



OpenTravel™ Alliance Message Specifications – Publication 2001A

Specifications Document

Abstract

This document presents the specifications for the exchange of messages in the travel industry, covering airlines, car rentals, hotels, and other travel services. It uses the Extensible Markup Language (XML) for the exchange of these messages transmitted under Internet protocols.

Part I of this document describes the base technical architecture used by the OpenTravel™ Alliance for message exchanges, including transport protocols, identification and connections to trading partners, security and privacy, and infrastructure required to identify the message content and route it to the proper application handler.

Part II of this document addresses the travel business content and exchanges of a customer profile, and identifies the message sets supported by this publication.

OpenTravel™ Alliance, Inc.

333 John Carlyle Street, Suite 600

Alexandria, Va. 22314 USA

+1 703-548-7005

Fax +1 703-548-1264

opentravel@disa.org

<http://www.opentravel.org/>

Prepared in partnership with Data Interchange Standards Association
(<http://www.disa.org>)

September 27, 2001

Copyright © 2001. OpenTravel™ Alliance, Inc.

License Agreement

OpenTravel™ Alliance 2001A Message Specifications

Copyright, OpenTravel Alliance (OTA), 2001.

All rights reserved, subject to the User License set out below.

USER LICENSE

IMPORTANT: The OpenTravel Alliance (“OTA”) “Travel Customer Profile Specification” (“Specification”), whether the document be in a paper or electronic format, is made available to you subject to the terms stated below. Please read the following carefully.

1. OTA openly provides this specification for voluntary use by individuals, partnerships, companies, corporations, organizations and any other entity for use at the entity’s own risk. This disclaimer and release is intended to apply to the OpenTravel Alliance, its officers, directors, agents, representatives, members, contributors, affiliates, contractors, or co-venturers (collectively OTA) acting jointly or severally.
2. All OTA Copyrightable Works are licensed for use only on the condition that the users agree to this license, and this work has been provided according to such an agreement. Subject to these and other licensing requirements contained herein, you may, on a non-exclusive basis, use the Specification to develop conforming software implementations and create appropriate documentation for such software. **YOU MAY NOT EDIT OR REPUBLISH THE SPECIFICATION OR USE ALL OR ANY PART OF THE SPECIFICATION TO CREATE A NEW OR ANY GENERAL OR SPECIFIC DERIVATIVE SPECIFICATION FOR AN INTERNET TRAVEL CUSTOMER PROFILE SYSTEM OR FOR ANY OTHER USES OTHER THAN THE USES STATED HEREIN.** You may copy and distribute the Specification so long as you include; commencing on the first page of all copies, the title, and copyright notices and User Licenses appearing herein.
3. **THE SPECIFICATION AND ANY RELATED OR ATTACHED MATERIALS ARE PROVIDED IN “AS IS” CONDITION. THE OPENTRAVEL ALLIANCE AND ITS OFFICERS, DIRECTORS, AGENTS, REPRESENTATIVES, MEMBERS, CONTRIBUTORS, AFFILIATES, CONTRACTORS, OR CO-VENTURERS (COLLECTIVELY, “OTA”), ACTING JOINTLY OR SEVERALLY, MAKES NO WARRANTIES, EXPRESS OR IMPLIED AS TO THE NATURE, PERFORMANCE OR SUITABILITY OF THE SPECIFICATION. OTA EXPRESSLY DISCLAIMS ANY WARRANTIES, INCLUDING, WITHOUT LIMITATION, WARRANTIES OF MERCHANTABILITY, FITNESS FOR PARTICULAR USE, OR NON-INFRINGEMENT OF PROPRIETARY RIGHTS.**
4. By using this specification in any manner or for any purpose, you release and covenant not to sue OTA, collectively or individually, from or regarding any and all liabilities, claims, causes of action, allegations, losses, injuries, damages, or detriments of any nature relating to your use of the Specification or any part thereof. You further agree not to file a lawsuit, make a claim, or take any other formal or informal action against OTA, resulting from your acquisition, use, duplication, distribution, or exploitation of the Specification or any portion thereof. Finally, you hereby agree that OTA is not liable for any direct, indirect, special or consequential damages arising from or relating to your acquisition, use, duplication, distribution, or exploitation of the Specification or any portion thereof.

5. This User License is perpetual subject to your conformance to the terms of this User License. The OpenTravel Alliance may terminate this User License immediately upon your breach of this agreement and, upon such termination you will cease all use duplication, distribution, and/or exploitation in any manner of the Specification.
6. This User License reflects the entire agreement of the parties regarding the subject matter hereof and supercedes all prior agreements or representations regarding such matters, whether written or oral. To the extent any portion or provision of this User License is found to be illegal or unenforceable, then the remaining provisions of this User License will remain in full force and effect and the illegal or unenforceable provision will be construed to give it such effect as it may properly have that is consistent with the intentions of the parties. This User License may only be modified in writing signed by an authorized representative of the OpenTravel Alliance. This User License will be governed by the law of the Commonwealth of Virginia, as such law is applied to contracts made and fully performed in Virginia. Any disputes arising from or relating to this User License will be resolved in the courts of the Commonwealth of Virginia. You consent to the jurisdiction of such courts over you and covenant not to assert before such courts any objection to proceeding in such forums.
7. Except as expressly provided herein, you may not use the name of OpenTravel, or any of its marks, for any purpose without the prior consent of an authorized representative of the owner of such name or mark.

YOUR ACCEPTANCE OF THIS USER LICENSE WILL BE INDICATED BY YOUR AFFIRMATIVE ACQUISITION, USE, DUPLICATION, DISTRIBUTION, OR OTHER EXPLOITATION OF THE SPECIFICATION. IF YOU DO NOT AGREE TO THESE TERMS PLEASE CEASE ALL USE OF THIS SPECIFICATION NOW. IF YOU HAVE ANY QUESTIONS ABOUT THESE TERMS, PLEASE CONTACT THE OFFICE OF THE EXECUTIVE DIRECTOR OF OPENTRAVEL ALLIANCE. AS OF THE DATE OF THIS REVISION OF THE SPECIFICATION YOU MAY CONTACT THE EXECUTIVE DIRECTOR'S OFFICE AT 703-548-7005, ext 160.

Executive summary

The OpenTravel™ Alliance Message Specification, Publication 2001A is a revision of the OTA version 1 Customer Profile Specification. The catalyst for this revision came from the agreement for cooperation and collaboration between two Standards Developer Organizations (SDOs), the OpenTravel™ Alliance (OTA) and the Hospitality Industry Technology Integration Standards (HITIS) project sponsored by the American Hotel & Lodging Association, (AH&LA). Bringing the two groups together has enabled this specification to profit from the best of both organizations, building upon their collective experience in the hospitality and travel industries.

This specification also benefits from the work of other standards developer organizations; specifically, the ebXML initiative sponsored by UN/CEFACT and OASIS (www.ebxml.org), and the work of the World Wide Web Consortium (W3C) and their advancement of XML Schema to Candidate Recommendation status (2000-10-24).

This document is separated into two parts: OTA Infrastructure and the revised Profile Specification, which is the first of the business content specifications that OTA intends to develop.

Part I - OTA Infrastructure

The OTA Infrastructure section defines the messaging components and how to securely and reliably exchange messages between trading partners. This definition establishes guidelines for OTA's message sets for the exchange of OTA 2001A profiles as well as future versions of messages in the travel industry.

The OTA specification uses the eXtensible Markup Language (XML) that allows for the exchange of structured data as well as processing instructions over the Internet or other means of transport. With XML, OTA defined a common vocabulary of data items in terms of elements, attributes, or reusable entities. Logical groups of data elements are represented by schema fragments that provide formal specification of semantics and data typing that allows the schema fragments to be combined into message structures as needed.

With the recent promotion of XML Schema as a Recommendation from the World Wide Web Consortium (W3C) on 2001-05-02, OTA has developed schema files for the messaging components and profile content data found in this specification, making it available for implementation.

- OTA 2001A Infrastructure specifies that parties send messages as pairs of requests with corresponding responses, and defines both synchronous and asynchronous messaging capability.
- OTA 2001A Infrastructure uses the ebXML Header and a modified version of the Manifest part of that header document, that identifies the version of the message and the operation to be performed as metadata about the document(s) in the payload.
- OTA infrastructure defines four basic message functions: (1) Create, (2) Read, (3) Update, and (4) Delete with examples given using the customer profile as a payload document. The Create, Read and Delete functions address the profile record as a whole. The complex update process used in OTA version 1 to address individual parts of the profile record has been revised to adopt an alternative representation using XPath to reference a specific element in a source document to modify a part of the contents, or to perform a replace operation.
- The specification merges the vocabulary of tag-naming conventions established by both standards developer organizations as defined in their respective version 1 (or 1.1), and drops the requirement to include the version and hierarchy of the elements in each tag.

Security Features

OTA 2001A retains security features defined in version 1 providing authentication of parties, confidentiality, and integrity of messages, and a control section for these security features, which has been separated from the business content of each message.

Part II - Profile Specification

The Profile Specification provides a common customer or company profile that can be exchanged among travel services over the Internet. The specification provides a uniform vocabulary that captures and exchanges data on about the identity of a person or a company, and their contacts and affiliations including loyalty programs, forms of payment, travel documents, and detailed travel preferences.

A key feature of the profile allows for the definition of collections of travel preferences in terms of specific travel plans and experiences, and includes preferences for various travel services (air, hotel, car rental, other) as well as common preferences across services. The specification allows for straightforward preferences as well as collections meeting complex conditions, including choices to avoid. It also allows for identification of related persons, such as family members, companions, or fellow business colleagues, to be linked to profiles, as well as identification of a travel arranger or other organization, such as a travel club or employer, that is related to the traveler in the context of the message. A link is provided to these entities by means of a globally unique identifier, comprised of an identifying string in combination with a URL that specifies the location of the identifier.

Privacy Requirements

The OTA 2001A specification retains the strict privacy requirements defined in OTA version 1 that allow both customers and companies to indicate which data to share with other parties. The privacy attributes in OTA 2001A can be used to determine whether this information may be shared for the synchronization of system information such as keeping all copies of the profile on remote sites identical with the original, or shared for marketing purposes.

Acknowledgments

This specification is based upon the original work done by the people who contributed to the OTA Version 1 specification, as compiled by the editor, Alan Kotok, Director of Education, DISA.

As OTA began the process of revising version 1 to accommodate the merger between the two standards developer organizations, others joined to pick up where their predecessors left off. The names and organizations of people who took part in meetings and conference calls, or provided input to the specification during the period from September, 2000 to December, 2000 are listed on the following page.

A few dedicated people should be mentioned for bringing forward continuity from Version 1 during the transition to Version 2. These people include: George Smith of ConXtra, Jim deBettencourt of McCord Travel Management, and Mary Manzer of United Airlines.

Special thanks go to David Marshall, VM Systems, Inc., for dutifully documenting the recommendations of the OTA/HITIS Profile Integration and OTA Architecture teams, and developing them into substantial proposals over the course of the integration process. Thanks also to Michael Guidone, Pegasus Solutions, for his efforts, along with David Marshall, in working out the recommendations for an alternative method of partial update.

The OTA thanks the Architecture Chair, Michael Townsie, Galileo, and especially the Architecture Co-Chair, Ron Kleinman, Sun Microsystems, for guiding OTA into alignment with the ebXML specifications and representing OTA's interests to that organization.

For hosting the meetings that took place, the OTA thanks Matt Guthrie, Hospitality Solutions International, for his assistance in the merger of the data elements in the two profiles, and his insight in the design model of the Customer Profile specification, and to Adam Athimuthu, Six Continent Hotels, Inc., for contributing his experience in implementing the HITIS standards to the development efforts.

Thanks also go to Mark Hoare and Paul McGrath of Pegasus Solutions, Ron Kleinman of Sun Microsystems, and Jim Martin and Scott Gibson of Cendant Corporation, for hosting the meetings that took place. Their hospitality made the task of getting the two groups together much easier and enabled the process of assimilating the two standards into one.

OTA created the sample messages using XML Spy. (<http://www.xmlspy.com>) and the document type definitions (DTD's) the XML Schema definition fragments and the element hierarchy charts seen throughout the document using XML Authority by Extensibility (<http://www.extensibility.com>), a division of Tibco Corporation.

Gloria Gordon Zimmer, Editor

The following is a list of names and organizations of people who took part in meetings and conference calls and contributed to the development of this specification.

Participants	Company Name
John Turato	Avis Rent-A-Car System, Inc.
Adam Athimuthu	Six Continent Hotels, Inc.
Mike Kistner	Best Western International, Inc.
Chris Heinen	Best Western International, Inc.
Maureen Murphy Kieffer	Best Western International, Inc.
Ernie Nishiseki	B-There.com
Robert Jacobsze	Cendant Corporation
James Martin	Cendant Corporation
Ron Spero	Cendant Corporation
Bill Geoghegan	Consultant to AH&MA
George Smith	ConXtra, LLC/ Highwire
Tim Cochran	DISA
Gloria Zimmer	DISA
Michael Townsie	Galileo International
Rahim Amlani	GetThere, Inc.
Sean Handel	GetThere, Inc.
David Mussa	GetThere, Inc.
Tal Petty	Goldenware Travel Technologies
Virginia Suliman	Hilton Hotels
Matt Guthrie	Hospitality Solutions International
Steve Reynolds	Innovata, LLC
Michael Belak	Marriott Corporation
Jim deBettencourt	McCord Travel Management
Ernest Edward Mindlin	Microsoft Corporation
David Merrill	Multi-Systems, Inc.
Mark Stevenson	Multi-Systems, Inc.
Mark Hoare	Pegasus Solutions
Paul McGrath	Pegasus Solutions
Mike Guidone	Pegasus Solutions
Dennis Robinson	Sabre
Christopher Ferris	Sun Microsystems, Inc.
Nicholas Kassem	Sun Microsystems, Inc.
Ron Kleinman	Sun Microsystems, Inc.
Mary Manzer	United Airlines
David Marshall	VM Systems, Inc.
Allan Chan	Worldres

OTA Code of Conduct

The OTA Code of Conduct governs the OpenTravel™ Alliance. A reading or reference to the OTA Code of Conduct begins every OTA activity, whether a meeting of the OTA Board of Directors or conference call to resolve a technical issue. The OTA Code of Conduct says:

Trade associations are perfectly lawful organizations. However, since a trade association is, by definition, an organization of competitors, OpenTravel™ Alliance (OTA) members must take precautions to ensure that we do not engage in activities which can be interpreted as violating anti-trust or other unfair competition laws.

For any activity which is deemed to unreasonably restrain trade, OTA, its members and individual representatives may be subject to severe legal penalties, regardless of our otherwise beneficial objectives. It is important to realize, therefore, that an action that may seem to make "good business sense" can injure competition and therefore be prohibited under the antitrust or unfair competition laws.

To ensure that we conduct all meetings and gatherings in strict compliance with any such laws and agreements in any part of the world, the OTA Code of Conduct is to be distributed and/or read aloud at all such gatherings.

- There shall be no discussion of rates, fares, surcharges, conditions, terms or prices of services, allocating or sharing of customers, or refusing to deal with a particular supplier or class of suppliers. Neither serious nor flippant remarks about such subjects will be permitted.
- OTA shall not issue recommendations about any of the above subjects or distribute to its members any publication concerning such matters. No discussions that directly or indirectly fix purchase or selling prices may take place.
- There shall be no discussions of members' marketing, pricing or service plans.
- All OTA related meetings shall be conducted in accordance with a previously prepared and distributed agenda.
- If you are uncomfortable about the direction that you believe a discussion is heading, you should say so promptly.

Members may have varying views about issues that OTA deals with. They are encouraged to express themselves in OTA activities. However, official OTA communications to the public are the sole responsibility of the OTA Board of Directors. To avoid creating confusion among the public, therefore, the Board must approve press releases and any other forms of official OTA communications to the public before they are released

Table of Contents

	Page
User license	iii
Executive summary	v
Acknowledgements	vii
OTA Code of Conduct	ix
 Volume 1, Table of Contents	 xi

Part I - OTA Infrastructure

Section	Page
1. Introduction	1
1.1 About the OpenTravel Alliance	
1.2 Collaboration between SDOs	
1.3 Design goals and decisions	
1.4 Relationships with other industry systems	
1.5 Relationships with other standards	
1.6 Status of this document	
 2. Overview of OTA 2001A	 5
2.1 Trading Partner Model	
2.2 Technical Architecture	
2.3 Message Structure	
2.4 Error and Non-Versioned Message Operations	
2.5 Security and Privacy	
2.6 Definitions and Conventions	
 3. Technical Architecture	 7
3.1 Reference Model	
3.2 Transport Protocols	
3.3 Use of XML Technology	
3.3.1 Message Composition and Grammar	
3.3.2 Guidelines for Tag Naming Conventions	
3.3.3 Element/ Attribute Guidelines	
3.4 OTA Infrastructure	
3.4.1 Versioning	
3.4.2 Logging	
3.4.3 Unique Identifier	
3.5 Infrastructure verbs	
3.5.1 Create Verb	
3.5.2 Read Verb	
3.5.3 Update Verb	
3.5.4 The simple "Replace" verb	
3.5.5 Delete Verb	
3.6 XML Extensions	

4. Message Structure	35
4.1 Header and Payload Documents	
4.2 ebXML Header Document	
4.3 ebXML Payload Envelope	
4.4 Control document	
4.5 Session vs. SendBy elements	
4.6 Session-based Security Model	
4.7 OTA Payload Document	
4.8 Non-session based (SendBy) Security	
4.9 Use of Encryption	
5 Error and non-versioned message operations	47
5.1 Non-versioned Base Messages	
5.2 Non-versioned message DTDs	
5.3 Standard Payload Attributes	
5.4 Standard Versioned Error Messages	
5.5 Version discovery messages	
6 Security and Privacy	57
6.1 Terminology	
6.2 Security and privacy requirements	
6.3 Security and privacy recommendations	

Part II - OTA Profile Specification

I. OTA Profile.	63
Introduction	
CustomerProfile Content and Scope	
Profile Message Flow and Operations	
Transmitting Empty Data Elements	
Elements and Attributes in the CustomerProfile	
Profile Element	
1.0 Accesses information.	74
2.0 Customer information.	76
3.0 Travel Preferences	103
3.1 Common Traveler Preferences	107
3.2 Vehicle Rental Preferences	119
3.3 Air Travel Preferences	130
3.4 Hotel Preferences	142
3.5 Other Travel Services	157
4.0 Affiliations.	162
5.0 Company Information.	174
6.0 Agreements	188
7.0 TPA_Extensions	193

OpenTravel™ Alliance Message Specification

Publication 2001A

Specifications Document

Part I - OTA Infrastructure

1. Introduction

1.1 About the OpenTravel Alliance

The OpenTravel™ Alliance (OTA), which began in May 1999, is a consortium of suppliers in all sectors of the travel industry, including air, car rental, hotel, travel agencies, and tour operators, as well as related companies that provide distribution and technology support to the industry. The OpenTravel Alliance now has over 125 members representing influential names in all sectors of the travel industry. The Alliance is comprised of five working groups – air, car, hotel, leisure supplier, and non-supplier – together with an interoperability committee to coordinate their efforts.

OTA defines open messages in eXtensible Markup Language (XML) that makes it possible to exchange business data seamlessly among different systems, companies, and industries over the World Wide Web. This specification defines the infrastructure on which OTA will build additional messages and functions, and represents the revision of OTA's first product, a common customer profile for the travel industry.

Companies in the travel industry recognize the importance of information management, both in terms of day-to-day operations and as a strategic tool. The travel industry pioneered online booking and registration systems, and in many respects, sets the expectation level for other industries in terms of responsiveness, reliability, and security of transaction processing. With the coming of the Internet and its ability to link millions of customers and companies in real-time computer communications, customer expectations for online services have increased further, and the travel industry seeks to continue to meet and exceed these expectations. Also, in keeping with the tradition of the Internet, even in its brief history, OTA specifications are open documents and will remain freely available to all industry participants.

OTA specifications provide a framework for companies in the travel industry to create new relationships with customers as well as create new partnerships with fellow companies in the business. As specifications for the industry, these documents provide a means by which companies can improve their interaction with trading partners as well as their level of service for customers. Lowering the demands for customized interfaces between trading partners will allow companies to concentrate their resources on developing innovative ways to communicate with their customers and provide increased levels of service to them.

The OpenTravel Alliance has a white paper that provides a detailed description and rationale for Internet-based exchanges and the value they bring to customers, travel suppliers, and companies in the distribution network. Readers may view the document on the public OTA web site at <http://www.opentravel.org/>. Note, however, that during the work leading to this specification many applications of it have been found in areas other than distribution of travel inventory.

1.2 Collaboration between Standards Developer Organizations (SDO's)

A word about the philosophies of the two Standards Developer Organizations that contributed to this product: readers will note that OTA uses the term *specification* rather than *standard*. OTA reserves the word standard for documents that undergo a rigorous, lengthy development and review process, such as those from International Organization of Standards (French acronym ISO) or American National Standards Institute (ANSI). This document, therefore, is called a specification, which takes broad standards such as the Extensible Markup Language (XML) and

applies them in particular industries or to particular problems. Several ISO standards used in the development of this specification are referenced in the text and listed in the Appendix.

As a specification, this document focuses on travel industry requirements and while developed in an open consensus process, does not represent itself as meeting the stringent requirements of standards.

An OTA specification, having gone through its own review process and recommended for adoption and broad implementation by the travel industry may be submitted as an ANSI standard as a result of an Agreement with the American Hotel & Motel Association (AH&MA), signed October 18, 2000. This agreement allows the AH&MA, an ANSI-accredited standards developer organization, to submit a specification for approval as an ANSI standard. That process, monitored by the American National Standards Institute, involves the publication of an intended standard for a period of commentary and consists of a canvass of those parties who would be materially affected by the standard.

AH&MA, founded in 1910, is a federation of state lodging associations throughout the United States, with some 11,000 property members worldwide, representing more than 1.4 million rooms. AH&MA provides operations, technical, educational, marketing, and communications services plus governmental affairs representation to the lodging industry.

1.3 Design goals and decisions

OTA specifications are designed using the following guidelines:

Openness - OTA specifications are publicly available to all organizations seeking to develop new or enhanced systems for improving travel services. Likewise membership in OTA for the purpose of developing these specifications is open to all organizations.

Support of exchanges among a broad number of parties - OTA specifications support communications among industry participants, including but not limited to:

- Travelers
- Airlines
- Car rental agencies
- Hotels
- Passenger rail services
- Leisure travel providers, such as cruise lines and tour services
- Travel arrangers
- Global distribution systems

Flexibility - OTA specifications provide travel service providers and their representatives with the flexibility they need to rapidly develop, test, and deploy new services. These specifications outline the minimum functionality necessary to provide reliable interactions between systems.

Extensibility OTA realizes that trading partners, using the minimum functionality defined in the specifications, may have a need for extensions to include proprietary data between them. OTA has defined a mechanism by which the specifications take advantage of the 'eXtensible' in XML to allow for these extensions to take place. Specific locations have been designated in logical

places throughout OTA specifications for trading partners to add their own data. Trading partners are encouraged to submit extensions used regularly between multiple trading partners for consideration for future revisions of OTA specifications.

Security - OTA recognizes the need to protect the information from unauthorized access as well as the need to give the individual customer or company control over the creation, update, and exchange of data with other parties.

Platform independence - This specification is intended to be implemented on any equipment or software that can support the common standards used in the specification.

International scope - OTA wrote this specification in English but intends to extend later versions to provide representation in character sets supporting the Unicode standard; as the eXtensible Markup Language (XML) used as the basis for this specification already supports Unicode. Wherever possible OTA designed the data elements in this specification to meet as many global conventions as possible.

Future Versions - OTA plans to add more services and functionality to its specifications in a way that minimizes incompatibility for those implementing this or future versions. However, this version of the specification incorporates significant changes, including incorporation of a modified Header envelope adapted from the ebXML specifications, and migration to the W3C XML-Schema candidate recommendation, that may not be retroactively compatible to OTA version 1. The OTA work groups that developed the version 1 specification realized this transition would take place, and designed the earlier version in such a way as to allow for migration to subsequent versions, intending to ease the burden for those implementing version 1.

1.4 Relationships with other industry systems

Electronic Data Interchange (EDI) - EDI is the computer-to-computer exchange of structured data between organizations according to a standard format. EDI can use any transport protocol that allows for exchanges of data between systems and organizations, but it generally uses standard formats such as X12 and UN/EDIFACT that pre-date the Web and Internet. While OTA specifications enable data exchanges among individuals and organizations, and create a common vocabulary of data elements and tags, they do not follow the established EDI standards.

Global Distribution Systems - GDSs centralize, consolidate and deliver travel supplier information for the online booking of reservations. Travel agencies are the primary users of GDSs, but direct-to-customer GDS services have recently become available over the Web. GDSs currently present information only on the companies and information that subscribe to their services and supply data to them. OTA's work can be expected to make it easier for GDSs to extend the range of the content they offer. OTA specifications offer the exchange of messages with profile data for implementation by any travel supplier or data service that adheres to these specifications based on open standards.

1.5 Relationships with other standards

Extensible Markup Language (XML) - OTA specifications use XML to represent the data used for travel transactions, as required by XML Recommendation 1.0 of the World Wide Web Consortium.

World Wide Web - The World Wide Web (WWW or the Web) is a collection of interconnected documents on the Internet that adhere to the accepted conventions for the identification and display of those documents. Most implementations of OTA specifications will use Web-related protocols to exchange XML messages as specified in this document.

XML Schema - OTA 2001A embraces the Candidate Recommendation of the World Wide Web Consortium (W3C) for XML schema (2000-10-24) and follows its conventions as guidelines for the implementation of XML messages

Hotel industry specifications - In the hotel industry, two organizations have defined common data elements for transactions among trading partners: the Hotel Electronic Distribution Network Association (HEDNA) and the Hospitality Industry Technology Integration Standards (HITIS). The HITIS standards used the work of the Windows Hospitality Interface Specifications (WHIS) for their base documents in the development of their standards. HEDNA made its descriptive content specifications available to both organizations, and OTA relied upon the work of WHIS and HEDNA in the development of OTA version 1. This 2001A specification represents the merger of the HITIS Profile Synchronization standard with OTA version 1 into one document. As a result, the portions of the HITIS standards that incorporate elements of HEDNA's work will remain compatible with this version of the profile.

1.6 Status of this document

This document is a draft publication of the OTA 2001A specification released for public review on May 15, 2001. It is anticipated that this document will undergo a period of commentary and resolution of comments may result in revisions prior to final publication. Individuals and companies implementing the specification or who would otherwise like to give their comments on the document should address them by e-mail to OTAccomments@disa.org.

OTA also welcomes comments by fax to +1 703-548-1264 or by mail to:

OpenTravel Alliance, Inc.
333 John Carlyle Street, Suite 600
Alexandria, Va. 22314 USA

2. Overview of OTA Version 2

This version of the Open Travel Alliance specification revises the overall message structure and basic technical architecture introduced in OTA Version 1. The progress made in the area of XML guidelines from standards organizations, such as W3C and ebXML, have enabled the revision of OTA Version 1 to conform with broad cross-industry practices that will ensure interoperability between multiple systems that exchange travel-related information.

Many of the changes that have occurred in these OTA specifications such as versioning, error handling, and header structure have been adopted directly from these organizations. Although these issues are not limited to the travel industry, this specification addresses them because of the business need for these functions to be defined in order for implementations within the travel industry to establish a baseline from which trading partners can begin their message exchange.

From a travel industry perspective, this version also addresses the detailed contents of a customer profile. See Part II–Profile Specifications.

2.1. Trading Partner Model

In order for individuals or organizations to begin exchanging data, each trading partner must first engage in a dialogue to learn the counterpart's support for various transport protocols, security, and required fields. In the 2001A specification, OTA has focused on messages that support a simple trading partner model, or exclusive trading partner agreement.

Support for exclusive trading partner relationships involves linking one requestor to one supplier. The OTA has specified Non-Versioned discovery messages that can be sent from partner to partner to assess capabilities, and has structured the 2001A messages to be used to exchange information.

OTA does not define any type of directory services that can be used to establish exclusive relationships at this time. While OTA supports dynamic discovery, it does not define any type of electronic contract document nor attempt to certify the acceptance of a trading partner relationship in either a predefined or dynamic situation.

As of the publication of OTA 2001A specification (late 2000) the means of dynamically discovering the characteristics of multiple trading partners has not been fully examined. OTA plans to address this issue in future versions in conjunction with other standards organizations as they bring forward their collective bodies of knowledge.¹

2.2. Technical Architecture

The technical architecture chapter presents the role and function of the various components of the messages that support the exchanges between trading partners defined in this specification. The chapter also discusses the use of basic technologies such as transport protocols, features inherent in the use of MIME for messaging services and defines important security features of the specification. This specification supports the message choreography of the connection to one trading partner per message, but supports multiple payloads that may include different operations or batch operations in one message.

¹ We thank Jim deBettencourt of McCord Travel Management for his contributions to this part of the document.

OTA has defined certain guidelines for the use of XML technology, including conventions for message composition and grammar as well as tag names, the use of DTDs for purposes of validation, and the use of XML Schema to provide a formal data typing capability.

2.3. Message Structure

OTA 2001A transactions use a request and response model that provides a matched pair of messages to help account for and manage the data flow. OTA messages use the message structure from the ebXML Header envelope that effectively separates the routing and packaging information from the payload content of the message. The structure also requires a Control section separate from the root Action Verb of the payload, (previously identified by the root tag name, Content, in OTA version 1) to separate security details and connection with the trading partner from the business content. (In future versions of the specification the Control section may be eliminated in favor of analogous structures available within the ebXML Header section.)

2.4. Error and Non-Versioned Message Operations

Some of the exchanges between trading partners will contain messages for administrative purposes rather than business context. These messages are intended to discover the versions supported by trading partners and for transmitting certain errors. Because this set of messages transcends individual versions, OTA has established a separate category and syntax for these messages that operates independently of the versioned content. Chapter 5 of this specification covers non-versioned messages.

The specification also makes a distinction between error responses in versioned messages – those that arise from business interactions with the processing application, from error messages resulting from the transport layer or infrastructure of the messaging process that do not carry version numbers.

2.5. Security and Privacy

Because of the sensitivity of the data contained in customer information in the travel industry, OTA 2001A gives special attention to issues of authentication, confidentiality, and message integrity. It also gives recommendations for non-repudiation of transactions in anticipation of standards that apply more directly to this issue. Chapter 6 of this specification addresses the security issues.

Part II of this document, Profile, focuses on specific privacy protections that require the customer to give explicit permission to share the profile data, even for routine synchronization of updates.

2.6. Definitions and Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in IETF document RFC 2119. For further information and the Abstract of the document, RFC 2119, see Volume 2, Appendices.

This text makes every attempt to accurately reflect the DTDs and XML Schema listed in the Appendices. In the case of a conflict between the document text, the DTD and the XML Schema, the XML Schema takes precedence.

3. Technical Architecture

As cross-industry XML standards organizations begin to complete development efforts that will define how Web-based commerce is exchanged, OTA has adopted portions of these efforts and thus reestablished some conventions for message definition and exchange. This section outlines the changes and guiding decisions for the OTA 2001A message structure.

3.1. Reference Model²

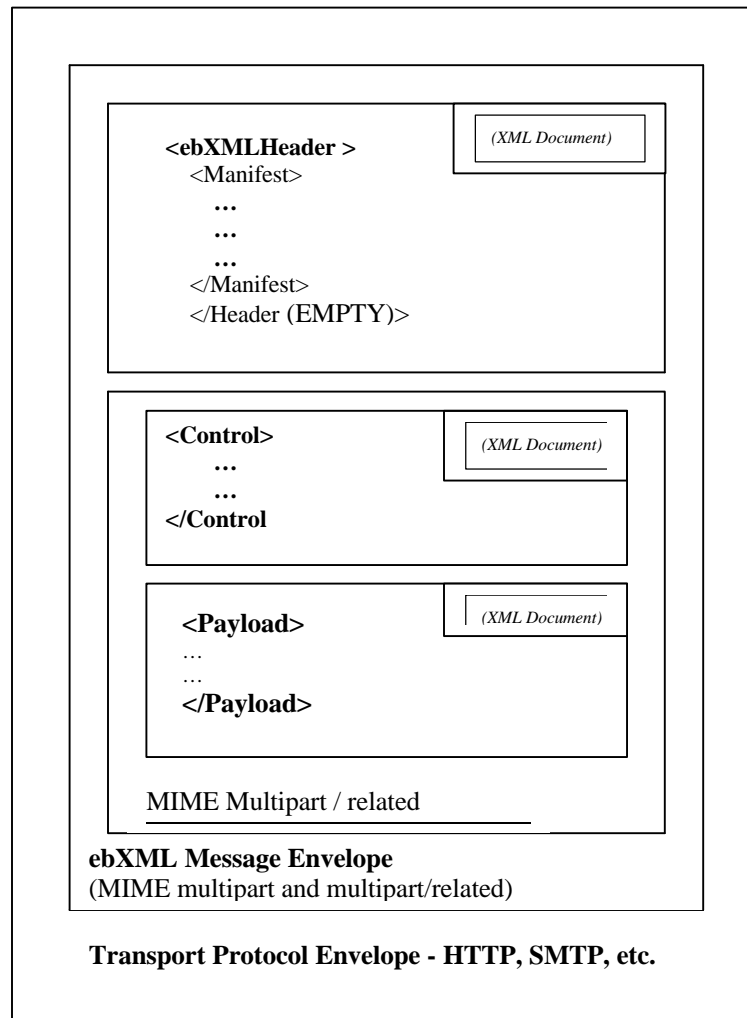


Figure 1. OTA 2001A reference model

The reference model for the OTA architecture can be viewed as a logical series of envelopes. Beginning at the bottom, the transport layer determines the communication protocol that governs the flow of messages. Inside the transport layer, OTA has adopted the ebXML Message Envelope which is a MIME multipart/related wrapper that provides an overall container and several non-travel-specific elements such as the transaction choreography and security of the messages.

² Much of the content in this section is adapted from the ebXML Transport, Routing and Packaging specification v0.8 (see <http://www.ebxml.org>)

The message envelope MUST contain two MIME headers:

- Content-Type
- Content-Length

The MIME Content-Type MUST be multipart/related for all OTA message envelopes. For example:

```
Content-Type: multipart/related;
```

The MIME Content-Type contains three attributes:

- Type
- Boundary
- Version

The MIME type attribute is used to identify the message envelope as an ebXML compliant structure. It conforms to MIME XML Media Type [XMLMedia] and MUST be set to "application/vnd.eb+xml".

For example:

```
type="application/vnd.eb+xml "
```

The MIME boundary attribute is used to identify the body part separator used to identify the start and end points of each body part contained in the message. The MIME boundary SHOULD be chosen carefully in order to ensure that it does not occur within the content area of a body part see [RFC 2045] for guidance on how to do this.

For example:

```
Boundary:="----vmguys.com321-----"
```

The MIME version attribute is used to identify the particular ebXML Message Envelope being used. All message headers SHOULD use "0.8"³.

For example:

```
version="0.8 "
```

The MIME Content-Length header is a decimal value used to identify the total number of OCTETS contained in all constituent message body parts, including boundary body parts.

The value of the Content-Length MIME header is computed by counting the total number of OCTETS starting with the first OCTET after the CRLF following the first MIME header and ending with the OCTET immediately before the MIME object's last boundary string.

For example:

```
Content-Length: 8941
```

³ Although OTA version 2 is not 100% compatible with ebXML 0.8 it is very close from a packaging perspective.

An example of a compliant ebXML/OTA Message envelope header appears below:

```
Content-Type: multipart/related; type="application/vnd.eb+xml";  
boundary:-----vmguys.com321-----  
Content-Length: 8941
```

The MIME message envelope encapsulates a minimum of three XML documents; the ebXML Header document, one Control document, and one or more payload documents, which defines the messages exchanged among trading partners based on an established set of elements and attributes prescribed by OTA.

The ebXMLHeader document is always the first within the message envelope container, and it is preceded by a MIME header envelope containing the following three MIME headers:

- Content-ID
- Content-Length
- Content-Type

The Content-ID MIME header identifies this instance of an ebXML/OTA message header body part. The value for Content-ID SHOULD be a unique identifier, in accordance with RFC 2045. For example:

```
Content-ID: 2000-1212-095613-1233000@vmguys.realm
```

The MIME Content-Length header is a decimal value used to identify the total number of OCTETS contained in the ebXML header MIME body part. For example:

```
Content-Length: 785
```

The MIME Content-Type for an ebXML header is identified with the value “application/vnd.eb+xml”. Content-Type MUST contain two attributes:

- version
- charset

The MIME version attribute indicates the version of the ebXML specification to which the header document conforms. Although not strictly conforming, OTA recommends the value “0.8” be used. For example:

```
version="0.8"
```

The MIME charset attribute identifies the character set used to create the ebXML Header document. The MIME charset SHALL be equivalent to the encoding attribute of the ebXML Header document. For maximum interoperability it is RECOMMENDED that UTF-8 be used. Note: this is not the default for MIME.

For example:

```
charset="UTF-8"
```

An example ebXML header document follows:

```
Content-ID: ebxmlheader-123@vmguys.com/OTAEngine
Content-Length: 781
Content-Type: application/vnd.eb+xml; version="0.8";
charset="UTF-8"

<?xml version="1.0" encoding="UTF-8"?>
<ebXMLHeader>
  <Manifest>
    ...
  </Manifest>
  <Header/>
</ebXMLHeader>
```

The XML payload document is the level in the OTA reference model that contains actionable OTA messages. These include authentication flows, infrastructure verbs that define non-travel specific elements and attributes, and travel verbs, elements and attributes.

Infrastructure verbs are actions such as the typical Create-Read-Update-Delete (CRUD) semantics, as well as sub-actions for more complex operations. These verbs do not represent travel-specific actions by themselves, but serve as a framework to act upon travel-specific content. The four main infrastructure verbs do not operate alone, but in conjunction with the request and response model described in Chapter 4, Message Structure.

Travel verbs define verbs and actions that are travel-specific (e.g., availability query), and pertain to travel industry business operations. The OTA 2001A Profile specification does not contain any travel-specific verbs, but future messages to be defined by OTA will contain travel verbs. Travel-specific actions may or may not use the infrastructure verbs.

The payload document(s) contains the OTA travel elements and attributes that make up the business content of the messages. **Section 2**, Profile, describes the content of the payload in detail. Business content documents utilize both travel verbs and infrastructure verbs as their root element.

OTA 2001A defines the XML infrastructure protocol and business content, in accordance with the logical view of the architecture as described above. This chapter discusses each of the layers in the architectural model in further detail.

3.2. Transport Protocols⁴

OTA messages are defined above any specific transport protocol and do not contain any invocation semantics specific to transport. However, as a practical matter, the Hypertext Transport Protocol (HTTP) is the underlying reference transport protocol. HTTP is used to establish initial communication between two parties in the absence of any other prior arrangement. As such, OTA messages **MUST** be able to flow on HTTP as a base and all OTA implementations **MUST** support exchange of versionless discovery messages via HTTP 1.1. One of the benefits of requiring HTTP for an initial message exchange enables the use of an HTTP-based URL, for example, as a reference to the location of a profile.

⁴ We thank Jim DeBettencourt of McCord Travel Management and David Marshall, VM Systems for their help in preparing this section of the chapter.

OTA Architecture however, does not preclude OTA standard messages from flowing over other transport protocols, such as File Transfer Protocol (FTP), Simple Mail Transfer Protocol (SMTP) or Internet Inter-ORB (Object Reference Broker) Protocol (IIOP). Through a prearranged agreement, two parties may begin transfer of messages on another protocol, and even if using HTTP as a way to make initial contact, parties may agree to change to another protocol after initial registration.

Specifics of transport protocols other than HTTP are negotiated outside of the OTA and beyond the realm of this version of the OTA 2001A architecture. OTA RECOMMENDS that each trading partner communicate the required transport protocol parameters to all of their other trading partners, and the remainder of the section provides guidance on discovering these requirements.

Trading partners can choose from a number of different Internet-based communications protocols that can support the exchange of OTA documents and requests/responses. Selection of a specific protocol depends upon the needs of the trading partners involved, and with any selection a variety of additional implementation issues will inevitably arise.

Parties implementing message exchanges will face the initial hurdle of understanding specific performance and message handling characteristics of the chosen protocol. For example, selection of HTTP, or even secure HTTP (HTTPS) protocol if security issues are important, still requires specifying the methods by which data are transferred to a server or host system. HTTP supports both a GET and a POST method for sending data to a server, but each method also has specific limitations and restrictions.

The HTTP GET method sends data to the server as name-value pairs, and requires URL encoding of any values to eliminate characters not understood by an HTTP server. HTTP POST returns a message to the originating client in the body of an HTML document, possibly the location for the response if originally an OTA request message was sent to the server.

Use of FTP-based message exchanges has other specific requirements. FTP requires some type of file naming convention be developed for files sent or received from a server, or placed in any FTP output area. And, as discussed below, selection of a specific message exchange protocol could have significant impact on the appropriate message security and coordination method (e.g. versioned versus non-versioned OTA messages) appropriate to a trading partner exchange.

Finally, each protocol adds a certain level of complexity to the overall error management tasks. Protocol, network, and server error messages and their handling can become a significant complication to the basic exchange of OTA messages among trading partners. In other words, actual implementation of a message exchange can be best viewed as a series of decisions on OTA message types, transport procedures, and security and error management that may easily differ from one trading partner exchange to another.

As cross-industry standard directories, such as the proposed Directory Services Markup Language (DSML), stabilize and provide the required capabilities, the OTA may adopt a distributed vendor directory model to allow for ad hoc communications between trading partners without requiring extensive manual processes. DSML (<http://www.dsml.org/>) is an initiative to represent directory services in XML, begun by Bowstreet (an XML software company) but now supported by leading companies in the industry: AOL/Netscape, IBM, Microsoft, Novell, Oracle, and Sun Microsystems. As of December 1999 DSML has been submitted to standards bodies for their consideration.

OTA may consider adopting the Internet-based Universal Description, Discovery, and Integration (UDDI) specification in order to enable trading partners to quickly, easily and dynamically find each other, negotiate protocols outside the OTA non-versioned message structure, and transact business between them. As of November 16, 2000, the UDDI Project, a broad coalition of business and technology companies, has announced the initial public beta testing of the UDDI Business Registry. The UDDI Business Registry is located at: <http://www.uddi.org/>.

Additionally, the ebXML technical architecture defines registry and repository services explicitly for purposes of trading partner and trading partner capability discovery. Once the ebXML Registry and Repository working group has published a specification, this is another technology that may be considered.

3.3. Use of XML Technology

OTA 2001A messages **MUST** meet the requirements of well-formed XML documents, as specified in XML Recommendation 1.0, published by the W3C. The document is available from the W3C online at <http://www.w3.org/TR/1998/REC-xml-19980210>.

OTA version 1 messages **SHOULD** begin with the XML declaration indicating use of XML version 1.0. That declaration reads:

```
<?xml version = "1.0"?>
```

OTA version 1 was based solely on validation to document type definitions (DTDs). With the recommendation from W3C (2000-10-24) for the advancement of XML Schema to Candidate Recommendation status, OTA recognizes that the use of schema will provide a formal, non-ambiguous data typing capability to the specifications, allowing trading partners to make use of schema fragments within their own implementations if they choose to do so. As XML Schema evolves as a standard, it will eventually be accommodated by XML tools that will provide capabilities for validation. As a result, this OTA 2001A publication includes both a DTD for validation and schema files for use by trading partners who may prefer the use of schema for messages exchanged between them. The DTDs and Schemas can be found in Volume 2, Appendices.

Until such time as the capability for validation to schema becomes generally available, valid OTA 2001A messages **MUST** continue to meet the grammar rules specified in the document type definitions or DTDs. An example of a document type declaration is:

```
<!DOCTYPE OTA_CreateProfilerQ SYSTEM "OTA_CreateProfilerQ.dtd">
```

Valid messages **SHOULD** include a DOCTYPE declaration that follows the XML declaration, and that DOCTYPE declaration **MAY** include a document type definition (DTD) or a schema definition as agreed between trading partners. While OTA does not require that every message be validated during real-time transaction processing, such messages must meet the requirements of being able to be validated.

Both the XML and DOCTYPE declarations precede the root element of the payload document.

OTA non-versioned messages **MUST** refer to a DTD in order to be validated. See Chapter 5 - Errors and Non-Versioned Messages.

3.3.1.Message Composition and Grammar

This section defines the rules for the order and structure of the OTA travel-specific business messages. The rules for grammar composition define a consistent vocabulary that includes both infrastructure and travel-specific verbs, as described in the reference model above, and tag names that follow certain guidelines for naming conventions.

Most versioned OTA messages will have at least two payload documents: a Control document and one or more payload document(s) that begins with the root tag of the verb defining the action to be performed. The Control document consists of information about the connection with the trading partner, and infrastructure protocol information such as authentication security and session information.

One advantage of having a Control document and one or more individual payload documents is that the Control document will be affected by the work of standards organizations such as ebXML. Information in the Control document is generic and not travel specific, and this data will be more likely to change as XML standards develop further.

The payload document(s) contains the travel-specific information. A payload document is a Request or a Response message, and the root element of the payload MUST consist of the Action verb. This action verb is identified in the ebXMLHeader <DocumentReference> element within the <Manifest> element by the attribute *xlink:role* which identifies the URL, as well as the action to be performed by the message. Both the root element tag name and the action that appends the URL entered in the *xlink:role* attribute must be identical strings in order for the parser to be able to route the payload document and to the appropriate application handler for processing. Chapter 4, Message Structure, gives examples of the Manifest and Action verb construction.

3.3.2.Guidelines for tag naming conventions

A key part of the XML grammar is consistent naming conventions for tags that represent the infrastructure and business-related elements. Tag name writers MUST follow these rules unless business requirements require other naming conventions.

- Use mixed case tag names, with the leading character of each word in upper case and the remainder in lower case.
Example: <PaymentForm>
- Acronyms are discouraged, but where needed, use all upper case.
Example: <PropertyURL>
- Where acronyms or single-letter words cause two upper case characters to be adjacent, separate them with an underscore (_).
Example: <PO_Box>
- Illegal characters cannot be used (e.g.: forward slash, etc.). Recommended characters in a tag name are basically limited to letters and underscores.
Example: (not allowed) <Date/Time>

- The use of periods, which appeared in OTA version 1 to indicate the version and hierarchy, is discouraged.

Tag writers SHOULD use these guidelines when constructing tag names.

- Use the same tag names with elements in a similar child structure

Example: <ContactAddress>
 <HomeAddress>
 <WorkAddress>

- Use plural tag names only for collections.

Example: <CreditCards>
 <CreditCard>

- Element and attribute names should not exceed 25 characters. Tag names should be spelled out except where they exceed 25 characters, then standardized abbreviations should be applied, using the abbreviations in OTA Version 1 as a baseline.

Example: <ShareSyncInd>

- Element and attribute names should incorporate the proposed list of suffixes for tag names as recommended by ebXML. The ebXML Data Element Representation Classes are the following (includes ebXML definition):

Amount	A number of monetary units specified in a currency where the unit of currency is explicit or it may be implied.
Code	A character string that represents a member of a set of values.
Boolean	An enumerated list of two, and only two, values which indicates a condition such as on/off; true/false etc. (It was the general consensus to use 'Flag' as a term to indicate a Boolean value.)
Date	A day within a particular calendar year. Note: Reference ISO 8601.
Time	The time within any day in public use locally, independent of a particular day. Reference ISO 8601:1988.
DateTime	A particular point in the progression of time. Note: This may incorporate dependent on the level of precision, the concept of date,
Identifier	(standard abbreviation Id, meaning a unique identifier) A character string used to identify and distinguish uniquely, one instance of an object within an identification scheme.
Name	A word or phrase that constitutes the distinctive designation of a person, object, place, event, concept etc.
Quantity	A number of non-monetary units. It is normally associated with a unit of measure.
Number	A numeric value which is often used to imply a sequence or a member of a series.
Rate	A ratio of two measures.
Text	A character string generally in the form of words.
Measure	A numeric value that is always associated with a unit of measure.

- OTA suggests the additional tag naming convention of 'Type' to be used for an enumeration.

Type	An enumerated list of values from which only one value can be chosen at a time.
------	---

Simple and Complex data types:

- A simple data type can be considered one of the basic data element representation types where only one tag is used and the data type is represented by the data within the tag (or an attribute within the element represented by the tag).

Example: `<CreateDateTime>2000-07-28T17:12:00Z</CreateDateTime>`

- A complex data type is comprised of multiple simple types nested within the complex type. A complex data type could be considered an 'object'. The W3C data type `timeDuration` which consists of any two of three values, `<startInstant>`, `<duration>`, and `<endInstant>` is an example of a complex data type.

Example: `<timeDuration>`

```

<startInstant>1999-05-31T13:20:00</startInstant>
<duration>+000000002T474000</duration>
</timeDuration>

```

OTA has designed its tag names to make them human-readable, which offers an advantage over traditional EDI coding that requires machine interpretation of even simple transactions or messages. With OTA's tags, parties can read the messages directly to see their content, which will make them accessible to many more trading partners than before, using no more than an XML-enabled Web browser.

3.3.3.Element/Attribute Guidelines

OTA has adopted guidelines for the use of attributes vs. elements, with the intention to reduce overall message size and avoid the potential of naming collisions. For a given data element, the preferred method is to represent that data-element as an attribute. The data-element is represented as an element if and only if:

- it is not atomic (i.e. It has attributes or child elements of its own) OR
- the anticipated length of the attribute value is greater than 64 characters OR
- presence or absence of the attribute represents a semantic 'choice' or branch within the schema OR
- the data element repeats (i.e. may have more than one instance)

3.4. OTA Infrastructure

This section describes a consistent approach for basic functions, including versioning, the maintenance of event logs, and the unique identification of a trading partner or specific record.

3.4.1.Versioning

To provide for both implementation stability and managed growth, OTA has adopted a practice of separating the infrastructure version from the version numbers of individual messages. Infrastructure changes, such as the addition of Header elements or changes in the method used to connect to trading partners and initiate authentication or security-related functions are included in an Infrastructure version. This specification, 2001A, documents the OTA infrastructure service interface in Chapter 4, Message Structure, which supports a modified version of the ebXML 0.21c header and the 2001A Control document. That infrastructure is not anticipated to change until OTA adopts the Messaging Specification published by ebXML in its entirety.

In order to isolate and contain the scope of change with the addition of new business content and messages, when OTA releases a specification each new message introduced will be versioned starting with the number 1. This places the versioning of the messages at the level of message interaction, or the action verb that is the root of the payload document. When changes to a schema or the addition of content elements in a business function of OTA messages occur, the message version number is incremented by one whole number.

The *Version* attribute in the <Schema> element within the <Manifest> indicates the version of the referenced document transmitted in the payload. The identification of such metadata about the version at this level allows a trading partner to immediately identify whether its system can accept the document(s) and process them appropriately. It is no longer necessary to parse the entire documents in order to return an error message about version compatibility.

OTA will identify their publications by the year of publication, and sequential publications within a calendar year further identified by the letter A, B, C, or D, etc. For example, the first specification published in the year 2001 will be 2001A, and the second publication 2001B, etc. The publication will list all messages supported in the version, which may include new messages or revisions of existing messages. New messages will be published as version 1, and revised message sets will be incremented by N+1. With each publication, a list of valid enumerations supported for the *Type* attribute will be included.

Vendors and operators of OTA implementations, upon examining a specification, may discover content and messages that have been added or upgraded that are of interest to them and identify others that are irrelevant to their business functions.

3.4.2. Logging

Organizations offering OTA transactions SHOULD provide logging capability, regardless of the type of transaction in the business message (e.g., travel verbs, infrastructure verbs), and trading partners MAY exchange event logs to provide audit trails.

Because of the lack of widely used standards or conventions for defining event logs, OTA does not require use of a specific log format, nor does the message architecture preclude any logging capability.

3.4.3. Unique Identifier on Profiles

Each record that identifies a unique account of travel information, such as a profile or reservation record, MUST have a unique identifier assigned by the system that creates it with the tag name <UniqueId>. The unique identifier on the record MUST contain both a *Type* and an *Id* attribute. It MAY optionally include a *URL* and an *Instance* attribute. The syntax for a <UniqueId> is formally defined in the following DTD fragment:

```
<!ELEMENT UniqueId EMPTY>
<!--ATTLIST UniqueId
      URL          CDATA #IMPLIED
      Type          (Profile)#REQUIRED
      Id            CDATA #REQUIRED
      Instance      CDATA #IMPLIED
-->
```

- *URL* - This optional attribute is what makes a <UniqueId> instance globally unique outside the context of a single bilateral conversation between known trading partners. OTA recommends that the URL referenced by the URL identifying the public HTTP v1.1 OTA discovery message implementation for each trading partner. **Note:** In the absence of having a public URL, the reference for this attribute could be determined by bilateral agreement.
- *Type* - This enumerated attribute references the type of object this <UniqueId> refers to, and gives this element its generality. By convention, the *Type* attribute value is the same as the OTA element tag name for the referenced object, for example, "Profile". As additional message types are defined in future versions of OTA specifications, the *Type* attribute enumeration will expand to include additional tag names values where a <UniqueId> applies.
- *Id* - This represents a unique identifying value assigned by the creating system, using the XML data type ID. The *Id* attribute might, for example, reference a primary-key value within a database behind the creating system's implementation.
- *Instance* - This optional attribute represents the record as it exists at a point in time. An *Instance* is used in update messages where the sender must assure the server that the update sent refers to the most recent modification level of the object being updated.

Possible implementation strategies for *Instance* values are:

- a timestamp
- a monotonically increasing sequence (incremented on each update)
- an md5 sum of the binary representation of the object in its persistent store

3.4.3.1. Examples of unique identifiers

A valid unique identifier MAY contain only the *Type* and *Id* (a unique string assigned by the system that created it) attributes:

```
<UniqueId Type="Profile" Id="1234567"/>
```

To ensure that a unique identifier is globally unique (in the universal namespace) add a *URL* attribute which includes a fully-qualified domain-name:

```
<UniqueId URL="http://vmguys.com/OTAengine/" Type="Profile"
Id="1234567"/>
```

This *Id* is assured of being globally unique in any namespace as the URL points to a vendor's version discovery implementation which, in turn, relies on the unique domain name for the vendor assigned via the INTERNIC. OTA has considered the potential effect a name change would have on globally unique identifiers, and noted that a change in URL or primary domain name should not present an issue unless another business entity assumes the previous URL unaltered.

3.4.3.2. Rationale

The following table summarizes the equivalence relationships between the <v1.UniqueId> element, its children and their attributes and the OTA 2001A <UniqueId>:

<i>V1 Element / Attribute</i>	<i>V2 Element /Attribute</i>	<i>Notes</i>
v1.UniqueId.TradingPartner.CompanyCode	UniqueId.URL	Used to help increase the likelihood of a unique id being truly unique. The introduction of a URL in v2 guarantees this.
v1.UniqueId.TradingPartner.CompanyCode.CodeContext	N/A	The codeContext was a qualifier to identify CompanyCode. In v2 this is implicit in the use of a URL and therefore not needed.
v1.ResourceId	UniqueId.Id	The unique id value assigned by the creating system
v1.ResourceId.ResourceContext	N/A	In v1 ResourceContext was used to increase the chances of uniqueness. No longer required in v2.

This approach to unique naming takes into consideration the following benefits:

- provides a simpler, more succinct representation
- guaranteed to be a globally unique identifier within the universal namespace (with the introduction of the URL attribute)
- becomes applicable and reusable in other OTA specifications

3.5. Infrastructure Verbs

Defining certain actions at the infrastructure level reduces the need for reconciling verb conflicts that may arise from business-specific verbs. Infrastructure verbs are the root elements of a payload document. The basic operations include Create, Read, Update, and Delete – memorably abbreviated CRUD. These four verbs provide consistent conventions for basic actions affecting both infrastructure and business elements in OTA specifications.

The Create, Read, and Delete actions **MUST** apply only to entire records. Updates allow for addressing one or more individual elements, and making changes to part(s) of a record.

3.5.1.Create verb

The Create infrastructure verb defines an operation that generates a new record with a unique identifier. The sequence follows these steps:

- Requestor sends a Create request along with the initial data, and optionally a unique identifier.
- Receiver creates a new record and assigns a unique identifier (e.g.: a Profile Id or Reservation Id).
- Receiver responds with a message providing a unique identifier for the new record created and optionally, any data entered by the requestor.

Example 1 - Create request message⁵

```
<OTA_CreateProfileRQ EchoToken="12345" TimeStamp="20001217T182248Z">
  <UniqueId Type="Profile"/>
  <Profile>
    <Customer Gender="Male">
      <PersonName>
        <GivenName>John</GivenName>
        <SurName>Smith</SurName>
      </PersonName>
      . . . . .
    </Customer>
  </Profile>
</OTA_CreateProfileRQ>
```

Example 2 - Create response message

```
<OTA_CreateProfileRS EchoToken="12345" TimeStamp="20001217T182259Z"
  <Success />
  <UniqueId
    URL="http://www.basshotels.com/OTAProcessor"
    Type="Profile"
    Id="1234567"/>
</OTA_CreateProfileRS>
```

3.5.2.Read verb

The Read infrastructure verb defines an operation that opens an existing record and transmits information contained in that record. The Read operation enables the user to identify a particular record and retrieve its entire contents. The basic operation has the following steps:

- Requestor queries the database where the record resides by sending a Read request message with the profile's unique identifier
- Receiver returns the record to the requestor

The use of the revised OTA <UniqueId> element allows for a generalized read transaction message. With the object type specified via the *Type* attribute, the action type is identified within a general read request.

Example 3 - OTA_ReadRQ message

```
<OTA_ReadRQ ReqRespVersion="2">
  <UniqueId
    URL="http://vmguys.com/OTAEngine/"
    Type="Profile"
    Id="0507-12345"/>
</OTA_ReadRQ>
```

⁵ We thank Adam Athimuthu for creating the sample Create " " messages.

- *ReqRespVersion* - The optional "Request Response Version" attribute allows the sender to indicate the version desired for the response message. For example, the *OTA_ReadRQ* message sample above indicates a request to return a 2001A OTA Customer Profile.

This request applies to all types of actions in addition to profiles. The type of the generated response depends, of course, on the *Type* specified in the request; for example, the *<OTA_ReadRQ>* shown in the example would generate a *<OTA_ReadProfileRS>* message that contains a 2001A OTA Profile as a response, as in the example below.

This generalization significantly reduces the maintenance burden for individual infrastructure verbs, with effectively zero loss in semantics.

This revision affects the original non-versioned version discovery messages published in the OTA version 1 specification, but with minor adjustments implementors of OTA specifications are able to take advantage of the significant long-term benefits that generalization provides.⁶

Example 4 - Read response message

```
<OTA_ReadProfileRS>
  <UniqueId
    URL="http://vmguys.com/OTAEngine/"
    Type="Profile"
    Id="0507-12345" />
    Instance="20000531172200"
  <Profile>
    ...
  </Profile>
</OTA_ReadProfileRS>
```

An *Instance* value returned in a Read response, if not implemented as a timestamp, may specify the same instance value to all requestors until the record is changed by a subsequent action to the record, such as an update.

3.5.3.Update verb

The Update infrastructure verb defines an operation that opens an existing record, identifies the information that needs changing, then transmits data corresponding to the appropriate elements in the tree, and adds or replaces those data in the record.

Because Update operations are more complex and can affect parts of the record rather than the entire record, handling update messages can generally be more difficult.⁷ As a result, two approaches to updating records are defined in this OTA 2001A specification.

The goals considered in the design of the Update operation include:

- Minimizing the size of a payload on the wire to represent an update transaction

⁶ See Chapter 5 - Errors and Non-Versioned Messages

⁷ We thank David Marshall, VM Systems, Inc. and Michael Guidone, Pegasus Solutions, for drafting the approach to performing the Update process presented here.

- Defining an explicit representation about what has changed
- Defining a representation with a clear and simple conceptual model
- Creating a representation that is content-independent and general-purpose in nature so as to be reusable throughout future OTA specifications
- Providing a simple-to-implement "replace" option to allow developers to get simpler implementations running quickly - at the expense of the first 2 goals (representation of change and size of message) above.

Because data to be modified may be stored in a database and not in an XML document format, it may not be possible to reconstruct the original document that transmitted the data. Therefore, it is RECOMMENDED that implementors utilizing the partial update process perform a Read request to obtain the structure of the XML tree prior to constructing an Update request.

3.5.3.1. Representing change in XML

Representing change is not a problem unique to the OTA. The approach presented in this specification includes current use of the following:

- library utilities that can compare before and after images of a particular XML document and then generate a succinct representation of the differences (conceptually similar to a GNU *diff* utility)
- a representation for differences that is both human readable and understandable by automated tools
- library utilities that can take an XML document, apply a differences document and generate the same after image (conceptually similar to a GNU *patch* utility)

Applying an Update action involves a tree-to-tree comparison, similar to well-known operations outside the XML arena. This specification assumes that prior to sending an <OTA_UpdateRQ> message, the sender has used standard libraries to generate the differences between the 'before' and desired 'after' images of the XML document. Implementors may wish to consult documentation of algorithms for tree-to-tree comparison and correction in computer science publications⁸, to provide the background for understanding this XML-specific approach.

3.5.3.2. Position Representation with XPath

The general concept of an OTA generic update request is to send a <Position> element followed by one or more operations to be applied at that position. XPath is a well-known W3C recommendation for representing a node in an XML document. The two best known applications of XPath are within XSL and Xlink (themselves XML recommendations).

Within the OTA update representation, XPath is used to reference a specific element within a source document. This XPath is encoded within the *XPath* attribute in the <Position> element. In OTA update messages, the *XPath* attribute within any given <Position> will always refer to an element, and this explanation therefore is restricted to that proper subset of XPath notation which refers to elements. The representation of <Position> followed by one or

⁸ For an excellent paper outlining the commonalties and differences between the best known algorithms see: "Tree-to-tree Correction for Document Trees", Technical Report 95-372 by David T. Barnard, Gwen Clarke and Nicholas Duncan (available from Queen's University, Kingston Ontario at: <ftp://ftp.qucis.queensu.ca/pub/reports/1995-372.ps>)

more operations is used, as updates will often consist of more than one operation to be performed at the same position. This representation will often be shorter than the alternative of embedding an XPath position explicitly within each operation.

It is important to note that when processing a differences document, insertion or deletion of nodes may invalidate subsequent XPath references. This is addressed in the section "Order of Representation and Application" (See Section 3.5.6.5).

3.5.3.3. Operands

In keeping with the design goals of minimal representation and explicit representation of change, changes are represented as occurring at four different levels, using the following operands:

- *Attribute* - performs an operation on the attribute *name* of the currently selected Element
- *Element* - performs an operation on the structure (i.e. content or children) of the currently selected element (but not its attributes)
- *Subtree* - allows for the grafting or pruning of an element which has its own children
- *Root* - this special attribute is ONLY used for a simple 'replace' alternative where an updated representation of the entire object is sent as opposed to sending the incremental differences

In the case of <Element> and <Subtree> operands, insert operations are performed on children of the selected Element (selection is made via the <Position> *Xpath* attribute). Children are always specified via the *Child* attribute, which is a zero-relative integer where children positions are specified left-to-right, with position zero indicating the leftmost child.

3.5.3.4. Operations

Operations are specified on an operand via the Operation attribute. Operations are "*insert*", "*modify*" and "*delete*", though the "*modify*" operation is not applicable to the operand <Subtree>. The operand <Root> is the only operand to have the special "*replace*" operation.

The following table describes the result of each operation on its associated operand:

Operand	Operation	Description
Attribute	insert	The attribute <i>name</i> with value <i>value</i> will be added to selected element
Attribute	modify	The value of the attribute <i>name</i> will be changed to <i>value</i> on the selected element
Attribute	delete	The attribute <i>name</i> will be deleted from the selected element
Element	insert	A child element will be inserted beneath the currently selected element. This child will be inserted at zero-relative position <i>Child</i> (see also Subtree insert)
Element	modify	The value of the currently selected element will be modified to the value specified. The text content of an element is removed by using a modify operation with a null replacement value.
Element	delete	The currently selected element is deleted. If this element

		has children of its own, these children will become children of their grandparent in the tree, with their position being equivalent to an insertion at the point previously occupied by the current element. Use Subtree delete to remove an Element and all its children from its currently selected parent
Subtree	insert	The subtree will be inserted beneath the currently selected element. This subtree will be inserted as a child at position <i>child</i> (see also Element insert)
Subtree	delete	The subtree beginning at the currently selected element will be deleted (use Element delete if an element is to be removed, but its children preserved)
Root	replace	A replacement representation of the entire document object follows. This operation allows implementors to avoid the complexity of the difference representation and send a full 'after' image of the updated document object

Although it is possible to replace the entire tree by performing a <Subtree> *delete* operation on the root, it is not possible to then insert a new subtree as a replacement. The <Root> *replace* operation is provided for this purpose.

The syntax for an <OTA_UpdateRQ> is formally defined in the following DTD fragment:

```
<!ELEMENT OTA_UpdateRQ (UniqueId,Position+)>
<!ELEMENT Position ((Attribute*,Element*,Subtree*)|Root)>
<!ATTLIST Position
    XPath CDATA #REQUIRED
>
<!ELEMENT Attribute EMPTY>
<!ATTLIST Attribute
    Operation (insert|modify|delete)#REQUIRED
    Name CDATA #REQUIRED
    Value CDATA #IMPLIED
>
<!ELEMENT Element ANY>
<!ATTLIST Element
    Operation (insert|modify|delete)#REQUIRED
    Child CDATA #IMPLIED
>
<!ELEMENT Subtree ANY>
<!ATTLIST Subtree
    Operation (insert|delete)#REQUIRED
    Child CDATA #IMPLIED
>
<!ELEMENT Root ANY>
<!ATTLIST Root
    Operation (replace)#REQUIRED
>
```

3.5.3.5. Order of Representation and Application

The XPath notation is used to determine the position in an XML document where operations are to be applied. As that position is specified via XPaths, it is entirely possible that application of an operation will invalidate subsequent XPath expressions if care is not taken explicitly in the order of presentation.

An XML document is inherently a tree, and when considering an XML document, the most natural order for presentation is a depth-first pre-order traversal (this is the way in which an XML document is represented in XML notation). Unfortunately, that order is the one that is most likely to invalidate XPaths when applying differences in order.

Therefore, for purposes of representing differences in an update message, positions and operations are presented in an order which favors a post-order traversal⁹. This ensures that differences may be applied sequentially to a source document to transform it into a target without invalidating the XPath of any subsequent unprocessed difference.

A brief illustration will help make this point clearer. Assuming the following abstract source document:

```
<A>
  <B>
    <C/>
    <D>
      <E/>
    </D>
  </B>
  <F>
    <G>
      <H/>
    </G>
    <I>
      <J/>
    </I>
  </F>
</A>
```

The depth-first pre-order traversal presents elements in the following sequence:

<A><C><D><E><F><G><H><I><J>

However, a post-order traversal presents elements in the following alternate sequence:

<C><E><D><H><G><J><I><F><A>

When considering tree-traversals in the context of XML, perhaps the easiest way to think of it is as follows:

- depth-first pre-order traversal presents elements in the order that opening tags occur
- post-order traversal presents elements in the order that element closure occurs (i.e. follow the closing tags)

An additional requirement for order is placed on repeating sequences of elements, as follows:

⁹ It should be noted that the fastest known algorithms for tree-to-tree correction operate on a post-order traversal

When operation(s) are to be performed on more than one element in a repeating sequence the <Position> for those elements in the sequence must be presented from right-to- left, i.e. from the largest index to the smallest.

By presenting and applying differences using a post-order traversal representation (and reverse index order for repeating sequences) operations applied to the current position cannot break the XPath to a subsequent position as later operations are deeper within the tree (closer to the root) down any given branch.

3.5.3.6. Update Examples

Given the difference representation above, the following examples illustrate this technique. These examples illustrate a chain of updates, showing before and after images at each step. All these changes are designed to be incremental, that is, the "after" image from the previous update becomes the "before" image for the next update.

Example 5 - Initial Document "Before" Image

```
<Profile>
  <Customer>
    <PersonName NameType="Default">
      <NameTitle>Mr.</NameTitle>
      <GivenName>George</GivenName>
      <MiddleName>A.</MiddleName>
      <SurName>Smith</SurName>
    </PersonName>
    <TelephoneInfo PhoneTech="Voice" PhoneUse="Work" >
      <Telephone>
        <AreaCityCode>206</AreaCityCode>
        <PhoneNumber>813-8698</PhoneNumber>
      </Telephone>
    </TelephoneInfo>
    <PaymentForm>
      ...
    </PaymentForm>
    <Address>
      <StreetNmbr POBox="4321-01">1200 Yakima St</StreetNmbr>
      <BldgRoom>Suite 800</BldgRoom>
      <CityName>Seattle</CityName>
      <StateProv PostalCode="98108">WA</StateProv>
      <CountryName>USA</CountryName>
    </Address>
  </Customer>
</Profile>
```

Example 6 - Update of AreaCode using <Root Operation="replace"/>

Note that the operation "modify" replaces the PCData within the element. In order to delete the data, the update request is sent with empty content in the element. (A "delete" operation deletes the entire element.)

Example 7 - Update of RelatedTraveler using <Subtree Operation="insert"/>

```

<OTA_UpdateRQ ReqRespVersion="2">
  <UniqueId    URL=" http://vmguys.com/OTAEngine/"
              Type="Profile"
              Id="12345678"
              Instance="8"/>
  <Position XPath="/Profile/Customer">
    <Subtree Operation="insert" Child="4">
      <RelatedTraveler Relation="Child" >
        <PersonName>
          <GivenName>Devin</GivenName>
          <MiddleName>R.</MiddleName>
          <SurName>Smith</SurName>
        </PersonName>
      </RelatedTraveler>
    </Subtree>
    <Subtree Operation="insert" Child="5">
      <RelatedTraveler Relation="Child" >
        <PersonName>
          <GivenName>Amy</GivenName>
          <MiddleName>E.</MiddleName>
          <SurName>Smith</SurName>
        </PersonName>
      </RelatedTraveler>
    </Subtree>
    <Subtree Operation="insert" Child="6">
      <RelatedTraveler Relation="Child" >
        <PersonName>
          <GivenName>Alfred</GivenName>
          <MiddleName>E.</MiddleName>
          <SurName>Newman</SurName>
        </PersonName>
      </RelatedTraveler>
    </Subtree>
  </Position>
</OTA_UpdateRQ>

```

Example 8 - Documents image after <Subtree Operation="insert"/> applied

```

<Profile>
  <Customer>
    <PersonName NameType="Default">
      <NameTitle>Mr.</NameTitle>
      <GivenName>George</GivenName>
      <MiddleName>A.</MiddleName>
      <SurName>Smith</SurName>
    </PersonName>
    <TelephoneInfo PhoneTech="Voice" PhoneUse="Work">
      <Telephone>
        <AreaCityCode>253</AreaCityCode>
        <PhoneNumber>813-8698</PhoneNumber>
      </Telephone>
    </TelephoneInfo>
    <PaymentForm>
      ...
    
```

```

    </PaymentForm>
    <Address>
      <StreetNmbr POBox="4321-01">1200 Yakima St</StreetNmbr>
      <BldgRoom>Suite 800</BldgRoom>
      <CityName>Seattle</CityName>
      <StateProv PostalCode="98108">WA</StateProv>
      <CountryName>USA</CountryName>
    </Address>
    <RelatedTraveler Relation="Child"
      <PersonName>
        <GivenName>Devin</GivenName>
        <MiddleName>R.</MiddleName>
        <SurName>Smith</SurName>
      </PersonName>
    </RelatedTraveler>
    <RelatedTraveler Relation="Child"
      <PersonName>
        <GivenName>Amy</GivenName>
        <MiddleName>E.</MiddleName>
        <SurName>Smith</SurName>
      </PersonName>
    </RelatedTraveler>
    <RelatedTraveler Relation="Child"
      <PersonName>
        <GivenName>Alfred</GivenName>
        <MiddleName>E.</MiddleName>
        <SurName>Newman</SurName>
      </PersonName>
    </RelatedTraveler>
  </Customer>
</Profile>

```

Example 9 - Update of document using <Subtree Operation="delete"/>

This operation will delete the first and third Related Travelers. (Note the required order of the operations):

```

<OTA_UpdateRQ ReqRespVersion="2">
  <UniqueId URL="http://vmguys.com/OTAEngine/"
    Type="Profile"
    Id="12345678"
    Instance="9"/>

  <Position XPath="/Profile/Customer/RelatedTraveler[3]">
    <Subtree Operation="delete"/>
  </Position>
  <Position XPath="/Profile/Customer/RelatedTraveler[1]">
    <Subtree Operation="delete"/>
  </Position>
</OTA_UpdateRQ>

```

Note: Use of the <Delete> element removes a designated element from the tree. Any children of the element removed will move up to the grandparent of the element. If the desired result is the removal of the element and all of its children, the element <Subtree> paired with the operation "delete" should be used, as in the example above.

Example 10 - Document image after <Subtree Operation="delete"/> applied.

```
<Profile>
  <Customer>
    <PersonName NameType="Default">
      <NameTitle>Mr.</NameTitle>
      <GivenName>George</GivenName>
      <MiddleName>A.</MiddleName>
      <SurName>Smith</SurName>
    </PersonName>
    <TelephoneInfo PhoneTech="Voice" PhoneUse="Work" >
      <Telephone>
        <AreaCityCode>253</AreaCityCode>
        <PhoneNumber>813-8698</PhoneNumber>
      </Telephone>
    </TelephoneInfo>
    <PaymentForm>
      ...
    </PaymentForm>
    <AddressInfo>
      <Address>
        <StreetNmbr POBox="4321-01">1200 Yakima St</StreetNmbr>
        <BldgRoom>Suite 800</BldgRoom>
        <CityName>Seattle</CityName>
        <StateProv PostalCode="98108">WA</StateProv>
        <CountryName>USA</CountryName>
      </Address>
    </AddressInfo>
    <RelatedTraveler Relation="Child" >
      <PersonName>
        <GivenName>Amy</GivenName>
        <MiddleName>E.</MiddleName>
        <SurName>Smith</SurName>
      </PersonName>
    </RelatedTraveler>
  </Customer>
</Profile>
```

Example 11 - Update of document using <Attribute Operation="modify"/>

This operation will change the TelephoneInfo PhoneUse attribute.

```
<OTA_UpdateRQ ReqRespVersion="2">
  <UniqueId URL="http://vmguys.com/OTAEngine/"
    Type="Profile"
    Id="12345678"
    Instance="10"/>

  <Position XPath="/Profile/Customer/TelephoneInfo">
    <Attribute Name="PhoneUse" Operation="modify" Value="Home"/>
  </Position>
</OTA_UpdateRQ>
```

Example 12 - Document image after <Attribute Operation="modify"/> applied

```

<Profile>
  <Customer>
    <PersonName NameType="Default">
      <NameTitle>Mr.</NameTitle>
      <GivenName>George</GivenName>
      <MiddleName>A.</MiddleName>
      <SurName>Smith</SurName>
    </PersonName>
    <TelephoneInfo PhoneTech="Voice" PhoneUse="Home">
      <Telephone>
        <AreaCityCode>253</AreaCityCode>
        <PhoneNumber>813-8698</PhoneNumber>
      </Telephone>
    </TelephoneInfo>
    <PaymentForm>
      ...
    </PaymentForm>
    <AddressInfo>
      <Address>
        <StreetNmbr PO_Box="4321-01">1200 Yakima St</StreetNmbr>
        <BldgRoom>Suite 800</BldgRoom>
        <CityName>Seattle</CityName>
        <StateProv PostalCode="98108">WA</StateProv>
        <CountryName>USA</CountryName>
      </Address>
    </AddressInfo>
    <RelatedTraveler Relation="Child">
      <PersonName>
        <GivenName>Amy</GivenName>
        <MiddleName>E.</MiddleName>
        <SurName>Smith</SurName>
      </PersonName>
    </RelatedTraveler>
  </Customer>
</Profile>

```

Example 13 - Update of document using <Attribute Operation="delete"/>

This operation will delete the PO_Box attribute of StreetNmbr.

```

<OTA_UpdaterQ ReqRespVersion="2">
  <UniqueId URL="http://vmguys.com/OTAEngine/"
    Type="Profile"
    Id="12345678"
    Instance="11"/>

  <Position Xpath="/Profile/Customer/AddressInfo/Address/StreetNmbr">
    <Attribute Name="PO_Box" Operation="delete"/>
  </Position>
</OTA_UpdaterQ>

```

Example 14 - Document image after <Attribute Operation="delete"/> applied

```

<Profile>
  <Customer>
    <PersonName NameType="Default">
      <NameTitle>Mr.</NameTitle>
      <GivenName>George</GivenName>
      <MiddleName>A.</MiddleName>
      <SurName>Smith</SurName>
    </PersonName>
    <TelephoneInfo PhoneTech="Voice" PhoneUse="Home">
      <Telephone>
        <AreaCityCode>253</AreaCityCode>
        <PhoneNumber>813-8698</PhoneNumber>
      </Telephone>
    </TelephoneInfo>
    <PaymentForm>
      ...
    </PaymentForm>
    <AddressInfo>
      <Address>
        <StreetNmbr>1200 Yakima St</StreetNmbr>
        <BldgRoom>Suite 800</BldgRoom>
        <CityName>Seattle</CityName>
        <StateProv PostalCode="98108">WA</StateProv>
        <CountryName>USA</CountryName>
      </Address>
    </AddressInfo>
    <RelatedTraveler Relation="Child">
      <PersonName>
        <GivenName>Amy</GivenName>
        <MiddleName>E.</MiddleName>
        <SurName>Smith</SurName>
      </PersonName>
    </RelatedTraveler>
  </Customer>
</Profile>

```

Example 15 - Update of document using <Attribute Operation="insert"/>, <Element Operation="delete"/>, and <Element Operation="insert"/>

These compound operations will insert the Gender attribute on Customer, delete the MiddleName, and insert NameTitle on the Related Traveler:

```

<OTA_UpdateRQ EchoToken="12345" TimeStamp="20001217T182259Z"
ReqRespVersion="2">
  <UniqueId URL="http://vmguys.com/OTAEngine/"
    Type="Profile"
    Id="12345678"
    Instance="12"/>

  <Position XPath="/Profile/Customer/PersonName/MiddleName">
    <Element Operation="delete"/>
  </Position>
  <Position
XPath="/Profile/Customer/RelatedTraveler/RelatedPerson/PersonName">
    <Element Operation="insert">

```

```

        <NameTitle>Ms.</NameTitle>
    </Element>
</Position>
<Position XPath="/Profile/Customer">
    <Attribute Name="Gender" Operation="insert" Value="Male"/>
</Position>
</OTA_UpdateRQ>

```

Example 16 - Document image after <Attribute Operation="insert"/>, <Element Operation="delete"/>, and < Element Operation="insert"/> applied

```

<Profile>
  <Customer Gender="Male">
    <PersonName NameType="Default">
      <NameTitle>Mr.</NameTitle>
      <GivenName>George</GivenName>
      <SurName>Smith</SurName>
    </PersonName>
    <TelephoneInfo PhoneUse="Home">
      <Telephone PhoneTech="Voice" >
        <AreaCityCode>253</AreaCityCode>
        <PhoneNumber>813-8698</PhoneNumber>
      </Telephone>
    </TelephoneInfo>
    <PaymentForm>
      ...
    </PaymentForm>
    <Address>
      <StreetNmbr>1200 Yakima St</StreetNmbr>
      <BldgRoom>Suite 800</BldgRoom>
      <CityName>Seattle</CityName>
      <StateProv PostalCode="98108">WA</StateProv>
      <CountryName>USA</CountryName>
    </Address>
    <RelatedTraveler Relation="Child">
      <PersonName>
        <NameTitle>Ms.</NameTitle>
        <GivenName>Amy</GivenName>
        <MiddleName>E.</MiddleName>
        <SurName>Smith</SurName>
      </PersonName>
    </RelatedTraveler>
  </Customer>
</Profile>

```

3.5.3.7. Validation of Update Messages

The update request message is a valid message within its own structure, but since it identifies only a portion of the content of an XML tree, it cannot be validated in the context of the business schema. Validation at the level of the business schema must occur after the update has been completed. Implementors may wish to validate the image of the document before changes have been applied and again after the changes have been applied in order to ascertain that the desired result has been obtained and is valid within the business context.

3.5.4. The Simple "Replace" verb

The Replace infrastructure verb defines an operation that updates an existing record by replacing the existing information with a complete overlay image of the document as it would appear after changes are applied. The Replace verb does not require a Read request to obtain an image of the current record prior to sending the message to replace it with the information being transmitted, although this may be desirable from the standpoint of good business practice.

The "replace" verb allows implementors greater flexibility in choosing how they wish to perform updates of information, since difference representation may be bypassed and a simple replacement image of the document object to be updated is sent. This reduces the complexity of handling updates for some implementations, but at the expense of size of the messages.

Example 17 - The following XML document fragment illustrates the "replace" type update:

```
<OTA_UpdaterQ ReqRespVersion="2">
  <UniqueId    URL="http://vmguys.com/OTAEngine/"
              Type="Profile"
              Id="12345678"
              Instance="7"/>
  <Position XPath="/Profile">
    <Root Operation="replace">
      <Profile>
        ...
        <!-- include entire updated 'after' image here -->
        ...
      </Profile>
    </Root>
  </Position>
</OTA_UpdaterQ>
```

3.5.5. The Delete verb

The Delete infrastructure verb defines an operation that identifies an existing record, and removes the entire record from the database. The use of the Delete verb depends upon the business rules of an organization. Alternative strategies, such as mapping a duplicate record to another by use of the UniqueId, may be considered.

The requestor MAY also verify the record before deleting it to ensure the correct record has been identified prior to deleting it. In this case, the use of the *Instance* attribute may also be useful in determining whether the record has been updated more recently than the information that is intended to be deleted. That choice, again, would be dictated by good business practices.

Steps in the Delete operation include:

- Requestor submits a Read request to view the record
- Receiver returns the record for the requestor to view
- Requestor submits a Delete request.
- Receiver removes the record and returns an acknowledgement

Example 18 - This example illustrates the Delete process, Read request

```
<?xml version="1.0" encoding="UTF-8"?>
<OTA_ReadRQ ReqRespVersion="2">
  <UniqueId>
    URL=http://vmguys.com/OTAEngine/
    Type="Profile"
    Id="0507-12345"
  </UniqueId>
</OTA_ReadRQ>
```

Example 19 - This example illustrates the Delete process, Read response

```
<?xml version="1.0" encoding="UTF-8"?>
<OTA_ReadProfileRS>
  <UniqueId>
    URL="http://vmguys.com/OTAEngine/"
    Type="Profile"
    Id="0507-12345"
    Instance="2"
  </UniqueId>
  <Profile>
    ...
  </Profile>
</OTA_ReadProfileRS>
```

Example 20 - This example illustrates the Delete request

```
<?xml version="1.0" encoding="UTF-8"?>
<OTA_DeleteRQ ReqRespVersion="2">
  <UniqueId>
    URL="http://vmguys.com/OTAEngine/"
    Type="Profile"
    Id="0507-12345"
    Instance="2"
  </UniqueId>
</OTA_DeleteRQ>
```

Example 21 - This example illustrates the Delete response

```
<?xml version="1.0" encoding="UTF-8"?>
<OTA_Deleters>
  <UniqueId>
    URL="http://vmguys.org/OTAEngine/"
    Type="Profile"
    Id="0507-12345"
    Instance="2"/>
  </UniqueId>
  <Success/>
</OTA_Deleters>
```

3.6. XML Extensions

Trading partners using the minimum functionality defined in OTA specifications, may have a need to include proprietary data within messages exchanged between them. By defining a specific element, <TPA_Extensions> of the data type, ANY, at sanctioned plug-in points for

bilaterally-agreed trading partner extensions, the OTA 2001A specification allows for extensibility within conforming implementations.

Trading partner extensions are formally noted by the use of the definition:

```
<!ELEMENT TPA_Extensions ANY>
```

Specific locations have been designated in logical places throughout OTA specifications for trading partners to add their own data. The OTA 2001A specification identifies extension points where the trading partners may add their own extensions within the framework of valid messages.

The customer profile element is redefined to include this extension as follows:

```
<!ELEMENT Profile (Accesses?, Customer, Preferences?,  
Affiliation*, TPA_Extensions? )>
```

This same element is found at specific locations throughout the DTD/Schema. This reusability avoids namespace clutter and retains conformance with the OTA 2001A specification.

Trading partners will need to write their own schemas to cover items not included in the 2001A specification. Trading partners are encouraged to submit extensions widely used between multiple trading partners for consideration to become part of future revisions of OTA specifications.

4. Message Structure

4.1. Header and Payload Documents

OTA endorses a modified version of the ebXML structure of a Header Document inside the ebXML Message Envelope (MIME multipart/related) wrapper. (See section 3.1 for a discussion of the use of MIME for an ebXML/OTA message envelope).

4.2. ebXML Header Document

[from Message Service Specification v0.8.doc]

The ebXML Header Envelope is the wrapper for an ebXMLHeader document that consists of two parts:

- **Manifest** - a single section that contains a list of references to the other parts of the message.
- **Header** - contains the information required by the recipient to process the message

Manifest - The OTA 2001A message structure consists of ONE Manifest section. The Manifest SHALL be the first element contained in the ebXMLHeader Envelope. It identifies the payload document(s) contained in the Payload Container. The purpose of the Manifest is to make it easier to directly extract a particular document associated with the Message.

Header - OTA has chosen to ignore the Header portion of the ebXMLHeader and use the OTA Control section to define the trading partner information required for connectivity, which includes the identification of synchronous or asynchronous messaging, and a capability to indicate distribution of messages to third parties.

4.2.1. Manifest element

[from Message Service Specification v0.8.doc]

The REQUIRED Manifest element consists of zero or more DocumentReference elements. If the Manifest is an empty XML element, a payload MUST NOT be present.

4.2.2. DocumentReference element

Each DocumentReference element identifies data associated with the message, whether included as part of the message, or remote resources accessible via a URL. The Document Reference is represented as an XLink simple link, an outbound link with exactly two participating resources.¹⁰ The use of the XLink namespace and its element and attribute semantics provides a more precise definition of the Manifest element.

The DocumentReference element is a composite element, and in the format adopted for OTA's use in 2001A messaging, contains six attributes and one subordinate element as follows:

- ***xlink:type*** = "simple" Denotes the document reference type is a simple XLink and has a fixed **value**.
- ***xlink:href*** = Identifies the URL of the Content-ID in the MIME header of the message, as defined in [RFC2392]. *XLink:href* is a REQUIRED attribute.
- ***xlink:role*** = a URI reference to the request message type for a specific business process. For OTA, this identifies the action verb that is the root element of the payload document. *XLink:role* is a REQUIRED attribute.

¹⁰ For complete explanation of XLinks, see <http://www.w3.org/TR/xlink>

- **sequence** = A one-relative sequence that defines the order of processing to be taken on a series of DocumentReference elements listed in the Manifest. *Sequence* is an optional attribute.
- **docref** = An optional attribute of XML data type ID used to cross-reference related pairs of documents to one another. The value of the *docref* is identical to the *xlink:href* attribute as it identifies the document within the DocumentReference.
- **docxref** = An optional attribute of XML data type IDREF used to cross-reference a related document to this document. The value of the *docxref* is identical to the *xlink:href* attribute of the related DocumentReference.

4.2.2.1. Schema sub-element

The Schema element is a REQUIRED child element of DocumentReference that identifies the version and location of the validating schema of the referenced payload document.

- **version** = The *version* attribute indicates the version of the payload message identified in this DocumentReference. *Version* is a REQUIRED attribute. This attribute allows the document's schema or DTD to be reflected externally to the referenced document and enables trading partners to determine whether they support the referenced version before attempting to process the document. (See Chapter 4, Section 4.3.1, for a description of the versioning scheme of OTA specifications.)
- **location** = The *location* attribute identifies the location of the validating DTD or schema and allows the software agent processing the message to access it externally to the payload document.

The DTD fragment for the Manifest element adopted from the ebXML Header is as follows:

```
<ELEMENT Manifest (DocumentReference*)>
<!ATTLIST Manifest
  xmlns:xlink CDATA #FIXED "http://www.w3.org/1999/xlink"
>
<ELEMENT DocumentReference (Schema)>
<!ATTLIST DocumentReference
  xlink:type CDATA #FIXED "simple"
  xlink:href CDATA #REQUIRED
  xlink:role CDATA #REQUIRED
  sequence CDATA #IMPLIED
  docref ID #IMPLIED
  docxref IDREF #IMPLIED
  a-dtype NMTOKENS "sequence int"
>
<ELEMENT Schema EMPTY>
<!ATTLIST Schema
  version CDATA #REQUIRED
  location CDATA #IMPLIED
>
```

4.3. ebXML Payload Envelope

If the OTA message contains a payload, OTA recommends the use of a single payload container to envelop it. OTA has adopted the ebXML Payload Envelope and follows these rules: If there is no payload within the OTA message then the payload container **MUST** not be present. The

contents of the payload container **MUST** be identified by the message Manifest element within the ebXML Header envelope (see Chapter 4, Section 4.2 above).

If the message Manifest is an empty XML element, the ebXML Payload Container **MUST NOT** be present in the Message. If an ebXML Payload Container is present, it **MUST** conform to MIME [RFC2045].

When one and only one payload document is present in the message (as in the case of OTA non-versioned messages) the payload document is preceded by a standard set of MIME headers (Content-ID, Content-Length and Content-Type), as defined in the ebXML Transport, Routing and Packing (TR&P) specification.

When two or more payload documents are present in the message all payload documents **MUST** be enveloped within a single MIME multipart/related container, each payload document representing a separate and distinct MIME part within that container. Each of these separate MIME parts shall have a standard set of MIME headers (Content-ID, Content-Length and Content-Type) whose definition and usage shall be as per the ebXML TR&P specification.

The use of a MIME envelope means that applications can identify the associated MIME type of each payload document. Since multiple documents are allowed within the payload envelope, there can be a mixture of encrypted and non-encrypted content, which are identified in the MIME header of each document.

The ebXML Payload Envelope is the wrapper for OTA documents and **MUST** contain at least two parts for all OTA versioned messages:

- **Control** - a single document that contains the elements needed to identify and authenticate the parties, and used for security of the connection, identifying whether the connection is to be maintained synchronously or asynchronously.
- **Payload Document(s)** - contains the travel content information required by the recipient to perform the operation requested and process the message. OTA payload documents begin with the Action Verb as the root element of the document.

By convention, within the <Manifest>, the Control document <DocumentReference> shall appear before any payload document <DocumentReference>s.

4.4. Control document

The OTA <Control> element contains the information required by the recipient to process the message, and is used in OTA 2001A to replace the Header element in the ebXML structure, supplying a static message structure that trading partners implement while allowing the ebXML TPA Info structure to evolve.

The Control document **MUST** begin with the tag <Control> and end with the tag </Control>.

The Control document contains one mandatory element (a choice of Session or SendBy), as represented in the following DTD fragment:

```
<!ELEMENT Control ((Session | SendBy))>
<!ATTLIST Control
    %OTA_PayloadStdAttributes; >
```

The <Control> document MAY contain the following (optional) payload standard attributes as defined by the entity %PayloadStdAttributes;

- **EchoToken** - The EchoToken attribute is a sequence number [or other unique identifying string] for additional message identification assigned by the host system. When a request message includes an EchoToken, the corresponding response message MUST include an EchoToken with an identical value.
 - **Timestamp** - A Timestamp attribute indicates the creation date and time of the message using the following format specified by ISO 8601: YYYY-MM-DDThh:mm:ss with time values using the 24-hour (military) clock. It is RECOMMENDED that the timestamp attribute be represented in UTC (Universal Coordinated Time).
Example: 29 May 2001, 2:45:30 p.m. UTC, becomes 2001-05-29T14:45:30Z
- Note:** Universal Time (UTC) notation has been chosen to take into consideration time zone anomalies, such as Arizona and Indiana that do not observe daylight savings time and a few countries have created time zones all their own (Iran, Liberia). The receiving system is tasked to convert the Timestamp into local time.
- **Target** - The Target attribute indicates if the message is a test or production message, with the default value of Production. Valid values: (Test | Production)
 - **Version** - The version of the Control document indicated by an integer value. The default value for this version of OTA Infrastructure is "2".
 - **SequenceNmbr** - The sequence number of the transaction as assigned by the sending system. Allows for an application to process messages in a certain order, or to request a resynchronization of messages in the event that a system has been offline and needs to retrieve messages that were missed.

The following is an example of the Control root tag with the following optional attributes:

```
<Control EchoToken = "4321" Timestamp = "2000-05-22T11:23:09"
Target = "Production" Version = "2" SequenceNmbr = "1234567">
```

4.5. Session vs. SendBy elements

This OTA specification defines two security models: Session-based, and Non-Session-based. The Control document will contain the elements needed to identify and authenticate the parties, and MUST contain one of the following elements:

- **Session** - Indicates a session-based, or synchronous, transaction that allow for both single and multiple system log-ons over a period of time. At the tradeoff of increased complexity with the OTA message structure and message protocols, Session-based security enables an OTA trading partner to provide single-log-in functionality to clients across multiple systems within an enterprise.
- **SendBy** - Indicates a non-session-based, or asynchronous, transaction that allow only for single-system log-ons as an integrated part of an OTA message. SendBy-based security does not require the caller to hold any session information.

Additionally, the Control element MAY contain any of the following four elements:

- **Credential** - A Credential element is a security-related construct that may include information such as a password, or public key certificate. A Credential element may be used with either Session-based or SendBy messages.
- **Principal** - A Principal element is only used with Session-based security model. It is a construct that represents some identity, and a Principal **MUST** have an Identity attribute associated with it. A Principal **MAY** also have associated Credentials, and a URI.
- **ReplyTo** - The ReplyTo element is not repeating, and must be filled out in order to have permission to push a message back to the requesting party or to push a message to a 3rd party. The ReplyTo element contains one mandatory attribute, OrigBodyReq, an abbreviation for "Original Body Requested", that indicates the body of the original request message should be included in the message pushed to the party of the ReplyTo. The following is an example:

```
<ReplyTo OrigBodyReq = "No"/>
```
- **CC_To** - The CC_To element **MAY** be repeating, and indicates a single third party listener or multiple additional parties to whom the response message should be pushed. The CC_To element contains one mandatory attribute, OrigBodyReq, an abbreviation for Original Body Requested, that indicates the body of the original request message should be included in the message pushed to the parties that are copied on the reply message, as a means to identify the context of the original message. The following is an example:

```
<CC_To OrigBodyReq = "Yes"/>
```

Figure 4 shows the relationship of the Control element to the Session OR SendBy elements.

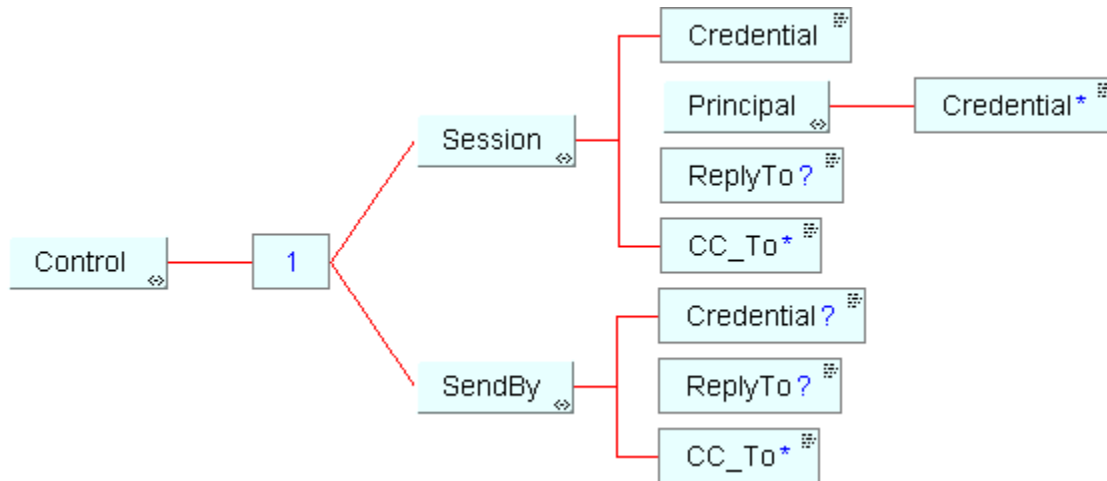


Figure 4. Relationship of Control section to Session and SendBy elements

4.6. Session-based Security model

Sessioned, or synchronous messages can be used for both single log-on, and for multiple log-ons over a period of time and, as a result, will have different authentication requirements from messages using the OTA Non-Session-based (SendBy, or asynchronous) security model.

A Session **MUST** have an Identity attribute. In fact, this is all that is required of an OTA Session.

An example of a minimum OTA Session is as follows:

```
<Control>
  <Session Identity="Tim3"/>
</Control>
```

An OTA Session typically also contains structures called “Principals” and “Credentials”. A Credential is a security-related construct that may include information such as a password, or public key certificate, with an optional Method attribute that identifies the type of encryption used for the Credential. A Session MAY contain multiple Credentials, but typically contains only one.

Further, a Session can contain a URI attribute, which essentially can be used by the authenticating trading partner for anything needed. The following example shows these constructs:

```
<Control>
  <Session Identity="Tim3" URI="122999_1156AM_5555-Our Web Portal">
    <Credential Method="Base64">676868777</Credential>
    <Credential Method="MD32">44422</Credential>
  </Session>
</Control>
```

A Principal is a construct that represents some identity, potentially associated Credentials, and a URI. A Principal MUST have an Identity associated with it. Session-based security allows for an OTA trading partner to authenticate a client against one or more authenticating systems, and return a Session to the Subject (client) that represents the authentication results. The Session structure enables this in two ways. Authenticating trading partners may simply imbed everything needed by back-end systems into the Session’s Credential(s), or by using multiple Principals within a Session, which may be useful to avoid Credential parsing upon every authenticated request.

An authenticating trading partner using Principals, will then generate, along with the described constructs above, Principals within a generated Session that represent authentication information for various (back-end) systems. In this case, the Identity and Credential(s) directly related to the Session would typically be used to identify the authenticated Subject.

Full authentication on several (back-end) systems, or even one system, may be a lengthy process to endure on every request (as prescribed by the non-Session-based, SendBy approach). Session-based authentication using multiple Principals in a Subject’s (client’s) Session facilitates implementing both a single login, and performance oriented designs where trading partner servers may optimize on holding or pooling system resources for authenticated clients.

A Principal MAY have associated Credentials, and MAY have an associated URI. The following example shows these constructs:

```
<Control>
  <Session Identity="Tim3" URI="Web Portal primary contact system">
    <Credential Method="Base64">676868777</Credential>
    <Credential Method="MD32">44422</Credential>
    <Principal Identity="TimC" URI="FFProgram1">
      <Credential Method="RAS32">FF569yt</Credential>
    </Principal>
    <Principal Identity="67ttt" URI="Vacation Program">
      <Credential Method="Clear Text">mypassword</Credential>
    </Principal>
  </Session>
</Control>
```

```

    </Principal>
    <ReplyTo OrigBodyReq="No">https://vmguys.com/OTAEngine/</ReplyTo>
    <CC_To/>
  </Session>
</Control>

```

It should be noted that Session-based security provides the ability to function without the optional Method attribute as shown in the above example, which is used to identify the encryption used for a specific Credential. OTA trading partners supporting Session-based security have the ability to keep this knowledge private, held within its process and not returned to the client, providing increased security. Because a definitive set of credentialing methods is not available, however, OTA does not include a specific enumerated list of methods.

4.7. OTA Payload

Most OTA 2001A messages will have one Manifest, one Control and one or more payload document sections. The payload MUST consist of an Action Verb as the root element, OR a 2001A Success, Warnings or Errors collection as defined in Chapter 5. The OTA Action Verb, or operation element is identified in the Document Reference element of the Manifest.

With the exception of OTA non-versioned messages (which do not have a Control document) all payloads (Control and actionable document(s)) must be enveloped within a multipart/related MIME envelope called the payload envelope.

Each payload document within the payload envelope is preceded by MIME headers that contain the following information:

- **Content-ID** - The Content-ID MIME Header is used to uniquely identify an instance of a document in the payload. The value of the Content-ID is the same as the *xlink:href* attribute inside the Manifest of the ebXML Header, and points to a URI. The value for Content-ID SHOULD be a unique identifier, in accordance with MIME [RFC 2045]. For example:

```
Content-ID: <2000-0722-161201-123456789@ebxmlhost.realm>
```

- **Content-Length** - The MIME Content-Length header is a decimal value used to identify the total number of OCTETS contained in the content portion of the Payload Container. The Content-Length appears at the highest level of the payload container only, and is not repeated for each document in the payload. For example:

```
Content-Length: 5012
```

- **Content-Type** - The MIME Content-Type for an ebXML payload is determined by the implementor and is used to identify the type of data contained in the content portion of the ebXML Payload Container. The MIME Content-Type must conform to [RFC2045]. For example:

```
Content-Type: application/xml
```

4.7.1. Authentication

Subjects (clients) must first attempt to become authenticated when using the Session-based security model. Subjects become authenticated by sending an OTA_AuthenticationRQ to a trading partner. As the Authentication request is a non-versioned message, no Control document is required in the message. An OTA_AuthenticationRQ is sent as a payload document. A single

Principal is provided within the request.

In the following example, the Principal indicates an ID of “Tim3”. The Credential is being passed with Clear Text encryption (in the clear), where the Subject uses some form of transport-level security (such as HTTPS).

```
<?xml version="1.0"?>
<!DOCTYPE AuthenticationRQ SYSTEM "AuthenticationRQ.dtd">
<AuthenticationRQ>
  <Principal Identity="Tim3">
    <Credential Method="Clear Text">SPOT
  </Credential>
  </Principal>
</AuthenticationRQ>
```

Given no errors, the OTA trading partner returns an AuthenticationRS message with a Success entity. A Session is returned in a Control document. In the example below, the OTA trading partner does not use any additional Principals with the returned Session (explained above). Again note that this does not necessarily mean that the authenticating trading partner does not authenticate against any back-end systems.

```
<?xml version="1.0"?>
<!DOCTYPE Control SYSTEM "Control.dtd">
<Control>
  <Session Identity="Tim3">
    <Credential Method="Base64">6785533</Credential>
  </Session>
</Control>

<?xml version="1.0"?>
<!DOCTYPE AuthenticationRS SYSTEM "AuthenticationRS.dtd">

<AuthenticationRS>
  <Success/>
</AuthenticationRS>
```

If the OTA trading partner does not support the Session-based security model a StandardError element is returned containing the “AuthenticationModel” ErrType attribute as follows:

```
<AuthenticationRS>
  <StandardError ErrType="AuthenticationModel"/>
</AuthenticationRS>
```

If the Subject’s provided Principal fails authentication, a StandardError element is returned containing the “Authentication” ErrType attribute as follows:

```
<AuthenticationRS>
  <StandardError ErrType="Authentication"/>
</AuthenticationRS>
```

4.7.1.1. Session-based Authentication

In OTA security, the word "Subject" is used to represent a client, or "caller" host system.

Once a Subject has become authenticated, the Session is passed, unaltered, to the target OTA trading partner on every business message.

The following examples show various Session structures that could be passed on business requests.

A. Session-based security with Session-level Identity, URI, and Credential with two Principals.

In this scenario, the Subject has previously become authenticated, which means it has obtained a Session. The Subject passes the Session along with a non-encrypted Content in an OTA message. The trading partner uses the Identity and URI attributes, along with the Credential element of the Session for its own purposes to be used for processing of subsequent message invocations from the authenticated Subject. Note that one of the Principals (generated by the authenticating Trading Partner) does not have a specified URI. Another Principal (also generated by the authenticating Trading Partner) is used to support a backend frequent flyer program and does have a specified URI.

```
<Control>
  <Session Identity="XYZ Co" URI="http://OTA.XYZ.COM">
    <Credential Method="RC4-SHA">foiu34iuhioaugho34uhgip
  </Credential>
  <Principal Identity="Scott9">
    <Credential Method="RC4SHA">3fff678sddd850</Credential>
  </Principal>
  <Principal Identity="SRH78" URI="FrequentFlyerProgram1">
    <Credential>777747777sd88888</Credential>
  </Principal>
</Session>
</Control>
```

B. Session with Session-level Identity, one Principal, and EncryptedContent

Here, the Subject has previously become authenticated, meaning it has obtained a Session. The Subject passes the Session along with an encrypted document in the payload of an OTA message. The trading partner has used only the Identity of the Session for its own purposes to be used for processing of subsequent message invocations from the authenticated Subject. Note that the Principal (generated by the authenticating trading partner) does not have a specified URI.

```
<Control>
  <Session Identity="XYZ Co">
    <Principal Identity="Scott9">
      <Credential Method="RC4SHA">3fff678sddd850</Credential>
    </Principal>
  </Session>
</Control>
```


4.7.1.2. Authentication Timeout and Re-Authentication

All Sessions eventually time out (exceed a specified period of time with no activity) in the authenticating trading partner. An OTA service that supports Session-based security MAY return a standard error from any given business request indicating that the current Session has timed out, as in the following example:

```
<?xml version="1.0" encoding="UTF-8"?>
<StandardError ErrType="AuthenticationTimeout"/>
```

In this case, the Subject can attempt the authentication sequence as before by utilizing the AuthenticationRQ message as described earlier. As an alternative, the ReAuthenticationRQ message structure MAY be used. The ReAuthenticationRQ structure may be more suited for implementations that do not hold the original authenticating Principal information, or would like to avoid relying on transport-level encryption for reauthentication.

```
<?xml version="1.0"?>
<!DOCTYPE Control SYSTEM "Control.dtd">

<Control>
  <Session Identity="Tim3">
    <Credential Method="Base64">6785533</Credential>
  </Session>
</Control>
```

```
<?xml version="1.0"?>
<!DOCTYPE ReAuthenticationRQ SYSTEM "ReAuthenticationRQ.dtd">

<ReAuthenticationRQ/>
```

The returned successful results of the ReAuthenticationRQ are the same as that of the AuthenticationRQ message except for the response using the ReAuthenticationRS construct. Subjects SHOULD use the new, reauthenticated Session in following business messages.

If the OTA trading partner does not support the Session-based security model a StandardError SHOULD be returned containing the "AuthenticationModel" ErrType attribute:

```
<AuthenticationRS/>
  </StandardError ErrType="AuthenticationModel"/>
</AuthenticationRS>
```

If the Subject's provided Session fails authentication, a StandardError is returned containing the "Authentication" ErrType attribute:

```
<AuthenticationRS/>
  </StandardError ErrType="Authentication"/>
</AuthenticationRS>
```

4.7.1.3. Unauthentication

All Sessions eventually time out in the authenticating trading partner. However, Subjects should use the UnAuthenticationRQ message to logout or become unauthenticated when authentication to a trading partner is no longer required. To unauthenticate, the Session is passed in within the Control document.

```
<?xml version="1.0"?>
<!DOCTYPE Control SYSTEM "Control.dtd">

<Control>
  <Session Identity="Tim3">
    <Credential Method="Base64">6785533</Credential>
  </Session>
</Control>
```

```
<?xml version="1.0"?>
<!DOCTYPE UnAuthenticationRQ SYSTEM "UnAuthenticationRQ.dtd">

<UnAuthenticationRQ/>
```

If the OTA trading partner does not support the Session-based security model a StandardError is returned containing the “AuthenticationModel” ErrType attribute.

If the Subject’s provided Session has never been authenticated from the target trading partner, a StandardError is returned containing the “Authentication” ErrType attribute.

4.8. The Non-session-based (SendBy) Security Model

In the Non-Session-based, asynchronous security model, the SendBy element identifies the sender of the message and MAY include a Credential element. The Subject (client) generates the SendBy construct. However, the SendBy element SHOULD have an Identity attribute and MAY have a URI attribute. The Credential element SHOULD have a Method attribute.

If the OTA trading partner supports only the Session-based security model and NOT the SendBy method discussed above, a StandardError is returned containing the “AuthenticationModel” ErrType attribute.

Given below are examples for several scenarios with different combinations of SendBy sessions and single log-ons as well as credentials and identities.

A) Non-sessioned single log-in transaction with Identity, URI, and Credential in the Control document

B) Single log-in transaction with Identity, and URI

```
<Control>
  <SendBy Identity="XYZ Co" URI="http://OTA.XYZ.COM" />
</Control>
```

C) Non-sessioned single log-in transaction with Identity only

Here the sender, indicated by SendBy, sends only the Identity without any XML-based authentication including URI or Credential.

```
<Control>  
  <SendBy Identity="XYZ Co" />  
</Control>
```

4.9. Use of Encryption

The use of specific security-related elements and attributes in the Control document depend on the use of encryption in the payload.

If the trading partners decide to encrypt the payload document of the message, the encrypted content by definition will have scrambled content and therefore will NOT follow the content model of the validating schema or DTD. The receiver of the message will need to validate the contents after decryption.

The use of the ebXML MIME wrapper provides the identification security of the encrypted content and the method used for encryption. Currently, there are two proposals that may be adopted:

- the use of SMIME with a signature for the encryption of the payload
- The use of XML digital signatures for encrypting the header, but not encrypting the payload

Because the determination of the method for providing security and encryption in XML documents has not been finalized by ebXML¹¹, the OpenTravel Alliance does not officially define how encryption should take place and recommends that trading partners negotiate this level of security between them in the interim.

¹¹ As of the writing of this specification, the ebXML TR&P team are actively working on the definition of security infrastructure, including methods to digitally sign and encrypt payloads

5. Error and Non-Versioned Messages¹²

Some of the exchanges between trading partners will contain messages for administrative rather than business purposes, specifically for the discovery of versions supported by the trading partners and for transmitting errors that occur at the parser. Because this set of messages transcends individual versions, OTA has established a separate category and syntax for these messages that operate independently of the versioned content. Because the messages do not have a version they must be structurally stable yet have operational flexibility.

5.1. Non-Versioned Base Messages

The OTA XML infrastructure defines several messages that are not versioned and are fixed over time by definition. All OTA XML compliant business systems **MUST** support all of the non-versioned OTA messages. The infrastructure defines separate DTDs for the non-versioned OTA payload messages (OTA_NVent.mod, OTA_StdErr.dtd, OTA_PayloadRQ.dtd, and OTA_PayloadRS.dtd)¹³.

Note: A .mod file is an external parameter entity reference that is to be included in a DTD, but cannot necessarily stand alone. When incorporated in a DTD, the entirety of the external entity must be included.

5.2. Non-versioned message DTDs

Non-versioned messages **MUST** use document type definitions (DTDs) separate from the DTDs used for versioned messages. Volume 2 - Appendices, lists the non-versioned DTDs specified in this section.

These DTDs are designed to be used individually (not combined), and as such the element and attribute names (or symbols) need only be unique within these DTD and the non-versioned OTA messages. Unlike versioned OTA messages, non-versioned OTA messages do not rely on a <Control> payload. As such, each of these payloads stands alone. However, the non-versioned OTA messages are anticipated to be encapsulated within an ebXML message envelope.

5.3. Non-versioned Standard Payload Attributes

Each OTA defined base payload will have three standard attributes defined on the root element. These attributes are the same standard attributes defined on the root element of other 2001A specific payloads, however as baseline payloads are versionless these attributes are defined via a separate and distinct entity to allow baseline payloads to remain unchanged should other payloads change in future versions. These standard attributes are defined via the following external entity declaration:

```
<!-- Copyright (C) 2000 Open Travel Alliance -->
<!-- Non-versioned messages, standard parameter entities OTA_nvent.mod
-->
<!ENTITY % OTA_PayloadStdAttributes
EchoToken CDATA #IMPLIED
```

¹² We thank David Marshall, VM Systems, Inc., and George Smith, ConXtra, for developing this part of the specification.

¹³ Previously, the v1 non-versioned message DTD was published under the name "OTA_Nv.dtd". As OTA is changing the Non-Versioned DTD, the name is also being changed for reasons of clarity as well as structurally requiring multiple DTDs.

```

TimeStamp CDATA #IMPLIED
Target (Test | Production) 'Production'
Version CDATA #IMPLIED
SequenceNmbr CDATA #IMPLIED

```

The meaning and usage of each of these standard attributes is as follows:

- *EchoToken*: a sequence number for additional message identification assigned by the requesting host system. When a request message includes an *EchoToken*, the corresponding response message MUST include an *EchoToken* with an identical value.
- *TimeStamp*: indicates the creation date and time of the message in UTC using the following format specified by ISO 8601: YYYY-MM-DDThh:mm:ssZ with time values using the 24-hour (military) clock. e.g.: 20 November 2000, 1:59:38pm UTC becomes 2000-11-20T13:59:38Z
- *Target*: indicates if the message is a test or production message, with a default value of Production. Valid values: (Test | Production)
- *Version*: This optional attribute is not used in OTA non-versioned messages. For all OTA versioned messages, the version of the message indicated by an integer value.
- *SequenceNmbr* - This optional attribute is used to identify the sequence number of the transaction as assigned by the sending system. Allows for an application to process messages in a certain order, or to request a resynchronization of messages in the event that a system has been offline and needs to retrieve messages that were missed.

5.4. StandardError

The OTA XML infrastructure defines a non-versioned standard error message. The non-versioned error messages are designed to accommodate errors that result from the parser, or from validation, before reaching the server. The set of errors that can use this non-versioned standard error (for operational flexibility) is constrained by its limited structure (for structural stability).

```

<!-- Copyright (C) 2000 OpenTravel Alliance-->
<!-- Non-versioned messages, standard error OTA_stderr.dtd -->

<!ENTITY % OTA_NVent SYSTEM "OTA_nvent.mod">
%OTA_NVent;
<!ELEMENT StandardError (#PCDATA)>
<!-- ATTLIST StandardError %OTA_PayloadStdAttributes;
Status (NotProcessed | Incomplete | Complete | Unknown) 'NotProcessed'
Type CDATA #REQUIRED
DocURL CDATA #IMPLIED
-->

```

The non-versioned standard error consists of three attributes (in addition to standard payload attributes) and optional additional information represented as parsed character data (PCDATA) returned by the trading partner's system. A StandardError element MUST contain a *Type* attribute. Attribute definitions are as follows:

Status - If the Status attribute is not present the default meaning SHOULD be "NotProcessed".

If the Status attribute is present, it SHOULD return one of the following responses:

- *NotProcessed*: The error occurred prior to processing of the request (or if during the processing of the request, before the intentions of the request were completed). This is an atomic failure.
- *Incomplete*: The error occurred during processing of the request. The request was partially completed. This is NOT an atomic failure.
- *Complete*: The error occurred after successful processing of the request. The requested process was completed.
- *Unknown*: The status of the request is unknown. The atomicity of the failure is also unknown.

Type - The *Type* attribute MUST be present and is defined as CDATA¹⁴ with a list of enumerated values recommended to indicate the error type. The validating DTD's can expect to accept values that have NOT been explicitly coded for and process them by already accepting a StandardError Type of 'Unknown'. Examples of the types of error that may generate an 'Unknown' response could be a character set that is not recognized, or a payload document that doesn't match the DocumentReference in the Manifest, etc.

The initial list of recommended values MUST contain:

- *Unknown*: indicates an unknown error. Additional information may be provided within the PCDATA.
- *Malformed*: indicates that the XML message was not well-formed. Additional information may be provided within the PCDATA.
- *Validation*: indicates that a well-formed XML message was sent, but did not pass the validation check. Additional information may be provided within PCDATA.
- *UnrecognizedRoot*: Indicates an unrecognized payload root element was received. Additional information may be provided within the PCDATA.
- *UnrecognizedVersion* - Indicates that the version attribute in the Schema element of the Manifest of the header is not recognized. Additional information, such as version number(s) supported, may be provided within the PCDATA.

DocURL - If present, this URL refers to an online description of the error that occurred.

Any OTA message request MAY result in a response message that consists of a StandardError baseline payload. A StandardError response may be matched to the original request via the standard *EchoToken* attribute.

```
<StandardError EchoToken="12345" TimeStamp="2000-11-20T20:14:10Z"
  Status="NotProcessed" Type="UnrecognizedRoot"
</StandardError>
```

¹⁴ Previously identified as an "evolving enumeration" in OTA version 1, the data type is designated as CDATA #REQUIRED as there is no way to formally specify an ambiguous enumeration in a DTD. Implementors should be careful to support the standard values specified here.

5.5. Versioned error messages

The OTA specification provides for error messages as part of the versioned messages, when those errors result from interactions with the trading partner's server. This section outlines specific requirements for the versioned error messages.

Typically, if a business message, such as updating a customer profile, fails for a business level reason, the business message itself should use the response message <xxxRS> that may declare a failure when it is returned. This response has meaning only in the context of the business message, based on the notion that a business content level error constitutes the response.

5.5.1. Versioned Standard Payload Attributes

The response message <xxxRS> contains the entity, %PayloadStdAttributes; on the root element. The meaning and usage of each of these standard attributes is as follows:

- *EchoToken*: a sequence number for additional message identification assigned by the requesting host system. When a request message includes an *EchoToken*, the corresponding response message MUST include an *EchoToken* with an identical value.
- *TimeStamp*: indicates the creation date and time of the message in UTC using the following format specified by ISO 8601: YYYY-MM-DDThh:mm:ssZ with time values using the 24-hour (military) clock. e.g.: 20 November 2000, 1:59:38pm UTC becomes 2000-11-20T13:59:38Z
- *Target*: indicates if the message is a test or production message, with a default value of Production. Valid values: (Test | Production)
- *Version*: This optional attribute is NOT used in OTA non-versioned messages. For all OTA versioned messages, the version of the message indicated by an integer value.
- *SequenceNmbr* - This optional attribute is used to identify the sequence number of the transaction as assigned by the sending system. Allows for an application to process messages in a certain order, or to request a resynchronization of messages in the event that a system has been offline and needs to retrieve messages that were missed.

Versioned error messages (and warnings), for any valid OTA payload response message provide a facility to help trading partners identify the outcome of a message.

Note: All OTA versioned message requests MAY result in a response message that consists of a non-versioned StandardError construct alone. When a <StandardError> is not returned, trading partners should be able to quickly determine whether the request succeeded, or had other errors identified by the application that processed the request.

Therefore, every <xxxRS> element MUST have an optional <Success/> element. The presence of the empty <Success/> element explicitly indicates that the OTA versioned message succeeded. In the absence of <Success/>, an implementation may return <Warnings> in the event of one or more business context errors, OR <Errors> in the event of a failure to process the message altogether.

5.5.2. Use of entity module for versioned message responses

Error and Warning elements share a common definition (with the exception of the tag name). These common elements and parameter entities provided for convenience when defining message response DTDs are defined in the standard module "*OTA_v2ent.mod*" shown below:

```
<!-- Copyright (C) 2000 OpenTravel Alliance -->
<!-- OTA v2 Specification - standard parameter entities -->

<!-- The following entity defines standard attributes -->
<!-- that appear on the root element for all OTA v2 payloads. -->
<!ENTITY % OTA_PayloadStdAttributes
    " EchoToken    CDATA #IMPLIED
      TimeStamp    CDATA #IMPLIED
      Target (Test | Production)'Production'
      Version CDATA #IMPLIED
      SequenceNmbr CDATA #IMPLIED
    >
<!-- The following parameter entities represent success and failure -->
<!-- in processing respectively. -->
<!ENTITY % OTA_SucceedDef "(Success|Warnings)">
<!ENTITY % OTA_FailureDef "Errors">

<!ENTITY % ErrorAttr
    " Type (Unknown|
          NoImplementation|
          BizRule|
          Authentication|
          AuthenticationTimeout|
          Authorization|
          ProtocolViolation|
          TransactionModel|
          AuthenticationModel|
          ReqFieldMissing) #REQUIRED
      Code          CDATA #IMPLIED
      DocURL        CDATA #IMPLIED
      Status        CDATA #IMPLIED
      Tag           CDATA #IMPLIED
      RecordId      CDATA #IMPLIED"
    >
<!-- Standard way to indicate success processing an OTA message -->
<!ELEMENT Success EMPTY>

<!-- Indicates successful processing, but warnings occurred -->
<!ELEMENT Warnings (Warning+)>

<!-- Indicates errors occurred during processing -->
<!ELEMENT Errors (Error+)>

<!ELEMENT Warning (#PCDATA)>
<!ATTLIST Warning %ErrorAttr;>
<!ELEMENT Error (#PCDATA)>
<!ATTLIST Error %ErrorAttr;>
```


This module may be then included in other payload definition DTDs¹⁵. An example will help illustrate. In this hypothetical example, we define an <ExampleRS> payload:

```
<!-- Example Response message using standard parameters -->

<!ENTITY % OTA_v2ent SYSTEM "OTA_v2ent.mod">
%OTA_v2ent;

<!ELEMENT ExampleRS ((%OTASucceedDef;,Example)|%OTAFailureDef;)>
<!ATTLIST ExampleRS %OTA_PayloadStdAttributes;>

<!-- Import content for the <Example> element -->
<!ENTITY % ExampleDef SYSTEM "example.dtd">
%ExampleDef;
```

Note how the definition of the <Example> element, the central element in this <ExampleRS> response message, is imported from another reusable DTD ("*example.dtd*" in this case).

The attributes of a <Warning> or <Error> element are identical and defined in the Error Attributes entity, %ErrorAttr; as follows:

Type - The Error element MUST contain the *Type* attribute that uses a recommended set of values to indicate the error type. The validating DTD can expect to accept values that it has NOT been explicitly coded for and process them in an acceptable way. This is facilitated by accepting the *Type* of "Unknown". The initial enumeration list MUST contain:

- *Unknown* - Indicates an unknown error. It is recommended that additional information be provided within the PCDATA, whenever possible.
- *NoImplementation* - Indicates that the target business system has no implementation for the intended request. Additional information may be provided within the PCDATA.
- *BizRule* - Indicates that the XML message has passed a low-level validation check, but that the business rules for the request, such as creating a record with a non-unique identifier, were not met. It is up to each implementation to determine when or if to use this error type or a more specific upper level content error. Additional information may be provided within the PCDATA.
- *AuthenticationModel* - Indicates the type of authentication requested is not recognized. Additional information may be provided within the PCDATA.
- *Authentication* - Indicates the message lacks adequate security credentials. Additional information may be provided within the PCDATA.
- *AuthenticationTimeout* - Indicates that the security credentials in the message have expired. Additional information may be provided within the PCDATA.
- *Authorization* - Indicates the sender lacks adequate security authorization to perform the request. Additional information may be provided within the PCDATA.

¹⁵ N.B. This is the essence of the "schema fragment" method of isolating change within a network of DTD documents forming a specification. Carefully followed, this method allows once-and-only-once definition of all elements and entities.

- *ProtocolViolation* - Indicates that a request was sent within a message exchange that does not align to the message protocols. Additional information may be provided within the PCDATA.
- *TransactionModel* - Indicates that the target business system does not support the intended transaction-oriented operation. Additional information may be found within the PCDATA.
- *Authentication Model* - Indicates that the type of authentication requested is not supported. Additional information may be provided within the PCDATA.
- *ReqFieldMissing* - Indicates that an element of attribute that is required by the DTD or Schema (or required by agreement between trading partners) is missing from the message.

Code - If present, this refers to a table of coded values exchanged between applications to identify errors or warnings.

DocURL - If present, this URL refers to an online description of the error that occurred.

Status - If present, recommended values are those enumerated in the non-versioned Standard Error message, (NotProcessed | Incomplete | Complete | Unknown) however, the data type is designated as CDATA for versioned messages responses, recognizing that trading partners may identify additional status conditions not included in the enumeration.

Tag - If present, this attribute may identify an unknown or misspelled tag that caused an error in processing. It is recommended that the *Tag* attribute use XPath notation to identify the location of a tag in the event that more than one tag of the same name is present in the document. Alternatively, the tag name alone can be used to identify missing data [*Type=ReqFieldMissing*].

RecordId - If present, this attribute allows for batch processing and the identification of the record that failed amongst a group of records.

The following is an example of a versioned error message in which a profile (identified by its UniqueId) was not found:

```
<OTA_ReadProfileRS
  EchoToken="12346" TimeStamp="2000-11-20T20:15:21Z" Version="2">
  <Errors>
    <Error Type="BizRule"> UniqueId 09782345768 not found</Error>
  </Errors>
</OTA_ReadProfileRS>
```

5.6. Version discovery messages

The OTA XML infrastructure supports version discovery message protocols in order for OTA businesses to transition through time with the standards, yet still communicate with multiple trading partners. To help with this transition, OTA businesses will likely need to support multiple versions of the OTA standards. As such, initially - and periodically - each OTA business will need to determine, for each trading partner, what version and protocol to use.

It is presumed that the version discovery messages are transported within an ebXML message envelope. The use of the OTA non-versioned discovery message assumes that the ebXML discovery has already taken place. From that perspective, the infrastructure of the envelope is transparent, and the level of discovery of the envelope is out of scope for this specification.

The OTA version discovery process consists of two questions:

1. What versions do you support?
2. Which payloads within a particular version do you support?

To answer this question, and to determine the payloads supported, trading partners MAY exchange a non-versioned OTA message, using the following simple payload for the request:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE PayloadsSupportedRQ SYSTEM "OTA_payloadRQ.dtd">
<PayloadsSupportedRQ/>
```

This request payload is formally defined via the following DTD:

```
<!-- Copyright (C) 2000 Open Travel Alliance -->
<!-- Non-versioned messages: OTA_payloadRQ.dtd -->

<!ENTITY % OTA_NVent SYSTEM "OTA_nvent.mod">
%OTA_NVent;

<!ELEMENT PayloadsSupportedRQ EMPTY>
<!ATTLIST PayloadsSupportedRQ
  %OTA_PayloadStdAttributes;
>
```

The <PayloadsSupportedRS> response defines a list of one or more <Payload> elements detailing which payloads and versions an implementation supports. The response may return a nested collection of one or more payloads supported. This response is defined as follows:

```
<!-- Copyright (C) 2000 Open Travel Alliance -->
<!-- Non-versioned messages: OTA_payloadRS.dtd -->

<!ENTITY % OTA_NVent SYSTEM "OTA_nvent.mod">
%OTA_NVent;

<!ELEMENT PayloadsSupportedRS (Payload+)>
<!ATTLIST PayloadsSupportedRS
  %OTA_PayloadStdAttributes;
>

<!ELEMENT Payload (#PCDATA)>
<!ATTLIST Payload
  XML_Root CDATA #REQUIRED
  Version CDATA #REQUIRED
  Protocol CDATA #REQUIRED
  Access CDATA #REQUIRED
  Params CDATA #IMPLIED
  DocURL CDATA #IMPLIED>
```

The Payload element has six attributes and optional element data (PCDATA) for additional information. The values of these <Payload> attributes are defined as follows:

- *XML_Root* - The root element tag name for a supported payload. This attribute identifies the root tag of the messages, or the action verb, that define which operations are supported. This is a REQUIRED attribute.
- *Version* - indicates the OTA version supported for this payload. This is a REQUIRED attribute.
- *Protocol* - indicates the communication protocol to be used to access the trading partner, using IETF defined protocol types. This is a REQUIRED attribute. (see example below)
- *Access* - indicates the access, path, ID or address to access the service that supports this payload type (for HTTP, FTP or SMTP based services, this represents a URL). This is a REQUIRED attribute.
- *Params* - this attribute is used to communicate protocol-specific additional information needed to access this version. This is an optional attribute.
- *DocURL* - refers to an online description of the support context and version. This is an optional attribute.

This approach reduces the need to grow the specification based on evolving protocols or access specification names. Trading partners may need, for other protocols, to encode additional options in either the Params attribute, or the optional element data.

An example of an <OTA_PayloadsSupportedRS> message follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE PayloadsSupportedRS SYSTEM "OTA_payloadRS.dtd">
<PayloadsSupportedRS>
  <Payload XML_Root="OTA_CreateProfileRQ"
    Version="2"
    Protocol="HTTPS 1.1 POST"
    Access="https://vmguys.com/OTAEngine/">
  <Payload XML_Root="OTA_CreateProfileRQ"
    Version="3"
    Protocol="HTTPS 1.1 POST"
    Access="https://vmguys.com/OTAEngine/">
  <Payload XML_Root="OTA_CreateProfileRQ"
    Version="2"
    Protocol="SMTP"
    Access="mailto:OTAEngine@vmguys.com">
  <Payload XML_Root="OTA_CreateProfileRQ"
    Version="3"
    Protocol="SMTP"
    Access="mailto:OTAEngine@vmguys.com">
  <Payload XML_Root="OTA_CreateProfileRQ"
    Version="2"
    Protocol="FTP"
    Access="ftp://ftp.vmguys.com/OTAEngine/">
  <Payload XML_Root="OTA_CreateProfileRQ"
    Version="2"
    Protocol="HTTPS 1.1 POST"
    Access="https://vmguys.com/OTAEngine/">
  <Payload XML_Root="OTA_CreateProfileRQ"
    Version="3"
    Protocol="HTTPS 1.1 POST"
    Access="https://vmguys.com/OTAEngine/">
```

```
...  
...  
...  
</PayloadsSupportedRS>
```

With a response like the one above, a trading partner can choose the most appropriate OTA payload, version and protocol to use to interact with the target trading partner. As shown above, a particular implementation may support several versions and several protocol/access methods for a particular payload type.

NOTE: It is necessary that for purposes of exchanging non-versioned messages, all implementations **MUST** support the "HTTP 1.1 POST" method. It is assumed that trading partners will know the correct access address for each others implementation to be able to perform version discovery.

6. Security and Privacy

Earlier chapters on message structure and the use of MIME wrappers discussed specific ways of addressing security and privacy in messaging. This chapter discusses the requirements for security and privacy, and provides additional guidance for these features in OTA specifications.

6.1. Terminology

This document uses several recognized security-related terms, borrowed from the Java Authentication and Authorization Service:

- **Subject:** user of a computing service; users and computing services are subjects. A subject has a set of principals.
- **Principal:** name associated with the subject. The name may be a conventional name or a public key. Subjects may have different names based on the service they are using. Principals can become associated with a subject upon successful authentication.
- **Authentication:** represents the process where one subject verifies the identity of another.
- **Credentials:** security related attributes such as passwords, public key certificates.

6.2. Security and privacy requirements

Conducting electronic business transactions over the Internet in any line of business carries a number of risks, but dealing with personal information, as in the case of profiles, or travel reservations, requires special precautions. While the measures recommended in this specification can reduce the risks, even the best of security programs cannot guarantee risk-free transactions. Successful security will require a combination of technical steps as well as sound policies implemented by the companies sending and receiving these data.

The implications for protecting personal information go beyond good business practice. Government authorities in the USA, Canada, Europe, and Japan have begun instituting regulations to restrict the movement of personal data, particularly when the consent of the individuals is needed to share the data with business partners. Since the travel business is a global industry, OTA must recognize its responsibilities in meeting these regulations.

The recommendations on security in OTA 2001A address the following factors:

6.2.1. Authentication

The function of Authentication seeks to identify and credential the parties in the transaction to ensure that the party engaged in a transaction is really the party they claim to be and not an imposter seeking to engage in fraud.

6.2.2. Confidentiality

Supplying adequate confidentiality in the exchange of messages limits the possibility of eavesdropping and provides assurance to the parties conducting a transaction that sensitive information remains known only to the parties intending to share the information.

6.2.3. Integrity

Integrity of message delivery limits the possibility of distortion of a message by sabotage or non-malicious intent, such as network errors, and assures the parties engaged in the transaction that the data received by one party is the same as the data sent by the other party.

6.2.4. Non-repudiation

Electronic transactions need to have the same level of commitment as a signed paper document to hold parties accountable to that commitment. OTA recommends that the parties provide a record that the transaction actually took place, to ensure that it could not have been a forgery.

6.2.5. Privacy

Authorities in many jurisdictions around the world have regulations on the collection and dissemination of personal and company data. The OTA specifications have taken into consideration the rights of individuals and companies to provide data and to have control over the disposition of data after its collection by the travel companies. OTA specifications provide the means for travel companies to be able to provide assurance to individuals and companies that their data will be used only for the purposes originally indicated on the collection forms. It assures individuals and companies, and that they will have the ability to restrict names or any other data from any sharing arrangements.

6.3. Security and privacy recommendations

This section discusses implementation of security and privacy, either by referencing detailed specifications or with recommended best practices.

6.3.1. Authentication

Chapter 4, Section 4.7.1 presents a recommended syntax for authenticating trading partners using the OTA specifications. Section 4.7.1.1B gives an example of the OTA message syntax, showing the relationship between authentication and encryption.

6.3.2. Confidentiality

To meet the needs of confidentiality OTA 2001A RECOMMENDS use of encryption to scramble the messages and protect the transfer of data against eavesdropping, as well as decryption by the recipient. The most common form of encryption/decryption with Web transactions is provided by the Secure Socket Layer (SSL) protocol developed by Netscape and now a component of the IETF transport layer security specifications. Because SSL operates at the transport layer, it encrypts the entire message, whether or not all the components of that message require it. As an option, the trading partner MAY encrypt the payload document in the message.

Trading partners need to resolve questions of encryption/decryption procedures before exchanging files with the OTA specifications. The non-versioned messages can be used for discovery but this section leaves open the potential for growth of this capability to include these topics. OTA expects that the Electronic Business XML (ebXML) initiative will address these questions, which OTA can implement in a future version.

6.3.3. Integrity

At this time, the World Wide Web Consortium is developing a digital signature recommendation based on XML that creates an encrypted hash message for verification. Once developed, it will provide a method of ensuring delivery of undistorted messages. OTA will review the development of the W3C digital signature for potential application.

The OTA 2001A specification uses a request and response model that returns some form of message to the sender, if only a simple acknowledgment. While not foolproof, this approach lets users and systems monitor the message flow and spot possible distortions or interruptions. If the parties have chosen to use encryption, it also provides a message integrity check, since the decryption routine performs a defacto hash function to correctly reassemble the content.

6.3.4. Non-repudiation

The combination of authentication and digital signature standards – in development by Internet and Web standards bodies – will provide powerful and effective tools to prevent non-repudiation of transactions. Until those standards become more widely available, trading partners **SHOULD** establish event logs to provide an audit trail for ascertaining that transactions took place.

6.3.5. Privacy

As with security, privacy connotes control over the information collected, and the OTA 2001A specification includes ways for the customer to determine the data they want shared with other parties. To address these concerns, OTA specifications provide a means by which the owner of the profile can determine the sharing of data overall, as well as for specific components of the profile. The OTA 2001A Profile specification assumes the customer or company does **NOT** want the data shared for any reason unless specifically authorized. As a result, when the data is provided for the profile, authorization **MUST** be given to share that data with other parties.

OTA specifications provide two attributes that address sharing of data. These attributes are present on Profile, the highest-level content element, which restricts sharing of data in the record as a whole.

The two privacy attributes included in OTA specifications are:

- *ShareAllSynchInd* - to permit sharing data for synchronization. Synchronization means updating records residing on other servers than the one that created it. Customers or companies may wish, for example, to have their profiles reside with vendors in which they exchange data in order to keep their respective records updated.
- *ShareAllMarketInd* - to permit sharing data for marketing purposes. Marketing purposes include providing the data to other vendors who can send related offers that cater to the interests of the customer or company.

Each of these attributes has values of YES and NO, with NO being the default, which means customers or companies must give their explicit permission -- change the value to YES -- in order to share the data.

Customers and companies can determine the data shared within a profile as well. The same two privacy attributes are present at other locations throughout the specification and allow control of privacy of specific data element groups that may include particularly sensitive information:

- *ShareSynchInd*, to permit sharing data for synchronization, as described above
- *ShareMarketInd*, to permit sharing data for marketing purposes

These attributes also have YES and NO values, but also the default INHERIT value, which means data lower in the hierarchy, inherit the value of *ShareSynchInd* or *ShareMarketInd* from higher levels in the tree. Therefore, if a customer says NO for sharing all of the data in the profile, it takes an explicit YES value to override that choice for individual elements.

OTA RECOMMENDS that travel service providers disclose their privacy policies and practices in accordance with the Standard for Internet Commerce, version 1.0, dated 14 December 1999. This standard outlines best practices for the conduct of business over the Web, but trading partners need to make themselves aware of legal requirements involving privacy in their jurisdictions as well. A copy of the relevant sections from this standard are included in the Appendix document.

Another standard involving privacy under development by the W3C is the Platform for Privacy Preferences or P3P; see <http://www.w3.org/P3P/>. This specification defines the semantics for an organization's privacy policies expressed in XML syntax. The specification allows organizations to indicate the types of data or precise data elements collected as well as policies for disclosure and sharing of that data. Because P3P uses XML syntax, it allows organizations to exchange this information electronically as well as making it available in human readable form. As of the date of this specification, W3C issued the second "last call" for comments for the P3P public working draft on October 18, 2000.

OpenTravel™ Alliance Message Specification

Publication 2001A

Specifications Document

Part II - Profile Specification

I. OTA Profile

Introduction

The OTA 2001A Profile specification defines the detailed business content of a customer profile from a travel industry perspective. This specification provides a set of common messages for transmitting customer profile data that customers provide to travel services to create these profiles, and for the exchange of profile information between travel services within the industry.

Profile Content and Scope

A profile includes basic information about a customer or a company for identification as well as financial transactions, memberships and contacts. The profile also defines collections of preferences for specific types of travel including key travel support services such as travel agencies and insurance. Profiles contain information about organizational affiliations, and identify certifications and alliances held by companies in their business relationships. No supplier pricing information is included, nor is data on the travel policies or requirements of an organization addressed in this specification.

The specification divides into two main categories – Business Content that has six sections and Maintenance with one section:

Business Content

1. **Customer** - information about the traveler needed to define and document the person's identity, means of contact, types of payment, and basic needs and interests for travel services.
2. **Preferences** - general and specific conditions to meet the needs of a customer or a company for travel based on identifiable purposes, such as business trips, family vacations, golf outings, international trips, or the annual church retreat. The profile can also be used to indicate preferences for specific kinds of travel services, such as for air travel, hotel stays, car rentals, and other types of travel services as part of these collections.
3. **Affiliations** - organizations with which the customer or company has a relationship and that convey travel benefits or privileges, such as employer, interest groups, membership organizations (e.g. AARP or AAA), travel arrangers, insurance companies, and vendor-sponsored travel clubs such as those provided by airlines. This section includes only data about the relationship to these organizations (e.g., employee identifier), not the policies of the companies themselves.
4. **CompanyInfo** - information about a company or business entity, including addresses, telephones, and other forms of contact, types of payment forms, loyalty programs etc.
5. **Agreements** - information about certifications and alliances which the company may hold, as well as information about commission arrangements.
6. **TPA_Extensions** - due to the proprietary nature of agreements between business relationships, the TPA_Extensions element is included in this section of the profile, and allows for companies to add selected information as needed.

Maintenance of the profile data

7. **Accesses** - a collection of elements that contains a chronicle of recent accesses to the profile, noting the create date and time and updates or subsequent actions performed on the profile record.

Each of the main sections has *elements* that represent one or more related fields for capturing data on specific logical components of the profile. Each of these elements may contain other elements or data, usually character strings of alpha or numeric characters.

In some cases the elements have identified specific properties or characteristics called *attributes*. A feature of attributes is the constraint of choices permitted defined by an enumeration of values. Many of the travel preferences, for example, allow for indicating a preference level (ex. PreferLevel) of Preferred, Only, or Unacceptable, with the default value being 'Preferred' in the absence of another choice being explicitly indicated. These are the only choices available for the PreferLevel values.

A Document Type Declaration (DTD) file contains rules for the message structure including the content of the profile. It spells out the required order of the data and which items are required in messages. Alternatively, an XML Schema (XSD) file provides an additional layer of validation including the ability to assign data types other than strings to certain data elements, such as datetimes and integers. As the OTA 2001A specification was finalized, the W3C eventually ratified XSD validation. As writing had already begun for this publication, the OTA 2001A specification therefore references a DTD for validation even so a set of XSD files is also included within this 2001A specification. Please note that validity in this case applies only to the structure of the message; neither method of validation would catch an incorrect credit card number or misspelled name.

In the course of modeling the data, OTA identified several groups of elements or attributes that repeat throughout the profile. In the travel profile, OTA calls these repeating groups reusable objects, entered as *entities* in the DTD, and *includes* in XML schema. Entities reduce the size of the DTD and allow for a more modular structure, and provide the same advantage in schema.

For the OTA 2001A specification, DTD fragments have been created for the components of a profile. The use of these fragments increases the ability to take advantage of reusable objects, particularly in those cases where customer information is likely to be reused throughout other OTA specifications. For example, information about membership in a loyalty program defined in the profile could be used to indicate the use of that program for a hotel stay, or airline reservation, using the same fragment.

The OTA 2001A DTD and XML Schema will be available as two separate documents from the OTA web site: <http://www.opentravel.org/>.

Profile Message flow and operations

The messages supported by this publication include the basic actions of creating, reading, updating a record partially, or replacing a record in its entirety, and deleting of records. These four request/response pairs underpin the role and function of the message exchanges for customer profiles between trading partners. The Create, Read, Replace and Delete verbs involve entire profile records, while the Update verb addresses parts of a profile.

A payload document that carries the profile information identifies the message as a request (RQ) or response (RS) which specifies the action required by the receiver to process the message, as well as providing the business data exchanged between trading partners. Payload documents are also used for the Authentication, ReAuthentication, and UnAuthentication actions described in OTA Infrastructure, Section 4, Message Structure.

XLink:role - The *xlink:role* attribute in the <DocumentReference> element of the Manifest identifies the business purpose of the referenced document. Its presence in the Manifest allows the actionable item to be determined without retrieving the referenced document. For OTA's purposes, the value entered for the *xlink:role* attribute MAY consist of the URI, but MUST consist of the OTA action verb, which is the root element of the payload document. Strict adherence to the syntax of the strings that define OTA action verbs should be observed, as parsers will use these strings to identify the action to be performed on the document.

With the publication of a set of messages, OTA enumerates the list of actions that can be performed upon specific business content e.g.: <OTA_CreateProfileRQ>, a request to create a profile. Additions to the enumerated list are anticipated as additional content and travel verbs become a part of the OpenTravel Alliance vocabulary.

Profile Messages Supported

The syntax of the action verbs to be used as root elements of a payload document used to exchange profile data are enumerated as follows:

- **OTA_CreateProfileRQ** - Identifies the initial creation of a profile.
- **OTA_CreateProfileRS** - Indicates the status of success in the creation of a profile.
- **OTA_ReadRQ** - Identifies a request to read a profile record when combined with the UniqueId of *Type="Profile"*.
- **OTA_ReadProfileRS** - Returns a customer profile in response to a Read request of *Type="Profile"*.
- **OTA_UpdateRQ** - Identifies a request to update a customer profile record when combined with the UniqueId of *Type="Profile"*. The specific action(s) to be applied to the profile record are indicated by the children of the <Position> element, as described in Part I - Section 3.5.4, Update verb.
- **OTA_UpdateRS** - Indicates the status of success in the updating of a profile.
- **OTA_DeleteRQ** - Identifies a request to delete a customer profile record when combined with the UniqueId of *Type="Profile"*.
- **OTA_DeleteRS** - Indicates the status of success in deleting the profile record.

Create verb

The Create infrastructure verb defines an operation that generates a new record. When used to create a profile, it identifies and populates a profile record, assigning it a unique identifier. The sequence follows these steps:

- Requestor sends a Create request of *Type*="Profile" and optionally, a unique identifier, along with the initial profile data.
- Receiver creates a new profile and assigns a unique identifier.
- Receiver responds with a message providing a unique identifier for the profile created and optionally, the data entered by the requestor.

The requestor can complete as much of the profile record as desired when submitting the create request.

- If the *ProfileType*="Customer" then at least one of the following elements: *PersonName*, *Telephone*, or *Email*, **MUST** be present.
- If the profile is being created for a company or business entity, the *CompanyName* element in the *CompanyInfo* section **MUST** be filled out.

To create a profile, the requestor sends a message with the root element `<OTA_CreateProfileRQ>`. The `<UniqueID>` element **MUST** indicate the *Type*="Profile" and optionally, a unique identifier in the *Id*= field if one is created by the requesting system. The message includes, at a minimum, the data required by the specification or trading partner to open a profile.

The trading partner returns a response with a root element `<OTA_CreateProfileRS>` and an identifier for the profile, usually generated by the responding system. The requestor can then add data using the Update process, which allows for updating specific parts of the profile record or by replacing the entire profile record, as agreed upon between trading partners.

Read verb

To read the profile, the requestor sends a message with the root element `<OTA_ReadRQ>` accompanied by a `<UniqueID>` element containing the identifier for the record, and *Type*="Profile". The trading partner returns a message with the root element `<OTA_ReadProfileRS>` and the current data in the profile.

Update verb

Updating the profile, because it allows for changing parts of the record, is a more complex process. Since OTA version 2001A messages will support updates from multiple sources, as well as single operations per message, the OTA Infrastructure specification incorporates a mechanism to inhibit potential corruption of data. To meet this objective, the process involves performing a read action that facilitates comparing before and after images of a profile, followed by an update message similar to the "update" approach in use by GDSs.

The process can be viewed as consisting of four steps:

- Requestor sends a Read request of *Type*="Profile" along with the unique identifier for the profile.
- Receiver responds with a copy of the current profile stored in the database.
- Requestor sends an Update message that identifies the position of the changes to be made, followed by one or more operations to be applied at that position.

- Receiver responds with a message indicating the success of the operation or the appropriate errors or warnings if the operation(s) were not completed successfully.

While the above steps may seem straightforward in concept, implementors should follow the steps outlined in Part I – OTA Infrastructure, Chapter 3, Section 3.5.4. - Update Verb.

To read the profile, the requestor sends a message with the root element `<OTA_ReadRQ>` of `Type="Profile"` and the identifier for the record, that triggers a return message with the `<OTA_ReadProfileRS>` element and the current contents of the profile. The requestor then sends a message with the root element `<OTA_UpdateRQ>` tag, with one or more operations from the choice of "insert", "modify", or "delete", to make the requested change. The message identifying the operations also contains the content affected by the update. The response message has the root element `<OTA_UpdateRS>` tag and indicates the success (or Warnings or Errors, if not successful) of the operation.

Replace sub-action verb

Updating specific parts of a profile is a more complex process than replacing a profile record in its entirety. For purposes of simplification, implementors may find it expedient to send an entire profile record instead of attempting to determine the path location for a modification.

Use of the Replace operation does not require a read action prior to sending the message to replace the contents of the profile, unless business practices require it. To replace the profile, the requestor sends a message with the root element `<OTA_UpdateRQ>` with the child element named `<Root>` that contains the attribute `Operation="replace"` along with the identifier for the record. The response message is a `<OTA_UpdateRS>` that indicates the relative success of the operation.

Delete verb

Deleting a profile record is a process that removes the entire profile record from the database. It may be preceded by a Read request to view the record and verify the identification of the profile to be deleted, or if the record can be identified with relative certainty, and business practices allow, the Delete action can be performed without reading the profile record first.

To delete a profile using the process of verifying the record prior to deleting it, the requestor sends a message with the root element `<OTA_ReadRQ>` of `Type="Profile"` and the identifier for the record, that returns a message with the `<OTA_ReadProfileRS>` root element and the current contents of the record. The requestor then sends a `<OTA_DeleteRQ>` message of `Type="Profile"` and the identifier of the profile. It is recommended that the `Instance=` attribute be filled in with a value to ensure that the latest update of the profile record is known to all parties. The response to the delete request is an `<OTA_DeleteRS>` message confirming deletion of the record along with the profile's identifier to verify which record was deleted.

Transmitting empty data elements

This specification makes only a minimum of data items mandatory in order to meet the widest possible array of business needs, and as a result it is technically possible to transmit empty elements and still meet XML validation requirements.

Empty elements can result from unrelated practices by companies using the profile. Systems may create placeholders, for example creating a <Preferences>, <Affiliations>, and/or <TPA_Extensions> element automatically under the Profile in anticipation of customers or companies providing data for those elements, or the inclusion of additional features anticipated between trading partners. But in this scenario, some of these elements are not filled in, e.g.: Preferences, or Affiliations, companies can send these empty elements as part of their messages. Update operations may also remove child elements from a profile but leave empty parent elements in their wake.

Sending empty elements adds to the message transmission size, increases the need for bandwidth, and puts an extra processing burden on the receivers of the messages. Trading partners need to establish policies and measures to minimize their occurrence. Because of the flexibility and the potential for sending empty elements, trading partners **SHOULD** transmit **ONLY** the data elements in the profile that are non-blank.

Elements and attributes in the Profile

In establishing the Profile data model, OTA defined data items to maximize the use of attributes, since the use of a data item as an attribute is unique within the namespace of the element. The use of attributes also reduces the size of the overall message, since an attribute name appears only once and does not require the use of both opening and closing tags surrounding the data.

The following criteria were used as guidelines for determining whether a data item should be an element or an attribute:

- Single occurrence, not a repeating element
- No child elements
- No attributes of its own
- Is not part of a 'choice' or branch within the schema
- Discrete code or limited text field (anticipated length not to exceed approximately 64 characters)

Elements and attributes tables

Each of the elements, element groups and their respective attributes of the data model of the OTA Profile, based on the Schema for <Profile>, are listed in the remainder of this section. The following is a key for the symbols, abbreviations, and terms used in the tables:

Element - name of the element used in XML tags

Occurrence -

R = Required, 1 occurrence allowed

? = Optional, 0 or 1 occurrences allowed

+ = Required, but multiples allowed

* = Optional, 0, 1, or more than one occurrence allowed

Content -

Element = another element

Text = Character data; see data type for details

Empty = No content in element, attributes used to convey meaning and provide data

Any = Any content allowed

Data type - applies to elements with content other than another element

string = Alphanumeric plus special punctuation and symbol characters

boolean = A value of Yes or No

timeInstant = Date and/or time format specified by W3C using ISO 8601

blank = Not applicable (element content)

integer = Any whole number - positive, negative, or zero.

decimal = Any number containing a fraction represented by a decimal point.

Please note that these data types are RECOMMENDED. The DTDs intended for validation of the XML exchanges use text string (CDATA) or enumeration data types. The XML Schema definitions allow more formal semantics of data typing, and should be used for reference to the datatypes assigned to the data, particularly when used for purposes of conversion into databases.

Attributes - gives the name of the attributes used with the element. Attribute tables will indicate the data type for the attribute, associated elements, default values, and descriptions. For attribute data types, Enumerations (abbreviated Enum in the charts) restrict the choices of values and one of the listed values MUST be used. For string attributes, any values listed are RECOMMENDED.

Description - provides a definition of the data element and includes references to specific code lists or standards, if used.

Privacy attributes

Two sets of attributes allow for customers and companies to indicate which data they would like to share for synchronization of their profiles at other locations and to receive marketing information. The specification calls for ShareAllSynchInd and ShareAllMarketInd attributes on the Profile element for permission to share all of the data in the profiles for synchronization and marketing purposes respectively. These attributes MUST have default values of No and require explicit approval from the customer to be changed to Yes.

The specification also gives the customer or company control over sharing of major elements within the profile. The ShareSynchInd and ShareMarketInd attributes on the major elements provide for the owner of the profile to indicate permission to share the data in these elements for synchronization or marketing. These attributes MUST have default values of Inherit for Inherited, which means the element inherits the permission value assigned to the next highest level. However, a value other than Inherit MUST over-ride the permission granted to the higher levels in the hierarchy for that given element.

For example, a value of No in ShareAllMarketInd in Profile will apply to the entire profile, unless it has been changed to Yes in ShareMarketInd for specific elements in the profile. Remember that with the default value of "Inherit" for Inherited, each subsequent element in the tree inherits the value of the next highest level, beginning with the root element, Profile.

In another example, the customer or company may indicate Yes in ShareAllSynchInd under the Profile element. If the customer or company wants to prevent the travel service provider holding the profile from sharing payment information, even for synchronization purposes, then the ShareSynchInd attribute for the PaymentForm element is set to a value of No.

Because the two attributes, ShareSynchInd and ShareMarketInd, occur frequently throughout the OTA 2001A specification, and they always appear together as a pair, they make up the Privacy entity, %Ent.Privacy;, which carries a default value of "Inherit". Note that the attributes in the Profile element, named ShareAllSynchInd and ShareAllMarketInd, have the default value of "No" which differ from the default value of the entity, "Inherit".

Reference Place Holder (RPH) attributes

This specification introduces the concept of a Reference Place Holder (RPH), an attribute that is used to identify a specific item in a collection of like items. For example, a Profile record may contain a series of payment forms, including credit cards, bank accounts or direct billing arrangements that a customer may use to pay for a travel experience; loyalty programs to which the customer belongs; or a list of contact persons that may include family members to be contacted in case of an emergency, travel arrangers who typically arrange for travel services on behalf of the customer, etc.

In a given travel situation, one of those payment forms may be used to pay for travel services; one or more contacts may be designated as an emergency contact during the trip; one or more loyalty programs could be used to accrue points for a travel service received in the course of travel.

The use of an RPH enables a reference to the location where information about an object is stored, so that the data does not have to be repeated in more than one place in a message transaction. Every time a collection of like elements is passed in a message to be stored in a database, they are numbered, or indexed, and assigned an RPH. The index (RPH) values **MUST** remain the same on multiple systems sharing the data. RPH values do not change and may not be re-used. The index number, or RPH of an item in a collection does not change when an item in that collection is deleted.

An RPH is a serial sequence number that may also serve as a unique ID to an element within a collection. RPH's are not restricted to a single use, but can be maintained after the message transaction in a flat file reference. If implementors choose to do so, an RPH can be used as a key in database.

The RPH in the Preferences collection of the Profile is a reference placeholder that refers to the index of the customer (or company) preference of a payment form, or loyalty program membership (in the Customer data) to be used in the context of the travel experience. Future use of RPHs in OTA specifications can be anticipated for attaching payment forms or loyalty programs to reservations for travel services, identifying contacts for customers or companies, etc.

Thus RPHs used in this context are locations of the source of data that are cross-referenced to one another. One is the RPH attribute of the object that houses the data and indexes the collection, and the other is the RPH used within the context of a message, such as a travel preference attached to a reservation request, as described above.

Profile element

The Profile element (Figure 1) is the root element of the business content of the profile. A Profile, along with the UniqueID and the profile identifier attributes: "URL", "Type", "Id", and "Instance" make up the OTA 2001A messages transmitted under the root tag that identifies the action to be performed. The attributes on the Profile element include the two privacy attributes described above, and the ProfileType. These attributes are required.

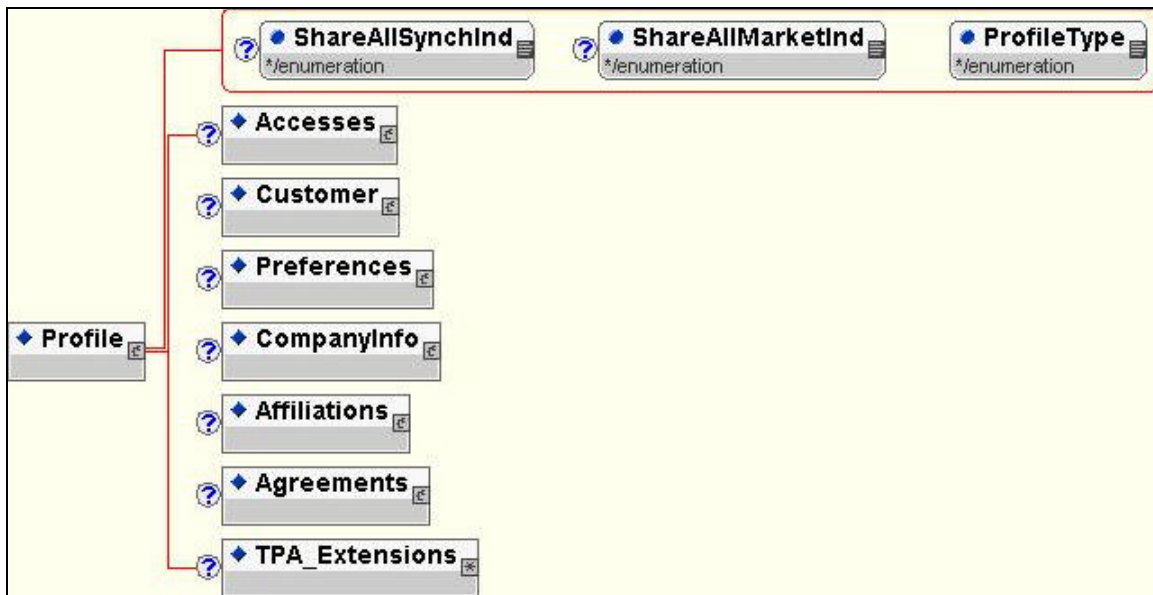


Figure 1. Profile element

Element	Occur.	Content	Data type	Attributes	Description
Profile	R	Element		ShareAllSynchInd, ShareAllMarketInd, ProfileType	Top-level element for profile
Accesses	?	Element		%Privacy, CreateDateTime	Element to capture creation and last update data
Customer	?	Element		Gender, Deceased, LockoutType, %Privacy, BirthDate, Currency	Contains basic data on the customer's identity, finances, and location
Preferences	?	Element		%Privacy	Preferences related to travel experiences
CompanyInfo	?	Element		CompanyCode, CodeContext	Location for company profile
Affiliations	?	Element		%Privacy	Companies or organizations connected with the customer that affect travel decisions
Agreements	?	Element		%Privacy	Section of a business profile that contains information about trading partner agreements
TPA_Extensions	?	Element			Due to the proprietary nature of agreements between business entities, this element allows for

					companies to add selected information as needed.
--	--	--	--	--	--

Profile

Attributes:

ShareAllSynchInd : (Yes | No)

Permission for sharing data for synchronization of profiles held by other travel service providers. Enumerated values: 'Yes', 'No'. Default at the Profile level = No.

ShareAllMarketInd : (Yes | No)

Permission for sharing access data for marketing information. Enumerated values: 'Yes', 'No'. Default at the Profile level = 'No'.

ProfileType : (Customer | GDS | Corporation | Trvl_Agnt | Wholesaler | Group | Tour Operator | CRO | RepCompany | InternetBroker | Airline | Hotel | CarRental | CruiseLine)

Indicates type of customer or company profile. Enumerated values: (Customer | GDS | Corporation | Trvl_Agnt | Wholesaler | Group | Tour Operator | CRO | RepCompany | InternetBroker | Airline | Hotel | CarRental | CruiseLine). This attribute is required.

Profile DTD fragment

```
<!ELEMENT Profile (Accesses?, Customer, Preferences?, Affiliations?, TPA_Extensions?)>
```

```
<!--ATTLIST Profile
```

```
  ShareAllSynchInd (Yes | No) "No"
```

```
  ShareAllMarketInd (Yes | No) "No"
```

```
  ProfileType (Customer | GDS | Corporation | Trvl_Agnt | Wholesaler | Group | TourOperator | CRO
| RepCompany | InternetBroker | Airline | Hotel | CarRental | CruiseLine) #REQUIRED >
```

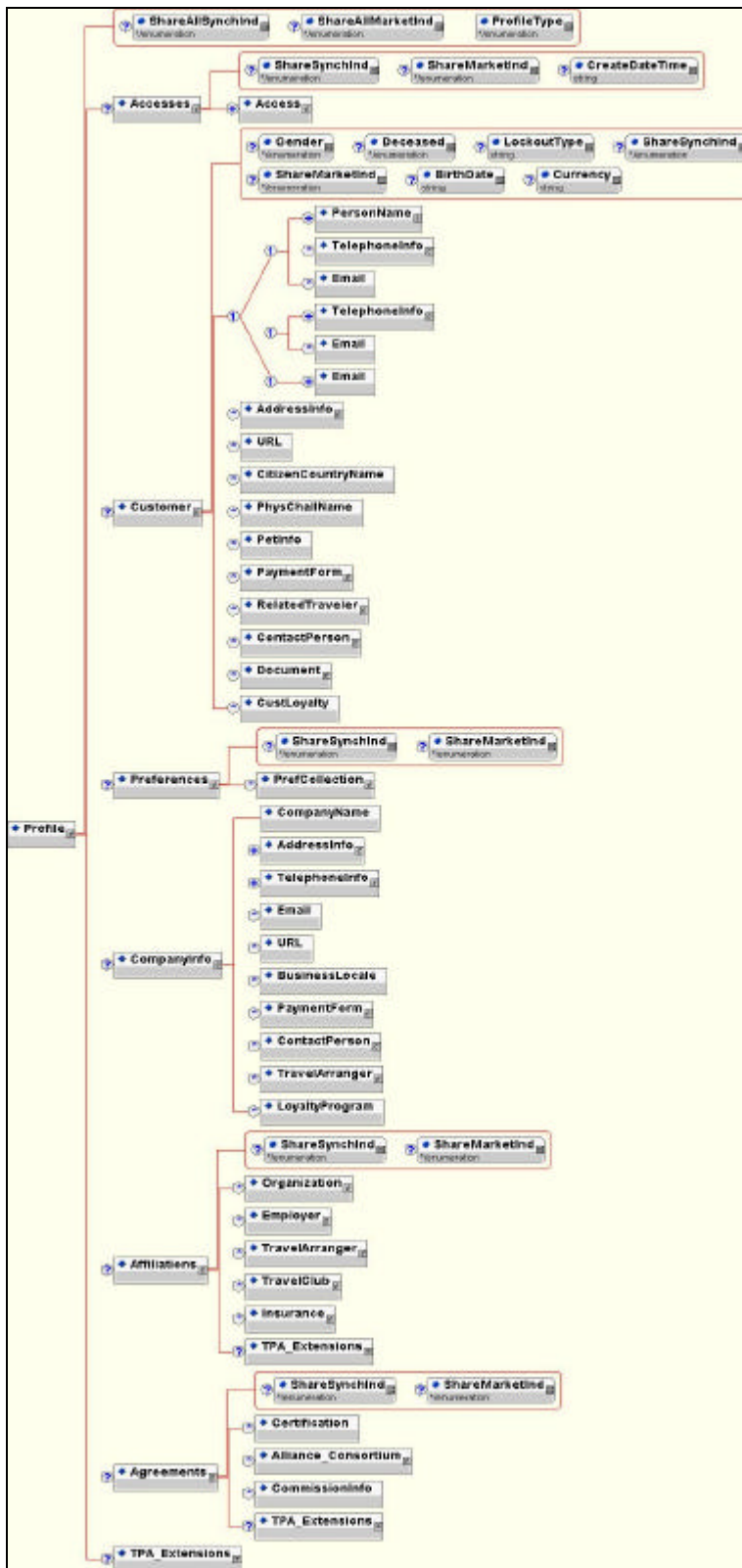


Figure 2. Profile element expanded

1.0 Accesses

The Profile element includes the optional Accesses element (Figure 3) that allows for tracking the creation date and time, and actions taken on the profile that may include the name of the person who accessed the profile and any additional comments entered by that person.

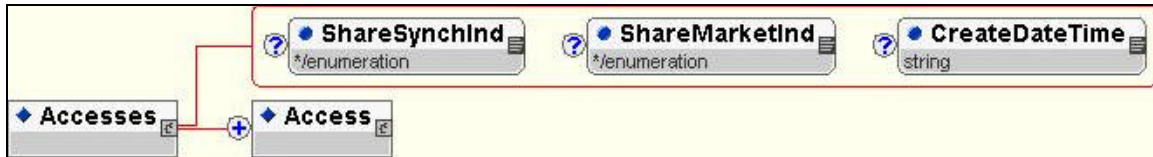


Figure 3. Accesses

Element	Occur.	Content	Data type	Attributes	Description
Accesses	?	Element		% Privacy CreateDateTime	Element to capture creation and last update data
Access	+	Element	String	ActionType, ActionDateTime, ActionSystemId	Describes the action taken on the record

Accesses

A record of the creation and recent accesses to the profile.

Attributes:

ShareSynchInd : (Yes | No | Inherit) = Inherit

Permission for sharing access information for synchronization of profiles held by other travel service providers. Enumerated values: 'Yes', 'No', 'Inherit'. Default = 'Inherit'.

ShareMarketInd : (Yes | No | Inherit) = Inherit

Permission for sharing access data for marketing information. Enumerated values: 'Yes', 'No', 'Inherit'. Default = 'Inherit'.

CreateDateTime : CDATA

Date and time this Profile record was created.

Accesses DTD fragment

```
<!ELEMENT Accesses (Access+ )>
<!ELEMENT Access (AccessPerson?, AccessComment* )>
<!ATTLIST Accesses
    %Ent.Privacy;
    CreateDateTime CDATA #IMPLIED
>
```

1.1 Access data (Figure 4)

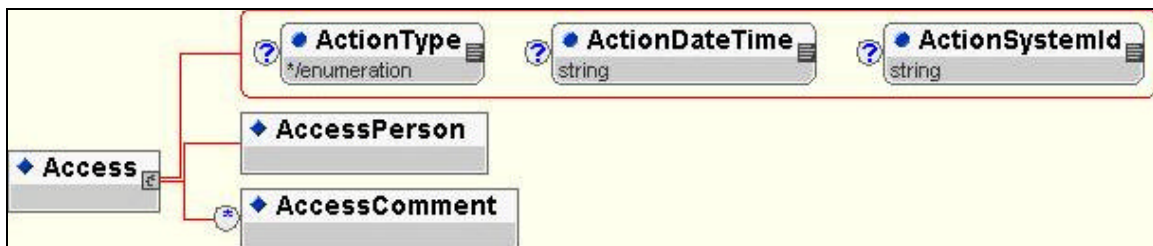


Figure 4. Access data

Element	Occur.	Content	Data type	Attributes	Description
Access	*	Element	String	ActionType, ActionDateTime, ActionSystemId	An action taken on the profile record.
AccessPerson	?	Text	String		Name of individual who originated record
AccessComment	*	Text	String		Free text description added by the person accessing the profile record.

Access

Record of an action taken on the profile record.

Attributes:

ActionType : (Create | Read | Update | Delete)

Defines the type of action performed on the Customer Profile record. Enumerated values:

Create, Read, Update, Delete

ActionDateTime : CDATA

Date and time that an action was performed on the profile.

ActionSystemId : CDATA

The identifier of the system that performed the action on the profile record.

Access DTD fragment

```
<!ELEMENT Access (AccessPerson?, AccessComment*)>
<!ATTLIST Access
  ActionType (Create | Read | Update | Delete) #IMPLIED
  ActionDateTime CDATA #IMPLIED
  ActionSystemId CDATA #IMPLIED
>
```

AccessPerson

Name of individual who originated or last updated record.

AccessPerson DTD fragment

```
<!ELEMENT AccessPerson (#PCDATA)>
>
```

AccessComment

Text description supplementing data from the LastAction and AccessPerson elements.

AccessComment DTD fragment

```
<!ELEMENT AccessComment (#PCDATA)>
```


2.0 Customer information

The Customer element and <Customer> tag covers the basic data about the customer including name and address, forms of contact, travel documents, forms of payment used, and references to related travelers, such as family members or business associates.

2.1 Required customer data

The specification defines a minimum set of data required for a profile, with that data located in the Customer section. Each profile **MUST** have entries in at least one of the following elements:

- PersonName, tag <PersonName>
- Telephone, tag <TelephoneInfo>
- Email, tag <Email>

PersonName and Telephone use the PersonName and Telephone objects that appear in several places throughout the profile, and are therefore, entities `%Ent.PersonName;` and `%Ent.Telephone;`. In the PersonName entity, the <Surname> element is **MANDATORY**. Likewise, if used, the Telephone entity and thus the TelephoneInfo element, **MUST** have entries in the AreaCityCode and PhoneNumber elements.

2.2 Customer elements and attributes (Figure 5)

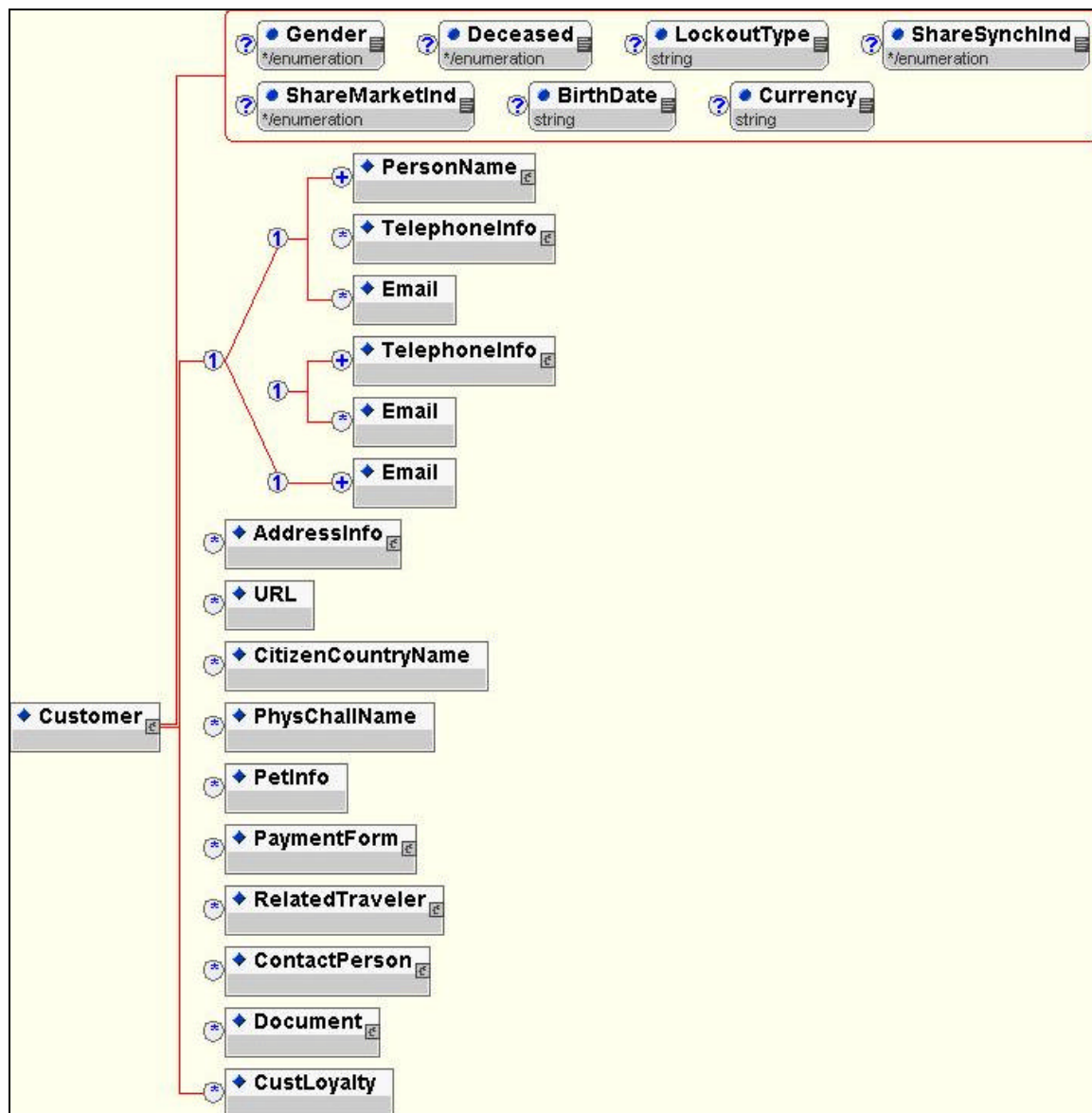


Figure 5. Customer elements

Element	Occur.	Content	Data type	Attributes	Description
Customer	R	Element		Gender, Deceased, LockoutType, % Privacy, BirthDate, Currency	Contains basic data on the customer's identity, finances, location,
PersonName	*	Element, % PersonName		DefaultInd, % Privacy, NameType	Names of the individual, includes former, nicknames, and alternate names
TelephoneInfo	*	Element, % Telephone		DefaultInd, % Privacy, PhoneUseType	Information about the telephone numbers of the individual in the profile.
Email	*	Element		DefaultInd, % Privacy,	Electronic mail addresses, in IETF specified format

				EmailType	
AddressInfo	*	Element		DefaultInd, %Privacy, AddressType	Address Information
URL	*	Element		DefaultInd, %Privacy, URL_Type	Defines the purpose of the URL address
CitizenCountry Name	*	Element		DefaultInd, CountryCode	Self-professed country or countries that customer claims for citizenship
PhysChallName	*	Element			Any special physical conditions that can affect travel choices, including allergies
PetInfo	*	Element			Information about pets that may accompany customer
PaymentForm	*	Element		%Privacy, CostCenterID, RPH	Ways of providing funds for travel by individual
RelatedTraveler	*	Element		%Privacy, Relation	Other travel profiles associated with individual
ContactPerson	*	Element		DefaultInd, %Privacy, ContactType, Relation, EmergencyFlag, RPH	Names of individual to contact
Document	*	Element		%Privacy, DocID, DocType, Gender, BirthDate, EffectiveDate, ExpireDate	Government issued documents for travel
CustLoyalty	*	Element		ProgramCode, %Privacy, MembershipID, SingleVendorInd, LoyalLevel, SignupDate, EffectiveDate, ExpireDate, RPH	Program rewarding frequent use by accumulating credits for services provided by vendors

Customer

Attributes:

Gender : (Male | Female)

Identifies the gender of the person identified in this profile. Values: Male | Female.

Deceased : (Yes | No) = No

Indicates if individual in this profile has died. Default = 'No'

LockoutType : CDATA

Indicates the reason for locking out record, such as Emergency, Incident, etc.

ShareSynchInd : (Yes | No | Inherit) = Inherit

Permission for sharing data for synchronization of profiles held by other travel service providers. Enumerated values: 'Yes', 'No', 'Inherit'. Default = 'Inherit'.

ShareMarketInd : (Yes | No | Inherit) = Inherit

Permission for sharing data for marketing information. Enumerated values: 'Yes', 'No', 'Inherit'. Default = 'Inherit'.

BirthDate : CDATA

Indicates self-professed date of birth, in ISO 8601 prescribed format, using the date only.

Currency : CDATA

Type of funds preferred for reviewing monetary values, ISO 4217 codes.

Customer DTD fragment

```
<!ELEMENT Customer (((PersonName+, TelephoneInfo*, Email*) | (TelephoneInfo+, Email*) |
(Email+)), AddressInfo*, URL*, CitizenCountryName*, PhysChallName*, PetInfo*, PaymentForm*,
RelatedTraveler*, ContactPerson*, Document*, CustLoyalty*)>
<!ATTLIST Customer
    Gender (Male | Female) #IMPLIED
    Deceased (Yes | No) "No" #IMPLIED
    LockoutType CDATA #IMPLIED
    %Ent.Privacy;
    BirthDate CDATA #IMPLIED
    Currency CDATA #IMPLIED
>
```

2.2.1 Person Name (Entity %PersonName) (Figure 6)

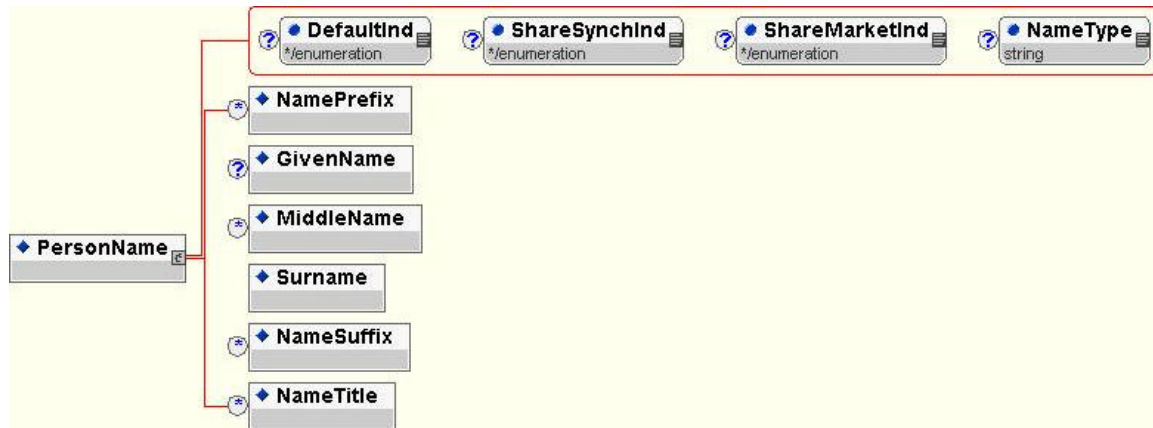


Figure 6. PersonName

Element	Occur.	Content	Data type	Attributes	Description
PersonName	*	Element		DefaultInd, %Privacy, NameType	Names of the individual, includes former, nicknames, and alternate names
NamePrefix	*	Element			Salutation of honorific
GivenName	?	Element			Given name, first name or names
MiddleName	*	Element			Middle name or names
SurName		Element			Family name, last name
NameSuffix	*	Element			Suffix (e.g. Jr. Sr. III)
NameTitle	*	Element			Degree or honors

PersonName

Attributes:

DefaultInd : (Yes | No)

Indicates that the receiving system should assume the default value if the user specifies no overriding value or action.

ShareSynchInd : (Yes | No | Inherit) = Inherit

Permission for sharing data for synchronization of profiles held by other travel service providers. Enumerated values: 'Yes', 'No', 'Inherit'. Default = 'Inherit'.

ShareMarketInd : (Yes | No | Inherit) = Inherit

Permission for sharing data for marketing information. Enumerated values: 'Yes', 'No', 'Inherit'. Default = 'Inherit'.

NameType : CDATA

Type of name of the individual, such as former, nickname, alternate or alias names.

PersonName DTD fragment

```
<!ELEMENT PersonName ( NamePrefix*, GivenName?, MiddleName*, Surname,
NameSuffix*, NameTitle*)>
<!ATTLIST PersonName
    DefaultInd (Yes | No) #IMPLIED
    %Ent. Privacy;
    NameType CDATA #IMPLIED
>
```

NamePrefix

Salutation or honorific (e.g., Mr., Mrs., Ms., Miss, Dr., etc.).

NamePrefix DTD fragment

```
<!ELEMENT NamePrefix (#PCDATA)>
>
```

GivenName

Given name, first name or names.

GivenName DTD fragment

```
<!ELEMENT GivenName (#PCDATA)>
>
```

MiddleName

Middle name or names.

MiddleName DTD fragment

```
<!ELEMENT MiddleName (#PCDATA)>
>
```

Surname

Family name, last name (REQUIRED ELEMENT).

Surname DTD fragment

```
<!ELEMENT Surname (#PCDATA)>
>
```

NameSuffix

Suffix that follows the surname (e.g., Jr., Sr., III, Ret., Esq., etc.).

NameSuffix DTD fragment

```
<!ELEMENT NameSuffix (#PCDATA)>
>
```

NameTitle

Degree or honors, (e.g.: Ph.D., M.D., etc.) often used with a person's name.

NameTitle DTD fragment

```
<!ELEMENT NameTitle (#PCDATA)>
```

2.2.2 Telephone Information

The TelephoneInfo element is used to store the telephone numbers of the customer identified in the profile, including home, business, mobile phone or other telephone number, defining the association with the individual in the context of the use of the phone.

Element	Occur.	Content	Data type	Attributes	Description
TelephoneInfo	*	Element, % Telephone		DefaultInd, %Privacy, PhoneUseType	Telephone numbers of the individual in the profile
%Telephone					

TelephoneInfo

Information about the telephone numbers of the individual or company in the profile.

Attributes:

DefaultInd : (Yes | No)

The value that the receiving system should assume if the user specifies no overriding value or action.

ShareSynchInd : (Yes | No | Inherit) = Inherit

Permission for sharing data for synchronization of profiles held by other travel service providers. Enumerated values: 'Yes', 'No', 'Inherit'. Default = 'Inherit'.

ShareMarketInd : (Yes | No | Inherit) = Inherit

Permission for sharing data for marketing information. Enumerated values: 'Yes', 'No', 'Inherit'. Default = 'Inherit'.

PhoneUseType : CDATA

Describes the type of telephone number, in the context of its general use, such as Home, Business, Emergency Contact, Travel Arranger, Day, Evening, etc.

TelephoneInfo DTD fragment

```
<ELEMENT TelephoneInfo (%Ent.Telephone;)>
<!ATTLIST TelephoneInfo
  DefaultInd (Yes | No) #IMPLIED
  %Ent.Privacy;
  PhoneUseType CDATA #IMPLIED
>
```

2.2.3 Telephone (Entity %Telephone) (Figure 7)



Figure 7. Telephone

Element	Occur.	Content	Data type	Attributes	Description
Telephone	*	Element		PhoneTechType, CountryAccessCode, AreaCityCode, PhoneNumber, Extension, PIN	Telephone numbers of the individual in the profile

Telephone

Telephone numbers for the parties in the profile.

Attributes:

PhoneTechType : CDATA

Indicates type of technology associated with this telephone number, such as Voice, Data, Fax, Pager, Cell, TTY.

CountryAccessCode : CDATA

Code assigned by telecommunications authorities for international country access identifier.

AreaCityCode : CDATA

Code assigned for telephones in a specific region, city, or area.

PhoneNumber : CDATA

Telephone number assigned to a single location.

Extension : CDATA

Extension to reach a specific party at the phone number.

PIN : CDATA

Additional codes used for pager or telephone access rights.

Telephone DTD fragment

```
<!ELEMENT Telephone EMPTY>
<!ATTLIST Telephone
  PhoneTechType CDATA #IMPLIED
  CountryAccessCode CDATA #IMPLIED
  AreaCityCode CDATA #REQUIRED
  PhoneNumber CDATA #REQUIRED
  Extension CDATA #IMPLIED
  PIN CDATA #IMPLIED>
```

2.2.4 Customer Email

Element	Occur.	Content	Data type	Attributes	Description
Email	*	Text	String	DefaultInd, %Privacy, EmailType	Electronic mail addresses, in IETF specified format

Email

Electronic mail addresses, in IETF specified format.

Attributes:**DefaultInd : (Yes | No)**

Indicates that the receiving system should assume the default value if the user specifies no overriding value or action.

ShareSynchInd : (Yes | No | Inherit) = Inherit

Permission for sharing data for synchronization of profiles held by other travel service providers. Enumerated values: 'Yes', 'No', 'Inherit'. Default = 'Inherit'.

ShareMarketInd : (Yes | No | Inherit) = Inherit

Permission for sharing data for marketing information. Enumerated values: 'Yes', 'No', 'Inherit'. Default = 'Inherit'.

EmailType : CDATA

Defines the purpose of the Email address, such as personal, business, listserve, etc.

Email DTD fragment

```
<!ELEMENT Email (#PCDATA)>
<!ATTLIST Email
  DefaultInd (Yes | No) #IMPLIED
  %Ent.Privacy;
  EmailType CDATA #IMPLIED
>
```

2.2.5 Customer Address

Element	Occur.	Content	Data type	Attributes	Description
AddressInfo	*	Element		DefaultInd %Privacy, AddressType	Precise location of the individual in the profile for mailing and delivery
% Address					

AddressInfo

Information about the physical addresses contained in the profile.

Attributes:

DefaultInd : (Yes | No)

Indicates that the receiving system should assume the default value if the user specifies no overriding value or action.

ShareSynchInd : (Yes | No | Inherit) = Inherit

Permission for sharing data for synchronization of profiles held by other travel service providers. Enumerated values: 'Yes', 'No', 'Inherit'. Default = 'Inherit'.

ShareMarketInd : (Yes | No | Inherit) = Inherit

Permission for sharing data for marketing information. Enumerated values: 'Yes', 'No', 'Inherit'. Default = 'Inherit'.

AddressType : (Home | Business | Delivery | Mailing | Billing | CreditCard | Other)

Defines what the purpose of the address is, such as home, business, mailing, delivery, shipping/receiving address, etc.

AddressInfo DTD fragment

```
<!ELEMENT AddressInfo (%Ent.Address;)>
<!ATTLIST AddressInfo
    DefaultInd (Yes | No) #IMPLIED
    %Ent.Privacy;
    AddressType (Home | Business | Delivery | Mailing | Billing | CreditCard | Other) #IMPLIED
>
```

ENTITY %Address

Uses the Address entity to define the mailing or delivery location where invoices are sent.

2.2.6 Address (Entity %Address) (Figure 8)

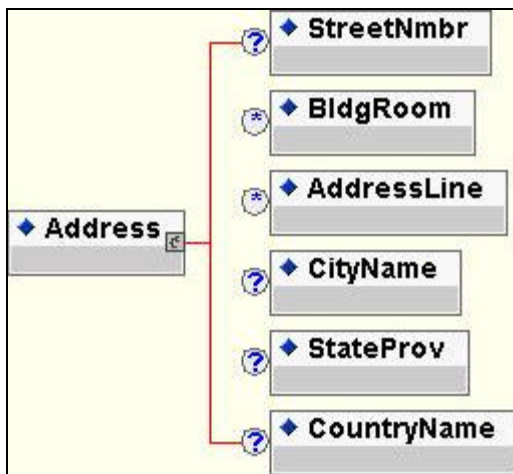


Figure 8. Address

Element	Occur.	Content	Data type	Attributes	Description
Address	*	Element			Defines a physical address of parites in the profile
StreetNmbr	?	Element			Street name, number on street
BldgRoom	*	Element			Building name, room, apartment, or suite number
AddressLine	*	Element			Additional line of an address as

					needed to define a physical location
CityName	?	Element			Name of city or town
StateProv	?	Element			State, province, or region name or code needed to identify
CountryName	?	Element			The name of the country specified in the address

Address

Defines a physical address of parties to the profile. May also be used as a postal address.

Address DTD fragment

```
<!ELEMENT Address (StreetNmbr?, BldgRoom*, AddressLine*, CityName?, StateProv?,
CountryName?,)>
```

StreetNmbr

Street name; number on street.

Attributes:

PO_Box : CDATA

Box, drawer, or route location reserved for postal delivery.

StreetNmbr DTD fragment

```
<!ELEMENT StreetNmbr (#PCDATA)>
<!ATTLIST StreetNmbr
    POBox CDATA #IMPLIED
>
```

BldgRoom

Building name; room, apartment, or suite number.

BldgRoom DTD fragment

```
<!ELEMENT BldgRoom (#PCDATA)>
>
```

AddressLine

Additional line of an address as needed to define a physical location. May be repeated.

Attributes:

ParsedInd : (Yes | No)

Indicates if the data in the AddressLine field is parsed. Systems should use the AddressLine element to send unparsed data, as the AddressLine element is repeatable.

AddressLine DTD fragment

```
<!ELEMENT AddressLine (#PCDATA)>
<!ATTLIST AddressLine
    ParsedInd (Yes | No) #IMPLIED
>
```

CityName

Name of city or town.

Attributes:

PostalCode : CDATA

Identifier for location assigned by postal authorities.

CityName DTD fragment

```
<!ELEMENT CityName (#PCDATA)>
```

```

<!ATTLIST CityName
    PostalCode CDATA #IMPLIED
>

```

StateProv

State, province, or region name or code that identifies a location.

Attributes:

StateCode : CDATA

The postal service standard code or abbreviation for the state, province or region.

StateProv DTD fragment

```

<!ELEMENT StateProv (#PCDATA)>
<!ATTLIST StateProv
    StateCode CDATA #IMPLIED
>

```

CountryName

The name of the country specified in the address.

Attributes:

CountryCode : CDATA

ISO 3166 code for country in address.

CountryName DTD fragment

```

<!ELEMENT CountryName (#PCDATA)>
<!ATTLIST CountryName
    CountryCode CDATA #IMPLIED
>

```

2.2.7 URL Element

Element	Occur.	Content	Data type	Attributes	Description
URL	*	Text	String	DefaultInd, %Privacy URL_Type	Web site addresses, in IETF specified format

URL

Web site address, in IETF specified format.

Attributes:

DefaultInd : (Yes | No)

Indicates that the receiving system should assume the default value if the user specifies no overriding value or action.

ShareSynchInd : (Yes | No | Inherit) = Inherit

Permission for sharing data for synchronization of profiles held by other travel service providers. Enumerated values: 'Yes', 'No', 'Inherit'. Default = 'Inherit'.

ShareMarketInd : (Yes | No | Inherit) = Inherit

Permission for sharing data for marketing information. Enumerated values: 'Yes', 'No', 'Inherit'. Default = 'Inherit'.

URL_Type : CDATA

Defines the purpose of the URL address, such as personal, business, public, etc.

URL DTD fragment

```

<!ELEMENT URL (#PCDATA)>
<!ATTLIST URL
    DefaultInd (Yes | No) #IMPLIED

```

```
%Ent.Privacy;
URL_Type CDATA #IMPLIED
>
```

2.2.8 Customer Country of Citizenship

Element	Occur.	Content	Data type	Attributes	Description
CitizenCountryName	*	Text	String	CountryCode DefaultInd,	Self-professed country or countries that customer claims for citizenship

CitizenCountryName

Name of the (self-professed) country or countries that customer claims for citizenship.

Attributes:

DefaultInd : (Yes | No)

Indicates that the receiving system should assume the default value if the user specifies no overriding value or action.

CountryCode : CDATA

ISO 3166 code for country in address.

CitizenCountryName DTD fragment

```
<ELEMENT CitizenCountryName (#PCDATA)>
<!ATTLIST CitizenCountryName
  DefaultInd (Yes | No) #IMPLIED
  CountryCode CDATA #IMPLIED
>
```

2.2.9 Physical Challenges

Element	Occur.	Content	Data type	Attributes	Description
PhysChallName	*	Text	String		Any special physical conditions that can affect travel choices, including allergies

PhysChallName

Any special physical conditions that can affect travel choices, including allergies.

PhysChallName DTD fragment

```
<ELEMENT PhysChallName (#PCDATA)>
```

2.2.10 Pet Information

Element	Occur.	Content	Data type	Attributes	Description
PetInfo	*	Text	String		Information about pets that may accompany customer while traveling

PetInfo

Information about pets that may accompany customer while traveling.

PetInfo DTD fragment

<ELEMENT PetInfo (#PCDATA)>

2.2.11 Payment Form and Loyalty Program elements

Several places in the profile use data on payment methods and loyalty programs, such as frequent flyer or frequent guest programs. Unlike reusable objects, such as `PersonName`, where only the structure gets re-used, in these cases the data as well as the structure have the potential for repeating in a number places in the profile as well as in other messages, such as reservations. As a result, the OTA profile specification holds information about payment data in the `PaymentForm` element (Figure 9) and information about loyalty program data in the `Loyalty` element, so that the customer's entries only need to appear once.

Each of the items that identify a payment form or loyalty program membership are assigned a Reference Place Holder (RPH) attribute that uniquely identifies the element within a collection of payment forms or loyalty programs. When it is necessary to refer to the data stored in the Customer section of the profile from other sections of the profile record, a reference is made to the RPH or index number of that item. Future use of RPHs in OTA specifications can be anticipated for attaching payment forms or loyalty programs to reservations for travel services, etc. In that way, such of these elements can be referenced in other parts of the profile, such as in the Preferences collection, rather than repeated.

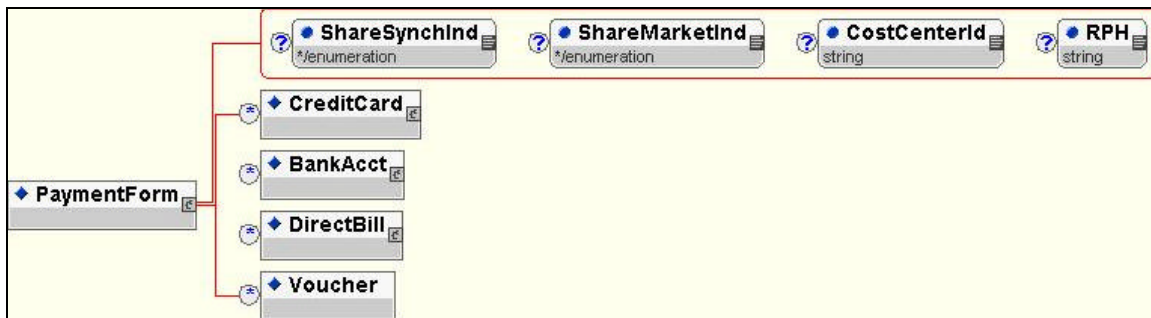


Figure 9. Payment Form

Element	Occur.	Content	Data type	Attributes	Description
PaymentForm	*	Element		DefaultInd, %Privacy CostCenterId, RPH	Ways of providing funds for travel by the individual
CreditCard	*	Element		%Privacy, CardType, CreditCardCode, CardNumber, SeriesCode, EffectiveDate, ExpireDate	Payment accounts authorizing purchases represented by plastic cards often with magnetic strips with data for identification
BankAcct	*	Element		%Privacy, BankID, AcctType, BankAcctNumber	Customer bank accounts for payments either for paper checks or electronic funds transfer
DirectBill	*	Element		%Privacy,	Company name and location

				DirectBill	for sending invoice for remittances for travel services
Voucher	*	Element		EffectiveDate, ExpireDate, SeriesCode	A form of payment using a voucher or coupon

PaymentForm

Attributes:

DefaultInd : (Yes | No)

Indicates that the receiving system should assume the default value if the user specifies no overriding value or action.

ShareSynchInd : (Yes | No | Inherit) = Inherit

Permission for sharing data for synchronization of profiles held by other travel service providers. Enumerated values: 'Yes', 'No', 'Inherit'. Default = 'Inherit'.

ShareMarketInd : (Yes | No | Inherit) = Inherit

Permission for sharing data for marketing information. Enumerated values: 'Yes', 'No', 'Inherit'. Default = 'Inherit'.

CostCenterId : CDATA

Code for allocating cost of travel to company accounts.

RPH : CDATA

A Reference Place Holder that serves as an index to the collection of payment forms stored in the customer's profile record. The RPH may be used to reference a method of payment from outside the profile.

PaymentForm DTD fragment

```
<!ELEMENT PaymentForm (CreditCard*, BankAcct*, DirectBill*, Voucher*)>
<!ATTLIST PaymentForm
  %Ent.Privacy;
  CostCenterId CDATA #IMPLIED
  RPH CDATA #IMPLIED >
```

2.2.12 CreditCard elements and attributes (Figure 10)

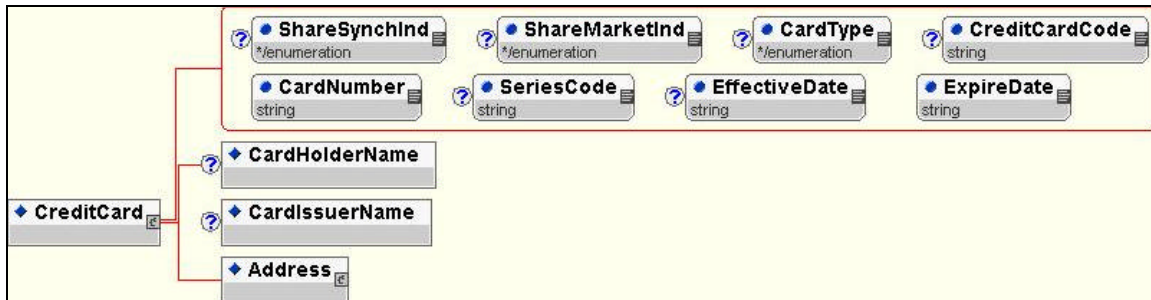


Figure 10. Credit Card

Element	Occur.	Content	Data type	Attributes	Description
CreditCard	*	Element		%Privacy CardType, CreditCardCode CardNumber, SeriesCode, EffectiveDate ExpireDate	Payment accounts authorizing purchases represented by plastic cards often with magnetic strips with data for identification

CardHolderName	?	Text	String	IssuingBankId	Name string of individual as embossed on the card
CardIssuerName	?	Text	String		Company or organization that issued the card
%Address					Mailing or delivery address that defines location to where invoices are sent.

CreditCard

Payment accounts authorizing purchases represented by plastic cards with magnetic strips with data for identification.

Attributes:

ShareSynchInd : (Yes | No | Inherit) = Inherit

Permission for sharing data for synchronization of profiles held by other travel service providers. Enumerated values: 'Yes', 'No', 'Inherit'. Default = 'Inherit'.

ShareMarketInd : (Yes | No | Inherit) = Inherit

Permission for sharing data for marketing information. Enumerated values: 'Yes', 'No', 'Inherit'. Default = 'Inherit'.

CardType : (Credit | Debit | CentralBill)

Indicates the type of magnetic striped card. Values: (Credit | Debit | CentralBill)

CreditCardCode : CDATA

Code indicating the type of credit card. (e.g., MasterCard, American Express, Visa, etc.), or the type of central bill/debit card.

CardNumber : CDATA

Credit card number embossed on the card.

SeriesCode : CDATA

Verification digits printed on the card following the embossed number.

EffectiveDate : CDATA

Date the card becomes valid for use in ISO 8601 prescribed format.

ExpireDate : CDATA

Date the card expires and can no longer be used in ISO 8601 prescribed format.

CreditCard_DTD fragment

```
<!ELEMENT CreditCard (CardHolderName?, CardIssuerName?, %Ent.Address;?)>
<!ATTLIST CreditCard
    %Ent.Privacy;
    CardType (Credit | Debit | CentralBill) #IMPLIED
    CreditCardCode CDATA #IMPLIED
    CardNumber CDATA #REQUIRED
    SeriesCode CDATA #IMPLIED
    EffectiveDate CDATA #IMPLIED
    ExpireDate CDATA #REQUIRED>
```

CardHolderName

Name of individual cardholder as embossed on the credit/debit card.

CardHolderName_DTD fragment

```
<!ELEMENT CardHolderName (#PCDATA)>
```

CardIssuerName

Name of bank or organization issuing the card. e.g.: alumni association, bank, fraternal organization, etc.

Attributes:**IssuingBankId** : CDATA

Code of bank issuing the card.

CardIssuerName_DTD fragment

```

<!ELEMENT CardIssuerName (#PCDATA)>
<!ATTLIST CardIssuerName
    IssuingBankId CDATA #IMPLIED
>

```

ENTITY %Address

Uses the Address entity to define the mailing or delivery location where invoices are sent.

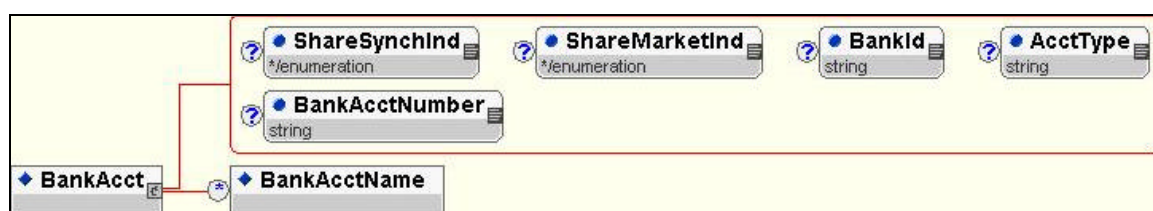
2.2.13 Bank Account (Figure 11)

Figure 11. Bank Account

Element	Occur.	Content	Data type	Attributes	Description
BankAcct	*	Element		%Privacy BankId, AcctType, BankAcctNumber,	Customer bank accounts for payments, either for paper checks or electronic funds transfers
BankAcctName	*	Text	String		Names of the individuals as they appear on the bank account.

BankAcct

Customer bank accounts for payments, either for paper checks or electronic funds transfers.

Attributes:**ShareSynchInd** : (Yes | No | Inherit) = Inherit

Permission for sharing data for synchronization of profiles held by other travel service providers. Enumerated values: 'Yes', 'No', 'Inherit'. Default = 'Inherit'.

ShareMarketInd : (Yes | No | Inherit) = Inherit

Permission for sharing data for marketing information. Enumerated values: 'Yes', 'No', 'Inherit'. Default = 'Inherit'.

BankId : CDATA

Code assigned by authorities to financial institutions; sometimes called bank routing number.

AcctType : CDATA

Describes the bank account used for financing travel, such as: Checking, Savings, Investment, etc.

BankAcctNumber : CDATA

Identifier for the account assigned by the bank.

BankAcct_DTD fragment

```
<!ELEMENT BankAcct (BankAcctName* )>
<!ATTLIST BankAcct
    %Ent.Privacy;
    BankId CDATA #IMPLIED
    AcctType CDATA #IMPLIED
    BankAcctNumber CDATA #IMPLIED
>
```

BankAcctName

Names of the individuals as listed on the bank account.

BankAcctName_DTD fragment

```
<!ELEMENT BankAcctName (#PCDATA)>
```

2.2.14 Direct Bill

Element	Occur.	Content	Data type	Attributes	Description
DirectBill	*	Element		%Privacy. DirectBill_Id	Company name and location for sending invoice for remittances for travel services.
%CompanyName	R	Entity	String		Name of organization to where invoices are sent
%Address	R	Entity	String		Precise mailing or delivery location to where invoices are sent

DirectBill

Company name and location for sending invoice for remittances for travel services.

Attributes:

ShareSynchInd : (Yes | No | Inherit) = Inherit

Permission for sharing data for synchronization of profiles held by other travel service providers. Enumerated values: 'Yes', 'No', 'Inherit'. Default = 'Inherit'.

ShareMarketInd : (Yes | No | Inherit) = Inherit

Permission for sharing data for marketing information. Enumerated values: 'Yes', 'No', 'Inherit'. Default = 'Inherit'.

DirectBill_Id : CDATA

Identifier for the organization to be billed directly for travel services.

DirectBill_DTD fragment

```
<!ELEMENT DirectBill (%Ent.CompanyName;, %Ent.Address;)>
<!ATTLIST DirectBill
    %Ent.Privacy;
    DirectBill_Id CDATA #REQUIRED
>
```

ENTITY %CompanyName

Uses the CompanyName entity to indicate the name of organization to which invoices are sent.

ENTITY %Address

Uses the Address entity to define the mailing or delivery location where invoices are sent.

2.2.15 Voucher

The Voucher element, a child element of PaymentForm, is used to identify a series of vouchers or coupons used for payment of travel services. The Voucher element is not likely to be used for a customer's personal profile record.

Element	Occur.	Content	Data type	Attributes	Description
Voucher	*	Element		EffectiveDate, ExpireDate, SeriesCode	Information about voucher program used to pay for travel services.

Voucher

A form of payment using a voucher or coupon.

Attributes:

EffectiveDate : CDATA

The date when a voucher becomes valid for use, if applicable.

ExpireDate : CDATA

The date when a voucher or series of coupons expires and is no longer valid.

SeriesCode : CDATA

Identification of a series of coupons or vouchers identified by serial number(s).

Voucher DTD fragment

```
<!ELEMENT Voucher (#PCDATA)>
<!ATTLIST Voucher
  EffectiveDate CDATA #IMPLIED
  ExpireDate CDATA #IMPLIED
  SeriesCode CDATA #IMPLIED
>
```

2.2.16 Related Traveler

Element	Occur.	Content	Data type	Attributes	Description
RelatedTraveler	*	Element		%Privacy, Relation	Other traveler profiles associated with the individual
%UniqueId	R	% UniqueId			Unique identifier assigned to individual related to the customer in the profile
%PersonName	?	%PersonName			Name of individual related to traveler

RelatedTraveler

Additional traveler profile associated with the individual in this profile.

Attributes:

ShareSynchInd : (Yes | No | Inherit) = Inherit

Permission for sharing data for synchronization of profiles held by other travel service providers. Enumerated values: 'Yes', 'No', 'Inherit'. Default = 'Inherit'.

ShareMarketInd : (Yes | No | Inherit) = Inherit

Permission for sharing data for marketing information. Enumerated values: 'Yes', 'No', 'Inherit'. Default = 'Inherit'.

Relation : CDATA

Indicates type of the relationship with the person in the profile, such as Spouse, Child, Family, Business Associate, Interest Group, Medical, Security, Other, etc.

RelatedTraveler DTD fragment

```
<!ELEMENT RelatedTraveler (%Ent.UniqueId;, %Ent.PersonName;)>
<!ATTLIST RelatedTraveler
    %Ent.Privacy;
    Relation CDATA #IMPLIED
>
```

ENTITY %UniqueID

Uses the UniqueID entity to uniquely identify a Profile record, identify an individual related to the primary entity in a profile.

ENTITY %PersonName

Uses the PersonName entity to provide names of the individual, includes former, nicknames, and alternate names.

2.2.17 UniqueId (Entity %UniqueId)

The UniqueId element (Figure 12), is used to identify a Profile record, can also be used to identify an individual related to the primary entity in a profile. The UniqueId may identify a profile for a related person.



Figure 12. Unique Id

Element	Occur.	Content	Data type	Attributes	Description
UniqueId	R	% UniqueId		Type, Id, URL, Instance	Unique identifier assigned to an individual customer profile

UniqueId

The UniqueId ENTITY is a globally unique identifier assigned to a profile record.

Attributes:

Type : CDATA

A reference to the type of object defined by the UniqueId element. By convention, the Type attribute value is the same as the element tag name for the referenced object, for example, "Profile".

Id : ID

A unique identifying value assigned by the creating system, using the XML data type ID. The Id attribute may be used to reference a primary-key value within a database or in a particular implementation.

URL : CDATA

URL that identifies the location of the record identified by the UniqueID. The URL referenced is recommended to be the public URL used for an OTA discovery message between trading partners.

Instance : CDATA

A snapshot of a record as it exists at a point in time, indicated by a value such as a timestamp, monotonically increasing sequence, etc.. An Instance is used in update messages where the sender must assure the server that the update sent refers to the most recent modification level of the object being updated.

UniqueId DTD fragment

```
<!ELEMENT UniqueId EMPTY>
<!ATTLIST UniqueId
  URL CDATA #IMPLIED
  Type CDATA #REQUIRED
  Id ID #IMPLIED
  Instance CDATA #IMPLIED
>
```

2.2.18 Contact Person elements and attributes

The Contact Person element (Figure 13) identifies a person whom travel services should contact on behalf of the customer. This may be a travel agent, employer, relative or other person not traveling with the customer, and if so indicated may be contacted in case of emergency.

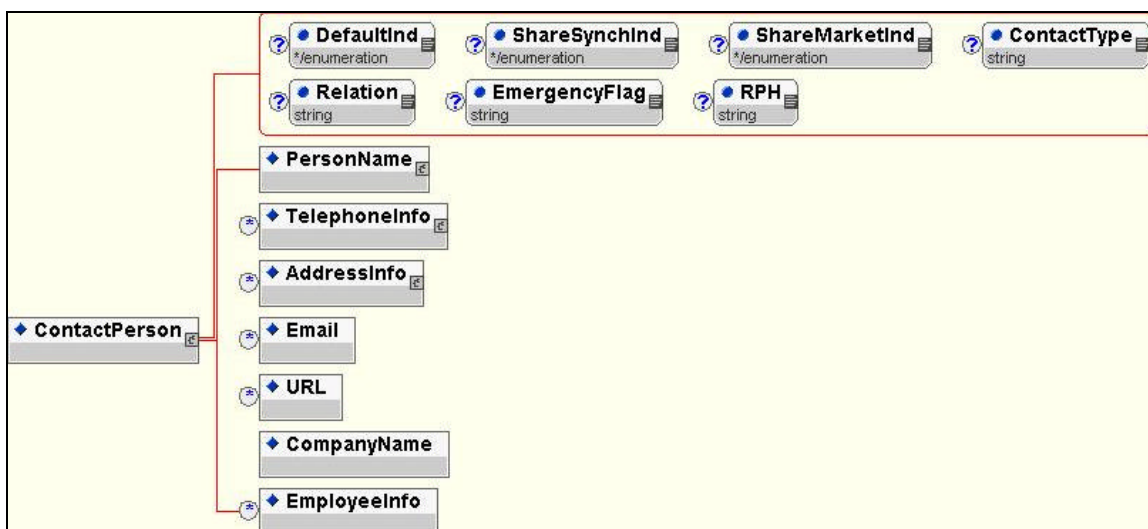


Figure 13. Contact Person

Element	Occur.	Content	Data type	Attributes	Description
ContactPerson	*	Element		DefaultInd %Privacy ContactType, Relation EmergencyFlag, RPH	Persons who travel services should contact on behalf of the individual in the profile
% PersonName	?	Element			Names of individuals to contact
TelephoneInfo	*	Element % Telephone		DefaultID, %Privacy, PhoneUseType	Telephone numbers, including, fax or pager, of persons to contact
AddressInfo	*	Element % Address		DefaultID, %Privacy,	Addresses of persons to contact

				AddressType	
Email	*	Text	String	DefaultInd, %Privacy, EmailType	Email addresses of contact person or entity
URL	*	Text	String	DefaultInd, %Privact, URL_Type	Web site addresses of contact person or entity
% CompanyName	*	Element		CompanyCode, CodeContext	Company associated with the contact.
EmployeeInfo	*	Element	String	EmployeeId, EmployeeLevel, EmployeeTitle, EmployeeStatusT ype	Employment information about the contact person.

ContactPerson

Person or entity with whom travel services should communicate when it is necessary to interact with a contact on behalf of the customer.

Attributes:

DefaultInd : (Yes | No)

Indicates the default among a choice of more than one. The receiving system should assume this default value if the user specifies no overriding value or action.

ShareSynchInd : (Yes | No | Inherit) = Inherit

Permission for sharing data for synchronization of profiles held by other travel service providers. Enumerated values: 'Yes', 'No', 'Inherit'. Default = 'Inherit'.

ShareMarketInd : (Yes | No | Inherit) = Inherit

Permission for sharing data for marketing information. Enumerated values: 'Yes', 'No', 'Inherit'. Default = 'Inherit'.

ContactType : CDATA

Type of contact in the context of use for the travel experience; such as permanent, temporary, affiliation, travel arranger, etc.

Relation : CDATA

Indicates type of the relationship with the person in the profile, such as Spouse, Children, Family, Business, InterestGroup, Medical, Security, Other.

EmergencyFlag : (Yes | No)

Indicates if this contact should be used in the case of an emergency.

RPH : CDATA

A Reference Place Holder that serves as an index to the collection of contacts stored in the customer's profile record. The RPH may be used to reference a contact from outside the profile, e.g.: in a reservation.

ContactPerson DTD fragment

```
<!ELEMENT ContactPerson (%Ent.PersonName;, TelephoneInfo*, AddressInfo*, Email*, URL*,
%Ent.CompanyName;*, EmployeeInfo*)>
<!ATTLIST ContactPerson
  DefaultInd (Yes | No) #IMPLIED
  %Ent.Privacy;
  ContactType CDATA #IMPLIED
  Relation CDATA #IMPLIED
  EmergencyFlag CDATA #IMPLIED
  RPH CDATA #IMPLIED
>
```

TelephoneInfo

Telephone numbers, including, fax or pager, of contact person.

Attributes:

DefaultInd : (Yes | No)

Indicates the default among a choice of more than one. The receiving system should assume this default value if the user specifies no overriding value or action.

ENTITY %Ent.Privacy;

The Privacy ENTITY contains attributes for determining permission to share data in the element for synchronization of profiles held by other travel service providers and for marketing information.

PhoneUseType : CDATA

Describes the type of telephone number, by its general use or time of day such as Home, Business, Day, Evening, Emergency Contact, Travel Arranger, etc.

TelephoneInfo DTD fragment

```
<!ELEMENT TelephoneInfo (%Ent.Telephone;)>
<!--ATTLIST TelephoneInfo
  DefaultInd (Yes | No) #IMPLIED
  %Ent.Privacy;
  PhoneUseType CDATA #IMPLIED
-->
```

AddressInfo

Information about the physical or mailing address of the contact person.

Attributes:

DefaultInd : (Yes | No)

Indicates that the receiving system should assume the default value if the user specifies no overriding value or action.

ENTITY %Ent.Privacy;

The Privacy ENTITY contains attributes for determining permission to share data in the element for synchronization of profiles held by other travel service providers and for marketing information.

AddressType : CDATA

Defines what the purpose of the address is, such as home, business, mailing, delivery, shipping/receiving address, etc.

AddressInfo DTD fragment

```
<!ELEMENT AddressInfo (%Ent.Address;)>
<!--ATTLIST AddressInfo
  DefaultInd (Yes | No) #IMPLIED
  %Ent.Privacy;
  AddressType (Home | Business | Delivery | Mailing | Billing | CreditCard | Other) #IMPLIED
-->
```

Email

Electronic mail addresses of the contact person.

Attributes:

DefaultInd : (Yes | No)

Indicates that the receiving system should assume the default value if the user specifies no overriding value or action.

ENTITY %Ent.Privacy;

The Privacy ENTITY contains attributes for determining permission to share data in the element for synchronization of profiles held by other travel service providers and for marketing information.

EmailType : CDATA

Defines the purpose of the Email address, such as personal, business, listserve, etc.

Email DTD fragment

```

<!ELEMENT Email (#PCDATA)>
<!--ATTLIST Email
DefaultInd (Yes | No) #IMPLIED
%Ent.Privacy;
EmailType CDATA #IMPLIED
-->

```

URL

Web site address of the contact person, in IETF specified format.

Attributes:**DefaultInd : (Yes | No)**

Indicates that the receiving system should assume the default value if the user specifies no overriding value or action.

ENTITY %Ent.Privacy;

The Privacy ENTITY contains attributes for determining permission to share data in the element for synchronization of profiles held by other travel service providers and for marketing information.

URL_Type : CDATA

Defines the purpose of the URL address, such as personal, business, public, etc.

URL DTD fragment

```

<!ELEMENT URL (#PCDATA)>
<!--ATTLIST URL
DefaultInd (Yes | No) #IMPLIED
%Ent.Privacy;
URL_Type CDATA #IMPLIED
-->

```

EmployeeInfo

Employment information about the contact person. The PCDATA field of EmployeeInfo can be used to record the exact name of the employee if it differs from that of the individual identified as a customer's contact person.

Attributes:**EmployeeId : CDATA**

Identifier assigned to the person by the employer.

EmployeeLevel : CDATA

Level in employer organization (e.g. seniority) that conveys privileges.

EmployeeTitle : CDATA

Title of employee in employer company that conveys rank or privileges.

EmployeeStatusType : (Active | Retired | Leave | Terminated)

Status of employment. Values: Active, Retired, Leave, Terminated.

EmployeeInfo DTD fragment

```

<!ELEMENT EmployeeInfo (#PCDATA)>
<!--ATTLIST EmployeeInfo
EmployeeId CDATA #IMPLIED
EmployeeLevel CDATA #IMPLIED
EmployeeTitle CDATA #IMPLIED
EmployeeStatusType (Active | Retired | Leave | Terminated) #IMPLIED
-->

```

ENTITY %CompanyName

Identifies a company or organization associated with the contact.

ENTITY %PersonName

Uses the PersonName entity to provide names of the individual, includes former, nicknames, and alternate names.

2.2.19 Document elements and attributes (Figure 14)

Information about documents used for travel, such as passports or driver's licenses, can be stored in the customer section of the profile, and used when needed for travel services.

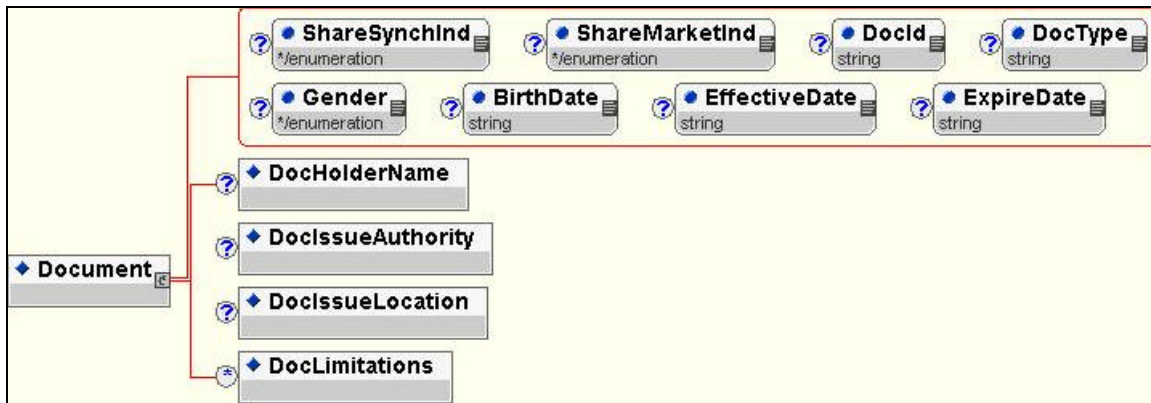


Figure 14. Document

Element	Occur.	Content	Data type	Attributes	Description
Document	*	Element	String	%Privacy, DocId, DocType, Gender, BirthDate, EffectiveDate, ExpireDate	Government-issued documents for travel
DocHolderName	?	Element	String		Name of person given on the document could be different than the PersonName.
DocIssueAuthority	?	Text	String		Government authority issuing document. If a national document use ISO 3166 code
DocIssueLocation	?	Text	String		City, state/province document in which the document was issued
DocLimitations	*	Text	String		Limitations on travel documents, such as visas for specific purposes or corrective lenses for driver's license

Document

A government-issued document used for travel.

Attributes:

ShareSynchInd : (Yes | No | Inherit) = Inherit

Permission for sharing data for synchronization of profiles held by other travel service providers. Enumerated values: 'Yes', 'No', 'Inherit'. Default = 'Inherit'.

ShareMarketInd : (Yes | No | Inherit) = Inherit

Permission for sharing data for marketing information. Enumerated values: 'Yes', 'No', 'Inherit'. Default = 'Inherit'.

DocId : CDATA

Unique identifying number assigned by authorities to document.

DocType : CDATA

Indicates the kind of travel documents held by the person, such as: Passport, Visa, Driver, VoterReg, BirthCert, SocialSecurity, MilitaryID, GovtID, Other.

Gender : (Male | Female)

Identifies the gender of person identified by the document. Values: Male | Female.

BirthDate : CDATA

Indicates date of birth as indicated in the document, in ISO 8601 prescribed format.

EffectiveDate : CDATA

Indicates the date on which the document takes effect (such as date of visa issuance), in ISO 8601 format.

ExpireDate : CDATA

Indicates the date after which the document is no longer valid, in ISO 8601 format.

Document DTD fragment

```
<!ELEMENT Document (DocHolderName?, DocIssueAuthority?, DocIssueLocation?,
DocLimitations*)>
<!ATTLIST Document
    %Ent.Privacy;
    DocId CDATA #IMPLIED
    DocType CDATA #IMPLIED
    Gender (Male | Female) #IMPLIED
    BirthDate CDATA #IMPLIED
    EffectiveDate CDATA #IMPLIED
    ExpireDate CDATA #IMPLIED
>
```

DocHolderName

Name of person as given on the document.

DocHolderName DTD fragment

```
<!ELEMENT DocHolderName (#PCDATA)
>
```

DocIssueAuthority

Government authority issuing document. If a national document, use ISO 3166 code to indicate the code of the country issuing the document.

DocIssueAuthority DTD fragment

```
<!ELEMENT DocIssueAuthority (#PCDATA)
>
```

DocIssueLocation

City, state/province, and/or country in which the document was issued.

DocIssueLocation DTD fragment

```
<!ELEMENT DocIssueLocation (#PCDATA)
>
```

DocLimitations

Limitations on travel documents, such as visas for specific purposes, restrictions for entry into certain countries, or corrective lenses for driver's license.

DocLimitations DTD fragment

```
<!ELEMENT DocLimitations (#PCDATA)>
```

2.2.20 CustLoyalty elements and attributes

The CustLoyalty element (Figure 15) identifies membership in a program rewarding frequent use by accumulating credits for services provided by vendors. Since a customer may have multiple memberships in such programs, the CustLoyalty element uses an RPH attribute to index a specific loyalty program among a collection. When used for travel services, the RPH can be used to reference a specific program stored in the profile

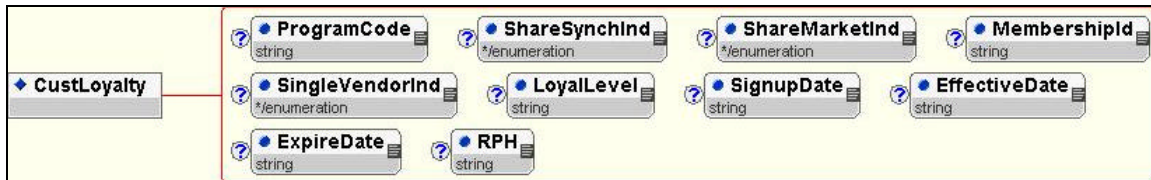


Figure 15. CustLoyalty

Element	Occur.	Content	Data type	Attributes	Description
CustLoyalty	*		String	ProgramCode %Privacy Membership Id SingleVendorIndL oyaltyLevel, SignupDate, EffectiveDate, ExpireDate, RPH	Name of the loyalty program.

CustLoyalty

Customer membership in loyalty programs such as frequent guest, frequent flier programs, etc.

Attributes:

ProgramCode : CDATA

The code of the loyalty or membership program.

ShareSynchInd : (Yes | No | Inherit) = Inherit

Permission for sharing data for synchronization of profiles held by other travel service providers. Enumerated values: 'Yes', 'No', 'Inherit'. Default = 'Inherit'.

ShareMarketInd : (Yes | No | Inherit) = Inherit

Permission for sharing data for marketing information. Enumerated values: 'Yes', 'No', 'Inherit'. Default = 'Inherit'.

MembershipId : CDATA

Unique identifier of the member in the program. ie: membership number, account number, etc.

SingleVendorInd : (SingleVndr | Alliance)

Indicates if program is affiliated with a group of related offers for accumulation of credits or points. Values: SingleVndr or Alliance.

LoyaltyLevel : CDATA

Indicates the status or level within the program, ie: VIP, Platinum, etc. This may carry certain privileges or represent a particular amount of credit awarded to the individual program member.

SignupDate : CDATA

Indicates when the member signed up for the loyalty program.

EffectiveDate : CDATA

Indicates when privileges of membership in the loyalty program starts.

ExpireDate : CDATA

Indicates when membership in the loyalty program expires.

RPH : CDATA

Reference number to be used to identify this loyalty program.

CustLoyalty DTD fragment

```
<!ELEMENT CustLoyalty (#PCDATA)>
<!ATTLIST CustLoyalty
  ProgramCode CDATA #IMPLIED
  %Ent.Privacy;
  MembershipId CDATA #IMPLIED
  SingleVendorInd (SingleVndr | Alliance) #IMPLIED
  LoyalLevel CDATA #IMPLIED
  SignupDate CDATA #IMPLIED
  EffectiveDate CDATA #IMPLIED
  ExpireDate CDATA #IMPLIED
  RPH CDATA #IMPLIED
>
```

3.0 Travel Preferences

The Preferences section (Figure 16) identifies the needs of travelers in various forms. It allows for the customer or company to gather these requirements into specified collections of preferences, and to designate a default group as well as name the groups for various kinds of travel experiences -- e.g., business, international, golf outings, overnight, family vacations, etc.. The Preferences section also provides for preferences related to individual travel services -- airlines, hotels, car rental agencies, and others. The Others category allows for extending the services into areas not yet defined.

The preference collections, element <PrefCollection>, allow the identification of common preferences as well as those associated with specific travel services. Profiles MAY have one or more of these collections. If a profile includes travel preferences, it SHOULD have services from one or more of the following element groups: <CommonPref>, <AirlinesPref>, <VehicleRentalPref>, <HotelPref>, <OtherSrvcPref> (other services) or <TPA_Extensions>.

The use of preference collections is a feature that enables travel services to be specified for each kind of travel identified. For example, a traveler may have requirements for onboard services on longer overseas flights that he or she would not need for shorter domestic flights. In this scenario, the collections could be defined for international and domestic travel, separate airline services identified in each collection. Travel to different destinations can require defining different collections. A petroleum engineer would likely need a different type of rental car when visiting wells in Alaska than when visiting the corporate headquarters in Houston.

OTA originally developed the hotel elements in OTA Customer Profile version 1 from the Windows Hospitality Interface Specifications (WHIS) and Hotel Electronic Distribution Network Association (HEDNA) element lists, as well as contributions from Hyatt and Cendant chains. This publication of Customer Profile, version 2, represents the integration of the HITIS Profile Synchronization standard with OTA into version 2. OTA work groups from the airline and car rental industries developed their respective elements over several months of effort. Their elements have been included in the modifications to the OTA model.

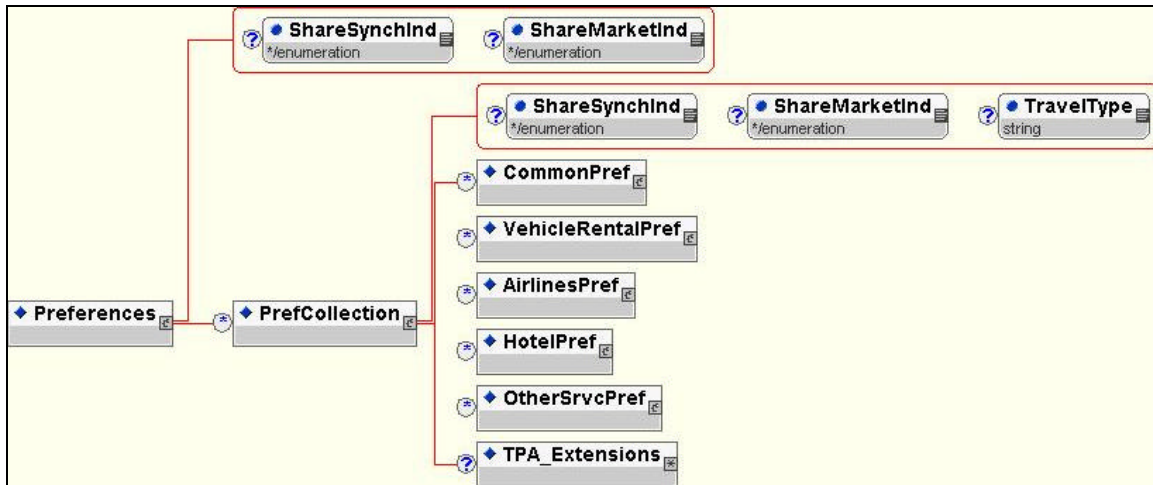


Figure 16. Preferences elements

Element	Occur	Content	Data type	Attributes	Description
Preferences	?	Element		%Privacy	Preferences related to travel experiences
PrefCollection	*	Element		%Privacy, TravelType	Unique aggregation of travel needs
CommonPref	*	Element		NamePref, PhonePref, AddressPref, PaymentFormPref, InterestPref, InsurancePref, SeatingPref, TicketDistribPref, MediaEntertainPref, PetInfoPref, MealPref, LoyaltyPref, SpecRequestPref, RelatedTravelerPref, TPA_Extensions?	Travel needs associated with a collection but independent of specific travel services
VehicleRentalPref	*	Element		LoyaltyPref, VendorPref, PaymentFormPref, VehicleCoveragePref, SpecRequestPref, VehicleTypePref, VehicleClassPref, AirConditionPref, TransmissionPref, CellPhonePref, GlobalNavPref, PhysChallEquipPref,	Vehicle rental preferences

				ChildSeatPref, TPA_Extensions	
AirlinesPref	*	Element		LoyaltyPref, VendorPref, PaymentFormPref, AirportOriginPref, AirportRoutePref, AirFareRestrictPref, FlightTypePref, AirEquipPref, AirCabinPref, AirSrvClassPref, AirSeatPref, TicketDistribPref, MediaEntertainPref, PetInfoPref, MealPref, SpecRequestPref, AirSSR_Pref, TPA_Extensions	Airlines preferences
HotelPref	*	Element		LoyaltyPref, PaymentFormPref, HotelChainPref, PropertyNamePref, PropertyLocationPref, PropertyTypePref, PropertyClassPref, PropertyAmenityPref, RoomAmenityPref, RoomLocationPref, BedTypePref, FoodSrvPref, MealPref, MediaEntertainPref, PetInfoPref, RecreationSrvPref, BusinessSrvPref, PersonalSrvPref, SecurityFeaturePref, PhysChallFeaturePref, SpecRequestPref, TPA_Extensions	Hotel preferences
OtherSrvPref	*	Element		OtherSrvName, VendorPref, LoyaltyPref, PaymentFormPref, SpecRequestPref, TPA_Extensions	Other service preferences

Preferences

The Preferences element identifies needs or preferences for a travel experience. Preferences can be gathered into collections for specific travel situations. Each group of preferences is named to identify the type of travel experience it applies to, and a default group can be designated.

Attributes:**ShareSynchInd : (Yes | No | Inherit) = Inherit**

Permission for sharing data for synchronization of profiles held by other travel service providers. Enumerated values: 'Yes', 'No', 'Inherit'. Default = 'Inherit'.

ShareMarketInd : (Yes | No | Inherit) = Inherit

Permission for sharing data for marketing information. Enumerated values: 'Yes', 'No', 'Inherit'. Default = 'Inherit'.

Preferences DTD fragment

```
<!ELEMENT Preferences (PrefCollection*)>
<!ATTLIST Preferences
    %Ent.Privacy;
>
```

PrefCollection

The element PrefCollection allows customers or companies to identify a group of preferences. Those preferences can be associated with specific travel services (airlines, car rentals, hotels) or common to all travel or non-travel experiences.

Attributes:**ShareSynchInd : (Yes | No | Inherit) = Inherit**

Permission for sharing data for synchronization of profiles held by other travel service providers. Enumerated values: 'Yes', 'No', 'Inherit'. Default = 'Inherit'.

ShareMarketInd : (Yes | No | Inherit) = Inherit

Permission for sharing data for marketing information. Enumerated values: 'Yes', 'No', 'Inherit'. Default = 'Inherit'.

TravelType : CDATA

Category of travel for this group of preferences, such as Base (non-travel), Business, Leisure, International, Domestic, Other.

PrefCollection DTD fragment

```
<!ELEMENT PrefCollection (CommonPref*, VehicleRentalPref*, AirlinesPref*, HotelPref*,
    OtherSrvcPref*, TPA_Extensions?)>
<!ATTLIST PrefCollection
    %Ent.Privacy;
    TravelType CDATA #IMPLIED
>
```

3.1 Common Preference elements and attributes

Common preferences element (Figure 17) represent those items that travel customers or companies would like associated with specific preference collections but are independent of travel services, such as airlines or hotels. These common preferences include name and contact information, related travelers, forms of payment, personal interests, loyalty programs, food and beverage needs, media and entertainment choices, pet information, and other special needs.

Data elements such as name, address, and telephone are also entered under customer information (in the Customer element). If, for example a traveler has a different name associated with this collection, such as an author's pen name used for book tours or, using the same analogy, a publisher's address and telephone rather than the author's, customers can capture those requirements with this structure. While all of the elements are OPTIONAL, any occurrence of these elements MUST occur in the designated order.

This specification structures the preferences from more general needs higher in the hierarchy to more specific requirements lower in the tree. As a rule, the lower the preferences appear in the hierarchy, the more precedence they take. For example, a loyalty program identified under car rental will take precedence for car rental preferences over loyalty programs identified under common preferences.

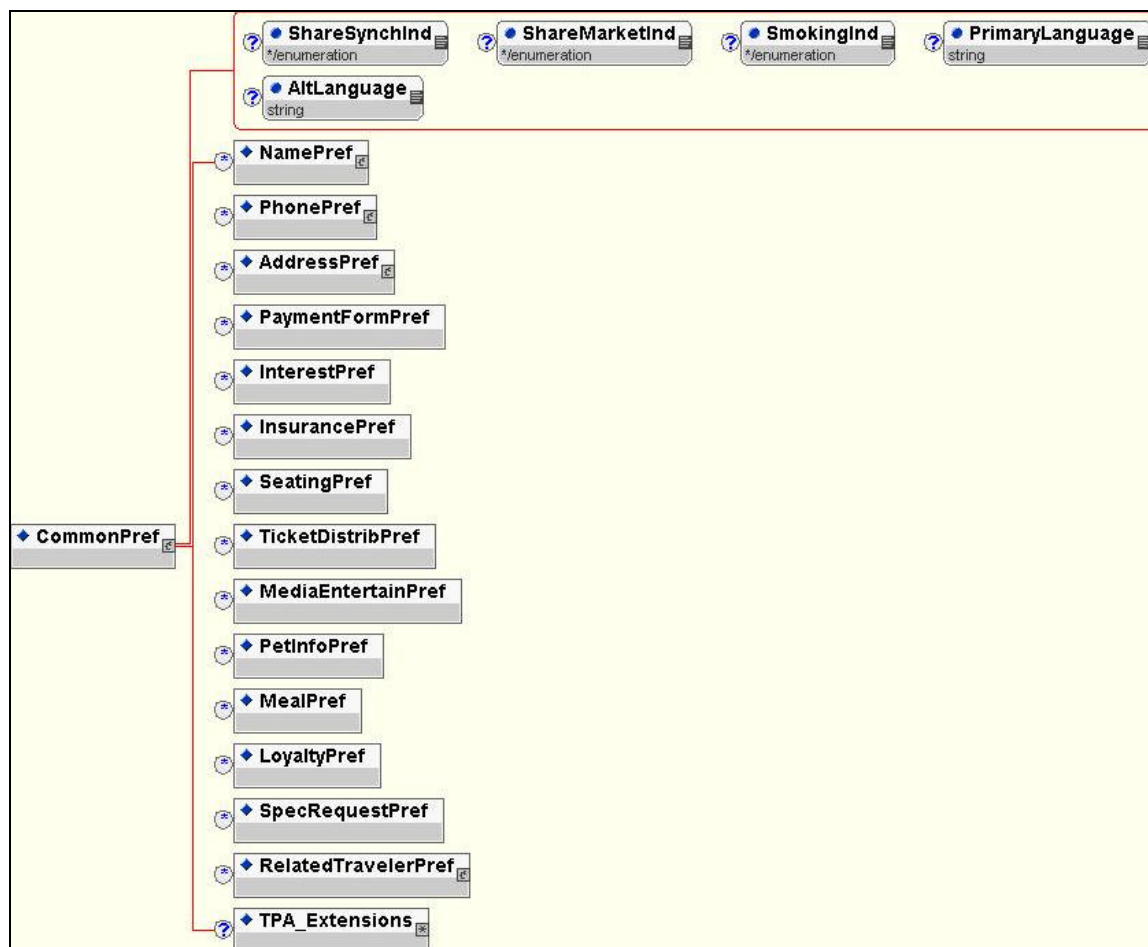


Figure 17. Common Preferences

Element	Occur.	Content	Data type	Attributes	Description
CommonPref	?	Element		%Privacy, SmokingInd, PrimaryLanguage, AltLanguage	Travel needs associated with a collection but independent of specific travel services
NamePref	*	Element		PreferLevel	Names to be used with this collection of preferences
PhonePref	*	Element		%Privacy, PhoneUseType	Telephone number(s) to be used with this collection of preferences
AddressPref	*	Element		%Privacy, AddressType	Address(s) to be used with this collection of preferences
PaymentFormPref	*	Element		PreferLevel, RPH	Form(s) of payment to be used with this collection of preferences
InterestPref	*	Element		PreferLevel	Traveler interests to be used with this collection of preferences
InsurancePref	*	Element		PreferLevel	Insurance policies to be used with this collection of preferences
SeatingPref	*	Element		PreferLevel	Seating preferences to be used with this collection of preferences
TicketDistribPref	*	Element		PreferLevel, DistribTime, TicketTime	Ticketing information to be used with this collection of preferences
MediaEntertainPref	*	Element		PreferLevel	Media and entertainment information preferences
PetInfoPref	*	Element		PreferLevel	Pet information to be used with this collection of preferences
MealPref	*	Element		PreferLevel, MealType, FavoriteFood, Beverage	Food and beverage preferences to be used with this collection of preferences
LoyaltyPref		Element		PreferLevel, LoyaltyRPH	Loyalty programs to be used with this collection of preferences
SpecRequestPref	*	Element		PreferLevel	Special requests to be used with this collection of preferences
RelatedTravelerPref	*	Element		PreferLevel	Name(s) of related travelers to be used with this collection of preferences
TPA_Extensions	?	Element			Due to the proprietary nature of agreements between business entities, this element allows for companies to add selected information as needed.

CommonPref

Travel needs associated with a collection but independent of specific travel services.

Attributes:

ShareSynchInd : (Yes | No | Inherit) = Inherit

Permission for sharing data for synchronization of profiles held by other travel service providers. Enumerated values: 'Yes', 'No', 'Inherit'. Default = 'Inherit'.

ShareMarketInd : (Yes | No | Inherit) = Inherit

Permission for sharing data for marketing information. Enumerated values: 'Yes', 'No', 'Inherit'. Default = 'Inherit'.

SmokingInd : (Yes | No)

Indicates preference of smoking or no smoking. Values: Yes | No.

PrimaryLanguage : CDATA

Identifies the primary language preference for the form of travel represented in this collection. The human language is identified by ISO 639 codes.

AltLanguage : CDATA

Identifies the alternate language preferred for the form of travel specified in this collection. The human language is identified by ISO 639 codes.

CommonPref DTD fragment

```
<!ELEMENT CommonPref (NamePref*, PhonePref*, AddressPref*, PaymentFormPref*,
InterestPref*, InsurancePref*, SeatingPref*, TicketDistribPref*, MediaEntertainPref*, PetInfoPref*,
MealPref*, LoyaltyPref*, SpecRequestPref*, RelatedTravelerPref*, TPA_Extensions?*)>
<!--ATTLIST CommonPref
    %Ent.Privacy;
    SmokingInd (Yes | No) #IMPLIED
    PrimaryLanguage CDATA #IMPLIED
    AltLanguage CDATA #IMPLIED
```

3.1.1. Name Preferences

Travelers may store more than one name in their customer profile, which allows them to use different names for travel situations. An author may use a 'pen name', or a famous person may use an assumed name that allows them to travel anonymously. The NamePref element allows for different names to be used by specifying a preference in different travel collections.

Element	Occur.	Content	Data type	Attributes	Description
NamePref	*	Element		PreferLevel	Name(s) to be used with this collection of preferences
%PersonName					
%UniqueID					

NamePref

Preferred name of traveler from profile to be used with this collection of preferences.

Attributes:**PreferLevel : (Only | Unacceptable | Preferred) = Preferred**

Identifies the level of preference for the payment form identified. Values: (Only | Unacceptable | Preferred). If Only, use only this preference for the travel situation; if Unacceptable, do NOT use this preference in the given situation. The default value is 'Preferred'.

NamePref DTD fragment

```
<!--ELEMENT NamePref (%Ent.UniqueId, %Ent.PersonName);-->
<!--ATTLIST NamePref
    PreferLevel (Only | Unacceptable | Preferred) "Preferred" #IMPLIED
-->
```

ENTITY %UniqueID

Uses the UniqueID entity to uniquely identify a Profile record, identify an individual related to the primary entity in a profile.

ENTITY %PersonName

Uses the PersonName entity to provide names of the individual, includes former, nicknames, and alternate names.

3.1.2. Telephone Preferences

The collection of telephone preferences allows for specific telephone numbers to be designated as a preference in different travel collections.

Element	Occur.	Content	Data type	Attributes	Description
PhonePref	*	Element, % Telephone		%Privacy, PhoneUseType	Telephone number(s) to be used with this collection of preferences

PhonePref

Preferred telephones (including fax and pager) to be used for this collection of preferences.

Attributes:

ShareSynchInd : (Yes | No | Inherit) = Inherit

Permission for sharing data for synchronization of profiles held by other travel service providers. Enumerated values: 'Yes', 'No', 'Inherit'. Default = 'Inherit'.

ShareMarketInd : (Yes | No | Inherit) = Inherit

Permission for sharing data for marketing information. Enumerated values: 'Yes', 'No', 'Inherit'. Default = 'Inherit'.

PhoneUseType : CDATA

Describes the type of telephone number, in the context of its general use, such as Home, Business, Emergency Contact, Travel Arranger, Day, Evening, etc.

PhonePref DTD fragment

```
<!ELEMENT PhonePref (%Ent.Telephone;)>
<!-- ATTLIST PhonePref
    %Ent.Privacy;
    PhoneUseType CDATA #IMPLIED -->
```

Entity %Telephone

Telephone numbers of the individual in the profile.

3.1.3. Address Preferences

The collection of address preferences allows for certain addresses to be designated in different collections, such as a company address for business travel and a home address for leisure travel.

Element	Occur.	Content	Data type	Attributes	Description
AddressPref	*	Element, % Address		%Privacy, AddressType	Address(es) to be used with this collection of preferences

AddressPref

Preferred address(es) to be used with this collection of preferences.

Public Attributes:**ShareSynchInd : (Yes | No | Inherit) = Inherit**

Permission for sharing data for synchronization of profiles held by other travel service providers. Enumerated values: 'Yes', 'No', 'Inherit'. Default = 'Inherit'.

ShareMarketInd : (Yes | No | Inherit) = Inherit

Permission for sharing data for marketing information. Enumerated values: 'Yes', 'No', 'Inherit'. Default = 'Inherit'.

AddressType : (Home | Business | Delivery | Mailing | Billing | CreditCard | Other)

Defines what the purpose of the address is, such as home, business, mailing, delivery, shipping/receiving address, etc.

AddressPref DTD fragment

```

<!ELEMENT AddressPref (%Ent.Address;)>
<!ATTLIST AddressPref
    %Ent.Privacy;
    AddressType (Home | Business | Delivery | Mailing | Billing | CreditCard | Other) #IMPLIED
>

```

ENTITY %Address

Uses the Address entity to define the mailing or delivery location where invoices are sent.

3.1.4. Payment Form Preferences

The collection of PaymentForm preferences allows for specific types of payment to be assigned to different preference collections, such as a corporate credit card for business travel and a personal bank account or credit card for leisure travel.

Element	Occur.	Content	Data type	Attributes	Description
PaymentFormPref	*	Text	String	PreferLevel, RPH	Form(s) of payment to be used with this collection of preferences

PaymentFormPref

Indicates preferences for the form of payment to be used for travel associated with this collection.

Attributes:**PreferLevel : (Only | Unacceptable | Preferred) = Preferred**

Identifies the level of preference for the payment form identified. Values: (Only | Unacceptable | Preferred). If Only, use only this preference for the travel situation; if Unacceptable, do NOT use this preference in the given situation. The default value is 'Preferred'.

RPH : CDATA

Index number to be used to reference the payment form to be used in this travel collection. Identifies a specific form of payment defined under Customer.

PaymentFormPref DTD fragment

```

<!ELEMENT PaymentFormPref (#PCDATA)>
<!ATTLIST PaymentFormPref
    PreferLevel (Only | Unacceptable | Preferred) "Preferred" #IMPLIED
    RPH CDATA #IMPLIED
>

```

3.1.5. Interest Preferences

The Interest preferences allows for designation of specific interests to be assigned to different preference collections, such as tickets to Broadway shows for travel to New York City, or preferences for types of recreational activities for leisure travel.

Element	Occur.	Content	Data type	Attributes	Description
InterestPref	*	Text	String	PreferLevel	Traveler interests to be used with this collection of preferences

InterestPref

Personal or special interests (i.e., hobbies, recreational activities, etc.) that enter into travel decisions, and are used as a preference for a given travel experience.

Attributes:

PreferLevel : (Only | Unacceptable | Preferred) = Preferred

Identifies the level of preference for the interest identified. Values: (Only | Unacceptable | Preferred). If Only, use only this interest preference for the travel situation; if Unacceptable, do NOT use this interest preference in the given situation. The default value is 'Preferred'.

InterestPref DTD fragment

```
<!ELEMENT InterestPref (#PCDATA)>
<!ATTLIST InterestPref
  PreferLevel (Only | Unacceptable | Preferred) "Preferred" #IMPLIED
>
```

3.1.6. Insurance Preferences

The Insurance preferences allows for designation of types of travel insurance to be procured in conjunction with different preference collections, such as additional medical coverage or transportation home from overseas travel, or insurance against loss of investment for a large travel purchase such as a cruise package.

Element	Occur.	Content	Data type	Attributes	Description
InsurancePref	*	Text	String	PreferLevel, InsuranceRPH	Insurance policies to be used with this collection of preferences

InsurancePref

Indicates preferences for the type of insurance policy to be used for travel associated with this collection.

Attributes:

PreferLevel : (Only | Unacceptable | Preferred) = Preferred

Identifies the level of preference for the insurance policy identified. Values: (Only | Unacceptable | Preferred). If Only, use only this preference for the travel situation; if Unacceptable, do NOT use this preference in the given situation. The default value is 'Preferred'.

InsuranceRPH : CDATA

Index number to be used for reference the insurance policy to be used in this travel collection.

InsurancePref DTD fragment

```
<!ELEMENT InsurancePref (#PCDATA)>
<!ATTLIST InsurancePref
```

PreferLevel (Only | Unacceptable | Preferred) "Preferred" #IMPLIED
InsuranceRPH CDATA #IMPLIED

>

3.1.7. Seating Preferences

The SeatingPref entity allows for designation of preferences for seat selection for airlines, rail transportation, or other modes of travel involving seat assignment. For example, a customer may prefer to have a window seat on a night flight, but an aisle seat on a short, domestic flight, or may wish to exclude seats that face backward regardless of the mode of transportation.

Element	Occur.	Content	Data type	Attributes	Description
SeatingPref	*	Text	String	PreferLevel, SeatDirection, SeatLocation, SeatPosition, SeatRow	Seating preferences to be used with this collection of preferences

%ENTITY SeatingPref

Indicates the customer's preference for seating arrangements.

Attributes:

PreferLevel : (Only | Unacceptable | Preferred) = Preferred

Identifies the level of preference for the seating arrangements indicated. Values: (Only | Unacceptable | Preferred). If Only, use only this seating preference for the travel situation; if Unacceptable, do NOT use this seating preference in the given situation. The default value is 'Preferred'.

SeatDirection : CDATA

Direction seat faces during travel, when conveyance allows.

SeatLocation : CDATA

Location of seat in cabin of conveyance. Suggested values include: Forward, Middle, Aft, ExitRow, Bulkhead, Right or Left Side, etc.

SeatPosition : CDATA

Preferred position of seat in a row, such as Aisle, Middle, Center, Window, etc.

SeatRow : CDATA

Preferred row for seating; indicates specific row number and/or seat identifier.

SeatingPref DTD fragment

```
<!ELEMENT SeatingPref (#PCDATA)>
<!ATTLIST SeatingPref
  PreferLevel (Only | Unacceptable | Preferred) "Preferred" #IMPLIED
  SeatDirection CDATA #REQUIRED
  SeatLocation CDATA #REQUIRED
  SeatPosition CDATA #REQUIRED
  SeatRow CDATA #REQUIRED
```

>

3.1.8. Ticket Distribution

The designation of preferences for ticket distribution specifies preferred methods of receiving tickets for travel-related services, such as use of courier, regular mail, or electronic ticketing, and provides the ability for a customer or company to specify the time frame for delivery of tickets.

Element	Occur.	Content	Data	Attributes	Description
---------	--------	---------	------	------------	-------------

			type		
TicketDistribPref	*	Text	String	PreferLevel, DistribType, TicketTime	Ticketing information to be used with this collection of preferences

TicketDistribPref

Indicates preferences for the type of ticket distribution to be used for the travel service product.

Attributes:

PreferLevel : (Only | Unacceptable | Preferred) = Preferred

Identifies the level of preference for the type of ticket distribution identified. Values: (Only | Unacceptable | Preferred). If Only, use only this preference for the travel situation; if Unacceptable, do NOT use this preference in the given situation. The default value is 'Preferred'.

DistribType : CDATA

Ticket distribution method; such as Fax, Email, Courier, Mail, Airport_Pickup, City_Office, Hotel_Desk, WillCall, etc.

TicketTime : CDATA

Ticket turnaround time desired, amount of time requested to deliver tickets.

TicketDistribPref DTD fragment

```
<ELEMENT TicketDistribPref (#PCDATA)>
<!ATTLIST TicketDistribPref
  PreferLevel (Only | Unacceptable | Preferred) "Preferred" #IMPLIED
  DistribType CDATA #IMPLIED
  TicketTime CDATA #IMPLIED
>
```

3.1.9. Media and Entertainment

The designation of media and entertainment preferences specifies certain entertainment or media-related services sought as part of a travel situation, which might include a preference for flights that offer movies or stereo entertainment, or hotels that offer Cable TV, etc.

Element	Occur.	Content	Data type	Attributes	Description
MediaEntertainPref	*	Text	String	PreferLevel	Media and entertainment information preferences

MediaEntertainPref

Identifies the media and entertainment preferences for a given travel situation, such as: books, magazines, movies, newspapers, radio, television, cinema, games, online, etc. This element could be used to indicate a preference for online connectivity, such as internet or modum dial-up connections when applicable to a given travel situation.

Attributes:

PreferLevel : (Only | Unacceptable | Preferred) = Preferred

Identifies the level of preference for the media and entertainment identified. Values: (Only | Unacceptable | Preferred). If Only, use only this preference for the travel situation; if Unacceptable, do NOT use this preference in the given situation. The default value is 'Preferred'.

MediaEntertainPref DTD fragment

```
<ELEMENT MediaEntertainPref (#PCDATA)>
<!ATTLIST MediaEntertainPref
```

PreferLevel (Only | Unacceptable | Preferred) "Preferred" #IMPLIED
>

3.1.10. Pet Information Preferences

The inclusion of pet information allows a customer to specify preferences for traveling with a pet, such as hotel accommodations that allow pets, boarding facilities, containers for shipping pets, etc.

Element	Occur.	Content	Data type	Attributes	Description
PetInfoPref	*	Text	String	PreferLevel	Pet information to be used with this collection of preferences

PetInfoPref

Indicates the preferences for information about pets that accompany the customer in a given travel situation.

Attributes:

PreferLevel : (Only | Unacceptable | Preferred) = Preferred

Identifies the level of preference for the pet information identified. Values: (Only | Unacceptable | Preferred). If Only, use only this pet information preference for the travel situation; if Unacceptable, do NOT use this pet information preference in the given situation. The default value is 'Preferred'.

PetInfoPref DTD fragment

```
<ELEMENT PetInfoPref (#PCDATA)>
<!ATTLIST PetInfoPref
  PreferLevel (Only | Unacceptable | Preferred) "Preferred" #IMPLIED
>
```

3.1.11. Food and Beverage Preferences (Entity %MealPref)

The Meal Preference entity allows a customer to indicate preferences for food and beverages, including dietary restrictions such as vegetarian or low fat meals, or favorite dining experiences, such as all-American or ethnic restaurants.



Figure 18. MealPref

Element	Occur.	Content	Data type	Attributes	Description
MealPref	*	Text	String	PreferLevel, MealType, FavoriteFood, Beverage	Food and beverage preferences to be used with this collection

MealPref

Indicates the customer's preference for meals, favorite foods, and beverages.

Attributes:**PreferLevel : (Only | Unacceptable | Preferred) = Preferred**

Identifies the level of preference for the meal or beverage type identified. Values: (Only | Unacceptable | Preferred). If Only, use only this preference for the travel situation; if Unacceptable, do NOT use this preference in the given situation. The default value is 'Preferred'.

MealType : CDATA

Identifies a type of meal preferred or dietary restrictions to be used for a given travel situation, such as vegetarian, or low-sodium meals.

FavoriteFood : CDATA

Indicates the customer's dining preferences, NOT dietary restrictions, to be used for a given travel situation.

Beverage : CDATA

Indicates the type of drink(s) preferred for a given type of travel situation.

MealPref DTD fragment

```
<ELEMENT MealPref (#PCDATA)>
<!ATTLIST MealPref
  PreferLevel (Only | Unacceptable | Preferred) "Preferred" #IMPLIED
  MealType CDATA #IMPLIED
  FavoriteFood CDATA #IMPLIED
  Beverage CDATA #IMPLIED
>
```

3.1.12. Loyalty Program Preferences

Preferences for loyalty programs allows for a customer to designate a specific loyalty program for different travel situations, such as an airline loyalty program for a travel collection that includes an airline segment, another loyalty program for a trip made exclusively in a rental car, etc.

Element	Occur.	Content	Data type	Attributes	Description
LoyaltyPref	*	Text	String	PreferLevel, LoyaltyRPH	Loyalty programs to be used with this collection of preferences

LoyaltyPref

Preferred loyalty program to be used for a given travel situation.

Attributes:**PreferLevel : (Only | Unacceptable | Preferred) = Preferred**

Identifies the level of preference for the loyalty program identified. Values: (Only | Unacceptable | Preferred). If Only, use only this loyalty program preference for the travel situation; if Unacceptable, do NOT use this loyalty program preference in the given situation. The default value is 'Preferred'.

LoyaltyRPH : CDATA

Index number to be used for reference the loyalty program to be used in this travel collection.

LoyaltyPref DTD fragment

```
<ELEMENT LoyaltyPref (#PCDATA)>
<!ATTLIST LoyaltyPref
  PreferLevel (Only | Unacceptable | Preferred) "Preferred" #IMPLIED
  LoyaltyRPH CDATA #IMPLIED>
```

3.1.13. Special Request Preferences

The special request preference element allows a special request to be associated with a given travel experience.

Element	Occur.	Content	Data type	Attributes	Description
SpecRequestPref	*	Text	String	PreferLevel	Special request to be used with this preference collection

SpecRequestPref

Indicates a preference for special service requests in a given travel situation.

Attributes:

PreferLevel : (Only | Unacceptable | Preferred) = Preferred

Identifies the level of preference for the special request identified. Values: (Only | Unacceptable | Preferred). If Only, use only this preference for the travel situation; if Unacceptable, do NOT use this preference in the given situation. The default value is 'Preferred'.

SpecRequestPref DTD fragment

```
<ELEMENT SpecRequestPref (#PCDATA)>
<!ATTLIST SpecRequestPref
  PreferLevel (Only | Unacceptable | Preferred) "Preferred" #IMPLIED
>
```

3.1.14. Related Traveler Preferences

The RelatedTravelerPref element allows a customer or company to designate a preference to be accompanied by another traveler on a given travel experience.

Element	Occur.	Content	Data type	Attributes	Description
RelatedTravelerPref	*	Element, % UniqueId, % PersonName		PreferLevel	Name(s) of related travelers to be used with this collection of preferences

RelatedTravelerPref

Indicates a preference for a related traveler identified in the customer profile to be used in a given travel situation.

Attributes:

PreferLevel : (Only | Unacceptable | Preferred) = Preferred

Identifies the level of preference for the special request identified. Values: (Only | Unacceptable | Preferred). If Only, use only this preference for the travel situation; if Unacceptable, do NOT use this preference in the given situation. The default value is 'Preferred'.

RelatedTravelerPref DTD fragment

```
<ELEMENT RelatedTravelerPref (%Ent.UniqueId;, %Ent.PersonName)>
<!ATTLIST RelatedTravelerPref
  PreferLevel (Only | Unacceptable | Preferred) "Preferred" #IMPLIED>
```

3.1.15. TPA_Extensions

Due to the proprietary nature of agreements between business relationships, the TPA_Extensions element is included in this section of the profile, and allows for companies to add selected information as needed.

Element	Occur.	Content	Data type	Attributes	Description
TPA_Extensions	?				Due to the proprietary nature of agreements between business entities, this element allows for companies to add selected information as needed.

TPA_Extensions

Due to the proprietary nature of agreements between business entities, this element allows for companies to add selected information as needed.

TPA_Extensions DTD fragment

```
<!ELEMENT TPA_Extensions ANY>
```

3.2 Vehicle Rental Preferences

Vehicle rental preferences can be specified for customers or companies to indicate their preferences for types of vehicles to be rented in specific travel situations. Companies may wish to specify certain class or types of cars that their employees are allowed to rent, or indicate special business needs for trucks for hauling goods, etc. Personal travelers may wish to indicate preferences for types of vehicles such as vans or SUVs for family vacations, or smaller, more sporty models when travelling alone. A customer may prefer to have a convertible or sun roof in a warm destination, but request a ski rack when vacationing in the wintertime.

The Vehicle Rental Preferences section, tag <VehicleRentalPref>, has entries for specific features on rental cars including vehicle type (major category such as car, truck, SUV) and vehicle class (more precise kind of vehicle), air conditioning, transmission, cell phone, global navigation equipment, physically challenged needs, and requirements for child seats. In addition, the specification captures preferences for vendors and loyalty programs, insurance coverage, forms of payment for rental cars, and other special requirements.

While each of the elements in the Vehicle Rental Preferences collection (Figure 19) is OPTIONAL, if any of these elements appear in an OTA message, they MUST appear in the designated order.

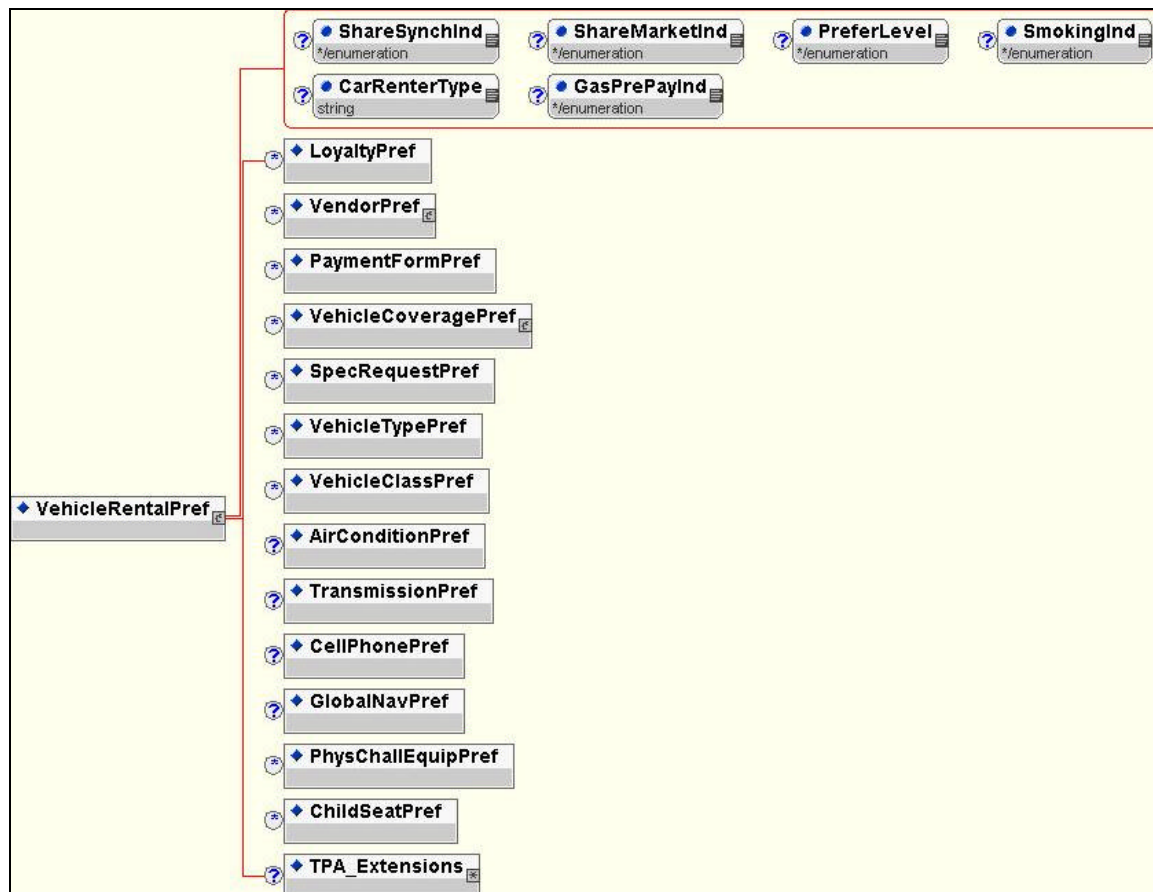


Figure 19. Vehicle Rental Preferences

3.2.1 Vehicle Rental Preferences Collection

Element	Occur.	Content	Data type	Attributes	Description
VehicleRentalPref	?	Element		PreferLevel, %Privacy, SmokingInd, CarRenterType GasPrePay	Vehicle Rental Preferences
LoyaltyPref	*	Element		PreferLevel, LoyaltyRPH	Loyalty programs to be used with this collection of preferences
VendorPref	*	Element		PreferLevel	Vendor(s) to be used with this collection of preferences
PaymentFormPref	*	Element		PreferLevel, RPH	Form(s) of payment to be used with this collection of preferences
VehicleCoveragePref	*	Element		%Privacy, CoverageType, CoverageCode	Insurance coverage to be used with this collection of preferences
SpecRequestPref	*	Element		PreferLevel	Special requests to be used with this collection of preferences
VehicleTypePref	*	Element		PreferLevel	Type of vehicle to be used with this collection of preferences
VehicleClassPref	*	Element		PreferLevel	Class of vehicle to be used with this collection of preferences
AirConditionPref	*	Element		PreferLevel	Air conditioning preference to be used with this collection of preferences
TransmissionPref	?	Element		PreferLevel	Transmission preference to be used with this collection of preferences
CellPhonePref	?	Element		PreferLevel	Cell phone preference to be used with this collection of preferences
GlobalNavPref	?	Element		PreferLevel	Global navigation preference to be used with this collection of preferences
PhysChallEquipPref	*	Element		PreferLevel	Equipment needed for drivers with special physical challenges
ChildSeatPref	*	Element		PreferLevel, ChildSeatType, Quantity	Child seat preference to be used with this collection of preferences
TPA_Extensions	?	Element			Due to the proprietary nature of agreements between business entities, this element allows for companies to add selected information as needed.

VehicleRentalPref

Preferences for car rentals associated with this collection

Attributes:

PreferLevel : (Only | Unacceptable | Preferred) = Preferred

Indicates the customer's level of preference for the car rental preferences identified in this collection. Values: (Only | Unacceptable | Preferred). If Only, use only this preference for the travel situation; if Unacceptable, do NOT use this preference in the given situation. The default value is 'Preferred'.

ShareSynchInd : (Yes | No | Inherit) = Inherit

Permission for sharing data for synchronization of profiles held by other travel service providers. Enumerated values: 'Yes', 'No', 'Inherit'. Default = 'Inherit'.

ShareMarketInd : (Yes | No | Inherit) = Inherit

Permission for sharing data for marketing information. Enumerated values: 'Yes', 'No', 'Inherit'. Default = 'Inherit'.

SmokingInd : (Yes | No)

Indicates the customer's preference for smoking. Values: (Yes | No).

CarRenterType : CDATA

Categories of renters - TBD.

GasPrePayInd : (Yes | No)

Indicates customer's preference of option to pre-pay for gasoline with rental.

VehicleRentalPref DTD fragment

```
<!ELEMENT VehicleRentalPref (LoyaltyPref*, VendorPref*, PaymentFormPref*, VehicleCoveragePref*,
SpecRequestPref*, VehicleTypePref*, VehicleClassPref*, AirConditionPref?, TransmissionPref?,
CellPhonePref?, GlobalNavPref?, PhysChallEquipPref*, ChildSeatPref*, TPA_Extensions?)>
<!--ATTLIST VehicleRentalPref
%Ent.Privacy;
PreferLevel (Only | Unacceptable | Preferred) "Preferred"
SmokingInd (Yes | No) #IMPLIED
CarRenterType CDATA #IMPLIED
GasPrePayInd (Yes | No) #IMPLIED-->
```

3.2.2 Vehicle Loyalty Preferences

Indicates a preference for the loyalty program to be used for vehicle rental. The RPH (Reference Place Holder) attribute designates a specific loyalty program from a collection stored in the profile. The same LoyaltyPref element is used in all preference collections.

Element	Occur.	Content	Data type	Attributes	Description
LoyaltyPref	*	Text	String	PreferLevel, LoyaltyRPH	Loyalty programs to be used with this collection of preferences

LoyaltyPref

Preferred loyalty program to be used for car rental services in a given travel situation.

Attributes:**PreferLevel : (Only | Unacceptable | Preferred) = Preferred**

Indicates the level of preference for the car rental loyalty program identified. Values: (Only | Unacceptable | Preferred). If Only, use only this preference for the travel situation; if Unacceptable, do NOT use this preference in the given situation. The default value is 'Preferred'.

LoyaltyRPH : CDATA

Index number to be used for reference the loyalty program to be used in this travel collection.

LoyaltyPref DTD fragment

```
<!ELEMENT LoyaltyPref (#PCDATA)>
<!--ATTLIST LoyaltyPref
PreferLevel (Only | Unacceptable | Preferred) "Preferred"
LoyaltyRPH CDATA #IMPLIED-->
```

3.2.3 Vehicle Vendor Preferences

Vendor Preferences indicates a preference for a specific car rental agency when used in a travel collection. The VendorPref element uses the Company Name entity to identify the preferred company by name and by vendor code. The same VendorPref element is used in all collections.

Element	Occur.	Content	Data type	Attributes	Description
VendorPref	*	Text	String	PreferLevel	Car rental vendor to be used with this preference collection
%CompanyName					

VendorPref

Indicates the preferred company to be used for a car rental.

Attributes:

PreferLevel : (Only | Unacceptable | Preferred) = Preferred

Indicates the level of preference for the car rental vendor identified. Values: (Only | Unacceptable | Preferred). If Only, use only this preference for the travel situation; if Unacceptable, do NOT use this preference in the given situation. The default value is 'Preferred'.

VendorPref DTD fragment

```
<!ELEMENT VendorPref (%Ent.CompanyName;)>
<!ATTLIST VendorPref
  PreferLevel (Only | Unacceptable | Preferred) "Preferred"
>
```

ENTITY %CompanyName

Identifies a company or organization associated with the contact.

3.2.4 PaymentForm Preferences

A preference for a payment indicates a specific type of payment, such as a credit card or direct bill, to be assigned to the car rental preference collection. The RPH (Reference Place Holder) attribute designates a specific payment form in a collection stored in the profile. The same PaymentFormPref element is used in all preference collections.

Element	Occur.	Content	Data type	Attributes	Description
PaymentFormPref	*	Text	String	PreferLevel, RPH	Form(s) of payment to be used with this vehicle rental

PaymentFormPref

Indicates preferences for the form of payment to be used for car rental services associated with this collection.

Attributes:

PreferLevel : (Only | Unacceptable | Preferred) = Preferred

Identifies the level of preference for the payment form identified. Values: (Only | Unacceptable | Preferred). If Only, use only this preference for the travel situation; if

Unacceptable, do NOT use this preference in the given situation. The default value is 'Preferred'.

RPH : CDATA

Reference to the payment form to be used in this travel collection. Identifies a specific form of payment defined under Customer.

PaymentFormPref DTD fragment

```
<ELEMENT PaymentFormPref (#PCDATA)>
<!ATTLIST PaymentFormPref
  PreferLevel (Only | Unacceptable | Preferred) "Preferred" #IMPLIED
  RPH CDATA #IMPLIED
>
```

3.2.5 Vehicle Insurance Coverage

A preference for a specific type of vehicle insurance coverage (Figure 20) to be used when renting a car in a given travel situation, along with preferences for supplemental insurance or limitations.

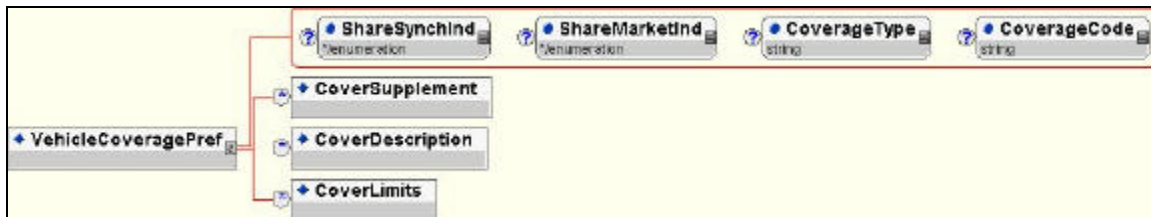


Figure 20. VehicleCoveragePref

Element	Occur.	Content	Data type	Attributes	Description
VehicleCoveragePref	*	Element		%Privacy, CoverageType, CoverageCode	Insurance coverage specified for this collection
CoverSupplement	*	Text	String		Supplemental insurance to be used for vehicle rental
CoverDescription	*	Text	String		Description of insurance coverage used for rental
CoverLimits	*	Text	String		Limitations on insurance coverage for rental vehicles

VehicleCoveragePref

Insurance coverage needed for car rentals associated with this collection.

Attributes:

ShareSynchInd : (Yes | No | Inherit) = Inherit

Permission for sharing data for synchronization of profiles held by other travel service providers. Enumerated values: 'Yes', 'No', 'Inherit'. Default = 'Inherit'.

ShareMarketInd : (Yes | No | Inherit) = Inherit

Permission for sharing data for marketing information. Enumerated values: 'Yes', 'No', 'Inherit'. Default = 'Inherit'.

CoverageType : CDATA

Categories of insurance preferred by the customer, such as; Effects, Personal, Baggage, VehicleDamage, Liability, Theft, YoungDriver

CoverageCode : CDATA

Industry code for type of insurance coverage.

VehicleCoveragePref DTD fragment

```
<!ELEMENT VehicleCoveragePref (CoverSupplement*, CoverDescription*, CoverLimits*)>
<!ATTLIST VehicleCoveragePref
    %Ent.Privacy;
    CoverageType CDATA #IMPLIED
    CoverageCode CDATA #IMPLIED
>
```

CoverSupplement

Supplemental insurance coverage for rental cars.

CoverSupplement DTD fragment

```
<!ELEMENT CoverSupplement (#PCDATA)>
```

CoverDescription

Description of insurance coverage for rental cars.

CoverDescription DTD fragment

```
<!ELEMENT CoverDescription (#PCDATA)>
```

CoverLimits

Limitations on car rental insurance coverage, such as minimums, maximums or per occurrence.

CoverLimits DTD fragment

```
<!ELEMENT CoverLimits (#PCDATA)>
```

3.2.6 Vehicle Special Requests

The special request preference element allows the customer to designate a special request to be associated with the vehicle rental.

Element	Occur.	Content	Data type	Attributes	Description
SpecRequestPref	*	Text	String	PreferLevel	Special request to be used with this preference collection

SpecRequestPref

Indicates a preference for special needs or requirements for car rental services.

Attributes:

PreferLevel : (Only | Unacceptable | Preferred) = Preferred

Identifies the customer's level of preference for the special request identified. Values: (Only | Unacceptable | Preferred). If Only, use only this preference for the travel situation; if Unacceptable, do NOT use this preference in the given situation. The default value is 'Preferred'.

SpecRequestPref DTD fragment

```
<!ELEMENT SpecRequestPref (#PCDATA)>
<!ATTLIST SpecRequestPref
    PreferLevel (Only | Unacceptable | Preferred) "Preferred" #IMPLIED
>
```

3.2.7 Vehicle Type Preferences

The vehicle type preference element allows the customer to designate a major category of vehicle preferred for rental.

Element	Occur.	Content	Data type	Attributes	Description
VehicleTypePref	*	Text	String	PreferLevel, VehicleType	Type of vehicle to be used with this preference collection

VehicleTypePref

Major category of vehicle, e.g., car, truck, van, SUV, etc.

Attributes:

PreferLevel : (Only | Unacceptable | Preferred) = Preferred

Indicates the customer's level of preference for the car rental vehicle type identified. Values: (Only | Unacceptable | Preferred). If Only, use only this preference for the travel situation; if Unacceptable, do NOT use this preference in the given situation. The default value is 'Preferred'..

VehicleType : (Car | Van | SUV | Convertible | Truck | Motorcycle | Limo | StationWagon | Pickup | MotorHome | Other)

Car, Van, SUV, Truck, Motorcycle, Limo, StationWagon, Pickup, MotorHome, Other

VehicleTypePref DTD fragment

```
<!ELEMENT VehicleTypePref (#PCDATA)>
<!ATTLIST VehicleTypePref
    PreferLevel (Only | Unacceptable | Preferred) "Preferred" #IMPLIED
    VehicleType (Car | Van | SUV | Convertible | Truck | Motorcycle | Limo | StationWagon |
    Pickup | MotorHome | Other) #REQUIRED
>
```

3.2.8 Vehicle Class Preferences

The vehicle class preference element allows the customer to designate specific detail about a vehicle type preferred for rental.

Element	Occur.	Content	Data type	Attributes	Description
VehicleClassPref	*	Text	String	PreferLevel, VehicleClass	Vehicle class to be used with this preference collection
VehicleClass			String		

VehicleClassPref

Detailed type of vehicle, such as Mini, Subcompact, Economy, Compact, Midsize, Intermediate, Standard, Full_Size, Luxury, Premium, Convertible, Minivan, etc.

Attributes:

PreferLevel : (Only | Unacceptable | Preferred) = Preferred

Indicates the level of preference for the car rental vehicle class identified. Values: (Only | Unacceptable | Preferred). If Only, use only this preference for the travel situation; if Unacceptable, do NOT use this preference in the given situation. The default value is 'Preferred'.

VehicleClass : CDATA

Additional modifier of vehicle class, such as; 12_Passenger_Van, Moving_Van, 15_Passenger Van, Cargo_Van, 12_Foot_Truck, 15_Foot_Truck, 20_Foot_Truck, 24_Foot_Truck, 26_Foot_Truck, Moped, Stretch, Regular, Unique, Exotic, Other

VehicleClassPref DTD fragment

```

<!ELEMENT VehicleClassPref (#PCDATA)>
<!--ATTLIST VehicleClassPref
    PreferLevel (Only | Unacceptable | Preferred) "Preferred" #IMPLIED
    VehicleClass (Mini | Subcompact | Economy | Compact | Midsize | Intermediate | Standard |
    Fullsize | Luxury | Premium | Minivan | 12_Passenger_Van | Moving_Van | 15_Passenger_Van |
    Cargo_Van | 12_Foot_Truck | 20_Foot_Truck | 24_Foot_Truck | 26_Foot_Truck | Moped |
    Stretch | Regular | Unique | Exotic | Other) #REQUIRED
-->

```

3.2.9 Air Conditioning Preferences

Element	Occur.	Content	Data type	Attributes	Description
AirConditionPref	*	Text	String	PreferLevel	Air condition preference to be used with this preference collection

AirConditionPref

Indicates preference for air conditioning in a rented car.

Attributes:

PreferLevel : (Only | Unacceptable | Preferred) = Preferred

Indicates the level of preference for the car rental air conditioning equipment identified. Values: (Only | Unacceptable | Preferred). If Only, use only this preference for the travel situation; if Unacceptable, do NOT use this preference in the given situation. The default value is 'Preferred'.

AirConditionPref DTD fragment

```

<!ELEMENT AirConditionPref (#PCDATA)>
<!--ATTLIST AirConditionPref
    PreferLevel (Only | Unacceptable | Preferred) "Preferred" #IMPLIED
-->

```

3.2.10 Transmission Preferences

Element	Occur.	Content	Data type	Attributes	Description
TransmissionPref	*	Text	String	PreferLevel	Transmission preference to be used with this preference collection

TransmissionPref

Describes the type of transmission preferred in a rented car, e.g. manual or automatic.

Attributes:

PreferLevel : (Only | Unacceptable | Preferred) = Preferred

Indicates the level of preference for the type of car transmission identified. Values: (Only | Unacceptable | Preferred). If Only, use only this preference for the travel situation; if Unacceptable, do NOT use this preference in the given situation. The default value is 'Preferred'.

TransmissionPref DTD fragment

```

<!ELEMENT TransmissionPref (#PCDATA)>
<!--ATTLIST TransmissionPref

```

PreferLevel (Only | Unacceptable | Preferred) "Preferred" #IMPLIED
>

3.2.11 Cell Phone or Mobile Phone Preferences

Element	Occur.	Content	Data type	Attributes	Description
CellPhonePref	*	Text	String	PreferLevel	Cell Phone preference to be used with this preference collection

CellPhonePref

Indicates the preference for a cellular phone in a rented car.

Attributes:

PreferLevel : (Only | Unacceptable | Preferred) = Preferred

Indicates the level of preference for the car rental cell phone equipment identified. Values: (Only | Unacceptable | Preferred). If Only, use only this preference for the travel situation; if Unacceptable, do NOT use this preference in the given situation. The default value is 'Preferred'.

CellPhonePref DTD fragment

```
<!ELEMENT CellPhonePref (#PCDATA)>
<!ATTLIST CellPhonePref
  PreferLevel (Only | Unacceptable | Preferred) "Preferred" #IMPLIED>
>
```

3.2.12 Global Navigation Equipment Preferences

Element	Occur.	Content	Data type	Attributes	Description
GlobalNavPref	*	Text	String	PreferLevel	Global navigation preference to be used with this preference collection

GlobalNavPref

Indicates a preference for a global navigation system in a rented car.

Attributes:

PreferLevel : (Only | Unacceptable | Preferred) = Preferred

Indicates the customer's level of preference for the car rental global navigation equipment identified. Values: (Only | Unacceptable | Preferred). If Only, use only this preference for the travel situation; if Unacceptable, do NOT use this preference in the given situation. The default value is 'Preferred'.

GlobalNavPref DTD fragment

```
<!ELEMENT GlobalNavPref (#PCDATA)>
<!ATTLIST GlobalNavPref
  PreferLevel (Only | Unacceptable | Preferred) "Preferred" #IMPLIED>
```

3.2.13 Physically Challenged Vehicle Rental Equipment

Element	Occur.	Content	Data type	Attributes	Description
PhysChallEquipPref	*	Text	String	PreferLevel	Physically challenged equipment

					preference to be used with this preference collection
--	--	--	--	--	---

PhysChallEquipPref

Describes equipment needed for drivers with special physical challenges.

Attributes:

PreferLevel : (Only | Unacceptable | Preferred) = Preferred

Indicates the customer's level of preference for the car rental physically challenged equipment identified. Values: (Only | Unacceptable | Preferred). If Only, use only this preference for the travel situation; if Unacceptable, do NOT use this preference in the given situation. The default value is 'Preferred'.

PhysChallEquipPref DTD fragment

```
<!ELEMENT PhysChallEquipPref (#PCDATA)>
<!ATTLIST PhysChallEquipPref
  PreferLevel (Only | Unacceptable | Preferred) "Preferred" #IMPLIED
>
```

3.2.14 Child Seat Equipment Preferences

Element	Occur.	Content	Data type	Attributes	Description
ChildSeatPref	*	Text	String	PreferLevel, ChildSeatType, Quantity	Child seat preference to be used with this preference collection

ChildSeatPref

Indicates requirements for child seats in a rented car.

Attributes:

PreferLevel : (Only | Unacceptable | Preferred) = Preferred

Indicates the level of preference for the car rental child seat type identified. Values: (Only | Unacceptable | Preferred). If Only, use only this preference for the travel situation; if Unacceptable, do NOT use this preference in the given situation. The default value is 'Preferred'.

ChildSeatType : CDATA

Type of child seat needed in a rented car, e.g.: Infant, Child, Booster, etc.

Quantity : CDATA

Number of child seats of the type indicated in ChildSeatType attribute.

ChildSeatPref DTD fragment

```
<!ELEMENT ChildSeatPref (#PCDATA)>
<!ATTLIST ChildSeatPref
  PreferLevel (Only | Unacceptable | Preferred) "Preferred" #IMPLIED
  ChildSeatType CDATA #IMPLIED
  Quantity CDATA #IMPLIED
>
```

3.2.15 TPA_Extensions

Due to the proprietary nature of agreements between business relationships, the TPA_Extensions element is included in this section of the profile, and allows for companies to add selected information as needed.

Element	Occur.	Content	Data type	Attributes	Description
TPA_Extensions	?				Due to the proprietary nature of agreements between business entities, this element allows for companies to add selected information as needed.

TPA_Extensions

Due to the proprietary nature of agreements between business entities, this element allows for companies to add selected information as needed.

TPA_Extensions DTD fragment

<!ELEMENT TPA_Extensions ANY>

3.3 Air Travel Preferences

In the Airline Preferences section customers or companies can indicate a collection of features relating to air travel to include in their profiles. The specification captures several airline specific data items including preferred airport for originating flights and connections, acceptable fare restrictions, and need for non-stop versus connecting flights. As a part of the air travel preferences, the customer has the ability to indicate preferred vendors and loyalty programs for air travel, forms of payment used, media and entertainment choices, meal types required, pet information, and special service requests (both free text fields and airline industry standard codes). This section also indicates preferred aircraft equipment, choices for airline seating, such as passenger cabin and service class, and preferences for position of seat in a row, and location of seat in the cabin, by reusing the <SeatingPref> object defined in the Common Preferences section.

While each of the elements in the AirlinesPref collection (Figure 21) is OPTIONAL, if any of these elements appear in an OTA message, they MUST appear in the designated order.

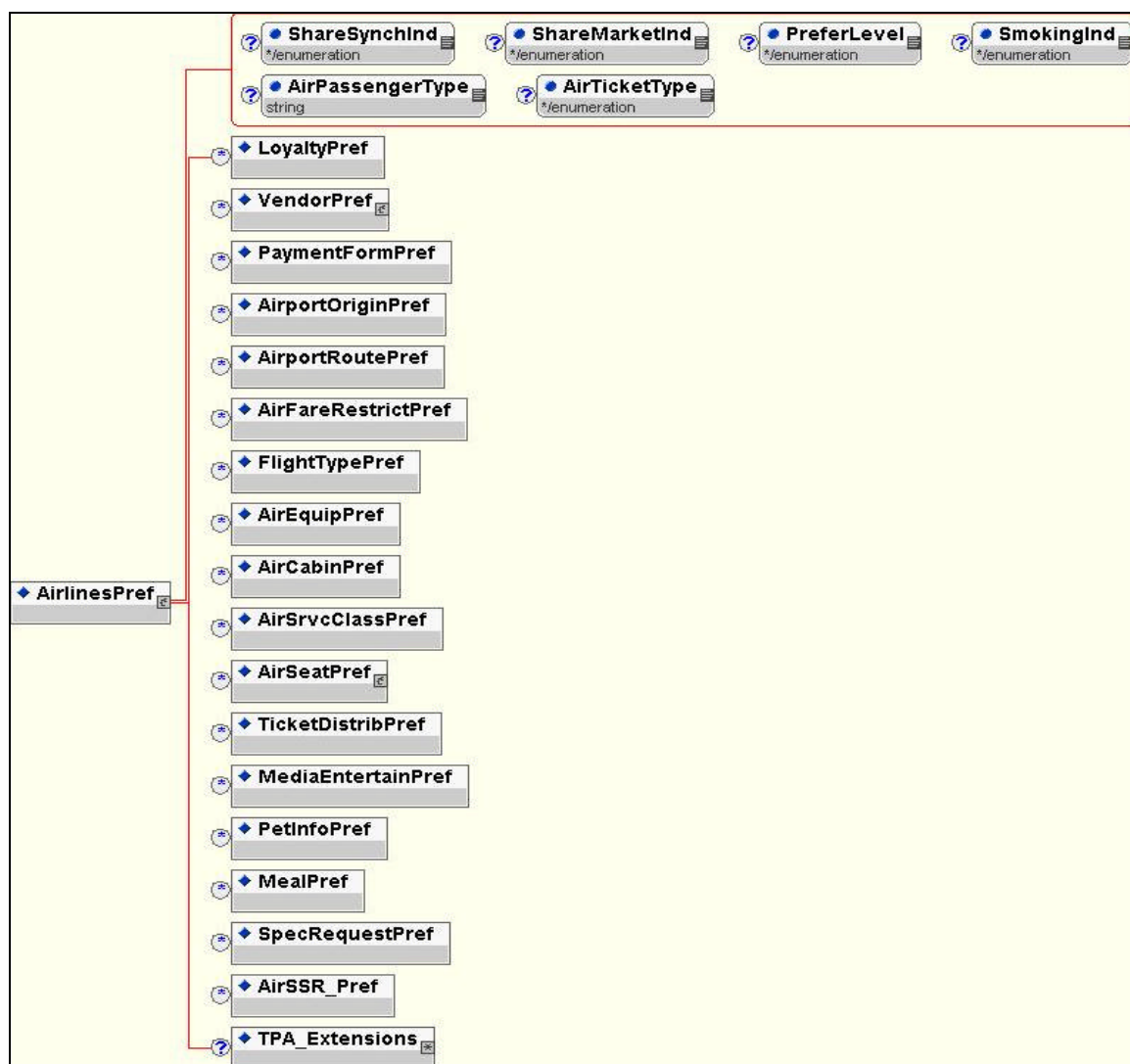


Figure 21. Airlines Preferences

3.3.1 Airline Preferences Collection

Element	Occur.	Content	Data type	Attributes	Description
AirlinesPref	?	Element		PreferLevel, ShareSynchInd, ShareMarketInd, SmokingInd, AirPassengerType AirTicketType	Air Travel Preferences
LoyaltyPref	*	Element		PreferLevel, LoyaltyRPH	Loyalty programs to be used with this collection of preferences
VendorPref	*	Element		PreferLevel	Vendor(s) to be used with this collection of preferences
PaymentFormPref	*	Element		PreferLevel, RPH	Form(s) of payment to be used with this collection of preferences
AirportOriginPref	*	Element		PreferLevel	Airport origin to be used with this collection of preferences
AirportRoutePref	*	Element		PreferLevel	Airport route preferences to be used with this collection of preferences
AirFareRestrictPref	*	Element		PreferLevel	Air fare restriction to be used with this collection of preferences
FlightTypePref	*	Element		PreferLevel	Type of flight to be used with this collection of preferences
AirEquipPref	*	Element		PreferLevel, AirEquipType	Air equipment preference to be used with this collection of preferences
AirCabinPref	*	Element		PreferLevel, CabinType	Air cabin preference to be used with this collection of preferences
AirSrvcClassPref	?	Element		PreferLevel, ServiceClassCode	Air service class preference to be used with this collection of preferences
AirSeatPref	?	Element		PreferLevel	Air seat preference to be used with this collection of preferences
TicketDistribPref	*	Element		PreferLevel, DistribTime, TicketTime	Ticketing information to be used with this collection of preferences
MediaEntertainPref	*	Element		PreferLevel	Media and entertainment information preferences
PetInfoPref	*	Element		PreferLevel	Pet information to be used with this collection of preferences
MealPref	*	Element		PreferLevel, MealType, FavoriteFood, Beverage	Food and beverage preferences to be used with this collection of preferences
SpecRequestPref	*	Element		PreferLevel	Special requests to be used with this collection of preferences
AirSSR_Pref	*	Element		PreferLevel, SSR_Code	Indicates preference for special services required for air travel using standard industry (SSR-OSI) code list.
TPA_Extensions	?	Element			Due to the proprietary nature of agreements between business entities, this element allows for

					companies to add selected information as needed.
--	--	--	--	--	--

AirlinesPref

Identifies a collection of preferences for air travel.

Attributes:

PreferLevel : (Only | Unacceptable | Preferred) = Preferred

Identifies the level of preference for the airline travel preferences identified in this collection. Values: (Only | Unacceptable | Preferred). If Only, use only this preference for the travel situation; if Unacceptable, do NOT use this preference in the given situation. The default value is 'Preferred'.

ShareSynchInd : (Yes | No | Inherit) = Inherit

Permission for sharing data for synchronization of profiles held by other travel service providers. Enumerated values: 'Yes', 'No', 'Inherit'. Default = 'Inherit'.

ShareMarketInd : (Yes | No | Inherit) = Inherit

Permission for sharing data for marketing information. Enumerated values: 'Yes', 'No', 'Inherit'. Default = 'Inherit'.

SmokingInd : (Yes | No)

Indicates the customer's preference for smoking accommodations. Values: (Yes | No).

AirPassengerType : CDATA

Category of airline passenger, using standard ATPCO codes.

AirTicketType : (Electronic | Paper)

Type of airline ticket preferred for this collection. Values: (Paper | Electronic).

AirlinesPref DTDfragment

```
<!ELEMENT AirlinesPref (LoyaltyPref*, VendorPref*, PaymentFormPref*, AirportOriginPref*,
AirportRoutePref*, AirFareRestrictPref*, FlightTypePref*, AirEquipPref*, AirCabinPref*,
AirSrvClassPref*, AirSeatPref*, TicketDistribPref*, MediaEntertainPref*, PetInfoPref*, MealPref*,
SpecRequestPref*, AirSSR_Pref*, TPA_Extensions?)>
<!ATTLIST AirlinesPref
  %Ent.Privacy;
  PreferLevel (Only | Unacceptable | Preferred) "Preferred"
  SmokingInd (Yes | No) #IMPLIED
  AirPassengerType CDATA #IMPLIED
  AirTicketType (Electronic | Paper) #IMPLIED
>
```

3.3.2 Airlines Loyalty Program Preferences

Indicates a preference for the loyalty program to be used for air travel. The RPH (Reference Place Holder) attribute designates a specific loyalty program from a collection stored in the profile. The same LoyaltyPref element is used in all preference collections.

Element	Occur.	Content	Data type	Attributes	Description
LoyaltyPref	*	Text	String	PreferLevel, LoyaltyRPH	Loyalty programs to be used with this preference collection

LoyaltyPref

Identifies preferences for an airline loyalty program to be used for a given travel situation.

Attributes:**PreferLevel : (Only | Unacceptable | Preferred) = Preferred**

Identifies the customer's level of preference for the airline loyalty program identified. Values: (Only | Unacceptable | Preferred). If Only, use only this preference for the travel situation; if Unacceptable, do NOT use this preference in the given situation. The default value is 'Preferred'.

LoyaltyRPH : CDATA

Index number to be used for reference the loyalty program to be used in this travel collection.

LoyaltyPref DTD fragment

```
<!ELEMENT LoyaltyPref (#PCDATA)>
<!ATTLIST LoyaltyPref
  PreferLevel (Only | Unacceptable | Preferred) "Preferred"
  LoyaltyRPH CDATA #IMPLIED
>
```

3.3.3 Airline Vendor Preferences

A vendor preferences designates a specific airline to be used in a travel collection. The VendorPref element uses the Company Name entity to identify the preferred company by name and by vendor code. The same VendorPref element is used in all preference collections.

Element	Occur.	Content	Data type	Attributes	Description
VendorPref	*	Text	String	PreferLevel	Airline to be used with this preference collection
%CompanyName					

VendorPref

Identifies preferences for an airline company to be used for a given travel situation.

Attributes:**PreferLevel : (Only | Unacceptable | Preferred) = Preferred**

Identifies the level of preference for the airline identified. Values: (Only | Unacceptable | Preferred). If Only, use only this preference for the travel situation; if Unacceptable, do NOT use this preference in the given situation. The default value is 'Preferred'.

VendorPref DTD fragment

```
<!ELEMENT VendorPref (%Ent.CompanyName;)>
<!ATTLIST VendorPref
  PreferLevel (Only | Unacceptable | Preferred) "Preferred"
>
```

ENTITY %CompanyName

Identifies a company or organization associated with the contact.

3.3.4 Airline Payment Form Preferences

This preference indicates a specific form of payment, such as a credit card, assigned to an airlines preference collection. The RPH (Reference Place Holder) attribute designates a specific payment form in a collection stored in the profile. The same PaymentFormPref element is used in all preference collections.

Element	Occur.	Content	Data type	Attributes	Description
PaymentFormPref	*	Text	String	PreferLevel, RPH	Form(s) of payment to be used for this airline travel.

PaymentFormPref

Indicates preferences for the means of paying for air travel associated with a given travel situation.

Attributes:

PreferLevel : (Only | Unacceptable | Preferred) = Preferred

Identifies the level of preference for the payment form identified. Values: (Only | Unacceptable | Preferred). If Only, use only this preference for the travel situation; if Unacceptable, do NOT use this preference in the given situation. The default value is 'Preferred'.

RPH : CDATA

Reference to the payment form to be used in this travel collection. Identifies a specific form of payment defined under Customer.

PaymentFormPref DTD fragment

```
<!ELEMENT PaymentFormPref (#PCDATA)>
<!ATTLIST PaymentFormPref
  PreferLevel (Only | Unacceptable | Preferred) "Preferred" #IMPLIED
  RPH CDATA #IMPLIED
>
```

3.3.5 Airport Origin Preferences

Element	Occur.	Content	Data type	Attributes	Description
AirportOriginPref	*	Text	String	PreferLevel	Airport origin preference to be used with this preference collection

AirportOriginPref

Departure airport preferences, IATA airport codes.

Attributes:

PreferLevel : (Only | Unacceptable | Preferred) = Preferred

Identifies the customer's level of preference for the airport identified. Values: (Only | Unacceptable | Preferred). If Only, use only this preference for the travel situation; if Unacceptable, do NOT use this preference in the given situation. The default value is 'Preferred'.

AirportOriginPref DTD fragment

```
<!ELEMENT AirportOriginPref (#PCDATA)>
<!ATTLIST AirportOriginPref
  PreferLevel (Only | Unacceptable | Preferred) "Preferred" #IMPLIED
>
```

3.3.6 Airport Route Preferences

Element	Occur.	Content	Data type	Attributes	Description
AirportRoutePref	*	Text	String	PreferLevel	Airport Route preference to be

					used with this preference collection
--	--	--	--	--	--------------------------------------

AirportRoutePref

Indicates the customer's preference for connecting airports, using IATA airport codes.

Attributes:

PreferLevel : (Only | Unacceptable | Preferred) = Preferred

Identifies the level of preference for the airport routing identified. Values: (Only | Unacceptable | Preferred). If Only, use only this preference for the travel situation; if Unacceptable, do NOT use this preference in the given situation. The default value is 'Preferred'.

AirportRoutePref DTD fragment

```
<!ELEMENT AirportRoutePref (#PCDATA)>
<!--ATTLIST AirportRoutePref
    PreferLevel (Only | Unacceptable | Preferred) "Preferred" #IMPLIED
-->
```

3.3.7 Air Fare Restrictions Preferences

Element	Occur.	Content	Data type	Attributes	Description
AirFareRestrictPref	*	Text	String	PreferLevel, FareRestrictType	Air fare restriction preference to be used with this preference collection

AirFareRestrictPref

Identifies preferences for airfare restrictions accepted or not acceptable for a given travel situation.

Attributes:

PreferLevel : (Only | Unacceptable | Preferred) = Preferred

Identifies the level of preference for the type of fare restriction identified. Values: (Only | Unacceptable | Preferred). If Only, use only this preference for the travel situation; if Unacceptable, do NOT use this preference in the given situation. The default value is 'Preferred'.

FareRestrictType : CDATA

Type of fare restriction: None, Advance, SatStopOver, StandBy, etc.

AirFareRestrictPref DTD fragment

```
<!ELEMENT AirFareRestrictPref (#PCDATA)>
<!--ATTLIST AirFareRestrictPref
    PreferLevel (Only | Unacceptable | Preferred) "Preferred" #IMPLIED
    FareRestrictType CDATA #IMPLIED
-->
```

3.3.8 Flight Type Preferences

Element	Occur.	Content	Data type	Attributes	Description
FlightTypePref	*	Text	String	PreferLevel	Flight type preference to be used with this preference collection

FlightTypePref

Indicates preferences for certain types of flights, such as connections or stopovers, when used for a specific travel situation.

Attributes:

PreferLevel : (Only | Unacceptable | Preferred) = Preferred

Identifies the customer's level of preference for the flight type identified. Values: (Only | Unacceptable | Preferred). If Only, use only this preference for the travel situation; if Unacceptable, do NOT use this preference in the given situation. The default value is 'Preferred'.

FlightTypePref DTD fragment

```
<!ELEMENT FlightTypePref (#PCDATA)>
<!ATTLIST FlightTypePref
  PreferLevel (Only | Unacceptable | Preferred) "Preferred">
```

3.3.9 Airline Equipment Preferences

Element	Occur.	Content	Data type	Attributes	Description
AirEquipPref	*	Text	String	PreferLevel, AirEquipType	Air equipment preference to be used with this preference collection

AirEquipPref

Identifies the type of make/model of aircraft preferred in a travel situation.

Attributes:

PreferLevel : (Only | Unacceptable | Preferred) = Preferred

Identifies the level of preference for the type of aircraft identified. Values: (Only | Unacceptable). If Only, use only this preference for the travel situation; if Unacceptable, do NOT use this preference in the given situation. The default value is 'Preferred'.

AirEquipType : CDATA

Identifies the type of make/model of aircraft equipment preferred for this travel collection.

AirEquipPref DTD fragment

```
<!ELEMENT AirEquipPref (#PCDATA)>
<!ATTLIST AirEquipPref
  PreferLevel (Only | Unacceptable | Preferred) "Preferred" #IMPLIED
  AirEquipType CDATA #IMPLIED>
```

3.3.10 Airline Cabin Preferences

Element	Occur.	Content	Data type	Attributes	Description
AirCabinPref	*	Text	String	PreferLevel, CabinType	Air cabin preference to be used with this preference collection

AirCabinPref

Indicates preferences for choice of airline cabin for a given travel situation.

Attributes:**PreferLevel : (Only | Unacceptable | Preferred) = Preferred**

Identifies the level of preference for the airline cabin type identified. Values: (Only | Unacceptable | Preferred). If Only, use only this preference for the travel situation; if Unacceptable, do NOT use this preference in the given situation. The default value is 'Preferred'.

CabinType : CDATA

Identifies the type of airline cabin, e.g.: First, Business, Coach, etc.

AirCabinPref DTD fragment

<!ELEMENT AirCabinPref (#PCDATA)>

<!--ATTLIST AirCabinPref

PreferLevel (Only | Unacceptable | Preferred) "Preferred" #IMPLIED

CabinType CDATA #IMPLIED>

3.3.11 Airline Service Class Preferences

Element	Occur.	Content	Data type	Attributes	Description
AirSrvcClassPref	*	Text	String	PreferLevel, ServiceClassCode	Air service class preference to be used with this preference collection

AirSrvcClassPref

Indicates preference for type of service class for air travel booking in a given situation, such as First Class, Economy, Super-Saver, etc.

Attributes:**PreferLevel : (Only | Unacceptable | Preferred) = Preferred**

Identifies the level of preference for the airline service class identified. Values: (Only | Unacceptable | Preferred). The default value is 'Preferred'.

ServiceClassCode : CDATA

Identifies the type of service class preferred for airline travel, using standard industry codes for service classes.

AirSrvcClassPref DTD fragment

<!ELEMENT AirSrvcClassPref (#PCDATA)>

<!--ATTLIST AirSrvcClassPref

PreferLevel (Only | Unacceptable | Preferred) "Preferred" #IMPLIED

ServiceClassCode CDATA #IMPLIED>

3.3.12 Airline Seating Preferences

Element	Occur.	Content	Data type	Attributes	Description
AirSeatPref	*	Text	String	PreferLevel	Indicates preferences for seating arrangements for air travel

AirSeatPref

Indicates preferences for seating arrangements for air travel.

Attributes:**PreferLevel : (Only | Unacceptable | Preferred) = Preferred**

Identifies the customer's level of preference for the airline seating identified. Values: (Only | Unacceptable | Preferred). The default value is 'Preferred'.

AirSeatPref DTD fragment

<!ELEMENT AirSeatPref (%Ent.SeatingPref;*)>

<!ATTLIST AirSeatPref

PreferLevel (Only | Unacceptable | Preferred) "Preferred" #IMPLIED>

3.3.13 Airline Ticket Distribution Preferences

The designation of preferences for ticket distribution specifies preferred methods of receiving tickets for travel-related services, such as use of courier, regular mail, or electronic ticketing, and provides the ability for a customer or company to specify the time frame for delivery of tickets.

Element	Occur.	Content	Data type	Attributes	Description
TicketDistribPref	*	Text	String	PreferLevel, DistribType, TicketTime	Ticketing information to be used with this collection of preferences

TicketDistribPref

Indicates preferences for the type of ticket distribution to be used for the travel service product.

Attributes:**PreferLevel : (Only | Unacceptable | Preferred) = Preferred**

Identifies the level of preference for the type of ticket distribution identified. Values: (Only | Unacceptable | Preferred). If Only, use only this preference for the travel situation; if Unacceptable, do NOT use this preference in the given situation. The default value is 'Preferred'.

DistribType : CDATA

Ticket distribution method; such as Fax, Email, Courier, Mail, Airport_Pickup, City_Office, Hotel_Desk, WillCall, etc.

TicketTime : CDATA

Ticket turnaround time desired, amount of time requested to deliver tickets.

TicketDistribPref DTD fragment

<!ELEMENT TicketDistribPref (#PCDATA)>

<!ATTLIST TicketDistribPref

PreferLevel (Only | Unacceptable | Preferred) "Preferred" #IMPLIED

DistribType CDATA #IMPLIED

TicketTime CDATA #IMPLIED>

3.3.14 Airline Entertainment Preferences

The designation of media and entertainment preferences specifies certain entertainment or media-related services sought as part of a travel situation, which might include a preference for flights that offer movies or stereo entertainment, or hotels that offer Cable TV, etc.

Element	Occur.	Content	Data type	Attributes	Description
MediaEntertainPref	*	Text	String	PreferLevel	Media and entertainment information preferences

MediaEntertainPref

Identifies the media and entertainment preferences for a given travel situation, such as: books, magazines, movies, newspapers, radio, television, cinema, games, online, etc.

Attributes:

PreferLevel : (Only | Unacceptable | Preferred) = Preferred

Identifies the level of preference for the media and entertainment identified. Values: (Only | Unacceptable | Preferred). If Only, use only this preference for the travel situation; if Unacceptable, do NOT use this preference in the given situation. The default value is 'Preferred'.

MediaEntertainPref DTD fragment

```
<ELEMENT MediaEntertainPref (#PCDATA)>
<!ATTLIST MediaEntertainPref
  PreferLevel (Only | Unacceptable | Preferred) "Preferred" #IMPLIED>
```

3.3.15 Airline Pet Information Preferences

The inclusion of pet information allows a customer to specify preferences for traveling with a pet, such as hotel accommodations that allow pets, boarding facilities, containers for shipping pets, etc.

Element	Occur.	Content	Data type	Attributes	Description
PetInfoPref	*	Text	String	PreferLevel	Pet information to be used with this collection of preferences

PetInfoPref

Indicates the preferences for information about pets that accompany the customer in a given travel situation.

Attributes:

PreferLevel : (Only | Unacceptable | Preferred) = Preferred

Identifies the level of preference for the pet information identified. Values: (Only | Unacceptable | Preferred). If Only, use only this pet information preference for the travel situation; if Unacceptable, do NOT use this pet information preference in the given situation. The default value is 'Preferred'.

PetInfoPref DTD fragment

```
<ELEMENT PetInfoPref (#PCDATA)>
<!ATTLIST PetInfoPref
  PreferLevel (Only | Unacceptable | Preferred) "Preferred" #IMPLIED>
```

3.3.16 Airline Meal Preferences

The Meal Preference entity allows a customer to indicate preferences for food and beverages, including dietary restrictions such as vegetarian or low fat meals, or favorite dining experiences, such as all-American or ethnic restaurants.

Element	Occur.	Content	Data type	Attributes	Description
MealPref	*	Text	String	PreferLevel, MealType,	Food and beverage preferences to be used with this collection

				FavoriteFood, Beverage	
--	--	--	--	---------------------------	--

MealPref

Indicates the customer's preference for meals, favorite foods, and beverages.

Attributes:

PreferLevel : (Only | Unacceptable | Preferred) = Preferred

Identifies the level of preference for the meal or beverage type identified. Values: (Only | Unacceptable | Preferred). If Only, use only this preference for the travel situation; if Unacceptable, do NOT use this preference in the given situation. The default value is 'Preferred'.

MealType : CDATA

Identifies a type of meal preferred or dietary restrictions to be used for a given travel situation, such as vegetarian, or low-sodium meals.

FavoriteFood : CDATA

Indicates the customer's dining preferences, NOT dietary restrictions, to be used for a given travel situation.

Beverage : CDATA

Indicates the type of drink(s) preferred for a given type of travel situation.

MealPref DTD fragment

```
<!ELEMENT MealPref (#PCDATA)>
<!ATTLIST MealPref
  PreferLevel (Only | Unacceptable | Preferred) "Preferred"
  MealType CDATA #IMPLIED
  FavoriteFood CDATA #IMPLIED
  Beverage CDATA #IMPLIED
>
```

3.3.17 Airline Special Requests

The special request preference element allows the customer to designate a special request to be associated with airline travel.

Element	Occur.	Content	Data type	Attributes	Description
SpecRequestPref	*	Text	String	PreferLevel	Special request to be used with this preference collection

SpecRequestPref

Identifies a special request associated with airline travel.

Attributes:

PreferLevel : (Only | Unacceptable | Preferred) = Preferred

Identifies the level of preference for the special request identified. Values: (Only | Unacceptable | Preferred). If Only, use only this preference for the travel situation; if Unacceptable, do NOT use this preference in the given situation. The default value is 'Preferred'.

SpecRequestPref DTD fragment

```
<!ELEMENT SpecRequestPref (#PCDATA)>
<!ATTLIST SpecRequestPref
  PreferLevel (Only | Unacceptable | Preferred) "Preferred" #IMPLIED
```

>

3.3.18 Airline Special Service Requests

Element	Occur.	Content	Data type	Attributes	Description
AirSSR_Pref	*	Text	String	PreferLevel, SSR_Code	Identifies preferences for special services required for air travel, using standard industry (SSR-OSI) code list

AirSSR_Pref

Identifies preferences for special services required for air travel, using standard industry (SSR-OSI) code list.

Attributes:

PreferLevel : (Only | Unacceptable | Preferred) = Preferred

Identifies the level of preference for the air travel special service request identified by the SSR_OSI code. Values: (Only | Unacceptable | Preferred). If Only, use only this preference for the travel situation; if Unacceptable, do NOT use this preference in the given situation. The default value is 'Preferred'.

SSR_Code : CDATA

Code of the special service request to be used for this air travel situation. Refers to standard industry (SSR-OSI) code list.

AirSSR_Pref DTD fragment

```
<ELEMENT AirSSR_Pref (#PCDATA)>
<!ATTLIST AirSSR_Pref
  PreferLevel (Only | Unacceptable | Preferred) "Preferred" #IMPLIED
  SSR_Code CDATA #IMPLIED
```

>

3.3.19 TPA_Extensions

Due to the proprietary nature of agreements between business relationships, the TPA_Extensions element is included in this section of the profile, and allows for companies to add selected information as needed.

Element	Occur.	Content	Data type	Attributes	Description
TPA_Extensions	?				Due to the proprietary nature of agreements between business entities, this element allows for companies to add selected information as needed.

TPA_Extensions

Due to the proprietary nature of agreements between business entities, this element allows for companies to add selected information as needed.

TPA_Extensions DTD fragment

```
<ELEMENT TPA_Extensions ANY>
```

3.4 Hotel Preferences

In the Hotel Preferences section (Figure 22) customers or companies can select their preferences for various amenities of hotel properties and rooms. Travelers may identify preferred hotel chains, properties, and loyalty programs, as well as the desired location, property type, property class, and room location. Customers can indicate their preferences for services provided by hotel properties, such as recreational and business services, as well as in-room amenities, including type of bed. Travelers may identify dietary restrictions and dining preferences, as well as indicate food services preferred in hotel properties. The specification allows for identification of media and entertainment preferences during hotel visits, as well as personal services and other special requests. Customers may indicate preferences for security features and requirements for features to meet the needs of physically challenged guests.

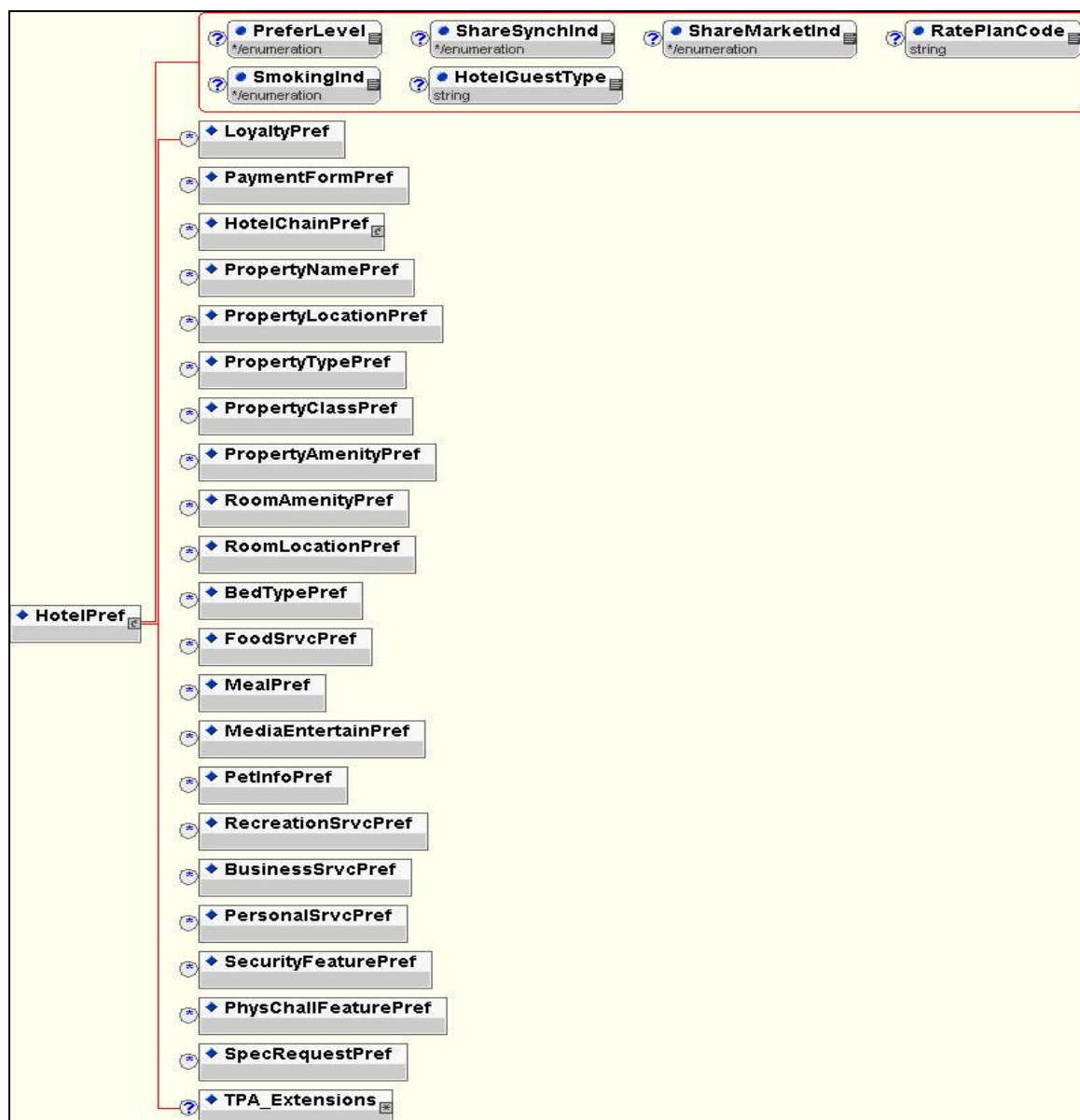


Figure 22. Hotel Preferences

3.4.1 Hotel Preferences Collection

Element	Occur.	Content	Data type	Attributes	Description
HotelPref	?	Element		PreferLevel, %Privacy RatePlanCode, SmokingInd, HotelGuestType	Hotel Preferences
LoyaltyPref	*	Element		PreferLevel	Loyalty programs to be used with this preference collection
PaymentFormPref	*	Element		PreferLevel, RPH	Form(s) of payment to be used with this preference collection
HotelChainPref	*	Element		PreferLevel	Hotel Chain to be used with this preference collection
PropertyNamePref	*	Element		PreferLevel	Hotel property to be used with this preference collection
PropertyLocationPref	*	Element		PreferLevel, PropertyLocationType	Location of property to be used with this preference collection
PropertyTypePref	*	Element		PreferLevel, PropertyType	Type of property to be used with this preference collection
PropertyClassType	*	Element		PreferLevel, PropertyClassType	Property class type to be used with this preference collection
PropertyAmenityPref	*	Element		PreferLevel, PropertyAmenityType	Property amenity(ies) to be used with this preference collection
RoomAmenityPref	*	Element		PreferLevel, RoomAmenityType	Room amenity(ies) to be used with this preference collection
RoomLocationPref	*	Element		PreferLevel, RoomLocationType	Location of room to be used with this preference collection
BedTypePref	*	Element		PreferLevel, BedType	Type of bed to be used with this preference collection
FoodSrvPref	*	Element		PreferLevel, FoodSrvType	Food and beverage preferences to be used with this preference collection
MealPref	*	Element		PreferLevel, MealType, FavoriteFood, Beverage	Meal preferences to be used with this preference collection
MediaEntertainPref	*	Element		PreferLevel	Media and entertainment information preferences
PetInfoPref	*	Element		PreferLevel	Pet information to be used with this preference collection
RecreationSrvPref	*	Element		PreferLevel, RecreationSrvType	Recreation service(s) to be used with this preference collection
BusinessSrvPref	*	Element		PreferLevel, BusinessSrvType	Business services(s) to be used with this preference collection
PersonalSrvPref	*	Element		PreferLevel, PersonalSrvType	Personal service(s) to be used with this preference collection
SecurityFeaturePref	*	Element		PreferLevel	Security feature(s) to be used with this preference collection
PhysChallFeaturePref	*	Element		PreferLevel, PhysChallFeature	Physically challenged feature preferences to be used with this

					preference collection
SpecRequestPref	*	Element		PreferLevel	Special requests to be used with this preference collection
TPA_Extensions	?	Element			Due to the proprietary nature of agreements between business entities, this element allows for companies to add selected information as needed.

HotelPref

Hotel preferences used for this collection.

Attributes:

PreferLevel : (Only | Unacceptable | Preferred) = Preferred

Identifies the level of preference for the collection of hotel preferences identified. Values: (Only | Unacceptable | Preferred). If Only, use only this preference for the travel situation; if Unacceptable, do NOT use this preference in the given situation. The default value is 'Preferred'.

ShareSynchInd : (Yes | No | Inherit) = Inherit

Permission for sharing data for synchronization of profiles held by other travel service providers. Enumerated values: 'Yes', 'No', 'Inherit'. Default = 'Inherit'.

ShareMarketInd : (Yes | No | Inherit) = Inherit

Permission for sharing data for marketing information. Enumerated values: 'Yes', 'No', 'Inherit'. Default = 'Inherit'.

RatePlanCode : CDATA

RatePlanCode identifying preferred hotel rate plan for this travel situation.

SmokingInd : (Yes | No)

Indicates the customer's preference for smoking accommodations. Values: (Yes | No).

HotelGuestType : CDATA

Categories of hotel guests, TBD.

HotelPref DTD fragment

```
<!ELEMENT HotelPref (LoyaltyPref*, PaymentFormPref*, HotelChainPref*, PropertyNamePref*,
PropertyLocationPref*, PropertyTypePref*, PropertyClassPref*, PropertyAmenityPref*,
RoomAmenityPref*, RoomLocationPref*, BedTypePref*, FoodSrvcPref*, MealPref*,
MediaEntertainPref*, PetInfoPref*, RecreationSrvcPref*, BusinessSrvcPref*, PersonalSrvcPref*,
SecurityFeaturePref*, PhysChallFeaturePref*, SpecRequestPref*, TPA_Extensions?)>
<!ATTLIST HotelPref PreferLevel (Only | Unacceptable | Preferred) "Preferred"
%Ent.Privacy;
RatePlanCode CDATA #IMPLIED
SmokingInd (Yes | No) #IMPLIED
HotelGuestType CDATA #IMPLIED
>
```

3.4.1.1 Hotel Loyalty Preferences

Indicates a preference for the loyalty program to be used for hotel accommodations. The RPH (Reference Place Holder) attribute designates a specific loyalty program from a collection stored in the profile. The same LoyaltyPref element is used in all preference collections.

Element	Occur.	Content	Data type	Attributes	Description
LoyaltyPref	*	Text	String	PreferLevel, LoyaltyRPH	Loyalty programs to be used with this preference collection

LoyaltyPref

Preferred loyalty program to be used for hotel services in a given travel situation.

Attributes:

PreferLevel : (Only | Unacceptable | Preferred) = Preferred

Identifies the level of preference for the loyalty program identified. Values: (Only | Unacceptable | Preferred). If Only, use only this loyalty program preference for the travel situation; if Unacceptable, do NOT use this loyalty program preference in the given situation. The default value is 'Preferred'.

LoyaltyRPH : CDATA

Reference to the loyalty program to be used in this travel collection.

LoyaltyPref DTD fragment

```
<!ELEMENT LoyaltyPref (#PCDATA)>
<!--ATTLIST LoyaltyPref
  PreferLevel (Only | Unacceptable | Preferred) "Preferred"
  LoyaltyRPH CDATA #IMPLIED-->
```

3.4.1.2 Hotel Payment Form Preferences

This preference indicates a specific type of payment, such as a credit card, to be assigned to a hotel preference collection. The RPH (Reference Place Holder) attribute designates a specific payment form in a collection stored in the profile. The PaymentFormPref element is used in all preference collections.

Element	Occur.	Content	Data type	Attributes	Description
PaymentFormPref	*	Text	String	PreferLevel, RPH	Form(s) of payment to be used with this hotel stay

PaymentForm

Indicates preferences for the form of payment to be used for hotel payment associated with this collection.

Attributes:

PreferLevel : (Only | Unacceptable | Preferred) = Preferred

Identifies the level of preference for the payment form identified. Values: (Only | Unacceptable | Preferred). If Only, use only this preference for the travel situation; if Unacceptable, do NOT use this preference in the given situation. The default value is 'Preferred'.

RPH : CDATA

Reference to the payment form to be used in this travel collection. Identifies a specific form of payment defined under Customer.

PaymentFormPref DTD fragment

```
<!ELEMENT PaymentFormPref (#PCDATA)>
<!--ATTLIST PaymentFormPref
  PreferLevel (Only | Unacceptable | Preferred) "Preferred" #IMPLIED
  RPH CDATA #IMPLIED
-->
```

3.4.1.3 Hotel Chain Preferences

Element	Occur.	Content	Data type	Attributes	Description
HotelChainPref	*	Text	String	PreferLevel	Hotel chain preference to be used with this preference collection

HotelChainPref

Company identifiers for hotel chains associated with this collection of preferences.

Attributes:

PreferLevel : (Only | Unacceptable | Preferred) = Preferred

Identifies the level of preference for the hotel chain identified. Values: (Only | Unacceptable | Preferred). If Only, use only this preference for the travel situation; if Unacceptable, do NOT use this preference in the given situation. The default value is 'Preferred'.

HotelChainPref DTD fragment

```
<!ELEMENT HotelChainPref (%Ent.CompanyName;)?>
<!--ATTLIST HotelChainPref
    PreferLevel (Only | Unacceptable | Preferred) "Preferred" #IMPLIED
-->
```

3.4.1.4 Hotel Property Name Preferences

Element	Occur.	Content	Data type	Attributes	Description
PropertyNamePref	*	Text	String	PreferLevel	Property name preference to be used with this preference collection

PropertyNamePref

Name of preferred hotel properties associated with this collection.

Attributes:

PreferLevel : (Only | Unacceptable | Preferred) = Preferred

Identifies the level of preference for the hotel property identified. Values: (Only | Unacceptable | Preferred). If Only, use only this preference for the travel situation; if Unacceptable, do NOT use this preference in the given situation. The default value is 'Preferred'.

PropertyNamePref DTD fragment

```
<!ELEMENT PropertyNamePref (#PCDATA)>
<!--ATTLIST PropertyNamePref
    PreferLevel (Only | Unacceptable | Preferred) "Preferred" #IMPLIED
-->
```

3.4.1.5 Hotel Location Preferences

Element	Occur.	Content	Data type	Attributes	Description
PropertyLocationPref	*	Text	String	PreferLevel, PropertyLocationType	Property location preference to be used with this preference collection

PropertyLocationPref

Indicates preferences for hotel property locations.

Attributes:

PreferLevel : (Only | Unacceptable | Preferred) = Preferred

Identifies the level of preference for the hotel location type identified. Values: (Only | Unacceptable | Preferred). If Only, use only this preference for the travel situation; if Unacceptable, do NOT use this preference in the given situation. The default value is 'Preferred'.

PropertyLocationType : CDATA

Type of property location, based on HEDNA property location types.

PropertyLocationPref DTD fragment

```
<!ELEMENT PropertyLocationPref (#PCDATA)>
<!ATTLIST PropertyLocationPref
  PreferLevel (Only | Unacceptable | Preferred) "Preferred"
  PropertyLocationType CDATA #IMPLIED
>
```

3.4.1.6 Hotel Property Type Preferences

Element	Occur.	Content	Data type	Attributes	Description
PropertyTypePref	*	Text	String	PreferLevel, PropertyType	Property type preference to be used with this preference collection

PropertyTypePref

Indicates preferences for hotel property types.

Attributes:

PreferLevel : (Only | Unacceptable | Preferred) = Preferred

Identifies the level of preference for the hotel property type identified. Values: (Only | Unacceptable | Preferred). If Only, use only this preference for the travel situation; if Unacceptable, do NOT use this preference in the given situation. The default value is 'Preferred'.

PropertyType : CDATA

Category of hotel property, as defined in the HEDNA list of hotel properties, All-Suite, All-Inclusive Resort, Apartment, Bed&Breakfast, Cabin/Bungalow, Corporate, Hostel, Inn, Health Spa, Holiday Resort, Hotel, Lodge, Condominium, Conference Center, Chalet, Campground, Guest House Limited Service, Guest Farm, Meeting Resort, Monastery, Motel, Ranch, Self-Catering Accommodation, Vacation Home, Villas, Wildlife Reserve.

PropertyTypePref DTD fragment

```
<!ELEMENT PropertyTypePref (#PCDATA)>
<!ATTLIST PropertyTypePref
  PreferLevel (Only | Unacceptable | Preferred) "Preferred" #IMPLIED
  PropertyType CDATA #IMPLIED
>
```

3.4.1.7 Hotel Property Class Preferences

Element	Occur.	Content	Data	Attributes	Description
---------	--------	---------	------	------------	-------------

			type		
PropertyClassPref	*	Text	String	PreferLevel, PropertyClassType	Property class preference to be used with this preference collection

PropertyClassPref

Indicates preferences for class of hotel property.

Attributes:

PreferLevel : (Only | Unacceptable | Preferred) = Preferred

Identifies the level of preference for the hotel property class identified. Values: (Only | Unacceptable | Preferred). If Only, use only this preference for the travel situation; if Unacceptable, do NOT use this preference in the given situation. The default value is 'Preferred'.

PropertyClassType : CDATA

Type of hotel property class, indicating general quality level preferred. Based on HEDNA list of property classes. Luxury, Moderate, Deluxe, Economy, 1st Class, Budget, Tourist, etc.

PropertyClassPref DTD fragment

```
<!ELEMENT PropertyClassPref (#PCDATA)>
<!ATTLIST PropertyClassPref
    PreferLevel (Only | Unacceptable | Preferred) "Preferred" #IMPLIED
    PropertyClassType CDATA #IMPLIED
>
```

3.4.1.8 Hotel Property Amenity Preferences

Element	Occur.	Content	Data type	Attributes	Description
PropertyAmenityPref	*	Text	String	PreferLevel, PropertyAmenityType	Property amenity preference to be used with this preference collection

PropertyAmenityPref

Indicates preferences for hotel property amenities.

Attributes:

PreferLevel : (Only | Unacceptable | Preferred) = Preferred

Identifies the level of preference for the hotel property amenity identified. Values: (Only | Unacceptable | Preferred). If Only, use only this preference for the travel situation; if Unacceptable, do NOT use this preference in the given situation. The default value is 'Preferred'.

PropertyAmenityType : CDATA

Type of features and service offered by a hotel property, based on specified list of property amenities.

PropertyAmenityPref DTD fragment

```
<!ELEMENT PropertyAmenityPref (#PCDATA)>
<!ATTLIST PropertyAmenityPref
    PreferLevel (Only | Unacceptable | Preferred) "Preferred" #IMPLIED
    PropertyAmenityType CDATA #IMPLIED
>
```

3.4.1.9 Hotel Room Amenity Preferences

Element	Occur.	Content	Data type	Attributes	Description
RoomAmenityPref	*	Text	String	PreferLevel, RoomAmenityType	Room amenity preference to be used with this preference collection

RoomAmenityPref

Indicates preferences for hotel room amenities.

Attributes:

PreferLevel : (Only | Unacceptable | Preferred) = Preferred

Identifies the customer's level of preference for the hotel room amenity identified. Values: (Only | Unacceptable | Preferred). If Only, use only this preference for the travel situation; if Unacceptable, do NOT use this preference in the given situation. The default value is 'Preferred'.

RoomAmenityType : CDATA

Special features of hotel rooms, based on specific list of hotel room amenities.

RoomAmenityPref DTD fragment

```
<!ELEMENT RoomAmenityPref (#PCDATA)>
<!ATTLIST RoomAmenityPref
    PreferLevel (Only | Unacceptable | Preferred) "Preferred" #IMPLIED
    RoomAmenityType CDATA #IMPLIED
>
```

3.4.1.10 Hotel Room Location Preferences

Element	Occur.	Content	Data type	Attributes	Description
RoomLocationPref	*	Text	String	PreferLevel, RoomLocationType	Room location preference to be used with this preference collection

RoomLocationPref

Indicates preferences for hotel room locations.

Attributes:

PreferLevel : (Only | Unacceptable | Preferred) = Preferred

Identifies the level of preference for the hotel room location identified. Values: (Only | Unacceptable | Preferred). If Only, use only this preference for the travel situation; if Unacceptable, do NOT use this preference in the given situation. The default value is 'Preferred'.

RoomLocationType : CDATA

Location or type of room in property e.g.: Higher Floor, Lower Floor, Quiet Room, Near Elevator, etc.

RoomLocationPref DTD fragment

```
<!ELEMENT RoomLocationPref (#PCDATA)>
<!ATTLIST RoomLocationPref
    PreferLevel (Only | Unacceptable | Preferred) "Preferred" #IMPLIED
    RoomLocationType CDATA #IMPLIED
>
```

3.4.1.11 Hotel Bed Type Preferences

Element	Occur.	Content	Data type	Attributes	Description
BedTypePref	*	Text	String	PreferLevel, BedType	Bed type preference to be used with this preference collection

BedTypePref

Indicates preferences for the size and features of hotel bed types.

Attributes:

PreferLevel : (Only | Unacceptable | Preferred) = Preferred

Identifies the level of preference for the hotel bed type identified. Values: (Only | Unacceptable | Preferred). If Only, use only this preference for the travel situation; if Unacceptable, do NOT use this preference in the given situation. The default value is 'Preferred'.

BedType : CDATA

Type of bed indicated; King, Queen, Double, Double/Twin, etc.

BedTypePref DTD fragment

```
<!ELEMENT BedTypePref (#PCDATA)>
<!ATTLIST BedTypePref
  PreferLevel (Only | Unacceptable | Preferred) "Preferred" #IMPLIED
  BedType CDATA #IMPLIED>
```

3.4.1.12 Hotel Food Service Preferences

Element	Occur.	Content	Data type	Attributes	Description
FoodSrvcPref	*	Text	String	PreferLevel, FoodSrvcType	Hotel food service preference to be used with this preference collection

FoodSrvcPref

Indicates preferences for type of restaurant facilities in a hotel.

Attributes:

PreferLevel : (Only | Unacceptable | Preferred) = Preferred

Identifies the level of preference for the hotel food service identified. Values: (Only | Unacceptable | Preferred). If Only, use only this preference for the travel situation; if Unacceptable, do NOT use this preference in the given situation. The default value is 'Preferred'.

FoodSrvcType : CDATA

Type of food services offered at a hotel property. Based on the HEDNA list of restaurant services. Full Service Restaurant, Coffee Shop, Buffet, Bar/Lounge, Snack Bar, Ice Cream/Dessert Shop, Fast Food.

FoodSrvcPref DTD fragment

```
<!ELEMENT HotelFoodSrvcPref (#PCDATA)>
<!ATTLIST HotelFoodSrvcPref
  PreferLevel (Only | Unacceptable | Preferred) "Preferred" #IMPLIED
  FoodSrvcType CDATA #IMPLIED>
```

3.4.1.13 Hotel Meal Type Preferences

The Meal Preference entity allows a customer to indicate preferences for food and beverages, including dietary restrictions such as vegetarian or low fat meals, or favorite dining experiences, such as all-American or ethnic restaurants.

Element	Occur.	Content	Data type	Attributes	Description
MealPref	*	Text	String	PreferLevel, MealType, FavoriteFood, Beverage	Food and beverage preferences to be used with this collection

MealPref

Identifies preferences for the type of meal preferred at a hotel in a given travel situation.

Attributes:

PreferLevel : (Only | Unacceptable | Preferred) = Preferred

Identifies the level of preference for the hotel meal information identified. Values: (Only | Unacceptable | Preferred). If Only, use only this preference for the travel situation; if Unacceptable, do NOT use this preference in the given situation. The default value is 'Preferred'.

MealPref DTD fragment

```
<ELEMENT MealPref (#PCDATA)>
<!ATTLIST MealPref
  PreferLevel (Only | Unacceptable | Preferred) "Preferred"
  MealType CDATA #IMPLIED
  FavoriteFood CDATA #IMPLIED
  Beverage CDATA #IMPLIED>
```

3.4.1.14 Hotel Entertainment and Media Preferences

The designation of media and entertainment preferences specifies certain entertainment or media-related services sought as part of a travel situation, which might include a preference for flights that offer movies or stereo entertainment, or hotels that offer Cable TV, etc.

Element	Occur.	Content	Data type	Attributes	Description
MediaEntertainPref	*	Text	String	PreferLevel	Media and entertainment information preferences

MediaEntertainPref

Identifies the media and entertainment preferences for a given travel situation, such as: books, magazines, movies, newspapers, radio, television, cinema, games, online, etc. This element could be used to indicate a preference for online connectivity, such as internet or modum dial-up connections when applicable to a given travel situation.

Attributes:**PreferLevel : (Only | Unacceptable | Preferred) = Preferred**

Identifies the level of preference for the media and entertainment identified. Values: (Only | Unacceptable | Preferred). If Only, use only this preference for the travel situation; if Unacceptable, do NOT use this preference in the given situation. The default value is 'Preferred'.

MediaEntertainPref DTD fragment

```
<!ELEMENT MediaEntertainPref (#PCDATA)>
<!ATTLIST MediaEntertainPref
  PreferLevel (Only | Unacceptable | Preferred) "Preferred" >
```

3.4.1.15 Hotel Pet Information Preferences

The inclusion of pet information allows a customer to specify preferences for traveling with a pet, such as hotel accommodations that allow pets, boarding facilities, containers for shipping pets, etc.

Element	Occur.	Content	Data type	Attributes	Description
PetInfoPref	*	Text	String	PreferLevel	Pet information to be used with this collection of preferences

PetInfoPref

Indicates the preferences for information about pets that accompany the customer in a given travel situation.

Attributes:**PreferLevel : (Only | Unacceptable | Preferred) = Preferred**

Identifies the level of preference for the pet information supplied. Values: (Only | Unacceptable | Preferred). If Only, use only this pet information preference for the travel situation; if Unacceptable, do NOT use this pet information preference in the given situation. The default value is 'Preferred'.

PetInfoPref DTD fragment

```
<!ELEMENT PetInfoPref (#PCDATA)>
<!ATTLIST PetInfoPref
  PreferLevel (Only | Unacceptable | Preferred) "Preferred">
```

3.4.1.16 Hotel Recreational Services Preferences

Element	Occur.	Content	Data type	Attributes	Description
RecreationSrvcPref	*	Text	String	PreferLevel, RecreationSrvcType	Indicates preferences for type of recreational services in a hotel

RecreationSrvcPref

Indicates preferences for type of recreational services in a hotel.

Attributes:**PreferLevel : (Only | Unacceptable | Preferred) = Preferred**

Identifies the customer's level of preference for the hotel recreational service identified. Values: (Only | Unacceptable | Preferred). If Only, use only this preference for the travel situation; if Unacceptable, do NOT use this preference in the given situation. The default value is 'Preferred'.

RecreationSrvcType : CDATA

Type of recreational services offered at a hotel property,

RecreationSrvcPref DTD fragment

```
<ELEMENT RecreationSrvcPref (#PCDATA)>
<!ATTLIST RecreationSrvcPref
    PreferLevel (Only | Unacceptable | Preferred) "Preferred" #IMPLIED
    RecreationSrvcType CDATA #IMPLIED
>
```

3.4.1.17 Hotel Business Services Preferences

Element	Occur.	Content	Data type	Attributes	Description
BusinessSrvcPref	*	Text	String	PreferLevel, BusinessSrvcType	Indicates preferences for type of business services in a hotel

BusinessSrvcPref

Indicates preferences for type of business services in a hotel.

Attributes:**PreferLevel : (Only | Unacceptable | Preferred) = Preferred**

Identifies the customer's level of preference for the hotel business service identified. Values: (Only | Unacceptable | Preferred). If Only, use only this preference for the travel situation; if Unacceptable, do NOT use this preference in the given situation. The default value is 'Preferred'.

BusinessSrvcType : CDATA

Type of business related services offered in a hotel, such as; Business Center, Copy Machine, Copy Center, Fax Machine, Computers, Computers available for rent, Computer printing available, Pager rental, Cell phone rental, Audio-Visual Equipment rental, Internet Connectivity, Computer modem hookups, Secretarial Services, Video Conferencing, etc.

BusinessSrvcPref DTD fragment

```
<ELEMENT BusinessSrvcPref (#PCDATA)>
<!ATTLIST BusinessSrvcPref
    PreferLevel (Only | Unacceptable | Preferred) "Preferred" #IMPLIED
    BusinessSrvcType CDATA #IMPLIED
>
```

3.4.1.18 Hotel Personal Services Preferences

Element	Occur.	Content	Data type	Attributes	Description
PersonalSrvcPref	*	Text	String	PreferLevel, PersonalSrvcType	Indicates preferences for type of personal services in a hotel

PersonalSvcPref

Indicates preferences for type of personal services in a hotel.

Attributes:

PreferLevel : (Only | Unacceptable | Preferred) = Preferred

Identifies the customer's level of preference for the hotel personal service identified. Values: (Only | Unacceptable | Preferred). If Only, use only this preference for the travel situation; if Unacceptable, do NOT use this preference in the given situation. The default value is 'Preferred'.

PersonalSvcType : CDATA

Type of personal service offered by a hotel property, such as: Express Checkin, Express Checkout, Concierge Service, Laundry/Valet Service, Childcare, Currency Exchange /Banking, Safety Deposit Boxes, Language Translation, Beauty Shop/ Barber/Hairdresser available, Beauty treatments, Massage Service, etc.

PersonalSvcPref DTD fragment

```
<!ELEMENT PersonalSvcPref (#PCDATA)>
<!ATTLIST PersonalSvcPref
  PreferLevel (Only | Unacceptable | Preferred) "Preferred" #IMPLIED
  PersonalSvcType CDATA #IMPLIED
>
```

3.4.1.19 Hotel Security Features Preferences

Element	Occur.	Content	Data type	Attributes	Description
SecurityFeaturePref	*	Text	String	PreferLevel	Indicates preferences for type of safety and security features in a hotel

SecurityFeaturePref

Indicates preferences for type of safety and security features in a hotel.

Attributes:

PreferLevel : (Only | Unacceptable | Preferred) = Preferred

Identifies the customer's level of preference for the hotel security feature identified. Values: (Only | Unacceptable | Preferred). If Only, use only this preference for the travel situation; if Unacceptable, do NOT use this preference in the given situation. The default value is 'Preferred'.

SecurityFeaturePref DTD fragment

```
<!ELEMENT SecurityFeaturePref (#PCDATA)>
<!ATTLIST SecurityFeaturePref
  PreferLevel (Only | Unacceptable | Preferred) "Preferred" #IMPLIED
>
```

3.4.1.20 Hotel Physically Challenged Features Preferences

Element	Occur.	Content	Data type	Attributes	Description
PhysChallFeaturePref	*	Text	String	PreferLevel, PhysChallFeature	Indicates preferences for type of features required to meet the needs of persons with physical challenges, such as handicapped parking, special

					pillows for allergies, etc
--	--	--	--	--	----------------------------

PhysChallFeaturePref

Indicates preferences for type of features required to meet the needs of persons with physical challenges, such as handicapped parking, special pillows for allergies, etc.

Attributes:

PreferLevel : (Only | Unacceptable | Preferred) = Preferred

Indicates the level of preference for the hotel physically challenged feature identified. Values: (Only | Unacceptable | Preferred). If Only, use only this preference for the travel situation; if Unacceptable, do NOT use this preference in the given situation. The default value is 'Preferred'.

PhysChallFeature : CDATA

Type of physically challenged feature needed. Such as; Closed-caption TV, Knock lights, Rails in bathroom, Wheelchair accessible, Elevators, Disabled parking, Television amplifier, Safety Bars in shower, Raised toilet with grab bars, Bathtub seat, Walk-in shower, Visual alarm, etc.

PhysChallFeaturePref DTD fragment

```
<ELEMENT PhysChallFeaturePref (#PCDATA)>
<!ATTLIST PhysChallFeaturePref
    PreferLevel (Only | Unacceptable | Preferred) "Preferred" #IMPLIED
    PhysChallFeature CDATA #IMPLIED
>
```

3.4.1.21 Hotel Special Request Preferences

The special request preference element allows a special request to be associated with hotel stays. This element may be repeated.

Element	Occur.	Content	Data type	Attributes	Description
SpecRequestPref	*	Text	String	PreferLevel	Special request to be used with this preference collection

SpecRequestPref

Indicates preferences for special requests for other services needed for hotel stays.

Attributes:

PreferLevel : (Only | Unacceptable | Preferred) = Preferred

Identifies the level of preference for the special request identified. Values: (Only | Unacceptable | Preferred). If Only, use only this preference for the travel situation; if Unacceptable, do NOT use this preference in the given situation. The default value is 'Preferred'.

SpecRequestPref DTD fragment

```
<ELEMENT SpecRequestPref (#PCDATA)>
<!ATTLIST SpecRequestPref
    PreferLevel (Only | Unacceptable | Preferred) "Preferred" #IMPLIED
>
```

3.4.1.22 TPA_Extensions

Due to the proprietary nature of agreements between business relationships, the TPA_Extensions element is included in this section of the profile, and allows for companies to add selected information as needed.

Element	Occur.	Content	Data type	Attributes	Description
TPA_Extensions	?				Due to the proprietary nature of agreements between business entities, this element allows for companies to add selected information as needed.

TPA_Extensions

Due to the proprietary nature of agreements between business entities, this element allows for companies to add selected information as needed.

TPA_Extensions DTD fragment

```
<!ELEMENT TPA_Extensions ANY>
```

3.5 Preferences for Other Travel Services

This specification allows customers and companies to identify services for other, as yet undefined travel services. The Other Preferences section includes items for the names and types of other services, as well as loyalty programs, forms of payment, and special requests related to these other services. Travel service vendors may wish to use this section to identify other travel services, such as golf outings, cruise industry preferences, etc., until such time as those industries develop specific profile information that is incorporated into OpenTravel Alliance specifications. Any number of other service preferences may be specified, along with the capability to add elements through the TPA_Extensions element that is part of each travel service collection and the Preferences collection as a whole.

While each of the elements in the Other Services Preferences section (Figure 23) is OPTIONAL, if any of these elements appear in an OTA message, they MUST appear in the designated order.

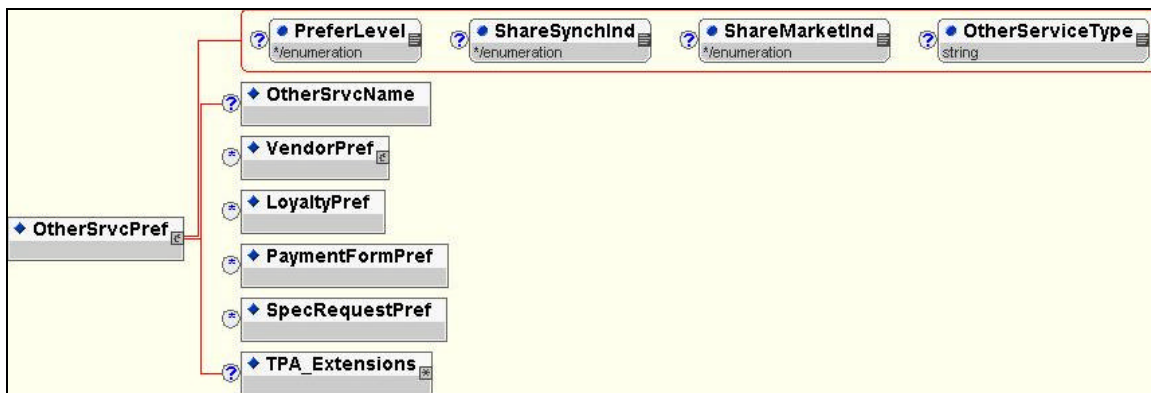


Figure 23. Other Preferences

3.5.1 Other Travel Services Collection

Element	Occur.	Content	Data type	Attributes	Description
OtherSrvcPref	*	Element		PreferLevel, %Privacy, OtherServiceType	Other Travel Service Preferences
OtherSrvcName	?	Element			Name of other travel service to be used with this collection preference
VendorPref	*	Element		PreferLevel	Vendor to be used with this collection preference
LoyaltyPref	*	Element		PreferLevel, LoyaltyRPH	Loyalty programs to be used with this collection preference
PaymentFormPref	*	Element		PreferLevel, RPH	Form(s) of payment to be used with this collection preference
SpecRequestPref	*	Element		PreferLevel	Special requests to be used with this collection preference
TPA_Extensions	?	Element			Due to the proprietary nature of agreements between business entities, this element allows for companies to add selected information as needed.

OtherSrvcPref

A collection to identify customer or company preferences for travel services outside the traditional industries of car rental companies, hotels, or airlines.

Attributes:

PreferLevel : (Only | Unacceptable | Preferred) = Preferred

Identifies the customer's level of preference for this collection of travel services. Values: (Only | Unacceptable | Preferred). If Only, use only this preference for the travel situation; if Unacceptable, do NOT use this preference in the given situation. The default value is 'Preferred'.

ShareSynchInd : (Yes | No | Inherit) = Inherit

Permission for sharing data for synchronization of profiles held by other travel service providers. Enumerated values: 'Yes', 'No', 'Inherit'. Default = 'Inherit'.

ShareMarketInd : (Yes | No | Inherit) = Inherit

Permission for sharing data for marketing information. Enumerated values: 'Yes', 'No', 'Inherit'. Default = 'Inherit'.

OtherServiceType : CDATA

Category of other travel service user - TBD.

OtherSrvcPref DTD fragment

```
<!ELEMENT OtherSrvcPref (OtherSrvcName?, VendorPref*, LoyaltyPref*, PaymentFormPref*,
SpecRequestPref*, TPA_Extensions?)>
<!ATTLIST OtherSrvcPref
  PreferLevel (Only | Unacceptable | Preferred) "Preferred"
  %Ent.Privacy;
  OtherServiceType CDATA #IMPLIED>
```

3.5.2 Other Travel Services

Specifies the name of the other travel services designated in a preference collection.

Element	Occur.	Content	Data type	Attributes	Description
OtherSrvcName	?	Text	String		Name of other travel service specified in this collection.

OtherSrvcName

Name of other travel services identified in this collection of preferences.

OtherSrvcName DTD fragment

```
<!ELEMENT OtherSrvcName (#PCDATA)
>
```

3.5.3 Other Travel Services Vendor Preferences

Designates a travel services supplier to be used in a specific travel collection. The VendorPref element uses the Company Name entity to identify the preferred company by name and by vendor code. The VendorPref element can be used in all preference collections.

Element	Occur.	Content	Data	Attributes	Description
---------	--------	---------	------	------------	-------------

			type		
VendorPref	*	Text	String	PreferLevel	Vendor to be used with this preference collection
%CompanyName					

VendorPref

Identifies preferences for travel service supplier to be used for a given travel situation.

Attributes:

PreferLevel : (Only | Unacceptable | Preferred) = Preferred

Identifies the customer's level of preference for the vendor identified. Values: (Only | Unacceptable | Preferred). If Only, use only this preference for the travel situation; if Unacceptable, do NOT use this preference in the given situation. The default value is 'Preferred'.

VendorPref DTD fragment

```
<!ELEMENT VendorPref (%Ent.CompanyName);>
<!ATTLIST VendorPref
  PreferLevel (Only | Unacceptable | Preferred) "Preferred"
>
```

ENTITY %CompanyName

Identifies a company or organization associated with the contact.

3.5.4 Other Loyalty Programs

Indicates a preference for the loyalty program to be used for other travel services specified. The RPH (Reference Place Holder) attribute designates a specific loyalty program from a collection stored in the profile. The LoyaltyPref element can be used in all preference collections.

Element	Occur.	Content	Data type	Attributes	Description
LoyaltyPref	*	Text	String	PreferLevel, LoyaltyRPH	Loyalty programs to be used with this preference collection

LoyaltyPref

Preferred loyalty program to be used for other travel services in a given travel situation.

Attributes:

PreferLevel : (Only | Unacceptable | Preferred) = Preferred

Indicates the customer's level of preference for the car rental loyalty program identified. Values: (Only | Unacceptable | Preferred). If Only, use only this preference for the travel situation; if Unacceptable, do NOT use this preference in the given situation. The default value is 'Preferred'.

LoyaltyRPH : CDATA

Index number to be used for reference the loyalty program to be used in this travel collection.

LoyaltyPref DTD fragment

```
<!ELEMENT LoyaltyPref (#PCDATA)>
<!ATTLIST LoyaltyPref
  PreferLevel (Only | Unacceptable | Preferred) "Preferred"
  LoyaltyRPH CDATA #IMPLIED
>
```

3.5.5 Other Payment Form Preferences

This preference indicates a specific type of payment to be assigned to a travel preference collection. The RPH (Reference Place Holder) attribute designates a specific payment form in a collection stored in the profile. The PaymentFormPref element may be used in all collections.

Element	Occur.	Content	Data type	Attributes	Description
PaymentFormPref	*	Text	String	PreferLevel, RPH	Form(s) of payment to be used with this travel service

PaymentFormPref

Indicates preferences for the form of payment to be used for other travel services associated with this collection.

Attributes:

PreferLevel : (Only | Unacceptable | Preferred) = Preferred

Identifies the customer's level of preference for the payment form identified. Values: (Only | Unacceptable | Preferred). If Only, use only this preference for the travel situation; if Unacceptable, do NOT use this preference in the given situation. The default value is 'Preferred'.

RPH : CDATA

Reference to the payment form to be used in this travel collection. Identifies a specific form of payment defined under Customer and CompanyInfo.

PaymentFormPref DTD fragment

```
<IELEMENT PaymentFormPref (#PCDATA)>
<!ATTLIST PaymentFormPref
  PreferLevel (Only | Unacceptable | Preferred) "Preferred" #IMPLIED
  RPH CDATA #IMPLIED
>
```

3.5.6 Other Special Requests

The special request preference element allows the a special request to be associated with a travel service.

Element	Occur.	Content	Data type	Attributes	Description
SpecRequestPref	*	Text	String	PreferLevel	Special request to be used with this preference collection

SpecRequestPref

Indicates preferences for special requests for other services needed for other travel services.

Attributes:

PreferLevel : (Only | Unacceptable | Preferred) = Preferred

Identifies the customer's level of preference for the special request identified. Values: (Only | Unacceptable | Preferred). If Only, use only this preference for the travel situation; if Unacceptable, do NOT use this preference in the given situation. The default value is 'Preferred'.

SpecRequestPref DTD fragment

```

<!ELEMENT SpecRequestPref (#PCDATA)>
<!--ATTLIST SpecRequestPref
  PreferLevel (Only | Unacceptable | Preferred) "Preferred" #IMPLIED
-->

```

3.5.7 TPA_Extensions

Due to the proprietary nature of agreements between business relationships, the TPA_Extensions element is included in this section of the profile, and allows for companies to add selected information as needed.

Element	Occur.	Content	Data type	Attributes	Description
TPA_Extensions	?				Due to the proprietary nature of agreements between business entities, this element allows for companies to add selected information as needed.

TPA_Extensions

Due to the proprietary nature of agreements between business entities, this element allows for companies to add selected information as needed.

TPA_Extensions DTD fragment

```

<!ELEMENT TPA_Extensions ANY>

```

4.0 Affiliations

Affiliations are defined as organizations that offer travel services or benefits with which the customer has contact. These organizations include employers and membership organizations, such as AAA, AARP, or alumni associations that provide travel programs, discounts, and benefits. This section also covers vendor-sponsored travel clubs offering special lounges and related services travel services. These clubs include those provided by airlines at airports or Amtrak for its first class and Metroliner passengers. The structure of information stored about travel arrangers is identical to that of the customer contacts listed under the Customer section, therefore, travel arrangers are identified by a Reference Place Holder (RPH) attribute that provides a reference to the data stored in another location in the Customer Profile.

Affiliations with insurance companies provide information about insurance policies carried by the individual. Insurance policies stored in this section may be used for specific travel situations and are assigned a Reference Place Holder so that they may be referenced from within a travel preference collection. In the section on car rental preferences, insurance refers to insurance coverage needed for car rentals.

In the Affiliations section (Figure 24) customers can have multiple Organizations, Employers, Insurance, Travel Arrangers, and Travel Clubs included in their profile information. While each of these elements is OPTIONAL, if any of these elements appear in an OTA message, they MUST appear in the designated order.

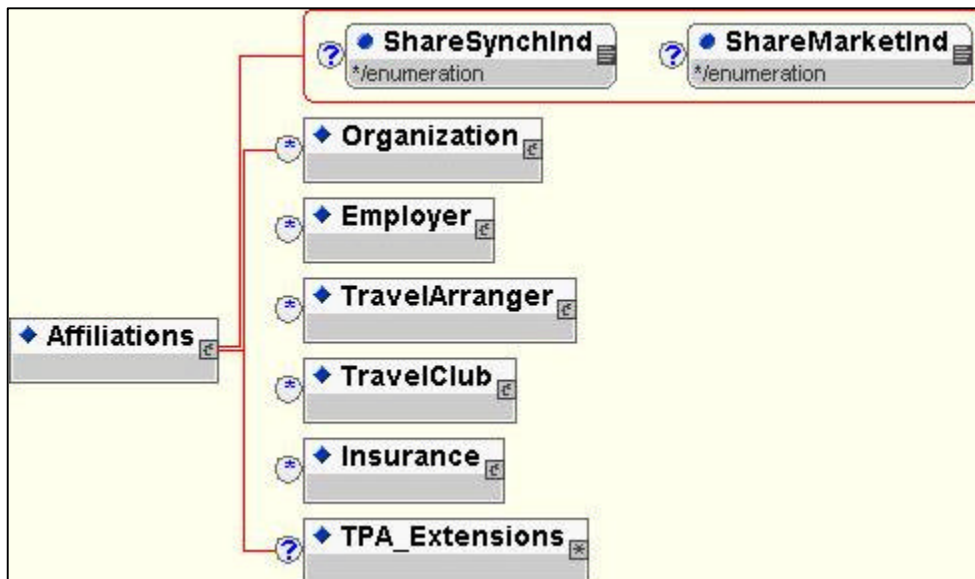


Figure 24. Affiliations

4.1 Affiliations Elements

Element	Occur.	Content	Data type	Attributes	Description
Affiliations	?	Element		%Privacy	Companies or organizations connected with the customer that

					affect travel decisions
Organization	*	Element		DefaultInd, ShareSynchInd, ShareMarketInd, OfficeType, ExpireDate, %Privacy	Membership organization that has travel benefits, programs, or discounts
Employer	*	Element		DefaultInd, ShareSynchInd, ShareMarketInd, OfficeType, ExpireDate	Company or organization that employs the customer
TravelArranger	*	Element		DefaultInd, ShareSynchInd, ShareMarketInd TravelArrangerType, ContactRPH, %Privacy	Individuals or companies affiliated with the customer that are responsible for making travel plans
TravelClub	*	Element		ShareSynchInd, ShareMarketInd ExpireDate, %Privacy	Special-privilege club room and related services offered by travel vendors
Insurance	*	Element		ShareSynchInd, ShareMarketInd InsuranceType, PolicyNumber, EffectiveDate, ExpireDate, RPH, %Privacy	Insurance for travel carried by customer

Affiliations

Defines the organizations that provide travel services and benefit with which the customer has contact.

Attributes:

ShareSynchInd : (Yes | No | Inherit) = Inherit

Permission for sharing data for synchronization of profiles held by other travel service providers. Enumerated values: 'Yes', 'No', 'Inherit'. Default = 'Inherit'.

ShareMarketInd : (Yes | No | Inherit) = Inherit

Permission for sharing data for marketing information. Enumerated values: 'Yes', 'No', 'Inherit'. Default = 'Inherit'.

Affiliations DTD fragment

```
<!ELEMENT Affiliations (Organization*, Employer*, TravelArranger*, TravelClub*, Insurance*,
TPA_Extensions?)>
<!--ATTLIST Affiliations
    %Ent.Privacy;
-->
```

4.2 Affiliated Organization

Element	Occur.	Content	Data type	Attributes	Description
Organization	*	Text	String	DefaultInd, %Privacy,	Membership organization that has travel benefits, programs or

				OfficeType, ExpireDate	discounts
--	--	--	--	---------------------------	-----------

Organization

Membership organization that has travel benefits, programs or discounts.

Attributes:

DefaultInd : (Yes | No)

Indicates if this is a default choice in the absence of the user specifying no overriding value or action.

ShareSynchInd : (Yes | No | Inherit) = Inherit

Permission for sharing data for synchronization of profiles held by other travel service providers. Enumerated values: 'Yes', 'No', 'Inherit'. Default = 'Inherit'.

ShareMarketInd : (Yes | No | Inherit) = Inherit

Permission for sharing data for marketing information. Enumerated values: 'Yes', 'No', 'Inherit'. Default = 'Inherit'.

OfficeType : CDATA

Main office, field office, or division of the organization.

ExpireDate : CDATA

Date when membership in organization ends, in ISO 8601 format. Date only.

Organization DTD fragment

```
<!ELEMENT Organization (OrgMemberName?, OrgName?, RelatedOrgName*,
TravelArranger*)>
<!--ATTLIST Organization
    DefaultInd (Yes | No) #IMPLIED
    %Ent.Privacy;
    OfficeType CDATA #IMPLIED
    ExpireDate CDATA #IMPLIED
-->
```

4.3 Organization Member

Element	Occur.	Content	Data type	Attributes	Description
OrgMemberName	*	Text	String	OrgMemberID, OrgLevel, OrgMemberTitle	Name of the member of the organization, as listed among the options of customer names. Uses the PersonName entity

OrgMemberName

Name of the member of the organization, as listed among the options of customer names. Uses the PersonName entity.

Attributes:

OrgMemberId : CDATA

Identifier assigned to the person by the organization.

OrgLevel : CDATA

Level in organization (e.g. seniority) that conveys privileges.

OrgMemberTitle : CDATA

Rank in organization or title of member. May convey privileges, e.g.: Chairman, President, etc.

OrgMemberName DTD fragment

```
<!ELEMENT OrgMemberName (%Ent.PersonName;)>
<!--ATTLIST OrgMemberName
```

```

    OrgMemberId CDATA #IMPLIED
    OrgLevel CDATA #IMPLIED
    OrgMemberTitle CDATA #IMPLIED
  >

```

4.4 Organization Name

Element	Occur.	Content	Data type	Attributes	Description
OrgName	*	Text	String		Name of the organization

OrgName

Name of the organization.

OrgName DTD fragment

```
<!ELEMENT OrgName (%Ent.CompanyName;)>
```

4.5 Related Organization

Element	Occur.	Content	Data type	Attributes	Description
RelatedOrgName	*	Text	String		Identifies a subsidiary, or entity related to the organization

RelatedOrgName

Identifies a subsidiary, or entity related to the organization.

RelatedOrgName DTD fragment

```
<!ELEMENT RelatedOrgName (#PCDATA)>
```

4.6 Organization's Travel Service Vendor

Identifies the travel agency or travel arranger used by the organization.

Element	Occur.	Content	Data type	Attributes	Description
TravelArranger	*	Text	String	DefaultInd, %Privacy, TravelArrangerType, ContactRPH	Identifies a company or individual responsible for making travel plans or transactions for the company and its employees. The ContactRPH is a reference to the contact information stored in the Customer and CompanyInfo section

TravelArranger

Identifies a company or individual responsible for making travel plans or transactions for the company and its employees. The ContactRPH is a reference to the contact information stored in the Customer and CompanyInfo section.

Public Attributes:**DefaultInd : (Yes | No)**

Indicates if this is a default choice in the absence of the user specifying no overriding value or action.

ShareSynchInd : (Yes | No | Inherit) = Inherit

Permission for sharing data for synchronization of profiles held by other travel service providers. Enumerated values: 'Yes', 'No', 'Inherit'. Default = 'Inherit'.

ShareMarketInd : (Yes | No | Inherit) = Inherit

Permission for sharing data for marketing information. Enumerated values: 'Yes', 'No', 'Inherit'. Default = 'Inherit'.

TravelArrangerType : CDATA

Type of service making travel plans or transactions, e.g.: travel agency, etc.

ContactRPH : CDATA

Reference number to the contact associated with the customer.

TravelArranger DTD fragment

```
<!ELEMENT TravelArranger (%Ent.CompanyName;)>
<!--ATTLIST TravelArranger
    DefaultInd (Yes | No) #IMPLIED
    %Ent.Privacy;
    TravelArrangerType CDATA #IMPLIED
    ContactRPH CDATA #IMPLIED-->
```

4.7 Affiliated Employer

Element	Occur.	Content	Data type	Attributes	Description
Employer	*	Text	String	DefaultInd, %Privacy, OfficeType, ExpireDate	Company or organization that employs the customer

Employer

Company or organization that employs the customer.

Attributes:**DefaultInd : (Yes | No)**

Indicates if this is a default choice in the absence of the user specifying no overriding value or action.

ShareSynchInd : (Yes | No | Inherit) = Inherit

Permission for sharing data for synchronization of profiles held by other travel service providers. Enumerated values: 'Yes', 'No', 'Inherit'. Default = 'Inherit'.

ShareMarketInd : (Yes | No | Inherit) = Inherit

Permission for sharing data for marketing information. Enumerated values: 'Yes', 'No', 'Inherit'. Default = 'Inherit'.

OfficeType : CDATA

Main office, field office, or division of the organization.

ExpireDate : CDATA

Date when employment ends, in ISO 8601 format. Date only.

Employer DTD fragment

```
<!ELEMENT Employer (%Ent.CompanyName;?, RelatedEmployer*, EmployeeInfo?,
InternalRefNmbr*, TravelArranger*, LoyaltyProgram*)>
<!--ATTLIST Employer
    DefaultInd (Yes | No) #IMPLIED
    OfficeType CDATA #IMPLIED
    ExpireDate CDATA #IMPLIED-->
```

>

4.8 Employer Name (Ent. %CompanyName)

Element	Occur.	Content	Data type	Attributes	Description
CompanyName	*	Text	String	CompanyCode, CodeContext	The Company Name entity is used to define the name of the customer's employer

Company Name

The Company Name entity is used to define the name of the customer's employer.

Attributes:

CompanyCode: CDATA

Main office, field office, or division of the organization.

CodeContext : CDATA

Date when employment ends, in ISO 8601 format. Date only.

CompanyName

```
<!ELEMENT CompanyName (#PCDATA)>
<!--ATTLIST CompanyName
      CompanyCode CDATA #IMPLIED
      CodeContext CDATA #IMPLIED
-->
```

4.9 Related Employer

Element	Occur.	Content	Data type	Attributes	Description
RelatedEmployer	*	Text	String		A company such as division or subsidiary, that is related to the employer for which the customer works. The CompanyName entity is used to define the related employer

RelatedEmployer

A company such as division or subsidiary, that is related to the employer for which the customer works. The CompanyName entity is used to define the related employer.

RelatedEmployer

```
<!ELEMENT RelatedEmployer (%Ent.CompanyName;)>
```

4.10 Employee Information

Element	Occur.	Content	Data type	Attributes	Description
EmployeeInfo	*	Text	String	EmployeeID, EmployeeLevel, EmployeeTitle, EmployeeStatusType	Employment information about the customer or person employed by the identified employer. The PCDATA field of EmployeeInfo can be used to record the exact name of the employee if it differs

					from that of the customer defined in the profile
--	--	--	--	--	--

EmployeeInfo

Employment information about the customer or person employed by the identified employer. The PCDATA field of EmployeeInfo can be used to record the exact name of the employee if it differs from that of the customer defined in the profile.

Attributes:

EmployeeId : CDATA

Identifier assigned to the person by the employer.

EmployeeLevel : CDATA

Level in employer organization (e.g. seniority) that conveys privileges.

EmployeeTitle : CDATA

Title of employee in employer company that conveys rank or privileges.

EmployeeStatusType : (Active | Retired | Leave | Terminated)

Status of employment. Values: Active, Retired, Leave, Terminated.

EmployeeInfo DTD fragment

```
<!ELEMENT EmployeeInfo (#PCDATA)>
<!ATTLIST EmployeeInfo
    EmployeeId CDATA #IMPLIED
    EmployeeLevel CDATA #IMPLIED
    EmployeeTitle CDATA #IMPLIED
    EmployeeStatusType (Active | Retired | Leave | Terminated) #IMPLIED
>
```

4.11 Employer's Internal Reference Numbers

Element	Occur.	Content	Data type	Attributes	Description
InternalRefNmbr	*	Text	String		Accounting code(s) assigned to travel for employer

InternalRefNmbr

Accounting code(s) assigned to travel for employer.

InternalRefNmbr DTD fragment

```
<!ELEMENT InternalRefNmbr (#PCDATA)>
```

4.12 Employer Loyalty Programs

Element	Occur.	Content	Data type	Attributes	Description
LoyaltyProgram	*	Text	String	LoyaltyRPH	Loyalty programs with travel service vendors to which the employer subscribes. Loyalty RPH refers back to loyalty programs defined under Customer

LoyaltyProgram

Loyalty programs with travel service vendors to which the employer subscribes. Loyalty RPH refers back to loyalty programs defined under Customer.

Attributes:

LoyaltyRPH : CDATA

Reference number used to identify this loyalty program.

LoyaltyProgram DTD fragment

```
<!ELEMENT LoyaltyProgram (#PCDATA)>
<!ATTLIST LoyaltyProgram
    LoyaltyRPH CDATA #IMPLIED>
```

4.13 Affiliated Travel Arranger

Identifies the travel agency or travel arranger used by the organization.

Element	Occur.	Content	Data type	Attributes	Description
TravelArranger	*	Text	String	DefaultInd, %Privacy, TravelArrangerType, ContactRPH	Identifies a company or individual responsible for making travel plans or transactions for the company and its employees. The ContactRPH is a reference to the contact information stored in the Customer and CompanyInfo section

TravelArranger

Companies or individuals responsible for making travel plans or transactions either for the customer, company, or affiliated organizations. The ContactRPH is a reference to the contact information stored in the Customer and CompanyInfo section.

Public Attributes:

DefaultInd : (Yes | No)

Indicates if this is a default choice in the absence of the user specifying no overriding value or action.

ShareSynchInd : (Yes | No | Inherit) = Inherit

Permission for sharing data for synchronization of profiles held by other travel service providers. Enumerated values: 'Yes', 'No', 'Inherit'. Default = 'Inherit'.

ShareMarketInd : (Yes | No | Inherit) = Inherit

Permission for sharing data for marketing information. Enumerated values: 'Yes', 'No', 'Inherit'. Default = 'Inherit'.

TravelArrangerType : CDATA

Type of service making travel plans or transactions, e.g.: travel agency, etc.

ContactRPH : CDATA

Reference number to the contact associated with the customer.

TravelArranger DTD fragment

```
<!ELEMENT TravelArranger (%Ent.CompanyName;)>
<!ATTLIST TravelArranger
    DefaultInd (Yes | No) #IMPLIED
    %Ent.Privacy;
    TravelArrangerType CDATA #IMPLIED
    ContactRPH CDATA #IMPLIED>
```

4.14 Affiliated Travel Club

Element	Occur.	Content	Data type	Attributes	Description
TravelClub	*	Text	String	%Privacy, ExpireDate	Identifies a travel club that offers special privileges and related services

TravelClub

Identifies a travel club that offers special privileges and related services.

Attributes:

ShareSynchInd : (Yes | No | Inherit) = Inherit

Permission for sharing data for synchronization of profiles held by other travel service providers. Enumerated values: 'Yes', 'No', 'Inherit'. Default = 'Inherit'.

ShareMarketInd : (Yes | No | Inherit) = Inherit

Permission for sharing data for marketing information. Enumerated values: 'Yes', 'No', 'Inherit'. Default = 'Inherit'.

ExpireDate : CDATA

Date club membership ends, in ISO 8601 format. Date only.

TravelClub DTD fragment

```
<!ELEMENT TravelClub (TravelClubName, ClubMemberName?)>
<!--ATTLIST TravelClub
    %Ent.Privacy;
    ExpireDate CDATA #IMPLIED
-->
```

4.15 Travel Club Name

Element	Occur.	Content	Data type	Attributes	Description
TravelClubName	*	Text	String		Name of the travel club
%CompanyName					

TravelClubName

Name of the travel club.

TravelClubName DTD fragment

```
<!ELEMENT TravelClubName (%CompanyName;)>
```

4.16 Travel Club MemberName

Element	Occur.	Content	Data type	Attributes	Description
ClubMemberName	*	Text	String	ClubMemberId	Name of individual registered with the travel club
%PersonName					

ClubMemberName

Name of individual registered with the travel club.

Attributes:**ClubMemberId : CDATA**

Identifier assigned to person registered with the travel club.

ClubMemberName DTD fragment

```
<!ELEMENT ClubMemberName (%Ent.PersonName;?)>
<!ATTLIST MemberName
    ClubMemberId CDATA #IMPLIED
>
```

4.17 Affiliated Insurance Company

Element	Occur.	Content	Data type	Attributes	Description
Insurance	*	Text	String	%Privacy, InsuranceType, PolicyNumber, EffectiveDate, ExpireDate, RPH	Insurance for travel carried by customer

Insurance

Insurance for travel carried by customer.

Public Attributes:**ShareSynchInd : (Yes | No | Inherit) = Inherit**

Permission for sharing data for synchronization of profiles held by other travel service providers. Enumerated values: 'Yes', 'No', 'Inherit'. Default = 'Inherit'.

ShareMarketInd : (Yes | No | Inherit) = Inherit

Permission for sharing data for marketing information. Enumerated values: 'Yes', 'No', 'Inherit'. Default = 'Inherit'.

InsuranceType : CDATA

Type of insurance policy carried by the individual.

PolicyNumber : CDATA

Identifier assigned by insurance company to the policy held by the customer.

EffectiveDate : CDATA

Date insurance coverage begins in ISO 8601 format. Date only.

ExpireDate : CDATA

Date insurance coverage ends in ISO 8601 format. Date only.

RPH : CDATA

Reference place holder or index number assigned to reference this insurance policy from Preferences collection or other specifications.

Insurance DTD fragment

```
<!ELEMENT Insurance (InsuredName?, InsuranceCompany?, Underwriter?)>
<!ATTLIST Insurance
    %Ent.Privacy;
    InsuranceType CDATA #IMPLIED
    PolicyNumber CDATA #REQUIRED
    EffectiveDate CDATA #IMPLIED
    ExpireDate CDATA #IMPLIED
    RPH CDATA #IMPLIED>
```

4.18 Insured Name

Element	Occur.	Content	Data	Attributes	Description
---------	--------	---------	------	------------	-------------

			type		
InsuredName	*	Text	String		Name of the insured party as listed on the insurance policy

InsuredName

Name of the insured party as listed on the insurance policy.

InsuredName

```
<!ELEMENT InsuredName (%Ent.PersonName;?)
>
```

4.19 Insurance Company Name

Element	Occur.	Content	Data type	Attributes	Description
InsuranceCompany	*	Text	String		Identification of the insurance company providing coverage

InsuranceCompany

Identification of the insurance company providing coverage.

InsuranceCompany DTD fragment

```
<!ELEMENT InsuranceCompany (#PCDATA)?>
```

4.20 Insurance Underwriter

Element	Occur.	Content	Data type	Attributes	Description
UnderWriter	*	Text	String		Underwriting company for insurance

Underwriter

Underwriting company for insurance.

Underwriter DTD fragment

```
<!ELEMENT Underwriter (#PCDATA)>
```

4.21 TPA_Extensions

The Customer Profile contains the <TPA_Extensions> element in logical places throughout the specification for trading partners to add their own data within the framework of valid messages.

Element	Occur.	Content	Data type	Attributes	Description
TPA_Extensions	?				Due to the proprietary nature of agreements between business entities, this element allows for companies to add selected information as

					needed.
--	--	--	--	--	---------

TPA_Extensions

Extensions allowed to OTA specifications per bilateral agreement between trading partners.

TPA_Extensions DTD fragment

<!ELEMENT TPA_Extensions ANY>

Trading partners will need to write their own schemas to cover items not included within conforming implementations of the Profile DTD/schema. Trading partners are encouraged to submit extensions widely used between multiple trading partners for consideration to become part of future revisions of OTA specifications.

5.0 Company Information

When a Profile is created for a company or business entity, as opposed to a personal or customer profile, that company is identified by Company Name and by a Company Code. The code assigned to a company may be assigned by the system that creates the company profile, or it may use a code assigned by an external identification system, such as a DUNs number. The Code Context identifies the source of the code.

The Company Name entity is found throughout the OTA Profile as an identifier for a company in relationship to its use, such as a travel arranger, employer, company associated with a contact person, etc. It is used in the CompanyInfo section as the primary identifier of the company.

A profile about a company is usually contains information about a trading partner, or other company with whom there is a business relationship. The CompanyInfo section shares many of the same elements with the Customer section of the specification. Traditional information about the address, telephone numbers email addresses, web pages (URLs) and other mailing or contact information is found in the Company Info section of the profile.

Pertinent information about a business also includes information about where that company is licensed to do business. Local, state, or national jurisdictions may have restrictions on what type of business can be conducted, and who that business can be conducted with. Therefore, the CompanyInfo section provides information about the primary place where a company is licensed to do business, and the opportunity to list alternate business locales, such as branch offices.

Companies have various means of paying for travel services which may range from company credit cards, direct payments from a business account, arrangements to pay for invoices through credit accounts or direct billing, or even by issuing vouchers for which compensation for travel services is paid. The Company Info section uses the Payment Form elements that are found in the Customer section. As with the customer data, company information is protected at the highest level and at each individual element group level with the Privacy attributes that are an inherent part of the OTA Profile.

Companies, as well as customers, have contact persons. Each of those contact persons have the potential for having individual telephone numbers and email addresses, as well as separate mailing addresses, even if those addresses are mailstops within the same building. A company contact person could even serve as an emergency contact. For example, the CEO may be on a business trip and need a contact, such as his secretary, who would be knowledgeable about his family's whereabouts, and capable of dispersing critical information to them as well as to business associates and other parties that have a need to know.

Companies, like customers, organizations, employers, or travel clubs can have travel arrangers who take care of the travel arrangements for their employees. The CompanyInfo section provides a means to identify the travel arranger, whether a person or an outside company such as a travel agency, to be identified and associated with the company information.

Companies can offer loyalty programs to their employees as well as sponsor membership programs that offer services to travelers. The information about a program offered by a company differs from the information about a member of a loyalty program, therefore, the loyalty program element in company info is intended to store information about the program offerings from the company viewpoint.

5.1.1 Required Company data

The specification defines a minimum set of data required for a company profile, with that data located in the CompanyInfo section (Figure 25). Each profile **MUST** have an entry in the following element:

- CompanyName, tag <CompanyName>

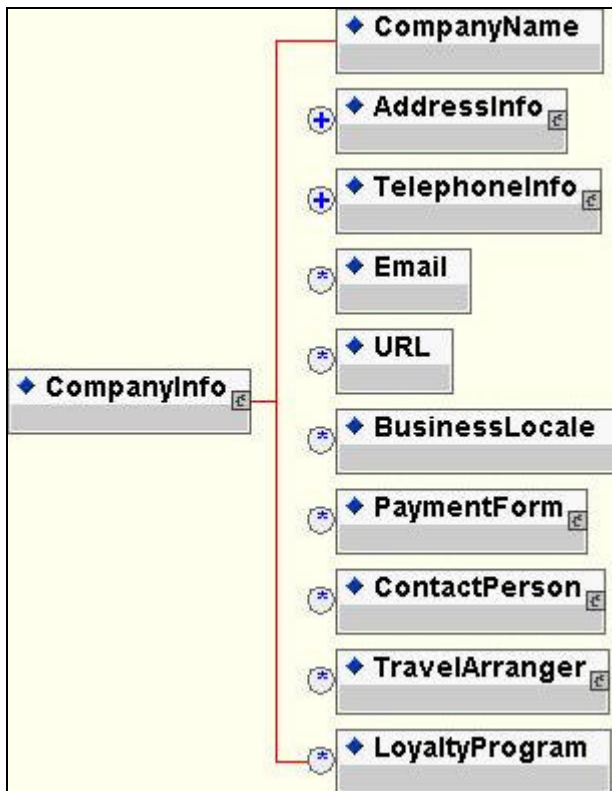


Figure 25. CompanyInfo

5.1.1.1 Company Name

Element	Occur.	Content	Data type	Attributes	Description
CompanyName	*	Text	String	CompanyCode, CodeContext	The Company Name entity is used to define the name of the customer's employer

CompanyName

Identifies a company by name and/or company code that may be associated with a membership or loyalty program, organization or travel service provider.

Attributes:

Company Code : CDATA

Identifies a company by the company code.

CodeContext : CDATA

Identifies the context of the identifying code, such as DUNS or IATA, ISO, etc.

%ENTITY CompanyName

```

<!ELEMENT CompanyName (#PCDATA)>
<!--ATTLIST CompanyName
      CompanyCode CDATA #IMPLIED
      CodeContext CDATA #IMPLIED
-->

```

5.1.1.2 Company Address

Identifies the address of the company. The AddressInfo element, which uses the %Address entity may be repeated to record more than one address, with the AddressType attribute indicating the type of address used. The AddressInfo element, as well as several other elements that define contact information for the company, contains the Privacy attributes, which can be used to control sharing of information for purposes of synchronization of data or for marketing purposes.

Element	Occur.	Content	Data type	Attributes	Description
AddressInfo	*	Element, % Address		DefaultInd %Privacy, AddressType	Precise location of the individual in the profile for mailing and delivery

AddressInfo

Information about the physical addresses contained in the profile.

Attributes:**DefaultInd : (Yes | No)**

Indicates that the receiving system should assume the default value if the user specifies no overriding value or action.

ShareSynchInd : (Yes | No | Inherit) = Inherit

Permission for sharing data for synchronization of profiles held by other travel service providers. Enumerated values: 'Yes', 'No', 'Inherit'. Default = 'Inherit'.

ShareMarketInd : (Yes | No | Inherit) = Inherit

Permission for sharing data for marketing information. Enumerated values: 'Yes', 'No', 'Inherit'. Default = 'Inherit'.

AddressType : (Home | Business | Delivery | Mailing | Billing | CreditCard | Other)

Defines what the purpose of the address is, such as main office, branch office, mailing, delivery, shipping/receiving address, etc.

AddressInfo DTD fragment

```

<!ELEMENT AddressInfo (%Ent.Address;)>
<!--ATTLIST AddressInfo
      DefaultInd (Yes | No) #IMPLIED
      %Ent.Privacy;
      AddressType (Home | Business | Delivery | Mailing | Billing | CreditCard | Other) #IMPLIED
-->

```

%ENTITY Address

Defines a physical address of parties to the profile. May also be used as a postal address.

5.1.1.3 Company Telephone

The TelephoneInfo element is used to store the company telephone numbers identified in the profile, including office, mobile phone or other telephone number, defining the association with the company in the context of the use of the phone.

Element	Occur.	Content	Data type	Attributes	Description
TelephoneInfo	*	Element, % Telephone		DefaultInd, %Privacy, PhoneUseType	Telephone numbers of the individual in the profile

TelephoneInfo

Information about the telephone numbers of the individual or company in the profile.

Attributes:

DefaultInd : (Yes | No)

The value that the receiving system should assume if the user specifies no overriding value or action.

ShareSynchInd : (Yes | No | Inherit) = Inherit

Permission for sharing data for synchronization of profiles held by other travel service providers. Enumerated values: 'Yes', 'No', 'Inherit'. Default = 'Inherit'.

ShareMarketInd : (Yes | No | Inherit) = Inherit

Permission for sharing data for marketing information. Enumerated values: 'Yes', 'No', 'Inherit'. Default = 'Inherit'.

PhoneUseType : CDATA

Describes the type of telephone number, in the context of its general use, such as Home, Business, Emergency Contact, Travel Arranger, Day, Evening, etc.

TelephoneInfo_DTD fragment

```
<ELEMENT TelephoneInfo (%Ent.Telephone;)>
<!ATTLIST TelephoneInfo
  DefaultInd (Yes | No) #IMPLIED
  %Ent.Privacy;
  PhoneUseType CDATA #IMPLIED
>
```

%ENTITY Telephone

Telephone numbers for the parties in the profile.

5.1.1.4 Company Email

Identifies the company email address(es). Note that contact persons within a company may be assigned their own individual e-mail addresses, therefore this element under CompanyInfo may be used for generic e-mail addresses such as "info@companyX" or "support@companyX", etc.

Element	Occur.	Content	Data type	Attributes	Description
Email	*	Text	String	DefaultInd, ShareSynchInd, ShareMarketInd, EmailType	Electronic mail addresses, in IETF specified format

Email

Electronic mail addresses, in IETF specified format.

Attributes:**DefaultInd : (Yes | No)**

Indicates that the receiving system should assume the default value if the user specifies no overriding value or action.

ShareSynchInd : (Yes | No | Inherit) = Inherit

Permission for sharing data for synchronization of profiles held by other travel service providers. Enumerated values: 'Yes', 'No', 'Inherit'. Default = 'Inherit'.

ShareMarketInd : (Yes | No | Inherit) = Inherit

Permission for sharing data for marketing information. Enumerated values: 'Yes', 'No', 'Inherit'. Default = 'Inherit'.

EmailType : CDATA

Defines the purpose of the Email address, such as personal, business, listserve, etc.

Email DTD fragment

```
<!ELEMENT Email (#PCDATA)>
<!ATTLIST Email
  DefaultInd (Yes | No) #IMPLIED
  %Ent.Privacy;
  EmailType CDATA #IMPLIED
>
```

5.1.1.5 Company Web Sites

Identifies the company web site address(es). The URL element may be repeated to record more than one web site address, using IETF specified format.

Element	Occur.	Content	Data type	Attributes	Description
URL	*	Text	String	DefaultInd, %Privacy URL_Type	Web site addresses, in IETF specified format

URL

Web site address, in IETF specified format.

Attributes:**DefaultInd : (Yes | No)**

Indicates that the receiving system should assume the default value if the user specifies no overriding value or action.

ShareSynchInd : (Yes | No | Inherit) = Inherit

Permission for sharing data for synchronization of profiles held by other travel service providers. Enumerated values: 'Yes', 'No', 'Inherit'. Default = 'Inherit'.

ShareMarketInd : (Yes | No | Inherit) = Inherit

Permission for sharing data for marketing information. Enumerated values: 'Yes', 'No', 'Inherit'. Default = 'Inherit'.

URL_Type : CDATA

Defines the purpose of the URL address, such as personal, business, public, etc.

URL DTD fragment

```
<!ELEMENT URL (#PCDATA)>
<!ATTLIST URL
  DefaultInd (Yes | No) #IMPLIED
  %Ent.Privacy;
  URL_Type CDATA #IMPLIED>
```

5.1.1.6 Business Locale

Identifies the address(es) where the company is licensed to do business. Uses the address entity to specify any of the components of an address, such as city, county, state or country.

Element	Occur.	Content	Data type	Attributes	Description
BusinessLocale	*	Text	String	DefaultInd,	Location where a company is authorized to do business
%Address					Identifies the address of the business, using the components of the Address entity.

BusinessLocale

The jurisdiction in which a company is authorized to do business. Uses the Address entity to indicate the location of the business.

Attribute:

DefaultInd : (Yes | No)

Indicates whether the business location is the default choice among several. The default business locale could be used to indicate the primary place of business or incorporation

BusinessLocale DTD fragment

```
<!ELEMENT BusinessLocale (%Ent.Address;)>
<!ATTLIST BusinessLocale
    DefaultInd (Yes | No) #IMPLIED
>
```

%ENTITY Address

Defines a physical address of parties to the profile. May also be used as a postal address.

5.1.2 Company Payment Forms

The CompanyInfo section of the profile provides a place to store data on the payment methods used by a company to pay for travel-related services. Company payment information may be used dynamically for payment of invoices in a direct billing situation, or for guarantees of reservations via company credit card, etc. Therefore, the OTA profile specification holds information about payment data in the PaymentForm element so that these entries only need to appear once, and can be referenced from other places in the profile or in message sets.

Each of the items that identify a payment form are assigned a Reference Place Holder (RPH) attribute that identifies the individual type of payment within a collection of payment forms. When it is necessary to refer to the data stored in the Company section of the profile from other sections of the profile record, a reference is made to the RPH or index number of that item.

Element	Occur.	Content	Data type	Attributes	Description
PaymentForm	*	Element		DefaultInd, %Privacy CostCenterId, RPH	Ways of providing funds for travel by the individual
CreditCard	*	Element		%Privacy,	Payment accounts

				CardType, CreditCardCode, CardNumber, SeriesCode, EffectiveDate, ExpireDate	authorizing purchases represented by plastic cards often with magnetic strips with data for identification
BankAcct	*	Element		%Privacy, BankID, AcctType, BankAcctNumber	Customer bank accounts for payments either for paper checks or electronic funds transfer
DirectBill	*	Element		%Privacy, DirectBill	Company name and location for sending invoice for remittances for travel services
Voucher	*	Element		EffectiveDate, ExpireDate, SeriesCode	A form of payment using a voucher or coupon

PaymentForm

Attributes:

DefaultInd : (Yes | No)

Indicates that the receiving system should assume the default value if the user specifies no overriding value or action.

ShareSynchInd : (Yes | No | Inherit) = Inherit

Permission for sharing data for synchronization of profiles held by other travel service providers. Enumerated values: 'Yes', 'No', 'Inherit'. Default = 'Inherit'.

ShareMarketInd : (Yes | No | Inherit) = Inherit

Permission for sharing data for marketing information. Enumerated values: 'Yes', 'No', 'Inherit'. Default = 'Inherit'.

CostCenterId : CDATA

Code for allocating cost of travel to company accounts.

RPH : CDATA

A Reference Place Holder that serves as an index to the collection of payment forms stored in the customer's profile record. The RPH may be used to reference a method of payment from outside the profile.

PaymentForm DTD fragment

```
<!ELEMENT PaymentForm (CreditCard*, BankAcct*, DirectBill*, Voucher*)>
<!ATTLIST PaymentForm
    %Ent.Privacy;
    CostCenterId CDATA #IMPLIED
    RPH CDATA #IMPLIED >
```

5.1.2.1 CreditCard elements and attributes

Element	Occur.	Content	Data type	Attributes	Description
CreditCard	*	Element		%Privacy CardType, CreditCardCode CardNumber, SeriesCode, EffectiveDate ExpireDate	Payment accounts authorizing purchases represented by plastic cards often with magnetic strips with data for identification
CardHolderName	?	Text	String	IssuingBankId	Name string of individual as

					embossed on the card
CardIssuerName	?	Text	String		Company or organization that issued the card
%Address					Mailing or delivery address that defines location to where invoices are sent.

CreditCard

Payment accounts authorizing purchases represented by plastic cards with magnetic strips with data for identification.

Attributes:

ShareSynchInd : (Yes | No | Inherit) = Inherit

Permission for sharing data for synchronization of profiles held by other travel service providers. Enumerated values: 'Yes', 'No', 'Inherit'. Default = 'Inherit'.

ShareMarketInd : (Yes | No | Inherit) = Inherit

Permission for sharing data for marketing information. Enumerated values: 'Yes', 'No', 'Inherit'. Default = 'Inherit'.

CardType : (Credit | Debit | CentralBill)

Indicates the type of magnetic striped card. Values: (Credit | Debit | CentralBill)

CreditCardCode : CDATA

Code indicating the type of credit card. (e.g., MasterCard, American Express, Visa, etc.), or the type of central bill/debit card.

CardNumber : CDATA

Credit card number embossed on the card.

SeriesCode : CDATA

Verification digits printed on the card following the embossed number.

EffectiveDate : CDATA

Date the card becomes valid for use in ISO 8601 prescribed format.

ExpireDate : CDATA

Date the card expires and can no longer be used in ISO 8601 prescribed format.

CreditCard_DTD fragment

```
<!ELEMENT CreditCard (CardHolderName?, CardIssuerName?, %Ent.Address;?)>
<!ATTLIST CreditCard
    %Ent.Privacy;
    CardType (Credit | Debit | CentralBill) #IMPLIED
    CreditCardCode CDATA #IMPLIED
    CardNumber CDATA #REQUIRED
    SeriesCode CDATA #IMPLIED
    EffectiveDate CDATA #IMPLIED
    ExpireDate CDATA #REQUIRED>
```

CardHolderName

Name of individual cardholder as embossed on the credit/debit card.

CardHolderName_DTD fragment

```
<!ELEMENT CardHolderName (#PCDATA)>
```

CardIssuerName

Name of bank or organization issuing the card. e.g.: alumni association, bank, fraternal organization, etc.

IssuingBankId : CDATA

Code of bank issuing the card.

CardIssuerName_DTD fragment

```
<!ELEMENT CardIssuerName (#PCDATA)>
<!ATTLIST CardIssuerName
    IssuingBankId CDATA #IMPLIED
>
```

ENTITY %Address

Uses the Address entity to define the mailing or delivery location where invoices are sent.

5.1.2.2 Bank Account

Element	Occur.	Content	Data type	Attributes	Description
BankAcct	*	Element		%Privacy BankId, AcctType, BankAcctNumber,	Customer bank accounts for payments, either for paper checks or electronic funds transfers
BankAcctName	*	Text	String		Names of the individuals as they appear on the bank account.

BankAcct

Customer bank accounts for payments, either for paper checks or electronic funds transfers.

Attributes:

ShareSynchInd : (Yes | No | Inherit) = Inherit

Permission for sharing data for synchronization of profiles held by other travel service providers. Enumerated values: 'Yes', 'No', 'Inherit'. Default = 'Inherit'.

ShareMarketInd : (Yes | No | Inherit) = Inherit

Permission for sharing data for marketing information. Enumerated values: 'Yes', 'No', 'Inherit'. Default = 'Inherit'.

BankId : CDATA

Code assigned by authorities to financial institutions; sometimes called bank routing number.

AcctType : CDATA

Describes the bank account used for financing travel, such as: Checking, Savings, Investment, etc.

BankAcctNumber : CDATA

Identifier for the account assigned by the bank.

BankAcct_DTD fragment

```
<!ELEMENT BankAcct (BankAcctName* )>
<!ATTLIST BankAcct
    %Ent.Privacy;
    BankId CDATA #IMPLIED
    AcctType CDATA #IMPLIED
    BankAcctNumber CDATA #IMPLIED
>
```

BankAcctName

Names of the individuals as listed on the bank account.

BankAcctName_DTD fragment

```
<!ELEMENT BankAcctName (#PCDATA)>
```

5.1.2.3 Direct Bill

Element	Occur.	Content	Data type	Attributes	Description
DirectBill	*	Element		%Privacy. DirectBill_Id	Company name and location for sending invoice for remittances for travel services.
%CompanyName	R	Entity	String		Name of organization to where invoices are sent
%Address	R	Entity	String		Precise mailing or delivery location to where invoices are sent

DirectBill

Company name and location for sending invoice for remittances for travel services.

Attributes:

ShareSynchInd : (Yes | No | Inherit) = Inherit

Permission for sharing data for synchronization of profiles held by other travel service providers. Enumerated values: 'Yes', 'No', 'Inherit'. Default = 'Inherit'.

ShareMarketInd : (Yes | No | Inherit) = Inherit

Permission for sharing data for marketing information. Enumerated values: 'Yes', 'No', 'Inherit'. Default = 'Inherit'.

DirectBill_Id : CDATA

Identifier for the organization to be billed directly for travel services.

DirectBill_DTD fragment

```
<!ELEMENT DirectBill (%Ent.CompanyName;, %Ent.Address;)>
<!ATTLIST DirectBill
    %Ent.Privacy;
    DirectBill_Id CDATA #REQUIRED
>
```

ENTITY %CompanyName

Uses the CompanyName entity to indicate the name of organization to which invoices are sent.

ENTITY %Address

Uses the Address entity to define the mailing or delivery location where invoices are sent.

5.1.2.4 Voucher

The Voucher element, a child element of PaymentForm, is used to identify a series of vouchers or coupons used for payment of travel services. The Voucher element is not likely to be used for a customer's personal profile record.

Element	Occur.	Content	Data type	Attributes	Description
Voucher	*	Element		EffectiveDate, ExpireDate, SeriesCode	Information about voucher program used to pay for travel services.

Voucher

A form of payment using a voucher or coupon.

Attributes:

EffectiveDate : CDATA

The date when a voucher becomes valid for use, if applicable.

ExpireDate : CDATA

The date when a voucher or series of coupons expires and is no longer valid.

SeriesCode : CDATA

Identification of a series of coupons or vouchers identified by serial number(s).

Voucher DTD fragment

```
<!ELEMENT Voucher (#PCDATA)>
<!ATTLIST Voucher
  EffectiveDate CDATA #IMPLIED
  ExpireDate CDATA #IMPLIED
  SeriesCode CDATA #IMPLIED
```

5.1.2.5 Contact Person

The Contact Person element identifies a person whom travel services should contact on behalf of the company. This may be an employee, or person that is amember of another company that provides travel arrangements for the company.

Element	Occur.	Content	Data type	Attributes	Description
ContactPerson	*	Element		DefaultInd %Privacy ContactType, Relation EmergencyFlag, RPH	Persons whom travel services should use as a company contact
%PersonName	?	Element			Names of individuals to contact
TelephoneInfo	*	Element % Telephone		PhoneTech PhoneUse,	Telephone numbers, including, fax or pager, of persons to contact
AddressInfo	*	Element % Address		ContactType,	Addresses of persons to contact
Email	*	Text	String	DefaultInd, EmailType	Email addresses of contact person or entity
URL	*	Text	String	DefaultInd, URL_Type	Web site addresses of contact person or entity
%CompanyName	*	Element:		DefaultInd	Company associated with the contact.
EmployeeInfo	*	Element	String	EmployeeId, EmployeeLevel, EmployeeTitle, EmployeeStatusT ype	Employment information about the contact person.

ContactPerson

Person or entity with whom travel services should communicate when it is necessary to interact with a contact on behalf of the company.

Attributes:

DefaultInd : (Yes | No)

Indicates the default among a choice of more than one. The receiving system should assume this default value if the user specifies no overriding value or action.

ShareSynchInd : (Yes | No | Inherit) = Inherit

Permission for sharing data for synchronization of profiles held by other travel service providers. Enumerated values: 'Yes', 'No', 'Inherit'. Default = 'Inherit'.

ShareMarketInd : (Yes | No | Inherit) = Inherit

Permission for sharing data for marketing information. Enumerated values: 'Yes', 'No', 'Inherit'. Default = 'Inherit'.

ContactType : CDATA

Type of contact in the context of use for the travel experience; such as permanent, temporary, affiliation, travel arranger, etc.

Relation : CDATA

Indicates type of the relationship with the person in the profile, such as Spouse, Children, Family, Business, InterestGroup, Medical, Security, Other.

EmergencyFlag : (Yes | No)

Indicates if this contact should be used in the case of an emergency.

RPH : CDATA

A Reference Place Holder that serves as an index to the collection of contacts stored in the customer's profile record. The RPH may be used to reference a contact from outside the profile, e.g.: in a reservation.

ContactPerson DTD fragment

```
<!ELEMENT ContactPerson (%Ent.PersonName;, TelephoneInfo*, AddressInfo*, Email*, URL*,
%Ent.CompanyName;*, EmployeeInfo*)>
<!ATTLIST ContactPerson
    DefaultInd (Yes | No) #IMPLIED
    %Ent.Privacy;
    ContactType CDATA #IMPLIED
    Relation CDATA #IMPLIED
    EmergencyFlag CDATA #IMPLIED
    RPH CDATA #IMPLIED
>
```

The following elements are children of the ContactPerson element, described elsewhere in this specification, as well as under the ContactPerson element in the Customer section. EmployeeInfo is fully described, as this element may be used frequently for company contacts.

TelephoneInfo

Telephone numbers, including, fax or pager, of contact person.

AddressInfo

Information about the physical or mailing address of the contact person.

Email

Electronic mail addresses of the contact person.

URL

Web site address of the contact person, in IETF specified format.

ENTITY %CompanyName

Identifies a company or organization associated with the contact.

EmployeeInfo

Employment information about the person employed by the company. The PCDATA field of EmployeeInfo can be used to record the exact name of the employee if it differs from that of the contact person identified for the company.

Attributes:**EmployeeId : CDATA**

Identifier assigned to the person by the employer.

EmployeeLevel : CDATA

Level in employer organization (e.g. seniority) that conveys privileges.

EmployeeTitle : CDATA

Title of employee in employer company that conveys rank or privileges.

EmployeeStatusType : (Active | Retired | Leave | Terminated)

Status of employment. Values: Active, Retired, Leave, Terminated.

EmployeeInfo DTD fragment

```
<!ELEMENT EmployeeInfo (#PCDATA)>
<!--ATTLIST EmployeeInfo
  EmployeeId CDATA #IMPLIED
  EmployeeLevel CDATA #IMPLIED
  EmployeeTitle CDATA #IMPLIED
  EmployeeStatusType (Active | Retired | Leave | Terminated) #IMPLIED
-->
```

5.1.3 Company Travel Arranger

Element	Occur.	Content	Data type	Attributes	Description
TravelArranger	*	Text	String	DefaultInd, %Privacy, TravelArrangerType, ContactRPH	Identifies a company or individual responsible for making travel plans or transactions for the company and its employees. The ContactRPH is a reference to the contact information stored in the Customer and CompanyInfo section

TravelArranger

Identifies a company or individual responsible for making travel plans or transactions for the company and its employees. The ContactRPH is a reference to the contact information stored in the Customer and CompanyInfo section.

Public Attributes:**DefaultInd : (Yes | No)**

Indicates if this is a default choice in the absence of the user specifying no overriding value or action.

ShareSynchInd : (Yes | No | Inherit) = Inherit

Permission for sharing data for synchronization of profiles held by other travel service providers. Enumerated values: 'Yes', 'No', 'Inherit'. Default = 'Inherit'.

ShareMarketInd : (Yes | No | Inherit) = Inherit

Permission for sharing data for marketing information. Enumerated values: 'Yes', 'No', 'Inherit'. Default = 'Inherit'.

TravelArrangerType : CDATA

Type of service making travel plans or transactions, e.g.: travel agency, etc.

ContactRPH : CDATA

Reference number to the contact associated with the customer.

TravelArranger DTD fragment

```
<!ELEMENT TravelArranger (%Ent.CompanyName;)>
<!--ATTLIST TravelArranger
    DefaultInd (Yes | No) #IMPLIED
    %Ent.Privacy;
    TravelArrangerType CDATA #IMPLIED
    ContactRPH CDATA #IMPLIED-->
```

5.1.4 Loyalty Programs

The LoyaltyProgram element identifies membership programs offered by the company that rewards frequent use by its members. The programs offered by the company are identified by name and by program code, and identify levels of service that they offer to subscribers. The SingleVendorInd attribute indicates whether the program is a part of an alliance of other similar reward programs. The LoyaltyProgram element uses an RPH attribute to index a specific loyalty program among a collection.

Element	Occur.	Content	Data type	Attributes	Description
LoyaltyProgram	*		String	ProgramCode SingleVendorInd LoyaltyLevel, RPH	Name of the loyalty program.

LoyaltyProgram

Loyalty program offered by the company.

Attributes:**ProgramCode : CDATA**

The code of the loyalty or membership program.

SingleVendorInd : (SingleVndr | Alliance)

Indicates if program is affiliated with a group of related offers for accumulation of credits or points. Values: SingleVndr or Alliance.

LoyaltyLevel : CDATA

Indicates the status or level offered by the program which carries certain privileges or represents a points or rewards available to members.

RPH : CDATA

Reference number to be used to identify this loyalty program.

LoyaltyProgram DTD fragment

```
<!ELEMENT LoyaltyProgram (#PCDATA)>
<!--ATTLIST LoyaltyProgram
    ProgramCode CDATA #IMPLIED
    SingleVendorInd (SingleVndr | Alliance) #IMPLIED
    LoyaltyLevel CDATA #IMPLIED
    RPH CDATA #IMPLIED
-->
```


6.0 Agreements

Profile information about a company often involves data about the business relationship, identifying information about trading partner agreements, methods or arrangements for earning commissions, or other affiliations and relationships to which that company belongs. That information sometimes implies the roles and privileges inherent to the trading partner relationship.

A company profile is identified by the UniqueId element in the same manner as a personal, customer profile. The four attributes, consisting of the Id, Type, URL and optionally, an Instance of the profile record, apply similarly to the company profile. The Id is a unique number assigned to identify the business, often using the identification number of a certifying body, such as ARC, IATA, for a travel arranger, or a DUN's number for a corporation, etc., depending upon the nature of the business entity. The list of types is enumerated in the Profile Type attribute. With the exception of Customer, all profile types refer to a business entity.

Because such information may be confidential between companies, the privacy attributes determine whether such information about the company can be shared for synchronization as well as for marketing purposes with other trading partners. The Privacy attributes are set by default to "No" at the highest level, with all other levels inheriting from the root level. Permission to share data must override the default at the specific element level.

The Agreements section of the OTA Profile identifies the certifications or accreditations held by a company, as well as any relationships with other groups, such as alliances or consortiums.

The CommissionInfo element allows for the identification of commission plans by code and associates rates to those plans. However, sensitive information about arrangements for monetary transactions is protected by the privacy built into the Agreements section of the OTA Profile specification, and at level of the CommissionInfo element.

6.1 Agreements data

Each of the elements in the Agreements section (Figure 26) is OPTIONAL, however, if any of these elements appear in an OTA message, they MUST appear in the designated order.

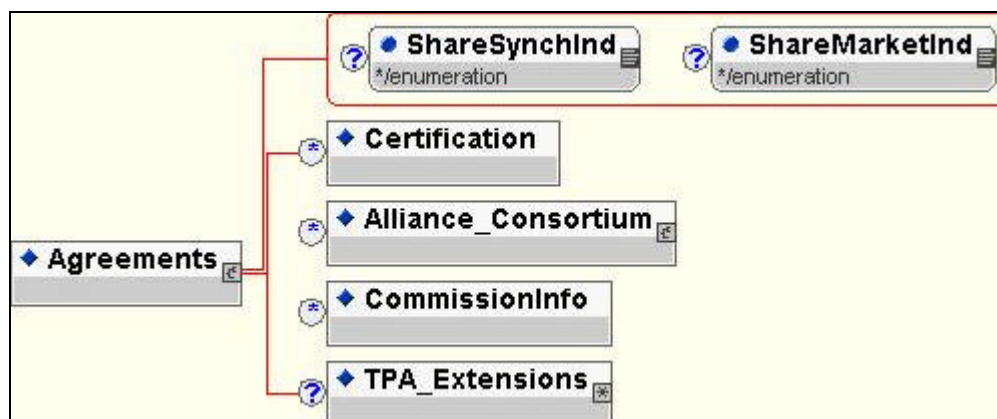


Figure 26. Agreements

6.2 Agreements Section

The Agreements Section contains information about agreements held by the company in an alliance or consortium and also information about commission information.

Element	Occur.	Content	Data type	Attributes	Description
Agreements	?	Element			Section of a business profile that contains information about trading partner agreements
Alliance_Consortium	*	Element			Information about membership in an alliance or consortium
CommisionInfo	*	Element			Information about commission agreements
TPA_Extensions	?	Element			Due to the proprietary nature of agreements between business entities, this element allows for companies to add selected information as needed.

Agreements

Section of a business profile that contains information about trading partner agreements.

Public Attributes:

%Ent.Privacy;

The privacy entity provides a means to control the sharing of information about trading partner agreements for system synchronization or marketing purposes.

Agreements DTD fragment

```
<!ELEMENT Agreements (Certification*, Alliance_Consortium*, CommissionInfo*, TPA_Extensions?)*>
<!ATTLIST Agreements
  %Ent.Privacy;>
```

6.3 Certifications

The Certification element provides information about certifications and accreditations held by the company. This element may be repeated for each certification recorded.

Element	Occur.	Content	Data type	Attributes	Description
Certification	*	Element		CertificationId, CertificationType, EffectiveDate, ExpireDate	Information about certifications held by the company

Certification

Information about a certification or accreditation held by the company.

Public Attributes:**CertificationId : CDATA**

The identification number assigned to a business entity. This number would presumably be the Id in the UniqueId.

CertificationType : (SingleVndr | Alliance)

An indication of whether a certification is part of a consortium or alliance partnership. Valid values: (Single Vendor | Alliance).

EffectiveDate : CDATA

The date when certification originated.

ExpireDate : CDATA

The date when the certification or accreditation expires and is no longer valid.

Certification DTD fragment

```
<!ELEMENT Certification (#PCDATA)>
<!ATTLIST Certification
  CertificationId CDATA #IMPLIED
  CertificationType (SingleVndr | Alliance) #IMPLIED
  EffectiveDate CDATA #IMPLIED
  ExpireDate CDATA #IMPLIED>
```

6.4 Alliance and Consortium Memberships

The Alliance_Consortium element contains information about membership held by the company in an alliance or consortium.

Element	Occur.	Content	Data type	Attributes	Description
Alliance_Consortium	*	Element		AllianceId EffectiveDate, ExpireDate,	Information about membership in an alliance or consortium
AllianceMember	*	Element			Identification of a company that participates in an alliance or consortium to which the primary business entity identified in this profile belongs. Uses the CompanyName entity

Alliance_Consortium

Provides information about alliance partnerships and consortiums of members grouped together to obtain trading partners agreements for travel services and privileges.

Attributes:**AllianceId : CDATA**

Identification of the alliance or partnership in which the business entity is a member.

EffectiveDate : CDATA

The date when membership in an alliance or consortium began.

ExpireDate : CDATA

The date when membership in the alliance or consortium ends.

Alliance_Consortium DTD fragment

```
<!ELEMENT Alliance_Consortium (AllianceMember*)>
<!ATTLIST Alliance_Consortium
  AllianceId CDATA #IMPLIED
  EffectiveDate CDATA #IMPLIED
  ExpireDate CDATA #IMPLIED>
```

AllianceMember

Identification of a company that participates in an alliance or consortium to which the primary business entity identified in this profile belongs. Uses the CompanyName entity.

Attribute:**MemberCode : CDATA**

Identifies the alliance or consortium member by code.

AllianceMember DTD fragment

```
<!ELEMENT AllianceMember (%Ent.CompanyName;)>
<!ATTLIST AllianceMember
  MemberCode CDATA #IMPLIED>
```

ENTITY %CompanyName

Identifies a company or organization associated with the contact.

6.5 Commission Information

The CommissionInfo element identifies information about arrangements between companies for commissions to be paid.

Element	Occur.	Content	Data type	Attributes	Description
CommissionInfo	*		String	%Privacy, CommissionPlanCode CommissionRate	Information about commission agreements

CommissionInfo

Contains information about agreements for commission arrangements with the business entity.

Attributes:**%Ent.Privacy; :**

The privacy entity provides a means to control the sharing of commission information for system synchronization or marketing purposes.

CommissionPlanCode : CDATA

Identifies a commission plan agreement between trading partners by plan code.

CommissionRate : CDATA

Identifies a rate for paying commissions; can be a decimal value based on percentage paid for the commission plan, or a flat rate.

CommissionInfo DTD fragment

```
<!ELEMENT CommissionInfo (#PCDATA)>
<!ATTLIST CommissionInfo
  %Ent.Privacy;
  CommissionPlanCode CDATA #IMPLIED
  CommissionRate CDATA #IMPLIED>
```

6.6 TPA_Extensions

Due to the proprietary nature of agreements between business relationships, the TPA_Extensions element is included in this section of the profile, and allows for companies to add selected information as needed.

Element	Occur.	Content	Data	Attributes	Description
---------	--------	---------	------	------------	-------------

			type		
TPA_Extensions	?				Due to the proprietary nature of agreements between business entities, this element allows for companies to add selected information as needed.

TPA_Extensions

Due to the proprietary nature of agreements between business entities, this element allows for companies to add selected information as needed.

TPA_Extensions DTD fragment

<!ELEMENT TPA_Extensions ANY>

7.0 TPA_Extensions

Due to the proprietary nature of agreements between business relationships, the TPA_Extensions element (Figure 27) is included in this section of the profile, and allows for companies to add selected information as needed.



Figure 27. TPA_Extensions

Element	Occur.	Content	Data type	Attributes	Description
TPA_Extensions	?				Due to the proprietary nature of agreements between business entities, this element allows for companies to add selected information as needed.

TPA_Extensions

Due to the proprietary nature of agreements between business entities, this element allows for companies to add selected information as needed.

TPA_Extensions DTD fragment

```
<ELEMENT TPA_Extensions ANY>
```