<!--************************************************************************-->
<!-- Specification of Transport Protocol                                    -->
<!--************************************************************************-->
  <Transport>
   <Communication>
    <HTTP>
     <HTTPNode>
      <OrgName Partyname="_obibuyer"/>
      <HTTPAddress>
        <URL URLName="requestURL"> </URL>
      </HTTPAddress>
...

     </HTTPNode>
     <HTTPNode>
      <OrgName Partyname="_obiseller"/>
      <HTTPAddress>
        <URL URLName="logOnURL"> </URL>
        <URL URLName="requestURL"> </URL>
        <URL URLName="responseURL"> </URL>
      </HTTPAddress>
...
     </HTTPNode>
    <NetworkDelay>300</NetworkDelay>
    </HTTP>
   </Communication>
<!--************************************************************************-->
<!-- Specification of Transport Security Protocol
-->
<!--************************************************************************-->
   <TransportSecurity>
      <Encryption>
       ...
        <Certificate>
...

          <Party>
            <OrgName Partyname="_obibuyer"/>
            <IssuerOrgName> </IssuerOrgName>
          </Party>
          <Party>
            <OrgName Partyname="_obiseller"/>
            <IssuerOrgName> </IssuerOrgName>
          </Party>
        </Certificate>
...
      <Authentication>
...
        <Certificate>
...
          <Party>
```

```
            <OrgName Partyname="_obibuyer"/>
            <IssuerOrgName> </IssuerOrgName>
          </Party>
          <Party>
            <OrgName Partyname="_obiseller"/>
            <IssuerOrgName> </IssuerOrgName>
          </Party>
        </Certificate>
       </CertificateAuthen>
      </Authentication>
    </TransportSecurity>
   </Transport>
<!--*************************************************************************-->
<!-- Specification of DocExchange                                            -->
<!--*************************************************************************-->

   <DocExchange>
...
<!--*************************************************************************-->
<!-- Specification of Message Security                                       -->
<!--*************************************************************************-->

    <MessageSecurity>
      <NonRepudiation>

...

        <Certificate>
          <Party>

...

            <OrgName Partyname="_obibuyer"/>
            <IssuerOrgName> </IssuerOrgName>
          </Party>
          <Party>
            <OrgName Partyname="_obiseller"/>
            <IssuerOrgName> </IssuerOrgName>
          </Party>
        </Certificate>
      </NonRepudiation>
    </MessageSecurity>
  </DocExchange>>

...

  <BusinessProtocol>
<!--*************************************************************************-->
<!-- Specification of Service Interface 01                                   -->
<!--*************************************************************************-->
  <ServiceInterface InterfaceId="interface01">
    <OrgName Partyname="_obibuyer"/>
    <Client>
      <OrgName Partyname="_obiseller"/>
    </Client>
    <ActionMenu>
      <Action ActionId="action01">

...

        <ResponseServiceTime>
          <ServiceTime> </ServiceTime>

...
```

```
        <ResponseServiceTime>
      </Action>
    </ActionMenu>
    <ServerServiceTime>
      <ServiceTime> </ServiceTime>

...

    </ServerServiceTime>

...

  </ServiceInterface>
<!--*********************************************************************-->
<!-- Specification of Service Interface 02                               -->
<!-- This interface below is for UnSolicited OBIPO from                  -->
<!-- buying organization to selling organization                         -->
<!--*********************************************************************-->
  <ServiceInterface InterfaceId="interface02">
    <OrgName Partyname="_obiseller"/>
    <Client>
      <PartyName Partyname="_obibuyer"/>
    </Client>
    <ActionMenu>
      <Action ActionId="action02">
...

     </Action>
    </ActionMenu>
    <ServerServiceTime>
      <ServiceTime> </ServiceTime>

...

    </ServerServiceTime>
...

  </ServiceInterface>
  </BusinessProtocol>
</TPA>
```

Appendix 5.2 Fully Qualified TPA

Following is a fully qualified TPA, the result of replacing the role names in the reusable TPA in the previous section with specific party names and filling in the party-specific information in various tags. Note that the <Role> tag and its subelements have been left in the fully qualified TPA for reference although they have no further function.

```
<?xml version="1.0"?>
<!DOCTYPE TPA SYSTEM "TPA.dtd" >
<!--*********************************************************************-->
<!--              OBI TPA between Large Co (buying company)              -->
<!--                      and Pens Are We (selling company)              -->
<!--*********************************************************************-->
<TPA xmlns="tpa.xsd">
<!--*********************************************************************-->
<!-- General TPA information                                             -->
<!--*********************************************************************-->
  <TPAInfo>
```

```
    <TPAName>OBIStandard</TPAName>
    <TPAType>
      <Protocol>OBI</Protocol>
      <Version>1.0</Version>?
      <Type>SS</Type>
    </TPAType>
<!--*************************************************************************-->
<!--Role parameters                                                          -->
<!--*************************************************************************-->
    <Role>
      <RoleDefn>
        <RoleName>OBIBuyer</RoleName>
        <RolePlayer>Large Co</RolePlayer>
       </RoleDefn>
      <RoleDefn>
        <RoleName>OBISeller</RoleName>
        <RolePlayer>Pens Are We</RolePlayer>
       </RoleDefn)
      </Role>
    <Participants>
<!--*************************************************************************-->
<!-- Specification of Buyer                                                  -->
<!--*************************************************************************-->
      <Member IdCodeType="ZZ" MemberId="77777777777777">
        <PartyName Partyname="_obibuyer">Large Co</PartyName>
        <CompanyTelephone>914-945-3000</CompanyTelephone>
        <Address>
          <AddressType>location</AddressType>
          <AddressLine>Large Co</AddressLine>
          <AddressLine>HQ Building</AddressLine>
          <AddressLine>1 Main Street</AddressLine>
          <City>SmallTown</City>
          <State>NY</State>
          <Zip>10000</Zip>
          <Country>USA</Country>
        </Address>
        <Address>
          <AddressType>billing</AddressType>
          <AddressLine>Large Co</AddressLine>
          <AddressLine>Accounting Department</AddressLine>
          <AddressLine>100 Bean Counters Road</AddressLine>
          <City>Any City</City>
          <State>CT</State>
          <Zip>06000</Zip>
          <Country>USA</Country>
        </Address>
        <Address>
          <AddressType>shipping</AddressType>
          <AddressLine>Large Co</AddressLine>
          <AddressLine>Procurement Department</AddressLine>
          <AddressLine>99 Purchase Road</AddressLine>
          <City>Buy City</City>
          <State>NY</State>
          <Zip>10001</Zip>
          <Country>USA</Country>
        </Address>
        <Contact Type = "primary">
          <LastName>Smith</LastName>
          <FirstName>John</FirstName>
          <MiddleName>L.</MiddleName>
          <Title>Senior Buyer</Title>
          <ContactTelephone Type = "primary">914-111-6789</ContactTelephone>
          <ContactTelephone Type = "secondary">914-111-6790</ContactTelephone>
          <EMail Type = "primary">jjsmith@largeco.com</EMail>
```

```
          <EMail Type = "secondary">
            http://www.largeco.com/procurement/jsmith.html
            </EMail>
          <Fax>914-111-6780</Fax>
        </Contact>
        <Contact Type = "secondary">
          <LastName>Blow</LastName>
          <FirstName>Joe</FirstName>
          <MiddleName>J.</MiddleName>
          <Title>Buyer</Title>
          <ContactTelephone Type = "primary">914-111-6722</ContactTelephone>
          <ContactTelephone Type = "secondary">914-111-6725</ContactTelephone>
          <EMail Type = "primary">jblow@largeco.com</EMail>
          <Fax>914-111-6780</Fax>
        </Contact>
      </Member>
<!--*********************************************************************-->
<!-- Specification of Seller                                             -->
<!--*********************************************************************-->
      <Member IdCodeType="ZZ" MemberId="888000009000000">
        <PartyName Partyname="_obiseller">Pens Are We</PartyName>
        <CompanyTelephone>945-123-1000</CompanyTelephone>
        <Address>
          <AddressType>location</AddressType>
          <AddressLine>Pens Are We</AddressLine>
          <AddressLine>Building 001</AddressLine>
          <AddressLine>123 High Street</AddressLine>
          <City>EarthQuake City</City>
          <State>CA</State>
          <Zip>94567</Zip>
          <Country>USA</Country>
        </Address>
        <Contact Type = "primary">
          <LastName>Doe</LastName>
          <FirstName>Jane</FirstName>
          <MiddleName>E.</MiddleName>
          <Title>Vice President of Internet Sales</Title>
          <ContactTelephone Type = "primary">945-123-4567</ContactTelephone>
          <ContactTelephone Type = "secondary">945-123-4570</ContactTelephone>
          <EMail Type = "primary">janedoe@pensarewe.com</EMail>
          <EMail Type = "secondary">
            http://www.pensarewe.com/sales/jdoe.html
            </EMail>
          <Fax>945-123-9999</Fax>
        </Contact>
      </Member>
        </Participants>
  </TPAInfo>

...

<!--*********************************************************************-->
<!-- Specification of Transport Protocol #01                             -->
<!--*********************************************************************-->
  <Transport>
   <Communication>
    <HTTP>
     <HTTPNode>
      <OrgName Partyname="_obibuyer"/>
      <HTTPAddress>
        <URL URLName="requestURL">
            https://www.largeco.com/jackal/servlet/OBIBuy</URL>
      </HTTPAddress>
...
```

```
     </HTTPNode>
     <HTTPNode>
      <OrgName Partyname="_obiseller"/>
      <HTTPAddress>
        <URL URLName="logOnURL">
            https://www.pensarewe.com/BPF/servlet/OBILogon</URL>
        <URL URLName="requestURL">
            https://www.pensarewe.com/BPF/servlet/OBIsell</URL>
        <URL URLName="responseURL">
            https://www.pensarewe.com/BPF/servlet/OBIsell</URL>
      </HTTPAddress>

...
    </HTTPNode>
    <NetworkDelay>300</NetworkDelay>
    </HTTP>
   </Communication>
<!--*************************************************************************-->
<!-- Specification of Transport Security Protocol                            -->
<!--*************************************************************************-->
   <TransportSecurity>
      <Encryption>
...
        <Certificate>
...

          <Party>
            <OrgName Partyname="_obibuyer"/>
            <IssuerOrgName>VeriSign, Inc.</IssuerOrgName>
          </Party>
          <Party>
            <OrgName Partyname="_obiseller"/>
            <IssuerOrgName>GTE, Inc.</IssuerOrgName>
          </Party>
         </Certificate>
        </Encryption>
       <Authentication>

...

          <Party>
            <OrgName Partyname="_obibuyer"/>
            <IssuerOrgName>VeriSign, Inc.</IssuerOrgName>
          </Party>
          <Party>
            <OrgName Partyname="_obiseller"/>
            <IssuerOrgName>GTE, Inc.</IssuerOrgName>
          </Party>
        </Certificate>
      </Authentication>
     </CertificateAuthen>
    </TransportSecurity>
   </Transport>
<!--*************************************************************************-->
<!-- Specification of DocExchange                                            -->
<!--*************************************************************************-->

   <DocExchange>
...

<!--*************************************************************************-->
<!-- Specification of Message Security                                       -->
<!--*************************************************************************-->
```

```
    <MessageSecurity>
      <NonRepudiation>

...

        <Certificate>
          <Party>

...

            <OrgName Partyname="_obibuyer"/>
            <IssuerOrgName>Verisign Inc.</IssuerOrgName>
          </Party>
          <Party>
            <OrgName Partyname="_obiseller"/>
            <IssuerOrgName>GTE Inc.</IssuerOrgName>
          </Party>
        </Certificate>
      </NonRepudiation>
    </MessageSecurity>
  </DocExchange>

...
  </BusinessProtocol>
<!--*************************************************************************-->
<!-- Specification of Service Interface 01                                   -->
<!--*************************************************************************-->
  <ServiceInterface InterfaceId="interface01">
    <OrgName Partyname="_obibuyer"/>
    <Client>
      <OrgName Partyname="_obiseller"/>
    </Client>
    <ActionMenu>
      <Action ActionId="action01">

...

        <ResponseServiceTime>
          <ServiceTime>3600</ServiceTime>

...

        </ResponseServiceTime>
       </Response>
...

      </Action>
    </ActionMenu>
    <ServerServiceTime>
       <ServiceTime>3660</ServiceTime>

...

    </ServerServiceTime>

...

  </ServiceInterface>
<!--*************************************************************************-->
<!-- Specification of Service Interface 02                                   -->
<!-- This interface below is for UnSolicited OBIPO from                      -->
<!-- buying organization to selling organization                             -->
<!--*************************************************************************-->
  <ServiceInterface InterfaceId="interface02">
```

```
    <OrgName Partyname="_obiseller"/>
    <Client>
      <OrgName Partyname="_obibuyer"/>
    </Client>
    <ActionMenu>
      <Action ActionId="action02">
...

      </Action>
    </ActionMenu>
    <ServerServiceTime>
      <ServiceTime>3660</ServiceTime>

...

    </ServerServiceTime>

...

  </ServiceInterface>
  </BusinessProtocol>
</TPA>
```

Appendix 6. Mapping of TPA Constructs to Message Header

This appendix will describe how specific constructs defined in the TPA are mapped to the proposed message header standard.