OpenTravel Alliance Message Specifications – Version 1

Volume 1, Specifications Document

Abstract

This document presents the specifications for customer profile 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. The specifications include mandatory requirements and recommendations for these messages, covering security and privacy, the base technical architecture, travel business content, and administrative exchanges.

OpenTravel Alliance, Inc.
333 John Carlyle Street, Ste. 600
Alexandria, Va. 22314 USA
+1 703-548-7005
Fax +1 703-836-4985
opentravel@disa.org
http://www.opentravel.org/

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

10 July 2000

Non-Exclusive License Agreement

OpenTravel Alliance Message Specifications – Version 1

Executive summary

The OpenTravel Alliance Message Specification, version 1 (OTA version 1) provides a common customer profile that travelers can fill out once and exchange among various travel services over the Internet. The specification provides a uniform vocabulary that captures and exchanges data on a traveler's identity, affiliations including employer, loyalty programs, forms of payment, travel documents, and detailed travel preferences.

A key feature of the profile allows customers to define collections of travel preferences in terms of their own 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. OTA version 1 also allows customers to identify related travelers, such as family members, companions, or fellow business colleagues, and link to their profiles.

The OTA specification uses the Extensible Markup Language (XML) that allows for the exchange of structured data – the kind of data stored in databases – as well as processing instructions over the Web. With XML, OTA defined a common vocabulary in terms of data items called elements, attributes, or reusable entities reflected in unique tags that identify the data in messages. The hierarchical structure of these data items in a set of electronic rules is called a document type definition (DTD) that allows parties exchanging customer profiles to validate the messages. A separate DTD specifies basic error conditions and administrative messages independent of OTA's specification versions.

OTA version 1 specifies that parties send profile messages as pairs of requests with corresponding responses. The messages contain four basic functions: (1) creating a profile, (2) reading a profile, (3) updating a profile, and (4) deleting a profile. The update process is the most complex and can address individual parts of the profile record. The other functions address the profile record as a whole. The specification provides tag-naming conventions, which include the version and hierarchy of the elements in each tag.

OTA version 1 recommends security features providing authentication of parties, confidentiality, and integrity of messages, and provides a control section in each message, separate from the business content, for these security features. The specification also has strict privacy requirements that lets the customer indicate which data to share with other parties, even for routine functions like keeping all copies of the profile on remote sites identical with the original.

Acknowledgments

To develop this specification over the course of a few short months meant a lot of work by a lot of dedicated people. You will find the names and organizations of people who took part in meetings and conference calls, or provided comments on the specification over the Fall and Winter of 1999 listed on the following page

A few people need a special mention for their substantial contributions to the development of this document: Scott Hinkelman and Sherry Shavor of IBM Corp., George Smith of ConXtra, Mary Manzer and Jay Whitehair of United Airlines, Robert Cole of Budget Group, Tim Kieschnick and Stuart Stevens of Pegasus Systems, Steve Quakenbush of Galileo, Paula Heilig of Worldspan, Bob Stockwell of Equinus (representing TTI), Ray Kopsa and Ray Niemeir of REZsolutions, Ellen Runyon of Continental Airlines, Steve Fohl of American Airlines, Nick Lanyon of Lanyon Inc., Jim deBettencourt of McCord Travel, and Don Scheer of Airline Automation Inc. DeBettencourt and Scheer also developed the demonstration of the OTA specification.

Thanks go also to the DISA staff and consultant who contributed mightily to this entire project: Tim Cochran, Victoria Day, Becky Podd, Bruce Peat, and John McGinnis.

OTA created the document type definitions and the element hierarchy charts seen throughout the document using XML Authority by Extensibility Inc. (http://www.extensibility.com).

Alan Kotok, Editor

Participants	Company Name
Don Scheer	Airline Automation Inc.
Scott Kimbriel	Airline Automation Inc.
Kim Lambert	Air Transport Association
Paula Williams	Alamo Rent A Car
Ed Sabala	Alamo/National
Molly O'Grady	Alaska Airlines
David Hardin	Alaska Airlines
Steve Fohl	American Airlines
Paul Stumbo	American Airlines
Joelle Cuvelier	ARC
Keith Venzke	ARC
Stephanie Kenyon	ASTA
Kim Lambert	ATA
Debbie Schneiderman	Avis Rent A Car System
Tony Fletcher	British Telecomm
Cheryl Karnes	Budget Rent A Car
Robert Cole	Budget Rent A Car
Colby Ellis	Cahners Travel Group
Glenda Brady	Cendant
Robert Lindhorst	Cendant
Scott Gibson	Cendant
Ellen Runyon	Continental Airlines, Inc.
George Smith	ConXtra
Robert Beiersdorfer	Delta Air Lines
Burnette, Bill	Delta Air Lines
Kevin Dunn	Delta Air Lines
Magnolia Daniels	Dollar Rent A Car
Richard Zemina	Dollar Rent A Car
Jerry Brown	EASYres Tech.
Rick Holofcener	EASYres Tech.
Brian Weingart	Enterprise Rent A Car
Russ Ditmar	Enterprise Rent A Car
Gina Miller	Enterprise Rent A Car
Bob Stockwell	Equinus, representing TTI
Bob Parsons	Equinus
Michael Townsie	Galileo
Steve Quackenbush	Galileo
Sean Handel	GetThere.com
James White	Hertz Corporation
John Prusnick	Hyatt International
Scott Hinkelman	IBM
Sherry Shavor	IBM
Nick Lanyon	Lanyon
Itzhak Baram	Leonardo
Jim deBettencourt	McCord Travel
Jim Broenniman	Midwest Express Airlines
Norman Sherlock	NBTA

Participants	Company Name
Paul Long	Northwest Airlines
Sharon Kadlec	Northwest Airlines
Rien Heald	OAG
Steve Raphaelson	OAG
Sue Wolsko	OAG
Josh de Witt	Pegasus Systems
Tim Kieschnick	Pegasus Systems
Stuart Stevens	Pegasus Systems
John Cross	Perot Systems
Pete Mathews	Preview Travel
Les Baker	Prism Group
Tim Clark	Proxicom
Ray Niemeir	REZSolutions
Ray Kopsa	REZsolutions
John MacDonald	Sabre
Patricia Larson	Sabre
Ken Hyde	Swissair/Atraxis
Doug Ritsema	Thrifty Car Rental
Regina Seago	Thrifty Car Rental
Bob Monroe	TWA
Dennis Shmigelsky	TWA
Tom Gilbert	TWA
Bob Tate	TWA
Kathie Lia	TWA
Mary Manzer	United Airlines
Sue Dominick	United Airlines
Jay Whitehair	United Airlines
Brendan Sherlock	US Airways
Gary Kresco	WizCom International
Glenn Fernandez	WizCom International
Paula Heilig	Worldspan

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

Volume 1, Table of Contents

1 Introduction

1.1 About the OpenTravel Alliance

The OpenTravel Alliance (OTA) is a consortium of suppliers of travel services – airline, hotel, car rental, passenger rail, travel arrangers, leisure – and companies that provide distribution and technology support to the industry. This specification represents the first product of OTA, a common customer profile for the industry, on which OTA will build additional services and functions.

Companies in the travel industry have long recognized 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.

OTA version 1 specifications and future OTA documents like it 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 jumping-off point from which companies can improve their current level of service for their customers and develop innovative ways for customers to communicate with them. Also, in keeping with the tradition of the Internet, even in its brief history, these specifications are indeed open documents freely available to all industry participants.

A word about words: readers will note that we use the term *specification* rather than *standard*. We reserve 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. We refer to several ISO standards in the text and list them in the references; see the Appendix. We call this document a specification, which often takes broad standards such as the Extensible Markup Language (XML) and applies them in particular industries or to particular problems. The OTA version 1 specification follows that approach. It focuses on travel industry requirements and while developed in an open manner does not meet the stringent requirements of standards.

1.2 Business requirements

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.

1.3 Reasons for selection of customer profile

OTA chose the customer profile as the first content work area not only because the profile is customer-centric, but also because it has relevance across the industry – to airlines, car rental companies, hotels, travel arrangers, travel management companies, Global Distribution Systems (GDSs), insurance companies, credit card companies, and others. It thus called for development of a process for soliciting requirements and reaching agreement on their implementation across OTA's membership of nearly 100 organizations. It also reduced requirements for the work being done in parallel on the associated technical infrastructure, particularly including security and privacy considerations. Finally, it enabled a simple demonstration of the coherence and interoperability of version 1 through implementation of a live (albeit simple) demonstration. See the OpenTravel Alliance Web site (http://www.opentravel.com/) for the link to this demonstration.

1.4 Objectives of a common travel customer profile

The customer profile collects important information used by the customer for travel arrangements, and provides the first benefits of computerized travel services developed around the customers' needs. Customers will see an immediate benefit from having to enter their basic customer data and preferences just once. OTA expects that availability of the data will help travel suppliers provide more customized services as well as new forms of distribution and even totally new services using the power of Internet-based technologies.

Customer travel profiles are nothing new. Online systems maintained by travel service providers, global distribution systems (GDSs), and travel arrangers have collected customer data in these profiles for several years. OTA version 1 however, specifies a customer profile for use throughout the entire travel industry.

OTA's common vocabulary for the travel industry extends the benefits of the profile even further. It enables the customer or companies acting on behalf of the customer to exchange information needed for delivery of travel services, but only when the customer gives explicit permission to share the data and thus respect the customer's privacy. The standard vocabulary allows for travel service providers to capture the information in their systems directly, making the process faster and with much less chance for errors. The common vocabulary also allows customers and travel arrangers to more easily find services that specialize in meeting special needs of the customers, such as recreational interests or requirements of physically challenged travelers.

The standard travel profile collects three types of data:

Customer information – important 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 business and personal travel services.

Preferences – collections of general and specific conditions to meet the needs of the customer for travel based on identifiable purposes, as defined by the customer, such as business trips, family vacations, golf outings, or the annual church retreat. Customers can also define preferences for specific kinds of travel services, such as for air travel, hotel stays, and car rentals.

Affiliations – organizations with which the customer has a relationship and that convey travel benefits or privileges, including one's employer, interest groups, membership organizations (e.g. AARP or AAA), financial institutions, travel arrangers, insurance companies, and supplier-provided travel clubs. Please note that OTA version 1 addresses only the customer's relationship with these organizations. It does not contain data on the organizations' travel policies or requirements.

1.5 Design goals and decisions

OTA designed this specification using the following guidelines:

Openness. The OTA specification is 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. The OTA version 1 specification supports 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. The OTA specification provides travel service providers and their representatives with the flexibility they need to rapidly develop, test, and deploy new services. The specification outlines the minimum necessary functionality to provide reliable interactions between customers and the systems owned and maintained by the companies serving them.

Extensibility. OTA plans to add more services and functionality to this specification in a way that minimizes incompatibility for those implementing this or other early versions. However, upcoming versions of the specification will likely require significant changes – migration to the emerging XML-Schema recommendation for example – that will not be compatible to OTA version 1. The OTA work groups that developed this specification have designed it with these transitions in mind, in order to ease the burden on those implementing version 1.

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

Platform independence. OTA developed this specification to work with 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; 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.

1.6 Relationships with other standards and industry systems

Extensible Markup Language (XML). OTA version 1 is an application of XML and represents data used for travel transactions, as required by XML Recommendation 1.0 of the World Wide Web Consortium. OTA version 1 also contains information and guidelines for the implementation of XML messages.

World Wide Web. The World Wide Web (WWW or the Web) is a collection of interconnected documents on the Internet that adhere to the commonly accepted conventions for the identification and display of those documents. Most users of OTA version 1 will likely have Web sites and use Web-related protocols to exchange XML messages as specified in this document.

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 version 1 enables data exchanges among individuals and organizations, and creates a common vocabulary of data elements and tags, it does 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-customer GDS services have recently become available over the Web. GDSs 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 version 1 offers specifications for the exchange of messages with customer travel data for implementation by any travel supplier or data service that adheres to these specifications based on open standards.

Hotel industry specifications. In the hotel industry, two specifications define common data elements for transactions among trading partners: Hotel Electronic Distribution Network Association (HEDNA) and Hospitality Industry Technology Integration Standards (HITIS). HEDNA has made available its descriptive content specifications for use by OTA. HITIS also incorporates elements of HEDNA's work in its standards. OTA and HITIS have agreed co-operate and merge their respective specifications. As a result the contents of this document may change as part of that process.

1.7 Status of this document

This document is the official OTA version 1 specification. 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 OTAcomments@disa.org . OTA also welcomes comments by fax to +1 703-836-4985 or by mail to:

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

2 Overview of OTA version 1

This first version of the Open Travel Alliance specification introduces the overall message structure and basic technical architecture principles such as versioning, error handling, and security. Although versioning, error handling, and security are not XML issues limited to the travel industry, this specification addresses them because of the business need for these functions and the lack of generic XML guidelines from standards organizations, at least at this time. From a travel industry perspective, this version also introduces the detailed contents of a customer profile.

2.1 Message flow and operations

OTA version 1 messages exchange customer-profile data between customers and travel services or among travel services themselves. Before individuals or organizations can begin exchanging data, however, each trading partner needs to learn the counterpart's support for various transport protocols, security, and required fields in the profile. Knowing these features in advance can save hours or days of trial and error. As of the publication of OTA version 1 specification (early 2000) the means of discovering these characteristics of trading partners have not fully emerged, but OTA plans to include them in future versions once they become available.

Once trading partners understand each other's requirements, they can begin exchanging messages. OTA version 1 messages use four main actions or verbs: Create, ReadAll, Update, and Delete. Create, ReadAll, and Delete verbs involve entire profile records, while Update addresses parts of a profile.

To open a profile, the requestor sends a message with the element `<v1.CustProfileCreateRQ>` that 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 `<v1.CustProfileCreateRS>` element and an identifier for the profile. The requestor can then add data using the Update process; see below.

To read the profile, the requestor sends a message with the `<v1.CustProfileReadAllRQ>` element and the identifier for the record. The trading partner returns a message with the `<v1.CustProfileReadAllRS>` element and the data in the profile at the time.

Updating the profile, because it allows for addressing parts of the record, has a more complex process. The requestor begins with a message containing the `<v1.CustProfileReadAllRQ>` tag and identifier that triggers a return message with `<v1.CustProfileReadAllRS>` tag and the current contents of the profile. The requestor then sends a `<v1.CustProfileUpdateRQ>` tagged message, with one or more sub-action verbs: Verify, Add, Remove, Insert, or Set (change). These sub-actions contain the content affected by the update. The response message has the `<v1.CustProfileUpdateRS>` tag and contents of the updated record.

Deleting a record involves sending a message with the `<v1.CustProfileReadAllRQ>` element and the profile's identifier, that returns a message with the `<v1.CustProfileReadAllRS>` element and the current contents of the record. The requestor then sends a message with the `<v1.CustProfileDeleteRQ>` tag and identifier, and an optional Verify sub-action, that returns a message with the `<v1.CustProfileDeleteRS>` element and the profile's identifier indicating deletion of the record.

2.2 Technical architecture

The technical architecture chapter presents the role and function of the various components of the messages defined in this specification. It defines the actions, called infrastructure verbs for the creation, reading, updating, and deletion of profile records, and which underpin the exchanges between trading partners. The chapter also discusses the role of basic technologies such as transport protocols and XML, and points out important security features of the specification. The specification supports one operation per message, but future versions should support multiple or batch operations per message.

2.3 Message structure

OTA version 1 transactions use a request and response model that provides a matched pair of messages to help account for and manage the data flow. The structure also spells out requirements for Control and Content elements in non-error messages to separate security details from the business content. The specification makes a distinction between error responses in versioned messages – those that arise from interactions with the host server – from error messages that do not carry version numbers. Chapter 6 of the specification covers non-versioned messages.

2.4 Customer profile content and its scope

Customer profile content includes basic information about the customer for identification as well as financial transactions and contacts. No supplier pricing information is collected. The profile also defines collections of preferences that allow the customer to create and identify preferences for specific types of travel. Profiles contain information about organizational affiliations such as employer and memberships, as well as key travel support services such as travel agencies and insurance. As noted in section 1.4 above, data on the travel policies or requirements of an individual's organization (e.g. employer) has not been fully addressed in OTA version 1.

2.5 Error and non-versioned message operations

Some of the exchanges between trading partners will contain messages for administrative purposes rather than customer profiles. 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.

2.6 Security and privacy

Because of the sensitivity of the data contained in customer profiles, OTA version 1 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. The document focuses on privacy protections that require the customer to give explicit permission in order to share the data, even for routine synchronization of updates.

2.7 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. See Appendix 5.

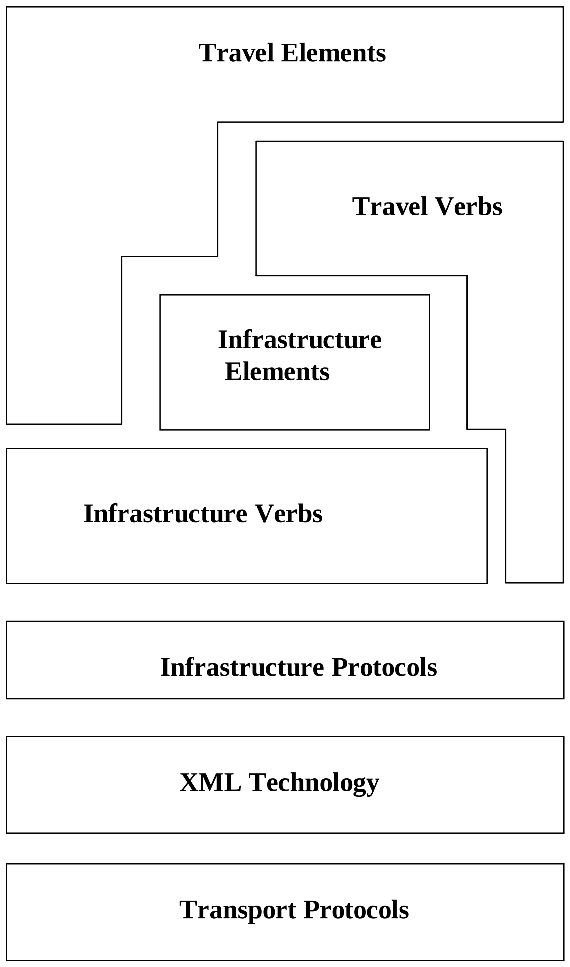
This text makes every attempt to accurately reflect the document type definitions (DTDs) listed in Appendix 1. In the case of a conflict between the document text and the DTD, the DTD takes precedence.

3 Technical Architecture

Because of the incomplete development of cross-industry XML standards, as well as Web-based commerce overall, OTA found it had to establish some of its own conventions for message definition and exchange. This section outlines the assumptions and decisions for the OTA version 1 message structure.

3.1 Reference model, logical view

Figure 1. OTA reference model
Source: Scott Hinkelman, IBM



The reference model for the overall OTA can be viewed as logical layers. Beginning at the bottom, the lowest layer is the transport protocols that govern the flow of messages. The layer above transport is the XML technology, from which OTA defined the elements and attributes that make up the messages exchanged among trading partners.

The layer above the XML technology layer of the OTA reference model defines non-travel-specific elements such as transaction, security, and versioning of the messages. Those message semantics define the infrastructure protocols layer. All OTA messages use the infrastructure protocols layer.

The layer for infrastructure verbs defines non-travel-specific verbs and actions such as typical Create-ReadAll-Update-Delete (CRUD) semantics, as well as sub-actions for more complex operations. These verbs do not represent travel-specific actions. Moreover, the four main infrastructure verbs do not operate alone, but in conjunction with the request and response model described in Chapter 4, Message Structure.

The travel verbs layer defines verbs and actions that are travel-specific (e.g., availability query). It defines the travel-specific business operations. OTA Version 1 that deals with customer profiles does not contain any travel-specific verbs, but one can expect future versions to have them. This layer overlaps with the infrastructure verbs layer in that the travel-specific verbs may or may not use the infrastructure-level verbs. All verbs are based on the infrastructure protocols layer.

The infrastructure element level contains non-travel elements and attributes. Infrastructure elements use infrastructure verbs.

The upper layer, travel elements, contains the OTA elements and attributes that are travel related. Chapter 5 on customer profile content describes this layer in detail. Travel elements utilize both travel verbs and infrastructure verbs.

The remainder of the chapter will discuss the layers in the model from bottom to top.

3.2 Transport protocols[1]

OTA version 1 is defined in terms of the infrastructure protocol layer and above, in accordance to the logical view of the architecture as described above. These levels represent some “system level” protocols, such as authentication flows, plus travel business-level messages.

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. As such, OTA messages MUST be able to flow on HTTP as a base. The OTA Architecture however, does not preclude OTA standard messages from flowing over other transport protocols, such as File Transfer Protocol (FTP) or Internet Inter-ORB (Object Reference Broker) Protocol.

Specifics of transport protocols are negotiated outside of the OTA and beyond the realm of this version of the OTA version 1 base 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 chose 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

[1] We thank Jim DeBettencourt of McCord Travel Management for his help in preparing this section of the chapter.

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 POST method sends data to the server as a name-value pair, and requires URL encoding of any values to eliminate characters not understood by an HTTP server. HTTP POST also 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.

3.3 XML technology

3.3.1 Schema direction

Until the World Wide Web Consortium (W3C) gives final approval to the XML-Schema recommendation, all versions published by the OTA will be built and based on document type definitions (DTDs). After W3C recommends XML-Schema, OTA will decide whether subsequent versions will be built and based on XML-Schema.

OTA version 1 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"?>
```

Valid OTA version 1 messages meet the grammar rules specified in the document type definitions or DTDs. Valid messages MUST include the document type declaration that follows the XML declaration. An example of a document type declaration is:

```
<!DOCTYPE OTA_v1 SYSTEM "OTA_v1.dtd">
```

Both the XML and document type declarations precede the OTA root element.

3.3.2 Message composition and grammar

This section defines the rules for the order and structure of the OTA version 1 business messages. The rules for grammar composition define a consistent vocabulary from various pieces of the logical view layers, such as message versions, verbs, and nouns described above in the reference model (section 3.1).

Most versioned OTA messages will have two main sections, Control and Content. The Control section consists of the infrastructure protocol information such as security and session. The Content section consists of travel business-specific information. Request and response messages MUST be placed in the Content section. The structure of non-versioned OTA messages is described in Chapter 6.

One advantage of splitting the messages into Control and Content, is that the information in the Control will more likely be affected by the work of standards organizations such as W3C. Since information in the Control is more generic and not travel specific, the data will more likely be subject to change as the XML standards develop further.

Chapter 4, Message Structure, gives precise specifications and examples of Control and Content construction, as well as the versioned standard error messages.

3.3.3 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: <v1.Cust.PaymentForm>

- Acronyms are discouraged, but where needed, use all upper case.

  Example: <v1.Cust.URL>

- Where acronyms or single-letter words cause two upper case characters to be adjacent, separate them with an underscore (_) .

  Example: <v1.Addr.PO_Box>

- Prefixes will use a period (.) as a separator.

  Example: <v1.Cust.Pay.CreditCard>

OTA's tag naming conventions include the specification version and content hierarchy as prefixes and, as a result, will require greater bandwidth to transmit than tags with more cryptic codes. OTA made the decision to include this much text in the tags – a decision that went through lengthy debate and several reviews – so OTA could convert the data model to XML-Schema as easily as possible. XML-Schema will not need the context or hierarchy included in the tag names, which will reduce their size.

The longer tag names also make the tags human-readable, 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. Also, in North America at least, bandwidth for transmitting messages is becoming more available and less expensive.

3.3.4 Tag naming guidelines

Tag writers SHOULD use these guidelines as good practices when constructing tag names.

- Use the same tag names with elements in a similar child structure
- Use plural tag names only for collections.
- Keep element and attribute names to 25 characters.

However, prefixes for use in DTDs do not count in the calculation. With XML-Schema, the context for tags will provide much of the meaning and help keep tag names short.

3.4 Infrastructure protocols

This section discusses versioning and related protocols. The next chapter on message structure offers details about security under the Control section.

3.4.1 Versioning

To provide for both implementation stability and managed growth, OTA has adopted versioning of messages. Changes to the schema or specification will be reflected by a change in the OTA message version.

3.4.1.1 Root tag versioning

Messages MUST be versioned by a version number attached to the end of the root element's tag name. This attachment MUST consist of an underscore, a lowercase letter V ('v'), and a whole number. The result for version 1, is to convert the OTA tag into:

OTA_v1

With the use of whole numbers only, subsequent versions will be: OTA_v2, OTA_v3, ..., OTA_v9, OTA_v10, etc...

3.4.1.2 Tag level versioning

One important reason for releasing a new version is to provide new functions, such as a new action under Content, or a new security model under Control. With changes like these, most of the schema or specification would probably not change and writing a completely new schema or specification for these reasons would waste time and energy. To meet these conditions, implementers will still need some way to differentiate portions of the XML schema tree by version. Or put another way, messages will need a version number attached to each portion of the tree.

Therefore, OTA has adopted versioning for all element tag names. With OTA version 1 that uses DTDs, versioning is put directly in the element tag name. In future versions that use XML-Schema, versioning can be expressed in the schema with the use of versioned archetype names.

In either case, except for the already versioned root OTA element (OTA_v*X*), this versioning is in the form of a prefix. Each XML tag in OTA version 1 MUST begin with a prefix consisting of a lowercase letter V ('v'), and a whole number. Any other prefixes and the text of the tag itself MUST follow the version tag. The result for a version 1 Content element, for example, is:

v1.Content

3.4.1.3 Implementation of versioning

The OTA versioned (OTA_v1, OTA_v2) root tag can be viewed as a sort of master switch. From this versioned tag, an implementer can select, at run-time, which code set to switch in to process the message.

When an element is changed from one version to the next, either by adding a child element or attribute, or by changing the names or data types, it should receive a new version number. This, of course, changes its parent element that now contains different child elements. These changes and new version numbers will flow all the way to the root tag (OTA_v*X*). Since the whole tree structure was probably not involved in this wave of changes, some portions will retain elements from older versions.

Therefore, after OTA version 1, each subsequent version will probably consist of multiple version elements. When OTA begins using XML-Schema, all the tags will change as the prefix versioning goes into the schema. This change will provide for a clean start with every tag graduating to the new (Schema-based) version of the OTA specification. Thereafter, subsequent versions will likely consist of multiple versions, but with versioning buried in the schema.

The need to discriminate between versions in OTA messages presents an extra burden on implementers, a burden that OTA acknowledges. OTA cannot guarantee that subsequent versions of its specifications will be backward-compatible with previous versions, despite the obvious desirability of building in this compatibility. As a result, OTA chose to err on the side of caution and include version in both the root tag and data tags so future messages could accommodate messages supporting multiple versions. Implementers will need to decide which version to express in the root tag based on the source of the tags in the business content of the message.

3.4.1.4 Actions per request or response

Each OTA version 1 message MUST contain only one business operation, defined as one verb expressed in a request or response. Future versions of the specification should support multiple or batch operations per message.

3.4.1.5 Logging

Organizations offering customer profile transactions SHOULD provide logging capability, regardless of composition used in the business message (e.g., travel verbs, infrastructure verbs). Also, 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 version 1 does not require use of a specific log format, nor does the message architecture preclude any logging capability.

3.5 Infrastructure verbs

This layer provides a cohesive approach for basic actions, and reduces the need for reconciling verb conflicts that may arise from business-specific verbs. The basic operations include Create, ReadAll, Update, and Delete – memorably abbreviated CRUD. These four verbs provide consistent conventions for basic OTA version 1 actions affecting both infrastructure and business elements.

In OTA version 1, Create, ReadAll, and Delete actions MUST apply only to entire profile records. Updates will allow for addressing one or more elements within a record.

3.5.1 Unique identifier on customer profiles

Each customer profile record MUST have a unique identifier assigned by the system that creates it with tag name `<v1.UniqueId>`. The unique identifier on the record MAY consist of a combination up to two elements:

- ResourceId, tag `<v1.ResourceId>`
- TradingPartner, tag. `<v1.TradingPartner>`

ResourceId. The unique identifier MUST contain the ResourceId, a unique string generated or assigned by the system that creates the profile. Examples of the ResourceId include employee numbers, organization's membership identification numbers, travel agency's account numbers, and travel supplier loyalty (e.g. frequent flyer) numbers.

The ResourceId has an optional attribute, *ResourceContext*, that describes the code or name represented by the ResourceId and when combined with the ResourceId creates a string with more chance of uniqueness for an identifier on customer profiles. Trading partners SHOULD use the ResourceContext if they have any question about potential duplication. Companies can use their IATA company codes, DUNS numbers, tax identification numbers, recognized corporate names or abbreviations (e.g. AOL, Wal-Mart), or other strings that can serve this purpose.

If the profile uses the ResourceContext, it MUST also have the ResourceId.

TradingPartner. Customer profile identifiers can add further guarantees of uniqueness by adding the TradingPartner element, representing the organizations that store profiles developed and shared by other entities. This element consists of the element CompanyCode (tag `<v1.CompanyCode>)` with the optional attribute CodeContext. The CompanyCode is a string identifying the organization, such as the IATA company identifier, DUNS number, or tax identification number. The CodeContext identifies the type of code with a string such as IATA, DUNS, or TAXID.

When more than one company stores the profile, the profile MAY use one or more TradingPartner elements. As with the ResourceId and ResourceContext, if a profile uses the CodeContext, then it MUST use the CompanyCode.

Figure 2 shows the relationship of the unique identifier elements with the Create, ReadAll, and Delete verbs that address the entire profile. The Update verbs are shown in Figure 3.

3.5.1.1 Examples of unique identifiers

A valid unique identifier MAY contain only the ResourceId, a unique string assigned by the system that created it …

```
<v1.UniqueId>
 <v1.ResourceId> 56139846 </v1.ResourceId>
</v1.UniqueId>

<v1.UniqueId>
 <v1.ResourceId> K817600 </v1.ResourceId>
</v1.UniqueId>
```

These strings may have meaning and uniqueness to the organizations creating and exchanging the profiles, but if the entities need to build more uniqueness into the identifiers, they can add a ResourceContext attribute to the ResourceId. When added to the ResourceId as a prefix, the resulting string would have a greater likelihood of uniqueness than the ResourceId itself.

```
<v1.UniqueId>
 <v1.ResourceId ResourceContext = "Hilton"> 56139846 </v1.ResourceId>
</v1.UniqueId>

<v1.UniqueId>
 <v1.ResourceId ResourceContext = "AvisAWD"> K817600 </v1.ResourceId>
</v1.UniqueId>
```

The TradingPartner element provides more protection against duplication. As described above, it contains the CompanyCode element with an optional CodeContext attribute. The following sample identifiers show a travel agency's DUNS number as the company code, with and without the CodeContext, for the two examples presented thus far.

```
<v1.UniqueId>
 <v1.TradingPartner>
      <v1.CompanyCode> 246813579 </v1.CompanyCode>
 </v1.TradingPartner>
 <v1.ResourceId ResourceContext = "Hilton"> 56139846 </v1.ResourceId>
</v1.UniqueId>

<v1.UniqueId>
 <v1.TradingPartner>
      <v1.CompanyCode CodeContext = "DUNS"> 246813579 </v1.CompanyCode>
 </v1.TradingPartner>
 <v1.ResourceId ResourceContext = "AvisAWD"> K817600 </v1.ResourceId>
</v1.UniqueId>
```

3.5.2 Create verb

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

- Requestor sends a Create request with the initial profile data, and optionally a unique identifier.
- 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.

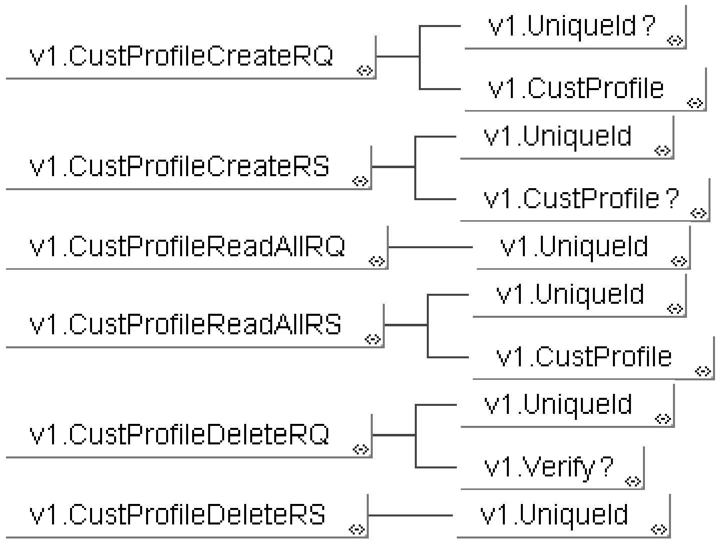
The requestor can complete as much of the profile record as desired when submitting the create request. However, The Customer section of the profile MUST contain at least one of the following elements:

- Cust.PersonName, tag `<v1.Cust.PersonName>`
- Cust.Telephone, tag `<v1.Cust.Telephone>`
- Cust.Email, tag `<v1.Cust.Email>`

See Chapter 5 on customer profile content for a discussion of the requirements of these elements.

Examples 1 and 2 show Create operations and are found on the following pages.

Figure 2. Main action verbs and profile identifier elements[2]



3.5.3 ReadAll verb

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

[2] This and other diagrams showing XML elements use DTD syntax symbols. The '?' indicates optional and single occurrence, '*' stands for optional and repeating occurrences, and '+' indicates required and repeating occurrences.

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

Examples 3 and 4 show a ReadAll exchange and follow the Create example.

Example 1. Create request message
See Chapter 4 for an explanation of the root (OTA_v1), Control, and Content elements

```
<?xml version="1.0"?>
<!DOCTYPE OTA_v1 SYSTEM "OTA_v1.dtd">
<OTA_v1 Timestamp="2001-10-08T21:10:30" EchoToken="87656" Target="Production"
GeoCode="+3851-7702">
 <v1.Control>
  <v1.Session Identity="XYZ Co" URI="http://ota.xyz.com">
   <v1.Credential Method="DeepSecret">asdfghjkl</v1.Credential>
    <v1.Principal Identity="Scott9">
    </v1.Principal>
  </v1.Session>
 </v1.Control>
 <v1.Content>
  <v1.CustProfileCreateRQ>
   <v1.CustProfile ShareAllMarketInd="No" ShareAllSynchInd="Yes">
    <v1.Customer Gender="Female">
     <v1.Cust.PersonName DefaultInd="Yes">
      <v1.PersonName>
       <v1.Person.NameTitle>Ms.</v1.Person.NameTitle>
       <v1.Person.GivenName>Phoebe</v1.Person.GivenName>
       <v1.Person.MiddleName>Peabody</v1.Person.MiddleName>
       <v1.Person.Surname>Beebe</v1.Person.Surname>
      </v1.PersonName>
     </v1.Cust.PersonName>
     <v1.Cust.Telephone DefaultInd="Yes" PhoneTech="Voice" PhoneUse="Work">
      <v1.Telephone CountryAccessCode="1">
       <v1.Phone.AreaCityCode>716</v1.Phone.AreaCityCode>
       <v1.Phone.Number>555-9999</v1.Phone.Number>
      </v1.Telephone>
     </v1.Cust.Telephone>
     <v1.Cust.Email EmailReferName="WorkEmail">PhoebePeabody@Beebe.Com
     </v1.Cust.Email>
    </v1.Customer>
   </v1.CustProfile>
  </v1.CustProfileCreateRQ>
 </v1.Content>
</OTA_v1>
```

Example 2. Create response message

```
<?xml version="1.0"?>
<!DOCTYPE OTA_v1 SYSTEM "OTA_v1.dtd">
<OTA_v1 Timestamp="2001-10-08T21:10:35" EchoToken="87656" Target="Production"
GeoCode="+4256-7844">
 <v1.Control>
  <v1.SendBy Identity="123 Co" URI="http://ota.onetwothree.com">
   <v1.Credential Method="RealPKI">poiuytr</v1.Credential>
  </v1.SendBy>
 </v1.Control>
 <v1.Content>
  <v1.CustProfileCreateRS Success="Yes">
   <v1.UniqueId>
    <v1.TradingPartner>
     <v1.CompanyCode CodeContext="DUNS"> 876543219 </v1.CompanyCode>
    </v1.TradingPartner>
    <v1.ResourceId ResourceContext="AAA_Member">050-7035487-005
    </v1.ResourceId>
   </v1.UniqueId>
  </v1.CustProfileCreateRS>
 </v1.Content>
</OTA_v1>
```

Example 3. ReadAll request message

```
<?xml version="1.0"?>
<!DOCTYPE OTA_v1 SYSTEM "OTA_v1.dtd">
<OTA_v1 Timestamp="2001-10-08T21:10:45" EchoToken="87658" Target="Production"
GeoCode="+3851-7702">
 <v1.Control>
  <v1.Session Identity="XYZ Co" URI="http://ota.xyz.com">
   <v1.Credential Method="DeepSecret">asdfghjkl</v1.Credential>
    <v1.Principal Identity="Scott9">
    </v1.Principal>
  </v1.Session>
 </v1.Control>
 <v1.Content>
  <v1.CustProfileReadAllRQ>
   <v1.UniqueId>
    <v1.TradingPartner>
     <v1.CompanyCode CodeContext="DUNS"> 876543219 </v1.CompanyCode>
    </v1.TradingPartner>
    <v1.ResourceId ResourceContext="AAA_Member">050-7035487-005
    </v1.ResourceId>
   </v1.UniqueId>
  </v1.CustProfileReadAllRQ>
 </v1.Content>
</OTA_v1>
```

Example 4. ReadAll response message

```
<?xml version="1.0"?>
<!DOCTYPE OTA_v1 SYSTEM "OTA_v1.dtd">
<OTA_v1 Timestamp="2001-10-08T21:10:50" EchoToken="87658" Target="Production"
GeoCode="+4256-7844">
 <v1.Control>
  <v1.SendBy Identity="123 Co" URI="http://ota.onetwothree.com">
   <v1.Credential Method="RealPKI">poiuytr</v1.Credential>
  </v1.SendBy>
 </v1.Control>
 <v1.Content>
  <v1.CustProfileReadAllRS Success="Yes">
   <v1.UniqueId>
    <v1.TradingPartner>
     <v1.CompanyCode CodeContext="DUNS"> 876543219 </v1.CompanyCode>
    </v1.TradingPartner>
    <v1.ResourceId ResourceContext="AAA_Member">050-7035487-005
    </v1.ResourceId>
   </v1.UniqueId>
   <v1.CustProfile ShareAllMarketInd="No" ShareAllSynchInd="Yes">
    <v1.Customer Gender="Female">
     <v1.Cust.PersonName DefaultInd="Yes">
      <v1.PersonName>
       <v1.Person.NameTitle>Ms.</v1.Person.NameTitle>
       <v1.Person.GivenName>Phoebe</v1.Person.GivenName>
       <v1.Person.MiddleName>Peabody</v1.Person.MiddleName>
       <v1.Person.Surname>Beebe</v1.Person.Surname>
      </v1.PersonName>
     </v1.Cust.PersonName>
     <v1.Cust.Telephone DefaultInd="Yes" PhoneTech="Voice" PhoneUse="Work">
      <v1.Telephone CountryAccessCode="1">
       <v1.Phone.AreaCityCode>716</v1.Phone.AreaCityCode>
       <v1.Phone.Number>555-9999</v1.Phone.Number>
      </v1.Telephone>
     </v1.Cust.Telephone>
     <v1.Cust.Email EmailReferName="WorkEmail">
      PhoebePeabody@Beebe.Com</v1.Cust.Email>
    </v1.Customer>
   </v1.CustProfile>
  </v1.CustProfileReadAllRS>
 </v1.Content>
</OTA_v1>
```

3.5.4 Update verb

The Update infrastructure verb defines an operation that opens an existing customer profile, identifies information that needs changing, transmits data corresponding to the appropriate elements in the profile, 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, the OTA version 1 specification devotes an extended discussion of this topic. We thank George Smith of ConXtra LLC for drafting this part of the document.

3.5.4.1 Update verb objectives

The objectives for the OTA version 1 Update process are:

- Provide for a mechanism to preserve and insure data integrity.
- Facilitate compound and complex updates in a single message pair exchange.
- Reuse the data/object models and XML representations generated for ReadAlls and Creates, in order to (a) minimize the changes to support Update and (b) not create a whole new syntax.

3.5.4.2 Update actions and syntax

Since OTA version 1 will support updates from multiple sources, as well as single operations per message, the specification needs a mechanism to inhibit potential corruption of data. To meet this objective, OTA version 1 will follow a compared-update-read process. This approach for updates is also used by GDSs.

The process can be viewed as consisting of two steps:

a) Verify the data in question, and on success
b) Modify the data

3.5.4.2.1 Sub-action verbs

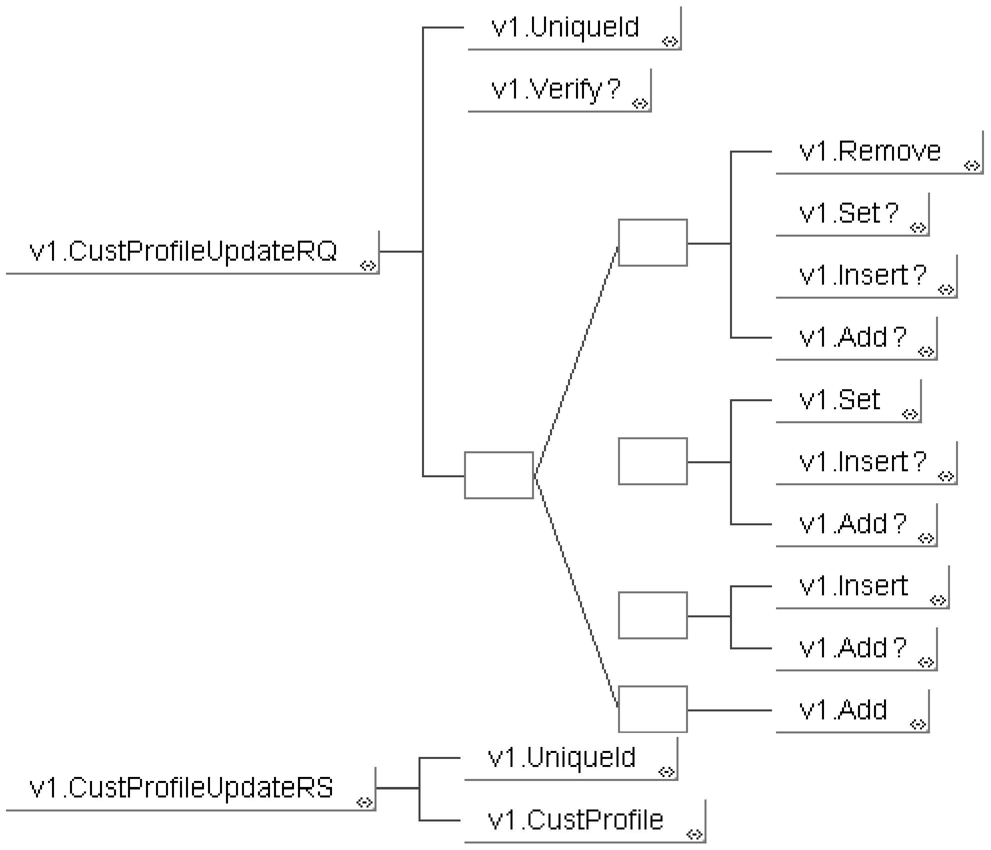
When an Update request (CustProfileUpdateRQ) is received, immediately under the element that indicates the UniqueId or primary key for the object being updated, the receiving system MUST allow the following five sub-actions, expressed as elements:

Verify,
Remove,
Set,
Insert – places items somewhere before the end of the record,
Add – appends the change to the end of the record.

Please note however that individually these sub-actions are optional. Under each of these verbs MUST go the actual structures to: Verify, Remove, Set, Insert, or Add.

Each of these sub-actions sits as an element under the main action verb request and response elements. When used, ONLY one of the sub-action verbs MUST precede the major content elements. However messages MAY contain multiple sub-action verbs, as described below. Figure 3 shows the configuration of the sub-actions and profile identifier under the Update verb.

Figure 3. Update verb, sub-actions, and profile identifiers



3.5.4.2.2 Conditions to accomplish updates

To accomplish the above process, two conditions MUST be met:

a) *Explicit Completeness*, defined as the principle that the data sent or received in an Update are complete and nothing else is left out. Unlike the ReadAll and Create verbs which have an implied completeness, the objects or element trees under (contained by) the sub-actions of Update would not normally represent all of the data. As such, with no implied completeness, an explicit completeness declaration must be supported.

Therefore, all elements in the data/object model from the Update sub-actions down (towards the leaf elements) that can contain elements MUST support an optional Boolean attribute of:

AllChildElements

The presence of this attribute indicates that from this element's view, the elements given immediately below (or contained by) it – its child elements – represent all the elements there are.

b) *Unique Element References.* Since identical attribute names are not allowed, then for non-repeatable elements and attributes the context allows for direct and unique references. For repeatable elements however, some uniqueness must be added to communicate which element is being referenced.

For all repeatable elements, the DynSeq -- for dynamic sequence – attribute MUST be used to give them an explicit order or sequence. Moreover these sequence numbers MUST not have gaps, MUST begin with a value of zero ('0') and MUST increment by a value of 1. Since the removal or deletion of one of these elements could cause the sequence numbers to be re-sequenced (upon completion of the action) they have a certain dynamic nature to them, and thus the name for this attribute. Implicitly adding these sequence numbers to the element names creates a contextually unique reference.

If a repeating element does not have a DynSeq number, the receiving system MUST assume it indicates the first and only appearance of the element, equivalent to a value of zero. (DynSeq = “0").

Because this attribute is required for the Update process and produced as part of the ReadAll process, and its dynamic nature, users SHOULD NOT use this attribute for any other purposes.

Please note that no matter how a system may generate a DynSeq value, if data are not updated by any party, the DynSeq numbers MUST stay the same from one Read operation to another over a reasonable period of time. Trading partners, as part of their implementation, will need to determine a reasonable length of time for the DynSeq numbers to remain unchanged from one Read operation to another.

The above approach would allow a server to:

A) Read and lock the object(s), and wait on a lock unavailable,

B) Verify that the data have not been changed since they were retrieved:

i) Verify that the Verify fields/sections exist (including current content if desired).
ii) Verify that the Remove fields/sections requested exist (including current content if desired).
iii) Verify that the Set fields/sections referenced exist.
iv) Verify that the Insert fields/sections referenced exist and that the new fields/sections being inserted do NOT exist (for repeatable elements this means having NO sequence number).
v) Verify that the Add fields/sections referenced exist and that the new fields/sections being added do NOT exist (for repeatable elements this means having NO sequence number).

If verification fails, the server SHOULD unlock the object(s) and report an error.

C) Modify the data, following this process. Following the Verify step, the sub-actions if used MUST appear in this order:

i) Remove the Remove fields/sections,
ii) Set the Set fields/sections,
iii) Insert the Insert fields/sections,
iv) Add the Add fields/sections (to the ends),
v) Re-sequence the changed sections, if the sequence numbers are explicitly stored

If modification fails, the server SHOULD unlock the object(s) and report an error.

D) Write and unlock the object(s),

E) Reply with success.

If the underlying legacy system does not support locking, then the parties MUST use some other mechanism to achieve the same result.

3.5.4.2.3 Insert sub-action requirements

To give the Insert sub-action the most flexibility for repeatable elements, it must be possible to insert an object relative to some other object. Therefore, for repeatable elements two additional attributes MUST be supported. These are:

BefDynSeq	(Before Dynamic Sequence), and
AftDynSeq	(After Dynamic Sequence).

Thus with a zero ('0') base to insert at the beginning, the BefDynSeq attribute would be a 0 (zero), to insert before element/item 0 (zero).

3.5.4.3 Update verb restrictions

As mentioned in section 3.5.2, a created profile MUST have entries in at least one of the following elements:

- Cust.PersonName, tag `<v1.Cust.PersonName>`
- Cust.Telephone, tag `<v1.Cust.Telephone>`
- Cust.Email, tag `<v1.Cust.Email>`

As a result, updates to the profile MUST NOT remove all of the required data, and any resulting profile record MUST meet the same requirements as a newly created profile.

Using the sub-action verbs -- Verify, Remove, Set, Insert, Add -- REQUIRES use of the Cust.Profile element and the same mandatory data requirements apply here as well. If the request message updates the required data in Cust.Profile, the message will be valid. However, if the request updates Cust.Profile data other than the required elements, the requestor MUST include the minimum required data from Cust.PersonName, Cust.Telephone, or Cust.Email. In this circumstance, the requestor SHOULD use the Validation Repeat Indicator (ValidRepeatInd) attribute, set to a value of

"Yes", on the data repeated in the message only for the purpose of meeting these validation requirements.

3.5.4.4 Summary of the Update verb

A) The Update action is divided into the following sub-actions:

Verify,
Remove,
Set,
Insert,
Add.

B) Under these sub-actions use the normal specified OTA version 1 object or data models.

C) Modify the normal specified object or data models to support the following attributes:

AllChildElements,
DynSeq,
BefDynSeq, &
AftDynSeq.

D) Require that repeatable elements have a DynSeq (Dynamic Sequence) number on Reads; the first and only of these elements may have its DynSeq value assumed as "0".

E) An updated profile record MUST meet the same mandated data requirements as a newly created profile.

3.5.4.5 Update verb examples

These examples, with the ones presented in the Appendix, illustrate all of the sub-action verbs described in this section. The last example in the Appendix shows the use of all sub-action verbs in one message.

In the following text and examples, the messages are NOT meant to imply any particular business condition, system configuration, or platform. The examples presented below consist of the profile data in XML given before and after the updates, as well as the messages themselves.

Example 5: Change a person's work voice telephone number area code, from 206 to 253
Example 5. XML object before:

```
<v1.CustProfile ShareAllMarketInd="No" ShareAllSynchInd="Yes">
 <v1.Customer>
  <v1.Cust.PersonName NameType="Default">
   <v1.PersonName>
    <v1.Person.NameTitle> Mr. </v1.Person.NameTitle>
    <v1.Person.GivenName> George </v1.Person.GivenName>
    <v1.Person.MiddleInitial> A. </v1.Person.MiddleInitial>
    <v1.Person.Surname> Smith</v1.Person.Surname>
   </v1.PersonName>
  </v1.Cust.PersonName>
  <v1.Cust.Telephone PhoneTech="Voice" PhoneUse="Work" DynSeq="0">
   <v1.Telephone>
    <v1.Phone.AreaCityCode>206</v1.Phone.AreaCityCode>  ← Data to change
     <v1.Phone.Number>813-8532</v1.Phone.Number>
   </v1.Telephone>
  </v1.Cust.Telephone>
  <v1.Cust.Telephone PhoneTech = "Fax" PhoneUse = "Work"
    DynSeq = "1">
   <v1.Telephone>
    <v1.Phone.AreaCityCode>253</v1.Phone.AreaCityCode>
    <v1.Phone.Number>813-8698</v1.Phone.Number>
   </v1.Telephone>
  </v1.Cust.Telephone>
  <v1.Cust.PaymentForm>
   <v1.Cust.Pay.CreditCard CardType = "Credit"
     PayFormReferName = "WorkMastercard" CardNumber="M:4356 ... 5677"
     CardExpiryDate="2001-12-31">
   </v1.Cust.Pay.CreditCard>
   <v1.Cust.Pay.DirectBill DirectBill_Id = "323453"
     PayFormReferName = "ConX_Invoice">
    <v1.Cust.Pay.Direct.CompanyName>ConXTravel Co
    </v1.Cust.Pay.Direct.CompanyName>
    <v1.Cust.Pay.Direct.Address>
     <v1.Address>
      <v1.Addr.StreetNmbr POBox = "4321-01">1200
        Yakima St</v1.Addr.StreetNmbr>
      <v1.Addr.BldgRoom>Suite 800</v1.Addr.BldgRoom>
      <v1.Addr.CityName>Seattle</v1.Addr.CityName>
      <v1.Addr.StateProv PostalCode = "98108">WA
      </v1.Addr.StateProv>
      <v1.Addr.CountryName>USA</v1.Addr.CountryName>
     </v1.Address>
    </v1.Cust.Pay.Direct.Address>
   </v1.Cust.Pay.DirectBill>
  </v1.Cust.PaymentForm>
 </v1.Customer>
</v1.CustProfile>
```

Note: The credit card number (v1.Cust.Pay.Card.Number) above shows an example of a possible representation for "munging" (changing it to communicate incomplete, but sufficient for the user to use) the data.

Example 5. XML object after:

```
<v1.CustProfile ShareAllMarketInd = "No" ShareAllSynchInd = "Yes">
 <v1.Customer>
  <v1.Cust.PersonName NameType = "Default">
   <v1.PersonName>
    <v1.Person.NameTitle> Mr. </v1.Person.NameTitle>
    <v1.Person.GivenName> George </v1.Person.GivenName>
    <v1.Person.MiddleInitial> A. </v1.Person.MiddleInitial>
    <v1.Person.Surname> Smith</v1.Person.Surname>
   </v1.PersonName>
  </v1.Cust.PersonName>
  <v1.Cust.Telephone PhoneTech = "Voice" PhoneUse = "Work"
    DynSeq = "0">
   <v1.Telephone>
    <v1.Phone.AreaCityCode>253</v1.Phone.AreaCityCode>  ← Data changed
    <v1.Phone.Number>813-8532</v1.Phone.Number>
   </v1.Telephone>
  </v1.Cust.Telephone>
  <v1.Cust.Telephone PhoneTech = "Fax" PhoneUse = "Work"
    DynSeq = "1">
   <v1.Telephone>
    <v1.Phone.AreaCityCode>253</v1.Phone.AreaCityCode>
    <v1.Phone.Number>813-8698</v1.Phone.Number>
   </v1.Telephone>
  </v1.Cust.Telephone>
  <v1.Cust.PaymentForm>
   <v1.Cust.Pay.CreditCard CardType = "Credit" PayFormReferName =
     "WorkMastercard"  CardExpiryDate="2001-12-31" CardNumber=
     "M:4356 ... 5677">
   </v1.Cust.Pay.CreditCard>
   <v1.Cust.Pay.DirectBill DirectBill_Id = "323453"
     PayFormReferName = "ConX_Invoice">
    <v1.Cust.Pay.Direct.CompanyName>ConXTravel Co
    </v1.Cust.Pay.Direct.CompanyName>
    <v1.Cust.Pay.Direct.Address>
     <v1.Address>
      <v1.Addr.StreetNmbr POBox = "4321-01">1200 Yakima St
      </v1.Addr.StreetNmbr>
      <v1.Addr.BldgRoom>Suite 800</v1.Addr.BldgRoom>
      <v1.Addr.CityName>Seattle</v1.Addr.CityName>
      <v1.Addr.StateProv PostalCode = "98108">WA
      </v1.Addr.StateProv>
      <v1.Addr.CountryName>USA</v1.Addr.CountryName>
     </v1.Address>
    </v1.Cust.Pay.Direct.Address>
   </v1.Cust.Pay.DirectBill>
  </v1.Cust.PaymentForm>
 </v1.Customer>
</v1.CustProfile>
```

Example 5. Update: ReadAll message exchange

ReadAll request:

```
<?xml version="1.0"?>
<!DOCTYPE OTA_v1 SYSTEM "OTA_v1.dtd">
<OTA_v1 Timestamp = "2000-11-08T09:30:00" EchoToken = "22334455"
  Target = "Production" GeoCode = "+4732-12218">
 <v1.Control>
  <v1.Session Identity = "ConXTravel Co"
    URI = "http://ota.ConXTravel.com">
   <v1.Credential Method = "MegaSecure">trewq</v1.Credential>
    <v1.Principal Identity = "CompanyPrincipal">
    </v1.Principal>
  </v1.Session>
 </v1.Control>
 <v1.Content>
  <v1.CustProfileReadAllRQ>
   <v1.UniqueId>
    <v1.ResourceId>XYA3:ConX_Smith/George(0)</v1.ResourceId>
   </v1.UniqueId>
  </v1.CustProfileReadAllRQ>
 </v1.Content>
</OTA_v1>
```

ReadAll response:

```
<?xml version="1.0"?>
<!DOCTYPE OTA_v1 SYSTEM "OTA_v1.dtd">
<OTA_v1 Timestamp="2000-11-08T09:30:05" EchoToken="22334455" Target="Production"
GeoCode="+3018-9742">
 <v1.Control>
  <v1.SendBy Identity="LoneStarAir" URI="http://ota.LoneStarAir.com">
   <v1.Credential Method="JunkYardDog">bvc098cxz</v1.Credential>
  </v1.SendBy>
 </v1.Control>
 <v1.Content>
  <v1.CustProfileReadAllRS Success="Yes">
   <v1.UniqueId>
    <v1.TradingPartner>
     <v1.CompanyCode CodeContext="FedTaxID">654-987123478
     </v1.CompanyCode>
    </v1.TradingPartner>
    <v1.ResourceId>XYA3:ConX_Smith/George(0)</v1.ResourceId>
   </v1.UniqueId>
   <v1.CustProfile ShareAllMarketInd="No" ShareAllSynchInd="Yes">
    <v1.Customer>
     <v1.Cust.PersonName NameType="Default">
      <v1.PersonName>
       <v1.Person.NameTitle> Mr. </v1.Person.NameTitle>
       <v1.Person.GivenName> George </v1.Person.GivenName>
       <v1.Person.MiddleInitial> A. </v1.Person.MiddleInitial>
       <v1.Person.Surname> Smith</v1.Person.Surname>
      </v1.PersonName>
     </v1.Cust.PersonName>
     <v1.Cust.Telephone PhoneTech="Voice" PhoneUse="Work" DynSeq="0">
      <v1.Telephone>
       <v1.Phone.AreaCityCode>206</v1.Phone.AreaCityCode>
        <v1.Phone.Number>813-8532</v1.Phone.Number>
      </v1.Telephone>
     </v1.Cust.Telephone>
     <v1.Cust.Telephone PhoneTech = "Fax" PhoneUse = "Work"
       DynSeq = "1">
      <v1.Telephone>
       <v1.Phone.AreaCityCode>253</v1.Phone.AreaCityCode>
       <v1.Phone.Number>813-8698</v1.Phone.Number>
      </v1.Telephone>
     </v1.Cust.Telephone>
     <v1.Cust.PaymentForm>
      <v1.Cust.Pay.CreditCard CardType = "Credit"
        PayFormReferName = "WorkMastercard" CardNumber="M:4356 ... 5677"
        CardExpiryDate="2001-12-31">
      </v1.Cust.Pay.CreditCard>
      <v1.Cust.Pay.DirectBill DirectBill_Id = "323453"
        PayFormReferName = "ConX_Invoice">
       <v1.Cust.Pay.Direct.CompanyName>ConXTravel Co
       </v1.Cust.Pay.Direct.CompanyName>
       <v1.Cust.Pay.Direct.Address>
        <v1.Address>
         <v1.Addr.StreetNmbr POBox = "4321-01">1200
           Yakima St</v1.Addr.StreetNmbr>
         <v1.Addr.BldgRoom>Suite 800</v1.Addr.BldgRoom>
         <v1.Addr.CityName>Seattle</v1.Addr.CityName>
         <v1.Addr.StateProv PostalCode = "98108">WA
         </v1.Addr.StateProv>
         <v1.Addr.CountryName>USA</v1.Addr.CountryName>
        </v1.Address>
       </v1.Cust.Pay.Direct.Address>
      </v1.Cust.Pay.DirectBill>
     </v1.Cust.PaymentForm>
    </v1.Customer>
   </v1.CustProfile>
  </v1.CustProfileReadAllRS>
 </v1.Content>
</OTA_v1>
```

Example 5: Update message exchange

Update message request

```
<?xml version="1.0"?>
<!DOCTYPE OTA_v1 SYSTEM "OTA_v1.dtd">
<OTA_v1 Timestamp = "2000-11-08T09:31:00" EchoToken = "22334457"
  Target = "Production" GeoCode = "+4732-12218">
 <v1.Control>
  <v1.Session Identity = "ConXTravel Co"
    URI = "http://ota.ConXTravel.com">
   <v1.Credential Method = "MegaSecure">trewq</v1.Credential>
    <v1.Principal Identity = "CompanyPrincipal">
    </v1.Principal>
  </v1.Session>
 </v1.Control>
 <v1.Content>
  <v1.CustProfileUpdateRQ>
   <v1.UniqueId>
    <v1.ResourceId>XYA3:ConX_Smith/George(0)</v1.ResourceId>
    </v1.UniqueId>
    <v1.Verify>
     <v1.CustProfile ShareAllMarketInd = "No" ShareAllSynchInd = "Yes">
      <v1.Customer>
       <v1.Cust.PersonName NameType = "Default">
        <v1.PersonName>
         <v1.Person.NameTitle> Mr. </v1.Person.NameTitle>
         <v1.Person.GivenName> George </v1.Person.GivenName>
         <v1.Person.MiddleInitial> A. </v1.Person.MiddleInitial>
         <v1.Person.Surname> Smith</v1.Person.Surname>
        </v1.PersonName>
       </v1.Cust.PersonName>
       <v1.Cust.Telephone PhoneTech = "Voice" PhoneUse = "Work">
        <v1.Telephone>
         <v1.Phone.AreaCityCode>206</v1.Phone.AreaCityCode>
          <v1.Phone.Number>813-8532</v1.Phone.Number>
        </v1.Telephone>
       </v1.Cust.Telephone>
      </v1.Customer>
     </v1.CustProfile>
    </v1.Verify>
    <v1.Set>
     <v1.CustProfile>
      <v1.Customer>
       <v1.Cust.Telephone PhoneTech = "Voice" PhoneUse = "Work">
        <v1.Telephone>
         <v1.Phone.AreaCityCode>253</v1.Phone.AreaCityCode>
         <v1.Phone.Number>813-8532</v1.Phone.Number>
        </v1.Telephone>
       </v1.Cust.Telephone>
      </v1.Customer>
     </v1.CustProfile>
    </v1.Set>
   </v1.CustProfileUpdateRQ>
  </v1.Content>
</OTA_v1>
```

Update message response

```
<?xml version="1.0"?>
<!DOCTYPE OTA_v1 SYSTEM "C:\My Documents\DISA\OTA\Testbed\OTA_v1.dtd">
<OTA_v1 Timestamp="2000-11-08T09:31:05" EchoToken="22334457" Target="Production"
GeoCode="+3018-9742">
 <v1.Control>
  <v1.SendBy Identity="LoneStarAir" URI="http://ota.LoneStarAir.com">
   <v1.Credential Method="JunkYardDog">bvc098cxz</v1.Credential>
  </v1.SendBy>
 </v1.Control>
 <v1.Content>
  <v1.CustProfileUpdateRS Success="Yes">
   <v1.UniqueId>
    <v1.TradingPartner>
     <v1.CompanyCode CodeContext="FedTaxID">654-987123478
     </v1.CompanyCode>
    </v1.TradingPartner>
    <v1.ResourceId>XYA3:ConX_Smith/George(0)</v1.ResourceId>
   </v1.UniqueId>
```

```
    <v1.CustProfile ShareAllMarketInd = "No" ShareAllSynchInd = "Yes">
     <v1.Customer>
      <v1.Cust.PersonName NameType = "Default">
       <v1.PersonName>
        <v1.Person.NameTitle> Mr. </v1.Person.NameTitle>
        <v1.Person.GivenName> George </v1.Person.GivenName>
        <v1.Person.MiddleInitial> A. </v1.Person.MiddleInitial>
        <v1.Person.Surname> Smith</v1.Person.Surname>
       </v1.PersonName>
      </v1.Cust.PersonName>
      <v1.Cust.Telephone PhoneTech = "Voice" PhoneUse = "Work"
        DynSeq = "0">
       <v1.Telephone>
        <v1.Phone.AreaCityCode>253</v1.Phone.AreaCityCode>
        <v1.Phone.Number>813-8532</v1.Phone.Number>
       </v1.Telephone>
      </v1.Cust.Telephone>
      <v1.Cust.Telephone PhoneTech = "Fax" PhoneUse = "Work"
        DynSeq = "1">
       <v1.Telephone>
        <v1.Phone.AreaCityCode>253</v1.Phone.AreaCityCode>
        <v1.Phone.Number>813-8698</v1.Phone.Number>
       </v1.Telephone>
      </v1.Cust.Telephone>
      <v1.Cust.PaymentForm>
       <v1.Cust.Pay.CreditCard CardType = "Credit" PayFormReferName =
         "WorkMastercard"  CardExpiryDate="2001-12-31" CardNumber=
         "M:4356 ... 5677">
       </v1.Cust.Pay.CreditCard>
       <v1.Cust.Pay.DirectBill DirectBill_Id = "323453"
         PayFormReferName = "ConX_Invoice">
        <v1.Cust.Pay.Direct.CompanyName>ConXTravel Co
        </v1.Cust.Pay.Direct.CompanyName>
        <v1.Cust.Pay.Direct.Address>
         <v1.Address>
          <v1.Addr.StreetNmbr POBox = "4321-01">1200 Yakima St
          </v1.Addr.StreetNmbr>
          <v1.Addr.BldgRoom>Suite 800</v1.Addr.BldgRoom>
          <v1.Addr.CityName>Seattle</v1.Addr.CityName>
          <v1.Addr.StateProv PostalCode = "98108">WA
          </v1.Addr.StateProv>
          <v1.Addr.CountryName>USA</v1.Addr.CountryName>
         </v1.Address>
        </v1.Cust.Pay.Direct.Address>
       </v1.Cust.Pay.DirectBill>
      </v1.Cust.PaymentForm>
     </v1.Customer>
    </v1.CustProfile>
   </v1.CustProfileUpdateRS>
  </v1.Content>
</OTA_v1>
```

Note: This Response with the object (CustProfile) could show additional changes that we did not verify or change

In Example 6, George Smith adds his children to his profile as related travelers.

Example 6. XML object before:

```
<v1.CustProfile ShareAllMarketInd="No" ShareAllSynchInd="Yes">
 <v1.Customer>
  <v1.Cust.PersonName NameType="Default">
   <v1.PersonName>
    <v1.Person.NameTitle> Mr. </v1.Person.NameTitle>
    <v1.Person.GivenName> George </v1.Person.GivenName>
    <v1.Person.MiddleInitial> A. </v1.Person.MiddleInitial>
    <v1.Person.Surname> Smith</v1.Person.Surname>
   </v1.PersonName>
  </v1.Cust.PersonName>
  <v1.Cust.Telephone PhoneTech="Voice" PhoneUse="Work" DynSeq="0">
   <v1.Telephone>
    <v1.Phone.AreaCityCode>253</v1.Phone.AreaCityCode>
    <v1.Phone.Number>813-8532</v1.Phone.Number>
   </v1.Telephone>
    </v1.Cust.Telephone>
    <v1.Cust.Telephone PhoneTech="Fax" PhoneUse="Work" DynSeq="1">
   <v1.Telephone>
    <v1.Phone.AreaCityCode>253</v1.Phone.AreaCityCode>
    <v1.Phone.Number>813-8698</v1.Phone.Number>
   </v1.Telephone>
  </v1.Cust.Telephone>
  <v1.Cust.PaymentForm>
   <v1.Cust.Pay.CreditCard CardType="Credit"
    PayFormReferName="WorkMastercard" CardNumber="M:4356 ... 5677"
    CardExpiryDate="2001-12-31">
   </v1.Cust.Pay.CreditCard>
   <v1.Cust.Pay.DirectBill DirectBill_Id="323453"
     PayFormReferName="ConXInvoice">
    <v1.Cust.Pay.Direct.CompanyName>ConXTravel Co
    </v1.Cust.Pay.Direct.CompanyName>
    <v1.Cust.Pay.Direct.Address>
     <v1.Address>
      <v1.Addr.StreetNmbr POBox="4321-01">1200
       Yakima St</v1.Addr.StreetNmbr>
      <v1.Addr.BldgRoom>Suite 800</v1.Addr.BldgRoom>
      <v1.Addr.CityName>Seattle</v1.Addr.CityName>
      <v1.Addr.StateProv PostalCode="98108">WA
      </v1.Addr.StateProv>
      <v1.Addr.CountryName>USA</v1.Addr.CountryName>
     </v1.Address>
    </v1.Cust.Pay.Direct.Address>
   </v1.Cust.Pay.DirectBill>
  </v1.Cust.PaymentForm>
 </v1.Customer>
</v1.CustProfile>
```

Example 6. XML object after:

```
<v1.CustProfile ShareAllMarketInd="No" ShareAllSynchInd="Yes">
 <v1.Customer>
  <v1.Cust.PersonName NameType="Default">
   <v1.PersonName>
    <v1.Person.NameTitle> Mr. </v1.Person.NameTitle>
    <v1.Person.GivenName> George </v1.Person.GivenName>
    <v1.Person.MiddleInitial> A. </v1.Person.MiddleInitial>
    <v1.Person.Surname> Smith</v1.Person.Surname>
   </v1.PersonName>
  </v1.Cust.PersonName>
  <v1.Cust.Telephone PhoneTech="Voice" PhoneUse="Work" DynSeq="0">
   <v1.Telephone>
    <v1.Phone.AreaCityCode>253</v1.Phone.AreaCityCode>
    <v1.Phone.Number>813-8532</v1.Phone.Number>
   </v1.Telephone>
  </v1.Cust.Telephone>
  <v1.Cust.Telephone PhoneTech="Fax" PhoneUse="Work" DynSeq="1">
   <v1.Telephone>
    <v1.Phone.AreaCityCode>253</v1.Phone.AreaCityCode>
    <v1.Phone.Number>813-8698</v1.Phone.Number>
   </v1.Telephone>
  </v1.Cust.Telephone>
  <v1.Cust.PaymentForm>
   <v1.Cust.Pay.CreditCard CardType="Credit"
     PayFormReferName="WorkMastercard"
     CardNumber="M:4356 ... 5677" CardExpiryDate="2001-12-31">
    </v1.Cust.Pay.CreditCard>
   <v1.Cust.Pay.DirectBill DirectBill_Id="323453"
    PayFormReferName="ConX_Invoice">
    <v1.Cust.Pay.Direct.CompanyName>ConXTravel Co
    </v1.Cust.Pay.Direct.CompanyName>
    <v1.Cust.Pay.Direct.Address>
     <v1.Address>
      <v1.Addr.StreetNmbr POBox="4321-01">1200 Yakima St
      </v1.Addr.StreetNmbr>
      <v1.Addr.BldgRoom>Suite 800</v1.Addr.BldgRoom>
      <v1.Addr.CityName>Seattle</v1.Addr.CityName>
      <v1.Addr.StateProv PostalCode="98108">WA
      </v1.Addr.StateProv>
      <v1.Addr.CountryName>USA</v1.Addr.CountryName>
     </v1.Address>
    </v1.Cust.Pay.Direct.Address>
   </v1.Cust.Pay.DirectBill>
  </v1.Cust.PaymentForm>
  <v1.Cust.RelatedTraveler Relation="Child" DynSeq="0">  ← New data added
   <v1.UniqueId>
    <v1.ResourceId>7005-548-703</v1.ResourceId>
   </v1.UniqueId>
   <v1.Cust.Related.PersonName>
    <v1.PersonName>
     <v1.Person.NameTitle>Mr.</v1.Person.NameTitle>
     <v1.Person.GivenName>Devin</v1.Person.GivenName>
     <v1.Person.MiddleInitial>R.</v1.Person.MiddleInitial>
     <v1.Person.Surname>Smith</v1.Person.Surname>
    </v1.PersonName>
```

```
      </v1.Cust.Related.PersonName>
     </v1.Cust.RelatedTraveler>
     <v1.Cust.RelatedTraveler Relation="Child" DynSeq="1">
      <v1.UniqueId>
       <v1.ResourceId>7005-548-704</v1.ResourceId>
      </v1.UniqueId>
      <v1.Cust.Related.PersonName>
       <v1.PersonName>
        <v1.Person.NameTitle>Ms.</v1.Person.NameTitle>
        <v1.Person.GivenName>Amy</v1.Person.GivenName>
        <v1.Person.MiddleInitial>E.</v1.Person.MiddleInitial>
        <v1.Person.Surname>Smith</v1.Person.Surname>
       </v1.PersonName>
      </v1.Cust.Related.PersonName>
     </v1.Cust.RelatedTraveler>
    </v1.Customer>
   </v1.CustProfile>
```

Example 6. Update: ReadAll message exchange

ReadAll request:

```
<?xml version ="1.0"?>
<!DOCTYPE OTA_v1 SYSTEM "OTA_v1.dtd">
<OTA_v1 Timestamp = "2000-11-08T09:32:00" EchoToken = "22334459"
  Target = "Production" GeoCode = "+4732-12218/">
 <v1.Control>
  <v1.Session Identity = "ConXTravel Co"
    URI = "http://ota.ConXTravel.com">
   <v1.Credential Method = "MegaSecure">trewq</v1.Credential>
    <v1.Principal Identity = "CompanyPrincipal">
    </v1.Principal>
  </v1.Session>
 </v1.Control>
 <v1.Content>
  <v1.CustProfileReadAllRQ>
   <v1.UniqueId>
    <v1.ResourceId>XYA3:ConX_Smith/George(0)</v1.ResourceId>
   </v1.UniqueId>
  </v1.CustProfileReadAllRQ>
 </v1.Content>
</OTA_v1>
```

ReadAll response:

```
<?xml version="1.0"?>
<!DOCTYPE OTA_v1 SYSTEM "OTA_v1.dtd">
<OTA_v1 Timestamp="2000-11-08T09:32:05" EchoToken="22334459" Target="Production"
GeoCode="+3018-9742">
 <v1.Control>
  <v1.SendBy Identity="LoneStarAir" URI="http://ota.LoneStarAir.com">
   <v1.Credential Method="JunkYardDog">bvc098cxz</v1.Credential>
  </v1.SendBy>
 </v1.Control>
 <v1.Content>
  <v1.CustProfileReadAllRS Success="Yes">
```

```
    <v1.UniqueId>
     <v1.TradingPartner>
      <v1.CompanyCode CodeContext="FedTaxID">654-987123478
      </v1.CompanyCode>
     </v1.TradingPartner>
     <v1.ResourceId>XYA3:ConX_Smith/George(0)</v1.ResourceId>
    </v1.UniqueId>
    <v1.CustProfile ShareAllMarketInd="No" ShareAllSynchInd="Yes">
     <v1.Customer>
      <v1.Cust.PersonName NameType="Default">
       <v1.PersonName>
        <v1.Person.NameTitle> Mr. </v1.Person.NameTitle>
        <v1.Person.GivenName> George </v1.Person.GivenName>
        <v1.Person.MiddleInitial> A. </v1.Person.MiddleInitial>
        <v1.Person.Surname> Smith</v1.Person.Surname>
       </v1.PersonName>
      </v1.Cust.PersonName>
      <v1.Cust.Telephone PhoneTech="Voice" PhoneUse="Work" DynSeq="0">
       <v1.Telephone>
        <v1.Phone.AreaCityCode>253</v1.Phone.AreaCityCode>
        <v1.Phone.Number>813-8532</v1.Phone.Number>
       </v1.Telephone>
        </v1.Cust.Telephone>
        <v1.Cust.Telephone PhoneTech="Fax" PhoneUse="Work" DynSeq="1">
       <v1.Telephone>
        <v1.Phone.AreaCityCode>253</v1.Phone.AreaCityCode>
        <v1.Phone.Number>813-8698</v1.Phone.Number>
       </v1.Telephone>
      </v1.Cust.Telephone>
      <v1.Cust.PaymentForm>
       <v1.Cust.Pay.CreditCard CardType="Credit"
        PayFormReferName="WorkMastercard" CardNumber="M:4356 ... 5677"
        CardExpiryDate="2001-12-31">
       </v1.Cust.Pay.CreditCard>
       <v1.Cust.Pay.DirectBill DirectBill_Id="323453"
         PayFormReferName="ConXInvoice">
        <v1.Cust.Pay.Direct.CompanyName>ConXTravel Co
        </v1.Cust.Pay.Direct.CompanyName>
        <v1.Cust.Pay.Direct.Address>
         <v1.Address>
          <v1.Addr.StreetNmbr POBox="4321-01">1200
           Yakima St</v1.Addr.StreetNmbr>
          <v1.Addr.BldgRoom>Suite 800</v1.Addr.BldgRoom>
          <v1.Addr.CityName>Seattle</v1.Addr.CityName>
          <v1.Addr.StateProv PostalCode="98108">WA
          </v1.Addr.StateProv>
          <v1.Addr.CountryName>USA</v1.Addr.CountryName>
         </v1.Address>
        </v1.Cust.Pay.Direct.Address>
       </v1.Cust.Pay.DirectBill>
      </v1.Cust.PaymentForm>
     </v1.Customer>
    </v1.CustProfile>
   </v1.CustProfileReadAllRS>
  </v1.Content>
</OTA_v1>
```

Example 6: Update message exchange

Update message request

```
<?xml version="1.0"?>
<!DOCTYPE OTA_v1 SYSTEM "OTA_v1.dtd">
<OTA_v1 Timestamp="2000-11-08T09:33:00" EchoToken="22334461" Target="Production"
GeoCode="+4732-12218">
 <v1.Control>
  <v1.Session Identity="ConXTravel Co" URI="http://ota.ConXTravel.com">
   <v1.Credential Method="MegaSecure">trewq</v1.Credential>
    <v1.Principal Identity="CompanyPrincipal">
    </v1.Principal>
  </v1.Session>
 </v1.Control>
 <v1.Content>
  <v1.CustProfileUpdateRQ>
   <v1.UniqueId>
    <v1.ResourceId>XYA3:ConX_Smith/George(0)</v1.ResourceId>
   </v1.UniqueId>
   <v1.Verify>
    <v1.CustProfile ShareAllMarketInd="No" ShareAllSynchInd="Yes"
      AllChildElements="Yes">
     <v1.Customer>
      <v1.Cust.PersonName NameType="Default">
       <v1.PersonName>
        <v1.Person.NameTitle> Mr. </v1.Person.NameTitle>
        <v1.Person.GivenName> George </v1.Person.GivenName>
        <v1.Person.MiddleInitial> A. </v1.Person.MiddleInitial>
        <v1.Person.Surname> Smith</v1.Person.Surname>
       </v1.PersonName>
      </v1.Cust.PersonName>
      <v1.Cust.Telephone PhoneTech="Voice" PhoneUse="Work" DynSeq="0">
       <v1.Telephone>
        <v1.Phone.AreaCityCode>253</v1.Phone.AreaCityCode>
        <v1.Phone.Number>813-8532</v1.Phone.Number>
       </v1.Telephone>
      </v1.Cust.Telephone>
      <v1.Cust.Telephone PhoneTech="Fax" PhoneUse="Work" DynSeq="1">
       <v1.Telephone>
        <v1.Phone.AreaCityCode>253</v1.Phone.AreaCityCode>
        <v1.Phone.Number>813-8698</v1.Phone.Number>
       </v1.Telephone>
      </v1.Cust.Telephone>
      <v1.Cust.PaymentForm>
       <v1.Cust.Pay.CreditCard CardType="Credit"
        PayFormReferName="WorkMastercard" CardNumber="M:4356 ... 5677"
        CardExpiryDate="2001-12-31">
       </v1.Cust.Pay.CreditCard>
       <v1.Cust.Pay.DirectBill DirectBill_Id="323453"
        PayFormReferName="ConXInvoice">
        <v1.Cust.Pay.Direct.CompanyName>ConXTravel Co
        </v1.Cust.Pay.Direct.CompanyName>
        <v1.Cust.Pay.Direct.Address>
         <v1.Address>
          <v1.Addr.StreetNmbr POBox="4321-01">
```

```
             1200 Yakima St</v1.Addr.StreetNmbr>
            <v1.Addr.BldgRoom>Suite 800</v1.Addr.BldgRoom>
            <v1.Addr.CityName>Seattle</v1.Addr.CityName>
            <v1.Addr.StateProv PostalCode="98108">WA
            </v1.Addr.StateProv>
            <v1.Addr.CountryName>USA</v1.Addr.CountryName>
           </v1.Address>
          </v1.Cust.Pay.Direct.Address>
         </v1.Cust.Pay.DirectBill>
        </v1.Cust.PaymentForm>
       </v1.Customer>
      </v1.CustProfile>
     </v1.Verify>
     <v1.Add>
      <v1.CustProfile>
       <v1.Customer>
        <v1.Cust.PersonName ValidRepeatInd="Yes">
         <v1.PersonName>
          <v1.Person.Surname> Smith</v1.Person.Surname>
         </v1.PersonName>
        </v1.Cust.PersonName>
        <v1.Cust.RelatedTraveler Relation="Child" DynSeq="0">
         <v1.UniqueId>
          <v1.ResourceId>7005-548-703</v1.ResourceId>
         </v1.UniqueId>
        <v1.Cust.Related.PersonName>
         <v1.PersonName>
          <v1.Person.NameTitle>Mr.</v1.Person.NameTitle>
          <v1.Person.GivenName>Devin</v1.Person.GivenName>
          <v1.Person.MiddleInitial>R.</v1.Person.MiddleInitial>
          <v1.Person.Surname>Smith</v1.Person.Surname>
         </v1.PersonName>
        </v1.Cust.Related.PersonName>
       </v1.Cust.RelatedTraveler>
        <v1.Cust.RelatedTraveler Relation="Child" DynSeq="1">
         <v1.UniqueId>
          <v1.ResourceId>7005-548-704</v1.ResourceId>
         </v1.UniqueId>
         <v1.Cust.Related.PersonName>
          <v1.PersonName>
           <v1.Person.NameTitle>Ms.</v1.Person.NameTitle>
           <v1.Person.GivenName>Amy</v1.Person.GivenName>
           <v1.Person.MiddleInitial>E.</v1.Person.MiddleInitial>
           <v1.Person.Surname>Smith</v1.Person.Surname>
          </v1.PersonName>
         </v1.Cust.Related.PersonName>
        </v1.Cust.RelatedTraveler>
       </v1.Customer>
      </v1.CustProfile>
     </v1.Add>
   </v1.CustProfileUpdateRQ>
  </v1.Content>
</OTA_v1>
```

Update message response

```
<?xml version="1.0"?>
<!DOCTYPE OTA_v1 SYSTEM "OTA_v1.dtd">
<OTA_v1 Timestamp="2000-11-08T09:33:05" EchoToken="22334461" Target="Production"
GeoCode="+3018-9742">
 <v1.Control>
  <v1.SendBy Identity="LoneStarAir" URI="http://ota.LoneStarAir.com">
   <v1.Credential Method="JunkYardDog">bvc098cxz</v1.Credential>
  </v1.SendBy>
 </v1.Control>
 <v1.Content>
  <v1.CustProfileUpdateRS Success="Yes">
   <v1.UniqueId>
    <v1.TradingPartner>
     <v1.CompanyCode CodeContext="FedTaxID">654-987123478
     </v1.CompanyCode>
    </v1.TradingPartner>
    <v1.ResourceId>XYA3:ConX_Smith/George(0)</v1.ResourceId>
   </v1.UniqueId>
   <v1.CustProfile ShareAllMarketInd="No" ShareAllSynchInd="Yes">
    <v1.Customer>
     <v1.Cust.PersonName NameType="Default">
      <v1.PersonName>
       <v1.Person.NameTitle> Mr. </v1.Person.NameTitle>
       <v1.Person.GivenName> George </v1.Person.GivenName>
       <v1.Person.MiddleInitial> A. </v1.Person.MiddleInitial>
       <v1.Person.Surname> Smith</v1.Person.Surname>
      </v1.PersonName>
     </v1.Cust.PersonName>
     <v1.Cust.Telephone PhoneTech="Voice" PhoneUse="Work" DynSeq="0">
      <v1.Telephone>
       <v1.Phone.AreaCityCode>253</v1.Phone.AreaCityCode>
       <v1.Phone.Number>813-8532</v1.Phone.Number>
      </v1.Telephone>
     </v1.Cust.Telephone>
     <v1.Cust.Telephone PhoneTech="Fax" PhoneUse="Work" DynSeq="1">
      <v1.Telephone>
       <v1.Phone.AreaCityCode>253</v1.Phone.AreaCityCode>
       <v1.Phone.Number>813-8698</v1.Phone.Number>
      </v1.Telephone>
     </v1.Cust.Telephone>
     <v1.Cust.PaymentForm>
      <v1.Cust.Pay.CreditCard CardType="Credit"
        PayFormReferName="WorkMastercard"
        CardNumber="M:4356 ... 5677" CardExpiryDate="2001-12-31">
       </v1.Cust.Pay.CreditCard>
      <v1.Cust.Pay.DirectBill DirectBill_Id="323453"
       PayFormReferName="ConX_Invoice">
       <v1.Cust.Pay.Direct.CompanyName>ConXTravel Co
       </v1.Cust.Pay.Direct.CompanyName>
       <v1.Cust.Pay.Direct.Address>
        <v1.Address>
         <v1.Addr.StreetNmbr POBox="4321-01">1200 Yakima St
         </v1.Addr.StreetNmbr>
         <v1.Addr.BldgRoom>Suite 800</v1.Addr.BldgRoom>
         <v1.Addr.CityName>Seattle</v1.Addr.CityName>
         <v1.Addr.StateProv PostalCode="98108">WA
         </v1.Addr.StateProv>
         <v1.Addr.CountryName>USA</v1.Addr.CountryName>
        </v1.Address>
       </v1.Cust.Pay.Direct.Address>
      </v1.Cust.Pay.DirectBill>
     </v1.Cust.PaymentForm>
     <v1.Cust.RelatedTraveler Relation="Child" DynSeq="0">
      <v1.UniqueId>
       <v1.ResourceId>7005-548-703</v1.ResourceId>
      </v1.UniqueId>
      <v1.Cust.Related.PersonName>
       <v1.PersonName>
        <v1.Person.NameTitle>Mr.</v1.Person.NameTitle>
        <v1.Person.GivenName>Devin</v1.Person.GivenName>
        <v1.Person.MiddleInitial>R.</v1.Person.MiddleInitial>
        <v1.Person.Surname>Smith</v1.Person.Surname>
       </v1.PersonName>
      </v1.Cust.Related.PersonName>
     </v1.Cust.RelatedTraveler>
     <v1.Cust.RelatedTraveler Relation="Child" DynSeq="1">
      <v1.UniqueId>
       <v1.ResourceId>7005-548-704</v1.ResourceId>
      </v1.UniqueId>
      <v1.Cust.Related.PersonName>
       <v1.PersonName>
        <v1.Person.NameTitle>Ms.</v1.Person.NameTitle>
        <v1.Person.GivenName>Amy</v1.Person.GivenName>
        <v1.Person.MiddleInitial>E.</v1.Person.MiddleInitial>
        <v1.Person.Surname>Smith</v1.Person.Surname>
       </v1.PersonName>
      </v1.Cust.Related.PersonName>
     </v1.Cust.RelatedTraveler>
    </v1.Customer>
   </v1.CustProfile>
  </v1.CustProfileUpdateRS>
 </v1.Content>
</OTA_v1>
```

Note: This Response with the object (CustProfile) could show additional changes that we did not Verify or change. You can find more Update examples in Appendix 2.

3.5.5 Delete verb

The Delete infrastructure verb defines an operation that identifies an existing customer profile, and removes the entire record from the database. The requestor MAY also verify the record before deleting it. Steps in the Delete operation include:

- Requestor submits a ReadAll request to view the record
- Receiver returns the record for the requestor to view
- Requestor submits a Delete request, and optionally verifies the record. As in the Update process, the Verify sub-action includes a sub-set of the record.
- Receiver removes the record and returns an acknowledgement

Examples 7 and 8 illustrate the Delete operations.

Example 7. Delete process, ReadAll request

```
<?xml version="1.0"?>
<!DOCTYPE OTA_v1 SYSTEM "OTA_v1.dtd">
<OTA_v1 Timestamp="2001-11-18T13:30:15" EchoToken="99887" Target="Production"
GeoCode="+3851-7702">
 <v1.Control>
  <v1.Session Identity="XYZ Co" URI="http://ota.xyz.com">
   <v1.Credential Method="DeepSecret">asdfghjkl</v1.Credential>
    <v1.Principal Identity="Scott9">
   </v1.Principal>
  </v1.Session>
 </v1.Control>
 <v1.Content>
  <v1.CustProfileReadAllRQ>
   <v1.UniqueId>
    <v1.TradingPartner>
     <v1.CompanyCode CodeContext="DUNS"> 876543219 </v1.CompanyCode>
    </v1.TradingPartner>
    <v1.ResourceId ResourceContext = "AAA_Member">050-7035487-005
    </v1.ResourceId>
   </v1.UniqueId>
  </v1.CustProfileReadAllRQ>
 </v1.Content>
</OTA_v1>
```

Example 7. Delete process, ReadAll response

```
<?xml version="1.0"?>
<!DOCTYPE OTA_v1 SYSTEM "OTA_v1.dtd">
<OTA_v1 Timestamp="2001-11-18T13:30:20" EchoToken="99887" Target="Production"
GeoCode="+4256-7844">
 <v1.Control>
  <v1.SendBy Identity="123 Co" URI="http://ota.onetwothree.com">
   <v1.Credential Method="RealPKI">poiuytr</v1.Credential>
  </v1.SendBy>
 </v1.Control>
 <v1.Content>
  <v1.CustProfileReadAllRS Success="Yes">
   <v1.UniqueId>
    <v1.TradingPartner>
     <v1.CompanyCode CodeContext="DUNS"> 876543219 </v1.CompanyCode>
    </v1.TradingPartner>
    <v1.ResourceId ResourceContext="AAA_Member">050-7035487-005
    </v1.ResourceId>
   </v1.UniqueId>
   <v1.CustProfile ShareAllMarketInd="No" ShareAllSynchInd="Yes">
    <v1.Customer Gender="Female">
     <v1.Cust.PersonName DefaultInd="Yes">
      <v1.PersonName>
       <v1.Person.NameTitle>Ms.</v1.Person.NameTitle>
       <v1.Person.GivenName>Phoebe</v1.Person.GivenName>
       <v1.Person.MiddleName>Peabody</v1.Person.MiddleName>
       <v1.Person.Surname>Beebe</v1.Person.Surname>
```

```
          </v1.PersonName>
         </v1.Cust.PersonName>
         <v1.Cust.Telephone DefaultInd="Yes" PhoneTech="Voice" PhoneUse="Work">
          <v1.Telephone CountryAccessCode="1">
           <v1.Phone.AreaCityCode>716</v1.Phone.AreaCityCode>
           <v1.Phone.Number>555-9999</v1.Phone.Number>
          </v1.Telephone>
         </v1.Cust.Telephone>
         <v1.Cust.Email EmailReferName="WorkEmail">PhoebePeabody@Beebe.Com
          </v1.Cust.Email>
        </v1.Customer>
        <v1.Preferences>
         <v1.Pref.Collection>
          <v1.Pref.Common SmokingInd="No">
           <v1.Pref.Common.InterestName>Cross country skiing
           </v1.Pref.Common.InterestName>
           <v1.Pref.Common.MealType PreferLevel="Only">Vegetarian
           </v1.Pref.Common.MealType>
           <v1.Pref.Common.FavoriteFood PreferLevel="Unacceptable">Beef
           </v1.Pref.Common.FavoriteFood>
           <v1.Pref.Common.MediaEntertain>New York Times
           </v1.Pref.Common.MediaEntertain>
           <v1.Pref.Common.Beverage>Skim milk</v1.Pref.Common.Beverage>
           <v1.Pref.Common.PetInfo>Dog, Murray</v1.Pref.Common.PetInfo>
          </v1.Pref.Common>
         </v1.Pref.Collection>
        </v1.Preferences>
       </v1.CustProfile>
      </v1.CustProfileReadAllRS>
     </v1.Content>
    </OTA_v1>
```

Example 8. Delete request

```
<?xml version="1.0"?>
<!DOCTYPE OTA_v1 SYSTEM "OTA_v1.dtd">
<OTA_v1 Timestamp="2001-11-18T13:30:25" EchoToken="99889" Target="Production"
GeoCode="+3851-7702">
 <v1.Control>
  <v1.Session Identity="XYZ Co" URI="http://ota.xyz.com">
   <v1.Credential Method="DeepSecret">asdfghjkl</v1.Credential>
   <v1.Principal Identity="Scott9">
   </v1.Principal>
  </v1.Session>
 </v1.Control>
 <v1.Content>
  <v1.CustProfileDeleteRQ>
   <v1.UniqueId>
    <v1.TradingPartner>
     <v1.CompanyCode CodeContext="DUNS"> 876543219 </v1.CompanyCode>
    </v1.TradingPartner>
    <v1.ResourceId ResourceContext="AAA_Member">050-7035487-005
    </v1.ResourceId>
   </v1.UniqueId>
   <v1.Verify>
    <v1.CustProfile ShareAllMarketInd="No" ShareAllSynchInd="Yes">
     <v1.Customer Gender="Female">
```

```
       <v1.Cust.PersonName DefaultInd="Yes">
        <v1.PersonName>
         <v1.Person.NameTitle>Ms.</v1.Person.NameTitle>
         <v1.Person.GivenName>Phoebe</v1.Person.GivenName>
         <v1.Person.MiddleName>Peabody</v1.Person.MiddleName>
         <v1.Person.Surname>Beebe</v1.Person.Surname>
        </v1.PersonName>
       </v1.Cust.PersonName>
       <v1.Cust.Telephone DefaultInd="Yes" PhoneTech="Voice" PhoneUse="Work">
        <v1.Telephone CountryAccessCode="1">
         <v1.Phone.AreaCityCode>716</v1.Phone.AreaCityCode>
         <v1.Phone.Number>555-9999</v1.Phone.Number>
        </v1.Telephone>
       </v1.Cust.Telephone>
       <v1.Cust.Email EmailReferName="WorkEmail">
         PhoebePeabody@Beebe.Com</v1.Cust.Email>
      </v1.Customer>
     </v1.CustProfile>
    </v1.Verify>
   </v1.CustProfileDeleteRQ>
  </v1.Content>
</OTA_v1>
```

Example 8. Delete response

```
<?xml version="1.0"?>
<!DOCTYPE OTA_v1 SYSTEM "OTA_v1.dtd">
<OTA_v1 Timestamp="2001-11-18T13:30:30" EchoToken="99889" Target="Production"
GeoCode="+4256-7844">
 <v1.Control>
  <v1.SendBy Identity="123 Co" URI="http://ota.onetwothree.com">
   <v1.Credential Method="RealPKI">poiuytr</v1.Credential>
  </v1.SendBy>
 </v1.Control>
 <v1.Content>
  <v1.CustProfileDeleteRS Success="Yes">
   <v1.UniqueId>
    <v1.TradingPartner>
     <v1.CompanyCode CodeContext="DUNS"> 876543219 </v1.CompanyCode>
    </v1.TradingPartner>
    <v1.ResourceId ResourceContext="AAA_Member">050-7035487-005
    </v1.ResourceId>
   </v1.UniqueId>
  </v1.CustProfileDeleteRS>
 </v1.Content>
</OTA_v1>
```

4 Message Structure

4.1 Root elements and tags

Each OTA version 1 message MUST begin with the tag `<OTA_v1>` and end with the tag `</OTA_v1>` .

4.1.1 Root element attributes

The OTA_v1 root element MAY contain the following optional attributes:

- Timestamp: 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.

  Example: 29 May 2001, 2:45:30 p.m., becomes 2001-05-29T14:45:30

- GeoCode: defined as the latitude and longitude of the server generating the timestamp, expressed in notation specified by ISO standard 6709. See http://www.ftp.uni-erlangen.de/pub/doc/ISO/english/ISO-6709-summary for a summary of the notation prescribed by the standard.

  Universal Time (UTC) notation does not deal well with 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). Thus OTA added the GeoCode attribute to indicate longitude and latitude, using ISO 6709.

- Target: indicates if the message is a test or production message, with a default value of Production

- EchoToken: sequence number 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.

An example of the root tag with optional attributes follows:

Date and time message created = 22 May 2000, 11:23:09
Echo Token = 4321
Production (not a test) message
GeoCode (Chicago) = 41° 47' N 87° 45' W

```
<OTA_v1 Timestamp = "2000-05-22T11:23:09" EchoToken = "4321"
         Target = "Production" GeoCode = "+4147-8745/">
```

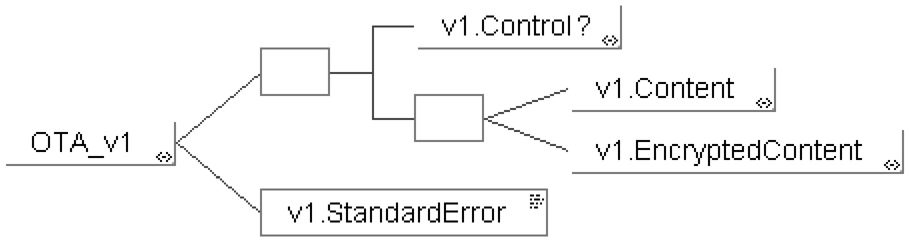
4.2 Control and Content

Most OTA version 1 messages will have one Control and one Content section, or a version 1 "Standard Error". If the OTA version 1 message is a non-standard error message, and has a Control section, the Control section MUST precede the Content section. The Control section in the OTA version 1 specification is used mostly for security purposes.

The Control element MUST begin with the tag `<v1.Control>` and end with the tag `</v1.Control>`. The Content element MUST begin with the tag `<v1.Content>` and end with tag `</v1.Content>`.

Figure 4 presents a diagram showing the relationship between the root tag (OTA_v1) and Control, Content, and StandardError elements.

Figure 4. Relationship of root tag to v1.Control, v1.Content, and v1.StandardError



4.2.1 Control section[3]

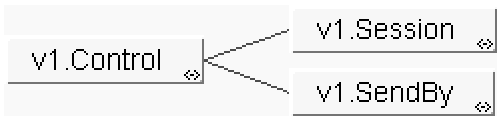
The OTA 1 specification defines two security models: Session-based, and Non-Session-based. The Control section, which is used mostly for security, will contain the elements needed to identify and authenticate the parties, and MUST contain one of the following elements:

- Session-based (tag v1.Session) for use with transactions 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 disjoint systems within an enterprise.
- Non-Session-based (tag v1.SendBy) for use with transactions 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.

Figure 5 shows the hierarchy for Control and Session or SendBy elements.

[3] We thank Scott Hinkelman of IBM Corp. for drafting this section.

Figure 5. Relationship of v1.Control to v1.Session and v1.Sendby



4.2.1.1 The Session-based Security Model

Sessioned 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) security model.

In OTA security, the word “Subject” can be used to refer to a client, or caller.

An OTA Session contains a variety of structures such as “Principals” and “Credentials”. A Session MUST have an Identity attribute. In fact, this is all that is required of an OTA Session. A minimum OTA Session follows:

```
<OTA_v1>
 <v1.Control>
   <v1.Session Identity="Tim3"/>
 </v1.Control>
 <v1.Content/>
     . . .
  </v1.Content>
</OTA_v1>
```

A Session MAY contain multiple Credentials, but typically contains only one. 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.

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:

```
<OTA_v1>
 <v1.Control>
   <v1.Session Identity="Tim3" URI="122999_1156AM_5555-Our Web Portal">
    <v1.Credential Method="Base64">676868777</v1.Credential>
    <v1.Credential Method="MD32">44422</v1.Credential>
   </v1.Session>
 </v1.Control>
 <v1.Content/>
     . . .
 </v1.Content>
</OTA_v1>
```

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:

```
<OTA_v1>
 <v1.Control>
   <v1.Session Identity="Tim3" URI="Web Portal primary contact system">
     <v1.Credential Method="Base64">676868777</v1.Credential>
     <v1.Credential Method="MD32">44422</v1.Credential>
       <v1.Principal Identity="TimC" URI="FFProgram1">
         <v1.Credential Method="RAS32">FF569yt</v1.Credential>
       </v1.Principal>
       <v1.Principal Identity="67ttt" URI="Vacation Program">
         <v1.Credential Method="Clear Text">mypassword</v1.Credential>
       </v1.Principal>
   </v1.Session>
 </v1.Control>
 <v1.Content/>
   . . .
 </v1.Content>
</OTA_v1>
```

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 could not include a specific enumerated list of methods in the DTD.

Further examples of the Session-based security model follow.

4.2.1.1.1 Subjects must become authenticated

Subjects (clients) must first attempt to become authenticated when using the Session-based security model. Subjects become authenticated by sending a v1.AuthenticationRQ to an OTA trading partner. For this message, no Control section is required in the message. An AuthenticationRQ structure is generated in the Content section. The Subject is responsible for providing a single Principal within the request. In the following example, the Subject provides a Principal that 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 OTA_v1 SYSTEM "OTA_v1.dtd">
<OTA_v1>
 <v1.Content>
  <v1.AuthenticationRQ>
   <v1.Principal Identity="Tim3">
    <v1.Credential Method="Clear Text">SPOT
    </v1.Credential>
   </v1.Principal>
  </v1.AuthenticationRQ>
 </v1.Content>
</OTA_v1>
```

Given no errors, the OTA trading partner returns a v1.AuthenticationRS message with a Success attribute = "Yes". A Session is returned to the Subject in the message's Control section. 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 OTA_v1 SYSTEM "OTA_v1.dtd">
<OTA_v1>
 <v1.Control>
  <v1.Session Identity="Tim3">
   <v1.Credential Method="Base64">6785533</v1.Credential>
  </v1.Session>
 </v1.Control>
 <v1.Content>
  <v1.AuthenticationRS Success="Yes"/>
 </v1.Content>
</OTA_v1>
```

If the OTA trading partner does not support the Session-based security model a v1.StandardError is returned containing the “AuthenticationModel” ErrType attribute as follows:

```
<OTA_v1>
     <v1.StandardError ErrType="AuthenticationModel"/>
</OTA_v1
```

If the Subject’s provided Principal fails authentication, a v1.StandardError is returned containing the “Authentication” ErrType attribute as follows:

```
<OTA_v1>
 <v1.StandardError ErrType="Authentication"/>
</OTA_v1>
```

4.2.1.1.2 Subjects pass the Session on business calls

Once a Subject (client) has become authenticated, the Subject passes the Session, 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.

```
<v1.Control>
      <v1.Session Identity="XYZ Co" URI="http://OTA.XYZ.COM">
         <v1.Credential Method="RC4-SHA">foiu34iuhioaugho34uhgip
          </v1.Credential>
         <v1.Principal Identity="Scott9">
            <v1.Credential Method="RC4-SHA">3fff678sddd850</v1.Credential>
         </v1.Principal>
         <v1.Principal Identity="SRH78" URI="BackendFrequentFlyerProgram1">
            <v1.Credential>7777477777sd88888</v1.Credential>
         </v1.Principal>
      </v1.Session>
  </v1.Control>
  <v1.Content>
      ...
  </v1.Content>
```

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 Content in 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.

```
<v1.Control>
   <v1.Session Identity="XYZ Co">
   <v1.Principal Identity="Scott9">
    <v1.Credential Method="RC4-SHA">3fff678sddd850</v1.Credential>
   </v1.Principal>
 </v1.Session>
</v1.Control>
<v1.EncryptedContent Method="EXP-RC4-MD5">
 <v1.Block>foiu34iuhioaugho34uhgip</v1.Block>
 ...
</v1.EncryptedContent>
```

C. Session with Session-level Identity and URI, with one Principal

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 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.

```
   <v1.Control>
      <v1.Session Identity="XYZ Co" URI="http://OTA.XYZ.COM">
        <v1.Principal Identity="Scott9">
           <v1.Credential Method="RC4-SHA">3fff678sddd850</v1.Credential>
        </v1.Principal>
      </v1.Session>
   </v1.Control>
   <v1.Content>
      ...
   </v1.Content>
```

4.2.1.1.3 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:

```
<OTA_v1>
  <v1.StandardError ErrType="AuthenticationTimeout"/>
</OTA_v1>
```

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.

```
<OTA_v1>
 <v1.Control>
   <v1.Session>
   . . .
  </v1.Session>
  </v1.Control>
 <v1.Content>
   <v1.ReAuthenticationRQ/>
 </v1.Content>
</OTA_v1>
```

If the OTA trading partner does not support the Session-based security model a v1.StandardError SHOULD be returned containing the “AuthenticationModel” ErrType attribute as follows:

```
<OTA_v1>
 <v1.StandardError ErrType="AuthenticationModel"/>
</OTA_v1
```

If the Subject’s provided Session fails authentication, a v1.StandardError is returned containing the “Authentication” ErrType attribute as follows:

```
<OTA_v1>
 <v1.StandardError ErrType="Authentication"/>
</OTA_v1
```

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. An example follows that does not use Principals.

```
<?xml version="1.0"?>
<!DOCTYPE OTA_v1 SYSTEM "OTA_v1.dtd">
<OTA_v1>
 <v1.Control>
  <v1.Session Identity="Tim3">
   <v1.Credential Method="Base64">6785533</v1.Credential>
  </v1.Session>
 </v1.Control>
 <v1.Content>
  <v1.ReAuthenticationRS Success="Yes"/>
 </v1.Content>
</OTA_v1>
```

4.2.1.1.4 Unauthentication

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

```
<OTA_v1>
  <v1.Control>
    <v1.Session>
    . . .
   </v1.Session>
  </v1.Control>
  <v1.Content>
    <v1.UnAuthenticationRQ/>
  </v1.Content>
</OTA_v1>
```

If the OTA trading partner does not support the Session-based security model a v1.StandardError is returned containing the “AuthenticationModel” ErrType attribute as follows:

```
<OTA_v1>
 <v1.StandardError ErrType="AuthenticationModel"/>
```

```
</OTA_v1
```

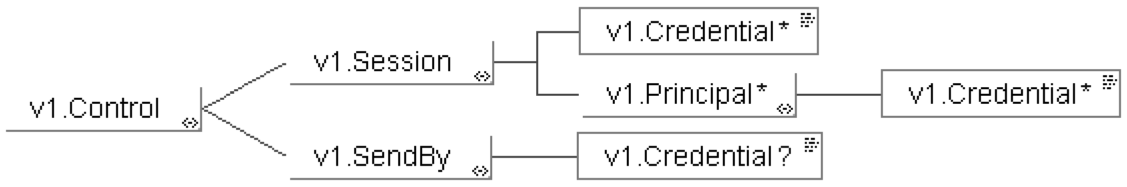
If the Subject's provided Session has never been authenticated from the target trading partner, a v1.StandardError is returned containing the "Authentication" ErrType attribute as follows:

```
<OTA_v1>
 <v1.StandardError ErrType="Authentication"/>
</OTA_v1
```

4.2.1.2 The Non-session-based (SendBy) Security Model

In the Non-Session-based 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. See Figure 6 for a diagram of the hierarchy for these elements.

Figure 6. Relationship among v1.Session, v1.Principal, SendBy, and v1.Credential tags



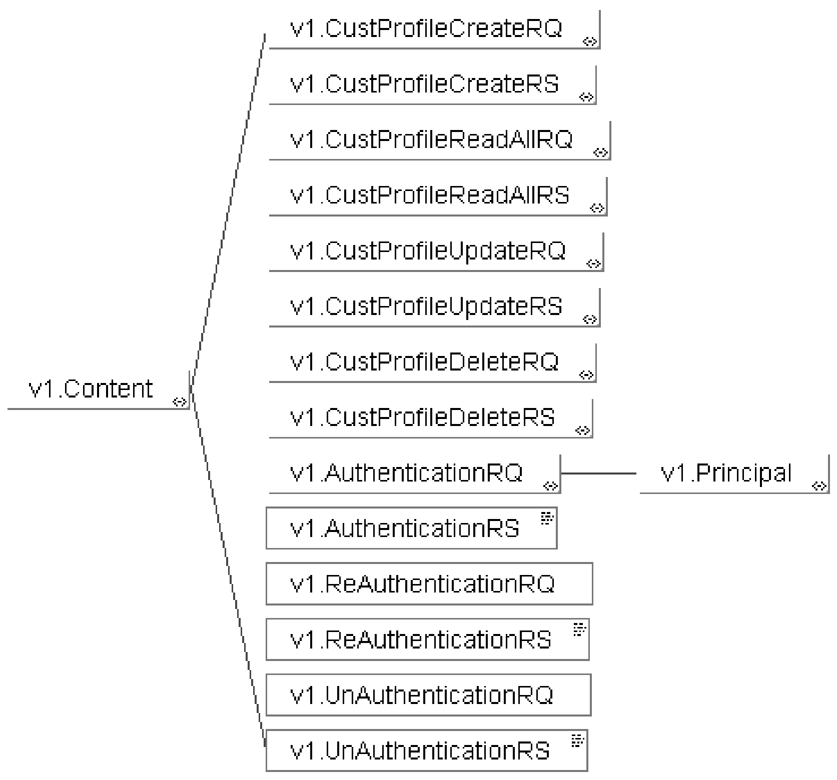
If the OTA trading partner supports only the Session-based security model and NOT the SendBy method discussed above a v1.StandardError is returned containing the "AuthenticationModel" ErrType attribute as follows:

```
<OTA_v1>
 <v1.StandardError ErrType="AuthenticationModel"/>
</OTA_v1
```

4.2.2 The Content section

The Content section identifies the message as a request or response and contains the actions required by the receiver to process the message, as well as the business data exchanged between trading partners. It also contains the Authentication, ReAuthentication, and UnAuthentication elements described above. Figure 7 shows the major elements and tag names in the Content section.

Figure 7. v1.Content with Major Verb Request/Response Tags



4.2.2.1 Non-sessioned security-related elements in Control and Content sections

As mentioned above in section 4.2.1 the use of specific security-related elements and attributes in the Control will depend on the use of encryption in the Content. Given below are examples for several scenarios with different combinations of sessions and single log-ons as well as credentials and encryption in the Content. We thank George Smith of ConXtra and Scott Hinkelman of IBM for preparing these examples.

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

In this scenario, the sender, indicated by SendBy, uses both the Identity and URI attributes, as well as Credential in the Control

```
 <v1.Control>
      <v1.SendBy Identity="XYZ Co" URI="http://OTA.XYZ.COM">
        <v1.Credential Method="RC4-SHA">foiu34iuhioaugho34uhgip
         </v1.Credential>
      </v1.SendBy>
</v1.Control>
<v1.Content>
     ...
</v1.Content>
```

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

In this scenario, the sender, indicated by SendBy, uses both the Identity and URI attributes, but carries the equivalent of the Credential for authentication in the EncryptedContent element.

```
 <v1.Control>
     <v1.SendBy Identity="XYZ Co" URI="http://OTA.XYZ.COM" />
 </v1.Control>
 <v1.Content>
    <v1.EncryptedContent >
      <v1.Block Method="EXP-RC4-MD5">foiu34iuhioaugho34uhgip</v1.Block>
        ...
    </v1.EncryptedContent>
</v1.Content>
```

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, in either the Control or EncryptedContent.

```
  <v1.Control>
     <v1.SendBy Identity="XYZ Co" />
  </v1.Control>
  <v1.Content>
      ...
  </v1.Content>
```

4.2.2.2 Major verb tags

OTA version 1 uses a request and response model for its messages. This approach means every message generates a corresponding exchange that helps manage the data flow. The major verbs as described in section 3.5 include the RQ and RS suffixes to indicate request and response message order. These tags use the following naming convention:

<v1.CustProfile*MajorVerb*R*X*>

where, MajorVerb = Create, ReadAll, Update, or Delete
R*X* indicates the Request (RQ) or Response (RS) suffix

and are listed below:

```
<v1.CustProfileCreateRQ>
<v1.CustProfileCreateRS>

<v1.CustProfileReadAllRQ>
<v1.CustProfileReadAllRS>

<v1.CustProfileUpdateRQ>
<v1.CustProfileUpdateRS>

<v1.CustProfileDeleteRQ>
```

```
<v1.CustProfileDeleteRS>
```

Figure 7 in section 4.2.2 shows the relationship of the main verbs to the v1.Content tag.

4.2.2.3 Success attribute

The Response verb elements (CustProfile*xxx*RS) SHOULD return a Success attribute with enumerated values of Yes, No, or Part for partial. The default value for this attribute is Yes.

4.2.2.4 Sub-action verb tags

The following tag names are valid for sub-actions undertaken as part of the Update verb, as described in section 3.5.4. The Verify element is also an option of the Delete function outlined in 3.5.5.

```
<v1.Verify>
<v1.Remove>
  <v1.Set>
  <v1.Insert>
  <v1.Add>
```

These verbs MUST appear within the major verb request and response tag sets. See section 3.5, Infrastructure Verbs for further definition and specification of these actions as well as examples.

4.2.3 EncryptedContent section

If the trading partners decide to encrypt the Content section of the message, the Content section MUST have an EncryptedContent element instead of the Content element. The EncryptedContent element by definition will have scrambled content and therefore will NOT follow the content model of the Content element. The receiver of the message will need to process the contents of the EncryptedContent element after decryption.

The EncryptedContent element MUST contain a Method attribute to describe the encryption method used by the sender. See section 4.2.1.1 that discusses this attribute as part of the Credential element. The EncryptedContent element MUST contain a repeating temporary Block element with the tag `<v1.Block>` to identify segments of the decrypted Content section. The Block element provides a way for breaking up decrypted data into manageable segments. Without the Block element, an EncryptedContent could return a message that, depending on the encryption technique, the receiver would find difficult to parse.

Because the Block element is temporary and not used for any other purpose, senders SHOULD NOT apply a dynamic sequence (DynSeq) attribute to this element. The DynSeq attribute is used to identify repeating elements and described in the section on Updates (3.5.4) above.

4.2.3.1 Examples of Content and Control section tags

Examples 9 and 10 illustrate high-level use of the tags in the Content and Control sections described up to this point.

Example 9. Session-based v1.Control and v1.Content

```
<?xml version="1.0"?>
<!DOCTYPE OTA_v1 SYSTEM "OTA_v1.dtd">
<OTA_v1 Timestamp = "2001-10-08T21:10:30" EchoToken = "87654"
Target = "Production" GeoCode = "+3851-7702">
 <v1.Control>
  <v1.Session Identity = "XYZ Co">
   <v1.Principal Identity = "Scott9" URI = "http://ota.xyz.com">
    <v1.Credential Method = "DeepSecret">asdfghjkl</v1.Credential>
   </v1.Principal>
  </v1.Session>
 </v1.Control>
 <v1.Content>
  <v1.CustProfileReadAllRQ>
   <v1.UniqueId>
    <v1.TradingPartner>
     <v1.CompanyCode CodeContext = "DUNS">876543219 </v1.CompanyCode>
    </v1.TradingPartner>
    <v1.ResourceId ResourceContext = "AAA">050-3854930-00</v1.ResourceId>
   </v1.UniqueId>
  </v1.CustProfileReadAllRQ>
 </v1.Content>
</OTA_v1>
```

Example 10. Non-sessioned (v1.SendBy) v1.Control and v1. EncryptedContent

```
<?xml version="1.0"?>
<!DOCTYPE OTA_v1 SYSTEM "OTA_v1.dtd">
<OTA_v1 Timestamp = "2001-10-08T21:10:35" EchoToken = "87654"
Target = "Production" GeoCode = "+4256-7844">
 <v1.Control>
  <v1.SendBy Identity = "XYZ Co" URI = "http://ota.xyz.com">
   <v1.Credential Method = "DeepSecret">asdfghjkl</v1.Credential>
  </v1.SendBy>
 </v1.Control>
 <v1.EncryptedContent Method = "RealPKI">
  <v1.Block>zxcvbnm</v1.Block>
 </v1.EncryptedContent>
</OTA_v1>
```

4.3 Standard versioned error messages[4]

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. The specification also defines non-versioned error messages that result from the parser, before reaching the server. The versioned standard errors will follow the same structure as the non-versioned standard messages, as presented in chapter 6 that gives the details on the overall structure. This section will outline specific requirements for the versioned error messages.

[4] Our thanks go to Scott Hinkelman of IBM Corp. and George Smith of ConXtra for preparing this section.

Versioned error messages are represented in the v1.StandardError element, an element under the root (OTA_v1) tag; see Figure 4 in section 4.2 .

The StandardError element MUST contain the ErrType attribute that uses an evolving enumeration to indicate the error type. An evolving enumeration is one where the validating DTD's enumeration list evolves (by adding, and only adding, new acceptable values), as such, the receiving application can expect to accept values that it has NOT been explicitly coded for and process them in an acceptable way. This is facilitated by already accepting an ErrType of "Unknown". The initial enumeration list MUST contain:

- Unknown. Indicates an unknown error. Additional information may be provided within the PCDATA.
- 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 standard error or a more specific upper level content error. 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.
- 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 provided within the PCDATA.

Version standard error messages SHOULD use the following syntax:

```
<v1.StandardError Status ="…" ErrType="…" DocURL="…">Additional Information
</v1.StandardError>
```

All OTA message requests MAY result in a response message that consists of a StandardError construct alone, directly within the OTA_v1 root. For example, a date field that reflects an air availability departure date can NOT be a past date. Depending on the parser used by a particular implementation, this error could be returned in any of the following ways:

- v1.StandardError ErrType="ProtocolViolation"
- v1.StandardError ErrType="BizRule"
- Hi-level context related error

Examples of versioned StandardError messages (returned from *xxx*RQ messages) follow:

```
<OTA_v1>
      <v1.StandardError ErrType="Authentication” />
</OTA_v1>

<OTA_v1>
      <v1.StandardError ErrType="NoImplementation">Unsupported:
      SomeReadRQ</v1.StandardError>
</OTA_v1>

<OTA_v1>
      <v1.StandardError ErrType="BizRule">Midnight not allowed, tag
      DepTM</v1.StandardError>
</OTA_v1>
```

5 Customer profile content[5]

5.1 Introduction

OTA version 1 provides a set of common messages for transmitting customer profiles in the travel industry. This section of the specification presents the business content of the profile — the nouns — that the customers provide to travel services creating these profiles.

The specification divides the business data into three main sections:

- Customer information: 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

- Preferences: general and specific conditions to meet the needs of the customer for travel based on identifiable purposes, as defined by the customer, such as business trips, family vacations, golf outings, or the annual church retreat. Customers can also define 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.

- Affiliations: organizations with which the customer 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 individual's relationship to these organizations (e.g., employee identifier), not the policies of the companies themselves. However, customers may wish to reflect those policies in the Preferences section of their profiles.

A fourth smaller collection of elements covers a record of recent accesses to the profile.

The document type definition or DTD 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. OTA version 1 messages use the document type declaration (see section 3.3.1) to reference the DTD and match the messages against those rules. Messages meeting the rules in the DTD are called *valid* messages. Please note that validity in this case applies only to the structure of the message; DTDs would not catch an incorrect credit card number or misspelled name. Appendix 1 lists the DTDs used in OTA version 1, but they are also available in machine-readable form from the OTA web site: http://www.opentravel.org/ .

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. All of the OTA version 1 elements begin with a version prefix, indicated by v1. See section 3.4.1 on versioning.

[5] Our thanks go to Mary Manzer of United Airlines for leading development of the overall data model represented in this chapter.

In some cases the elements have identified specific properties or characteristics called *attributes*. A feature of attributes is they constrain the choices of permitted values in the elements. Many of the travel preferences, for example, allow for indicating a preference level of Only or Unacceptable. A value of "OK I guess" in this element would not be permitted.

A few of the attributes provide valuable management functions, such as the Dynamic Sequence or DynSeq attribute attached to repeating elements, as described in section on the Update process (3.5.4). The Update process requires unique identification of the elements in order to update the correct elements. Those elements that allow for repeating occurrences need to have unique identification for this process, and the DynSeq attribute assigns a sequence number to these elements as they occur. See the sample messages in the section 3.5.4 and the Appendix, such as the multiple telephones and related travelers that use the DynSeq attribute.

The AllChildElements attribute applies to all elements with sub-elements. The BefDynSeq and AftDynSeq attributes apply to all repeating elements, those indicated with DynSeq in the tables. See the update section (3.5.4) for a description of these attributes.

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* contained in the document type definition or DTD. Entities reduce the size of the DTD and allow for a more modular structure.

5.2 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. Because of this flexibility and the potential for sending empty elements, trading partners SHOULD transmit ONLY the data elements in the profile that are non-blank.

Empty elements can result from unrelated practices by companies using the profile. Systems may create placeholders, for example automatically creating v1.Customer, v1.Preferences, and v1.Affiliation under v1.CustProfile in anticipation of customers providing data for those elements. But in this scenario, if customers do not fill in v1.Preferences or v1.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.

5.3 Understanding the element group and element tables

Each of the element groups and elements follow in the remainder of this section. Here is a key for the symbols, abbreviations, and terms in the tables.

Element, name of the element used in XML tags

Occurrence (Occurc.)

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 often provide meaning
Any = Any content allowed

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

String = Alphanumeric plus special punctuation and symbol characters
Boolean = Returns a value of Yes or No
Date = Date format specified by ISO 8601
Blank = Not applicable (element content)

Please note that these data types are RECOMMENDED. The document type definitions to validate the XML exchanges use text string or enumeration data types.

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. Those attributes listed as Shared are used by more than one element and are described in their first occurrence. Under attribute data types, Enumerations (abbreviated Enum in the charts) MUST use the values listed. For string attributes, any values listed are RECOMMENDED.

Description, includes references to specific standard code lists.

5.3.1 Element and attributes in the data model

The XML 1.0 Recommendation from the W3C offers basic guidance, but hardly the final word, on the use of elements or attributes in a data model. In establishing the customer profile data model, OTA defined data items to maximize the use of attributes, since in the OTA model attributes do not include version or hierarchy. This liberal use of attributes will make it easier to convert the OTA data model to XML Schema, once the W3C approves it as a Recommendation. See section 3.3.1, Schema direction.

Data items meeting the following criteria qualified as attributes:

- Single occurrence
- No child elements
- No attributes
- Discrete code or limited text field

5.4 CustProfile

The CustProfile element covers the business content transmitted in OTA version 1 messages and includes identifiers and accesses to the record. CustProfile with the profile identifiers — TradingPartner, ResourceId, Resource Context — come under the main Content element as well as the sub-action verbs. CustProfile also includes the optional Accesses element that allows the profile to track creation date and time, last actions, and the name of the person who last accessed the profile. See Figure 3 in section 3.5.1 that shows the configuration of these elements.

5.4.1 Privacy attributes

Two sets of attributes allow for customers 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 CustProfile 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 become Yes.

The specification also gives customers control over sharing of major elements within the customer profile. The ShareSynchInd and ShareMarketInd attributes on the major elements provide for the customer 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 for that given element the permission granted to the higher levels in the hierarchy.

For example, a value of No in ShareAllMarketInd in CustProfile will apply to the entire profile, unless the customer indicates 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 CustProfile.

In another example, the customer indicates Yes in ShareAllSynchInd under CustProfile. If the customer wants to keep the travel service provider creating the profile from sharing payment information in the v1.Cust.PaymentForm element, even for synchronization purposes, the customer needs to give the ShareSynchInd attribute for that element a value of No.

Element	Occurc.	Content	Data type	Attributes	Description
CustProfile	R	Element		ShareSynchAllInd, ShareMarketAllInd, AllChildElements	Top-level element for customer profile

Attribute Name	Element	Data Type	Default	Values	Description
AllChildElements	Shared	Enum		Yes, No	Used in the Update process to indicate all subordinate elements in the structure are included. See section 3.5.4
ShareAllSynchInd	Shared	Enum	No	Yes, No	Permission for sharing all data in profile for synchronization of profiles held by other travel service providers
ShareAllMarketInd	Shared	Enum	No	Yes, No	Permission for sharing all data in profile for marketing information

Element	Occurc.	Content	Data type	Attributes	Description
Accesses	?	Element		ShareSynchInd, ShareMarketInd	Element to capture creation and last update data
Access.PersonName	?	Text	String		Name of individual who originated record
Access.LastAction	?	Text	String	Action, LastActionDate, LastActionTime, ActionSystemId	Describes the last action taken on the record
Access.Comment	*	Text	String	DynSeq, BefDynSeq, AftDynSeq	Free text description supplementing data in the previous elements

Attribute Name	Element	Data Type	Default	Values	Description
Action	Access.LastAction	Enum		Create, ReadAll, Update, Delete	Previous function undertaken on the profile
ActionSystemId	Access.LastAction	String			Identifier assigned to server that last accessed this profile
AllChildElements	Shared	Enum		Yes, No	Used in the Update process to indicate all subordinate elements in the structure are included. See section 3.5.4
CreateDate	Accesses	String			Month, day, year the profile originated, in ISO 8601 format
CreateTime	Accesses	String			Hour, minute, second of day the profile originated, in ISO 8601 format
DynSeq, BefDynSeq, AftDynSeq	Shared	String	0		Indicates the appearance sequence of repeating elements. Use with * or + elements, increments by 1. See section 3.5.4
LastActionDate	Access.LastAction				Month, day, year in ISO 8601 format of most recent action since Create: ReadAll, Update, Delete
LastActionTime	Access.LastAction				Hour, minute, second of day in ISO 8601 format of last action
ShareMarketInd	Shared	Enum	Inherit	Yes, No, Inherit	Permission for sharing data in element for marketing information
ShareSynchInd	Shared	Enum	Inherit	Yes, No, Inherit	Permission for sharing data in element for synchronization of profiles held by other travel service providers

5.5 Customer information

In OTA version 1, the Customer element and `<v1.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. Elements under v1.Customer begin with the prefix Cust. Figure 8 shows the element configuration under v1.Customer.

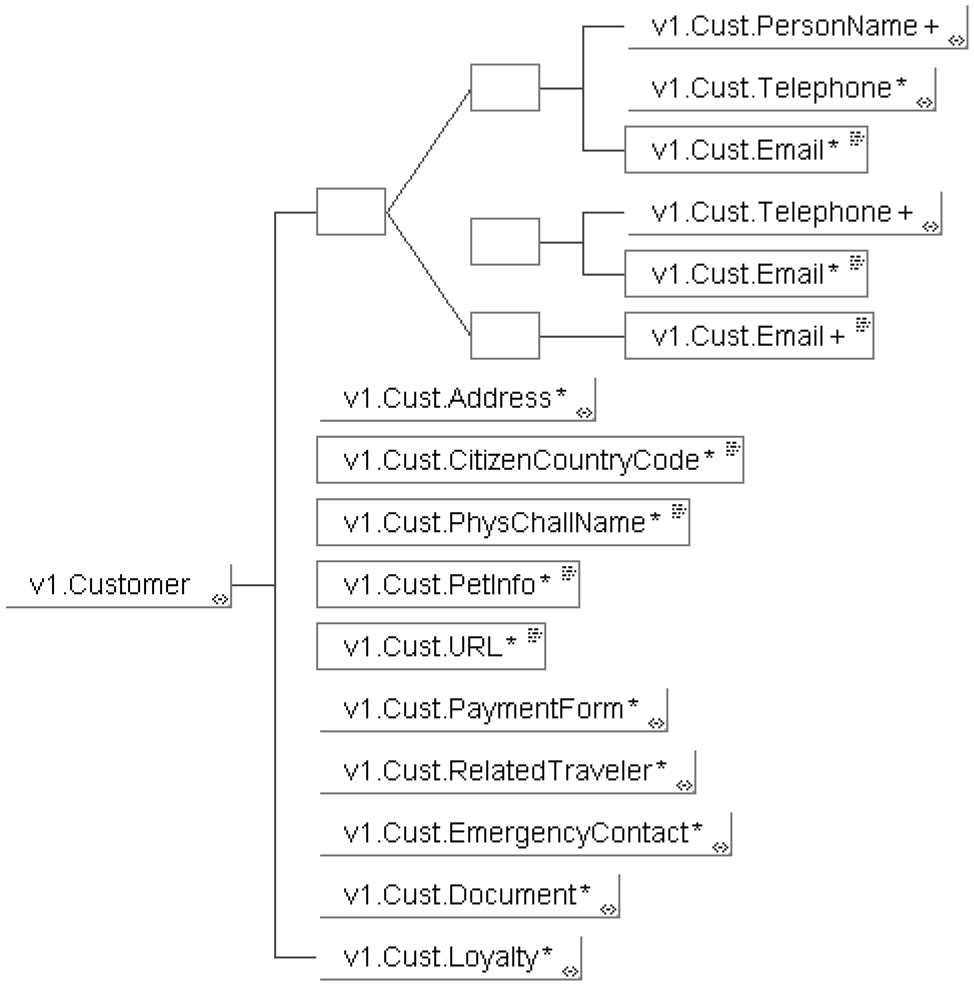
5.5.1 Required customer data

The specification defines a minimum set of data in a profile that fall under the Customer section. Each profile MUST have entries in at least one of the following elements:

- Cust.PersonName, tag `<v1.Cust.PersonName>`
- Cust.Telephone, tag `<v1.Cust.Telephone>`
- Cust.Email, tag `<v1.Cust.Email>`

Cust.PersonName and Cust.Telephone use the v1.PersonName and v1.Telephone objects that appear in several places throughout the profile. In the v1.PersonName object, and thus in Cust.PersonName, the v1.Person.Surname element is MANDATORY. Likewise, if used, the v1.Telephone object and thus the Cust.Telephone element, MUST have entries in the v1.Phone.AreaCityCode and v1.Phone.Number elements.

Figure 8. v1.Customer elements



5.5.2 Payment form and loyalty program elements and attributes

Several places in the profile use data on payment methods and loyalty programs, such as frequent flyer programs. Unlike reusable objects, such as v1.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. As a result, the OTA profile specification holds all payment data in the Cust.PaymentForm element and all loyalty program data in the Cust.Loyalty element, so that the customer's entries only need to appear once. Each of these elements has attributes for identifying the individual entries and therefore can be referenced in other parts of the profile, rather than repeated.

5.5.3 Customer elements and attributes

Element	Occurc.	Content	Data type	Attributes	Description
Customer	R	Element		AllChildElements ShareSynchInd, ShareMarketInd, Gender, Deceased, LockoutType, BirthDate,	Contains basic data on the customer's identity, finances, location,
Cust.PersonName	*	Element, Entity: PersonName		DynSeq, BefDynSeq, AftDynSeq, NameType, DefaultInd, ShareSynchInd, ShareMarketInd, ValidRepeatInd	Names of the individual, includes former, nicknames, and alternate names
Cust.Address	*	Element, Entity: Address		ShareSynchInd, ShareMarketInd, DynSeq, BefDynSeq, AftDynSeq, AddressType, ContactType, DefaultInd	Precise location of the individual in the profile for mailing and delivery
Cust.Telephone	*	Element, Entity: Telephone		ShareSynchInd, ShareMarketInd DefaultInd, ContactType, PhoneTech, PhoneUse, DynSeq, BefDynSeq, AftDynSeq, AllChildElements, ValidRepeatInd	Telephone numbers of the individual in the profile

Attribute Name	Element	Data Type	Default	Values	Description
NameType	Shared	String		Default, Former, Nickname, Alternate	Describes purpose of the name associated with the person
ContactType	Shared	String		Emergency, Home, Work, Affiliation, Arranger	Type of address or telephone given
AddressType	Shared	Enum		Delivery, Mailing, Billing, CreditCard	Describes the purpose of the address
PhoneTech	Shared	String		Voice, Data, Fax, Pager, Cell, TTY	Indicates type of technology associated with this telephone number
PhoneUse	Shared	String		Work, Home, Day, Night	Describes the general use or preferred time of day for telephone number listed
Gender	Shared	Enum		Male, Female	
Currency	Customer	String			Type of funds preferred for reviewing monetary values, ISO 4217 codes
Deceased	Customer	Enum	No	Yes, No	Notes if individual has died
LockoutType	Customer	String		Emergency, Incident	Indicates reason for locking out record
DefaultInd	Shared	Enum		Yes, No	The value that the receiving system should assume if the user specifies no overriding value or action.
ValidRepeatInd	Shared	Enum	No	Yes, No	Indicates if the appearance of the element is solely for repetition to satisfy validation requirements of the DTD.
BirthDate	Shared	String			Indicates self-professed date of birth, in ISO 8601 prescribed format

Element	Occurc.	Content	Data type	Attributes	Description
Cust.CitizenCountryCode	*	Text	AN	DefaultInd, DynSeq, BefDynSeq, AftDynSeq	Self-professed country or countries that customer claims for citizenship, ISO 3166 codes
Cust.PhysChallName	*	Text	AN	DynSeq, BefDynSeq, AftDynSeq	Any special physical conditions that can affect travel choices, including allergies
Cust.PetInfo	*	Text	String	DynSeq, BefDynSeq, AftDynSeq	Information about pets that may accompany customer while traveling
Cust.Email	*	Text	String	DefaultInd, DynSeq, BefDynSeq, AftDynSeq, ValidRepeatInd, EmailReferName	Electronic mail addresses, in IETF specified format
Cust.URL	*	Text	String	DefaultInd, DynSeq, BefDynSeq, AftDynSeq	Web site addresses, in IETF specified format

Element	**Occurc.**	**Content**	**Data type**	**Attributes**	**Description**
Cust.PaymentForm	*	Element		CostCenterId, DefaultInd, DynSeq, BefDynSeq, AftDynSeq, PreferLevel, AllChildElements, ShareSynchInd, ShareMarketInd	Ways of providing funds for travel by the individual

Element	**Occurc.**	**Content**	**Data type**	**Attributes**	**Description**
Cust.Pay.CreditCard	*	Element		CardType, PreferLevel, DefaultInd, PayFormReferName, BefDynSeq, AftDynSeq, DynSeq, ShareSynchInd, ShareMarketInd	Payment accounts authorizing purchases represented by plastic cards often with magnetic strips with data for identification
Cust.Pay.Card.PersonName	?	Text	String		Name string of individual as embossed on the card
Cust.Pay.Card.UserDefinedName	?	Text	String		Common-use name of the card, e.g., University of Iowa Alumni Mastercard
Cust.Pay.Card.IssueCompName	?	Text	String		Company or organization that issued the card
Cust.Pay.Card.Address	?	Entity: v1.Address	Element		

Element	Occurc.	Content	Data type	Attributes	Description
Cust.Pay.BankAcct	*	Element		AcctType, BankId, PreferLevel, DefaultInd, PayFormReferName, BefDynSeq, AftDynSeq, DynSeq, ShareSynchInd, ShareMarketInd	Customer bank accounts for payments, either for paper checks or electronic funds transfers
Cust.Pay.Bank.PersonName	*	Entity: v1.PersonName		BefDynSeq, AftDynSeq, DynSeq	Names of the individuals on the bank account
Cust.Pay.Bank.AcctNumber	?	Text	String		Identifier for the account assigned by the bank
Cust.Pay.DirectBill	*	Element		DirectBill_Id, PreferLevel, DefaultInd, PayFormReferName, BefDynSeq, AftDynSeq, DynSeq, ShareSynchInd, ShareMarketInd	Company name and location for sending invoice for remittances for travel services.
Cust.Pay.Direct.CompanyName	R	Text	String		Name of organization to where invoices are sent
Cust.Pay.Direct.Address	R	Entity: v1.Address			Precise mailing or delivery location to where invoices are sent

Attribute Name	Element	Data Type	Default	Values	Description
CostCenterId	Cust.PaymentForm	String			Code for allocating cost of travel to company accounts
PreferLevel	Shared	Enum		Only, Unacceptable	Identifies the customer's level of preference for the content of the element; if the attribute is absent assume a value of "preferred"
PayFormReferName	Shared	String			A unique name to identify the form of payment, and used elsewhere in the profile to refer back to this element.
AcctType	Cust.Pay.BankAcct	String		Checking, Savings, Investment	Describes the bank account used for financing travel
BankId	Cust.Pay.BankAcct	String			Code assigned by authorities to financial institutions; sometimes called bank routing number
CardNumber	Cust.Pay.CreditCard	String			Identifier embossed on card
CardExpiryDate	Cust.Pay.CreditCard	String			Last date of use for the card in ISO 8601 format
SeriesCode	Cust.Pay.CreditCard	String			Verification digits sometimes printed (but not embossed) on the card

Element	Occurc.	Content	Data type	Attributes	Description
Cust.RelatedTraveler	*	Element		ShareSynchInd, ShareMarketInd, DynSeq, BefDynSeq, AftDynSeq, RelatedServerId Relation	Other traveler profiles associated with the individual
UniqueId	R	Entity: v1.UniqueId			Unique identifier assigned to individual related to the customer in the profile
Cust.Related.PersonName	?	Element Entity = v1.PersonName			Name of individual related to traveler

Attribute Name	Element	Data Type	Default	Values	Description
Relation	Shared	String		Spouse, Children, ExtendedFamily, Business, InterestGroup, Medical, Security, Other	Indicates type of the relationship with the person in the profile
RelatedServerId	Shared	String			Identifier for the server that created the related traveler's profile

Element	Occurc.	Content	Data type	Constraints	Attributes	Description
Cust.EmergencyContact	*	Element			ShareSynchInd, ShareMarketInd, DynSeq, BefDynSeq, AftDynSeq, Relation, Email, URL	Persons who travel services should call in an emergency
Cust.Contact.PersonName	?	Element: Entity: v1.PersonName			Relation	Names of individuals to contact
Cust.Contact.Phone	*	Element Entity: v1.Telephone			PhoneTech PhoneUse, DynSeq, BefDynSeq, AftDynSeq,	Telephone numbers, including, fax or pager, of persons to contact
Cust.Contact.Addr	*	Element Entity: v1.Address			AddressType, DynSeq, BefDynSeq, AftDynSeq	Addresses of persons to contact

Element	Occurc.	Content	Data type	Attributes	Description
Cust.Document	*			DocType, Gender, BirthDate, EffectDate, ExpiryDate, DynSeq, BefDynSeq, AftDynSeq, ShareSynchInd, ShareMarketInd	Government-issued documents for travel
Cust.Doc.PersonName	?	Entity: v1.PersonName			Name of person given on the document
Cust.Doc.IssueAuthority	?	Text	String		Government authority issuing document. If a national document use ISO 3166 code
Cust.Doc.IssueLocation	?	Text	String		City, state/province document in which the document was issued
Cust.Doc.Limits	*	Text	String	DynSeq, BefDynSeq, AftDynSeq	Limitations on travel documents, such as visas for specific purposes or corrective lenses for driver's license

Attribute Name	Element	Data Type	Default	Values	Description
DocType	Cust.Document	String		Passport, Visa, Driver, VoterReg, BirthCert, SocialSecurity, MilitaryID, GovtID, Other	Indicates the kind of travel documents held by the person
EffectDate	Shared	String			Indicates the date on which the document takes effect (such as date of visa issuance), in ISO 8601 format
ExpiryDate	Shared	String			Indicates the date after which the document is no longer valid, in ISO 8601 format
DocId	Cust.Document	String			Unique number assigned by authorities to document
Gender	Shared	Text	Enum	Male, Female	Gender is sometimes used to identify documents

Element	Occurc.	Content	Data type	Attributes	Description
Cust.Loyalty	*	Element		LoyalLevel, SingleVendorInd, DynSeq, BefDynSeq, AftDynSeq, PreferLevel, ShareSynchInd, ShareMarketInd	Program rewarding frequent use by accumulating credits for services provided by vendors
CompanyCode	R	Text	String	CodeContext	Identifier for the company
Cust.Loyal.ProgramName	?	Text	String		Name of program accumulating the credits
Cust.Loyal.PersonName	?	Entity: v1.PersonName			Name of person registered with program

Attribute Name	Element	Data Type	Default	Values	Description
LoyalLevel	Cust.Loyalty	String			Indicates special privileges in program assigned to individual
SingleVendorInd	Cust.Loyalty	Enum		SingleVndr, Alliance	Indicates if program is affiliated with a group of related offers accumulating credits

5.6 Travel preferences

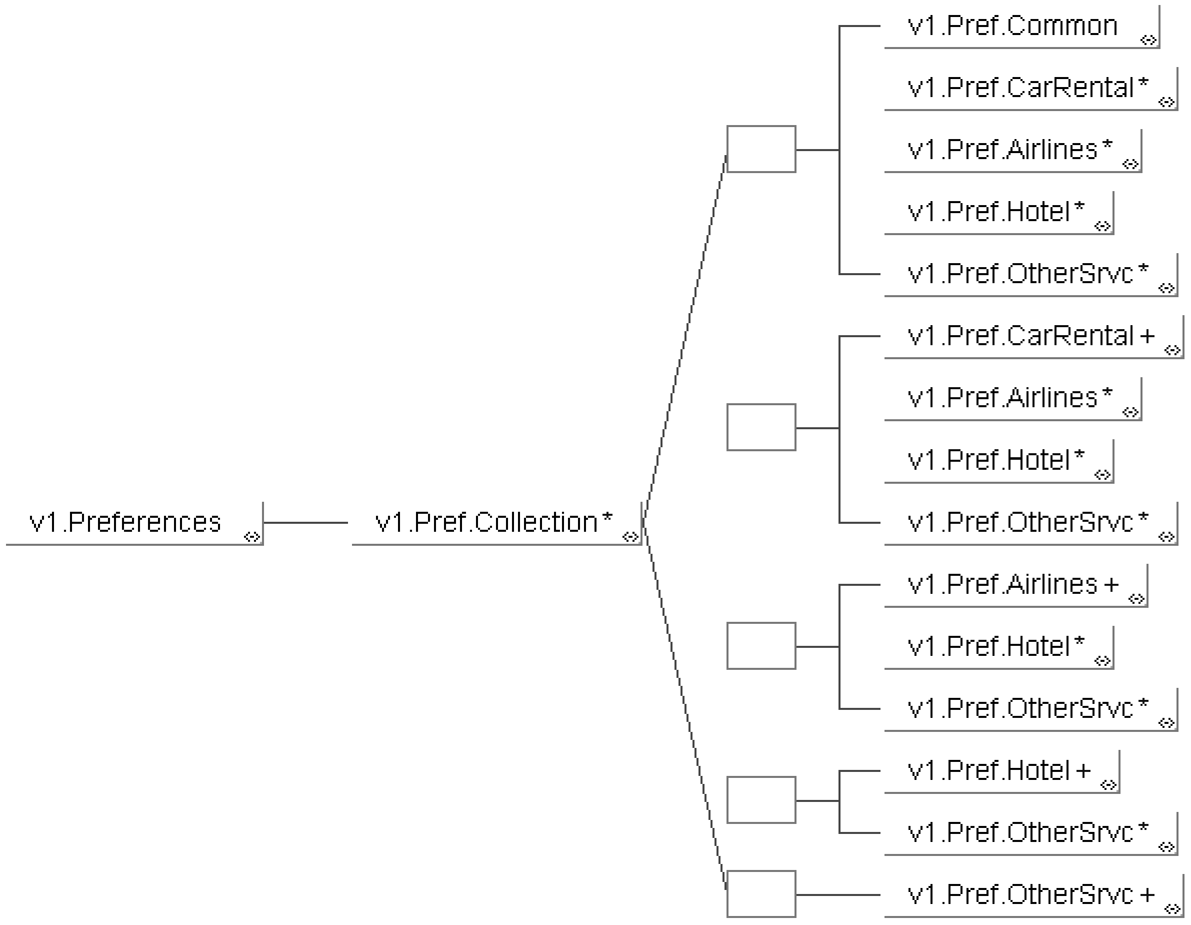
The Preferences section (tag prefix v1.Pref.) lists the needs of the traveler in various forms. It allows for the customer to gather these requirements into specified collections of preferences, and to designate a default group as well as name the groups for various kinds of travel experiences -- e.g., business, international, golf outings, or the annual church retreat. The section also provides preferences for 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 Pref.Collection) allow customers to identify common preferences as well as those associated with specific travel services. Customer profiles MAY have one or more of these collections. If a profile includes travel preferences, it MUST have services from one or more of the following element groups: Common, CarRental, Airlines, Hotel, or OtherSrvc (other services). As a result, a customer profile CANNOT include an empty Collections element. It MUST have valid data from one or more of the Common, CarRental, Airlines, Hotel, or OtherSrvc elements. See Figure 9 for the configuration of these main elements under Preferences.

The preference collections feature enables customers to specify features and services for each kind of travel identified. For example, the 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 customer would define collections for international and domestic travel, and identify separate airline services needed in each collection. Likewise, business 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 the corporate headquarters in Houston.

OTA developed the hotel elements 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. OTA work groups from the airline and car rental industries developed their respective elements over several months of effort.

Figure 9. Travel preferences elements



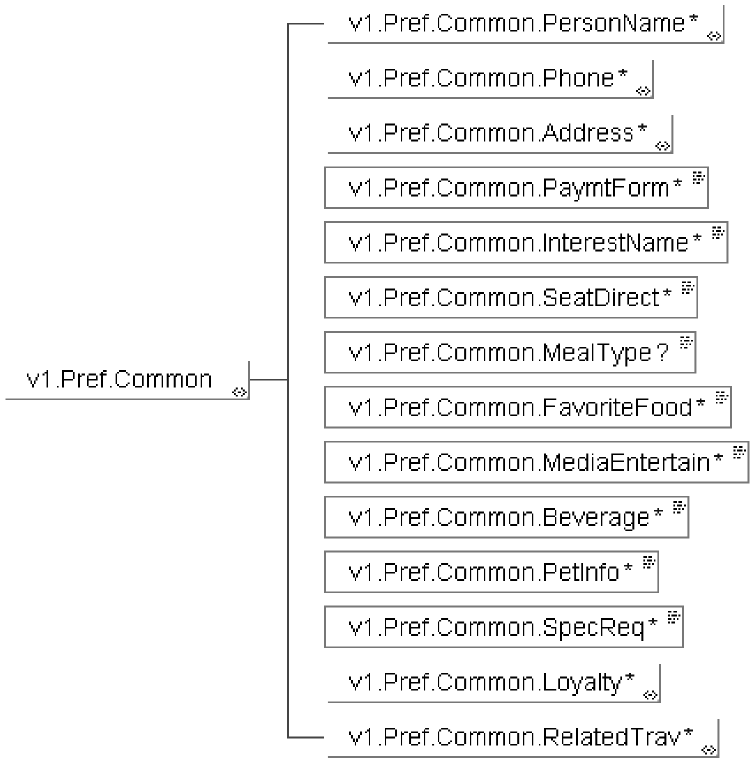
5.6.1 Common preference elements and attributes

Common preferences -- element Pref.Common -- represent those items that travel customers want associated with specific preference collections and are independent of travel services, such as airlines or hotels. These common preferences include the person's 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. Figure 10 shows the elements given under Pref.Common. While all of the elements are OPTIONAL, any occurrence of these elements MUST occur in the order shown in Figure 10.

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 10. Common travel preference elements


v1.Pref.Common
v1.Pref.Common.PersonName*
v1.Pref.Common.Phone*
v1.Pref.Common.Address*
v1.Pref.Common.PaymtForm*
v1.Pref.Common.InterestName*
v1.Pref.Common.SeatDirect*
v1.Pref.Common.MealType?
v1.Pref.Common.FavoriteFood*
v1.Pref.Common.MediaEntertain*
v1.Pref.Common.Beverage*
v1.Pref.Common.PetInfo*
v1.Pref.Common.SpecReq*
v1.Pref.Common.Loyalty*
v1.Pref.Common.RelatedTrav*

Element	Occurc	Content	Data type	Attributes	Description
Preferences	*	Element		ShareSynchInd, ShareMarketInd, AllChildElements	Needs of the traveler related to travel experiences
Pref.Collection	*	Element		PrefCollectName, TravelType, ShareSynchInd, ShareMarketInd, DynSeq, BefDynSeq, AftDynSeq, AllChildElements	Unique aggregation of customer travel needs

Attribute Name	Element	Data Type	Default	Values	Description
PrefCollectName	Pref.Collection	String			Unique name assigned to identify the collection
TravelType	Pref.Collection	String		Base (non-travel), Business, Leisure, International, Domestic, Other	Category of travel for this group of preferences

Element	Occurc	Content	Data type	Attributes	Description
Pref.Common	?	Element		SmokingInd, Language, TicketTime, ShareSynchInd, ShareMarketInd, AllChildElements	Travel needs associated with a collection but independent of specific travel services
Pref.Common.PersonName	*	Entity: v1.PersonName		NameType, ShareSynchInd, ShareMarketInd DynSeq, BefDynSeq, AftDynSeq	Name of traveler for profile, if required
Pref. Common.Phone	*	Entity: v1.Telephone		PhoneTech, PhoneUse, ContactType, ShareSynchInd, ShareMarketInd DynSeq, BefDynSeq, AftDynSeq	Telephones (including fax and pager) to use for this specific collection
Pref. Common.Address	*	Entity: V1.Address		AddressType, ContactType, ShareSynchInd, ShareMarketInd DynSeq, BefDynSeq, AftDynSeq	Addresses to use for this collection of preferences
Pref.Common.PaymtForm	*	Text	String	PreferLevel, PayFormReferName, DynSeq, BefDynSeq, AftDynSeq	Means of paying for travel services associated with this collection
Pref.Common.InterestName	*	Text	String	PreferLevel, DynSeq, BefDynSeq, AftDynSeq	Personal interests (i.e., hobbies, avocations) that enter into travel decisions
Pref.Common.SeatDirect	*	Text	String	SeatDirect, DynSeq, BefDynSeq, AftDynSeq, PreferLevel	Direction seat faces during travel, when conveyance allows
Pref.Common.MealType	?	Text	String	PreferLevel	Dietary restrictions during this travel collection , such as vegetarian or low-sodium
Pref.Common.FavoriteFood	*	Text	String	PreferLevel, DynSeq, BefDynSeq, AftDynSeq	Dining preferences, NOT dietary restrictions, for this travel collection
Pref.Common.MediaEntertain	*	Text	String	PreferLevel, DynSeq, BefDynSeq, AftDynSeq	Media and entertainment preferences for this travel: books, magazines, newspapers, radio, television, cinema, games, online

Pref.Common.Beverage	*	Text	String	PreferLevel, DynSeq, BefDynSeq, AftDynSeq	Drinks preferred during this kind of travel

Element	**Occurc**	**Content**	**Data type**	**Attributes**	**Description**
Pref.Common.PetInfo	*	Text	String	PreferLevel, DynSeq, BefDynSeq, AftDynSeq	Describes the pets that accompany the customer for this type of travel
Pref.Common.SpecReq	*	Text	String	PreferLevel, DynSeq, BefDynSeq, AftDynSeq	Special service requests for this type of travel
Pref.Common.Loyalty	*	Element Entity: v1.NameOrCode		PreferLevel, DynSeq, BefDynSeq, AftDynSeq	Program rewarding frequent use of service for this collection. CompanyCode refers back to loyalty programs defined under Customer
CompanyCode (Ent.v1.NameOrCode)	?	Text	String	CodeContext	Identifier for the company
FreeFormName (Ent.v1.NameOrCode)	?	Text	String		Name of company in free text

Attribute Name	**Element**	**Data Type**	**Default**	**Values**	**Description**
TicketTime	Shared	String			Ticket turnaround time desired, amount of time requested to deliver tickets
Language	Pref.Common	String			Preferred language for the form of travel represented in this collection, using ISO 639 codes
SeatDirect	Pref.Common.SeatDirect	String		Forward, Backward, Both	Direction seat faces during travel, when conveyance allows
PayFormReferName	Shared	String			A unique name to identify a form of payment, and refers back to a specific Cust.PaymentForm entry defined earlier.
PreferLevel	Shared	Enum		Only, Unacceptable	Identifies the customer's level of preference for the content of the element; if the attribute is absent assume a value of "preferred"

Element	Occurc	Content	Data type	Attributes	Description
Cust.CommonRelatedTrav	*	Element		DynSeq, BefDynSeq, AftDynSeq, RelatedServerId Relation	Other traveler profiles associated with the individual
UniqueId	R	Entity: v1.UniqueId			Unique identifier assigned to individual associated with this collection related to the customer in the profile
PersonName	R	Element Entity = v1.PersonName			Name of individual associated with this collection related to the customer in the profile

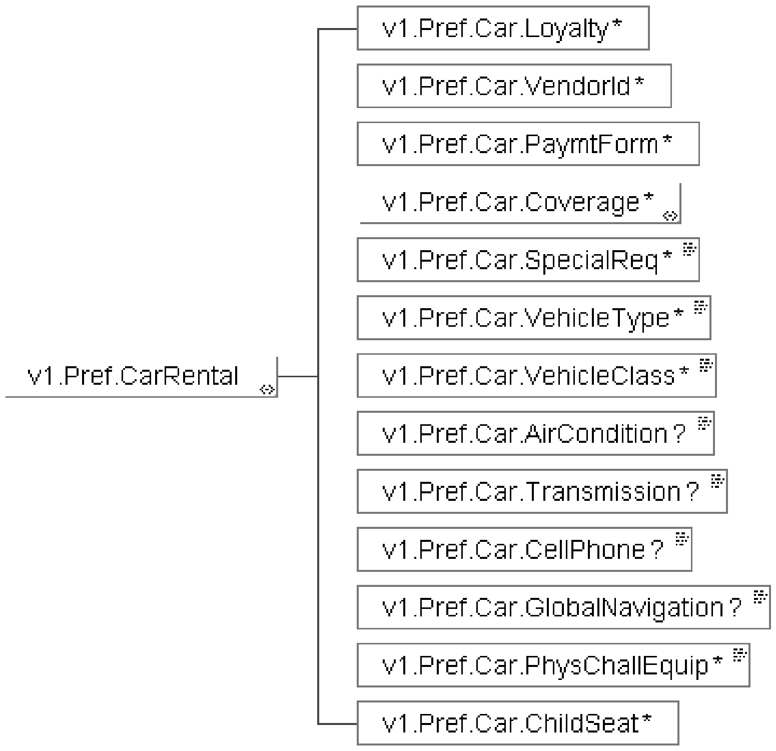
Attribute Name	Element	Data Type	Default	Values	Description
Relation	Shared	String		Spouse, Children, ExtendedFamily, Business, InterestGroup, Medical, Security, Other	Indicates type of the relationship with the person in the profile
RelatedServerId	Shared	String			Identifier for the server that created the related traveler's profile

5.6.2 Car rental preferences

The OTA version 1 specification captures preferences for rental cars including vendor and loyalty programs, forms of payment for rental cars, insurance coverage needed, and special requirements. This section also has entries for specific features on rental cars such as 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, physical-challenged needs, and child seats.

Figure 11 shows the elements included in the car rental section of the specification. While each of these elements is OPTIONAL, if any of these elements appear in an OTA version 1 message, they MUST appear in the order given in Figure 11.

Figure 11. Car rental preference elements



Element	Occur	Content	Data type	Attributes	Description
Pref.CarRental	*	Element		PreferLevel, CarRenterType, SmokingInd, TicketTime, ShareSynchInd, ShareMarketInd DynSeq, BefDynSeq, AftDynSeq	Preferences for car rentals associated with this collection
Pref.Car.Loyalty	*	Element Entity: v1.NameOrCode	String	PreferLevel, DynSeq, BefDynSeq, AftDynSeq	Car rental loyalty program preferred for this collection. CompanyCode element refers back to loyalty programs defined under Customer
CompanyCode (Ent.v1.NameOrCode)	?	Text	String	CodeContext	Identifier for the company
FreeFormName (Ent.v1.NameOrCode)	?	Text	String		Name of company in free text
Pref.Car.VendorId	*	Element Entity: v1.NameOrCode, Text	String	PreferLevel, DynSeq, BefDynSeq, AftDynSeq	Company identifiers for preferred car rental companies associated with this collection
CompanyCode (Ent.v1.NameOrCode)	?	Text	String	CodeContext	Identifier for the company
FreeFormName (Ent.v1.NameOrCode)	?	Text	String		Name of company in free text

Attribute Name	Element	Data Type	Default	Values	Description
TicketTime	Shared	String			Ticket turnaround time desired, amount of time requested to deliver tickets
CarRenterType	Pref.CarRental	String			For future use -- Categories of renters
SmokingInd	Shared	Enum	No	Yes, No	Describes customer as smoker or non-smoker, or preference for smoking or non-smoking facilities

Element	Occurc	Content	Data type	Attributes	Description
Pref.Car.PaymtForm	*	Text	String	PreferLevel, PayFormReferName, DynSeq, BefDynSeq, AftDynSeq	Means of paying for car rentals associated with this collection. PayFormReferName refers back to forms of payment defined under Customer.

Element	Occurc	Content	Data type	Attributes	Description
Pref.Car.Coverage	*	Element		CoverageType, CoverageCode, ShareSynchInd, ShareMarketInd, DynSeq, BefDynSeq, AftDynSeq, AllChildElements	Insurance coverage needed for car rentals associated with this collection
Pref.Car.Cov.Supplement	*	Text	String		Supplemental insurance coverage needed for rental cars
Pref.Car.Cov.Description	*	Text	String		Description of coverage needed for rental cars
Pref.Car.Cov.Limits	*	Text	String		Limitations on car rental insurance coverage, such as minimums, maximums or per occurrence

Attribute Name	Element	Data Type	Default	Values	Description
CoverageType	v1.PrefCar.Coverage	String		Effects, Personal, Baggage, VehicleDamage, Liability, Theft, YoungDriver	Categories of insurance preferred by the customer
CoverageCode	v1.PrefCar.Coverage	String			Industry code for insurance coverage

Element	Occurc	Content	Data type	Attributes	Description
Pref.Car.SpecialReq	*	Text	String	CarSpecialReq, DynSeq, BefDynSeq, AftDynSeq	Special needs or requirements for car rental services
Pref.Car.VehicleType	*	Text	String	DynSeq, BefDynSeq, AftDynSeq, PreferLevel, VehicleType	Major category of vehicle, e.g., car, truck, SUV
Pref.Car.VehicleClass	*	Text	String	DynSeq, BefDynSeq, AftDynSeq, PreferLevel, VehicleClass	Detailed type of vehicle, e.g. compact, midsize, passenger_van
Pref.Car.AirCondition	?	Text	String		Indicates need for air conditioning in a rented car
Pref.Car.Transmission	?	Text	String		Describes the type of transmission needed in a rented car, e.g. manual or automatic
Pref.Car.CellPhone	?	Text	String		Indicates the need for a cellular phone in a rented car
Pref.Car.GlobalNavigation	?	Text	String		Indicates need for a global navigation system in a rented car
Pref.Car.PhysChallEquip	*	Text	String	DynSeq, BefDynSeq, AftDynSeq, PreferLevel, ChallEquip	Describes equipment needed for drivers with special physical challenges
Pref.Car.ChildSeat	*	Element		DynSeq, BefDynSeq, AftDynSeq, PreferLevel, ChildSeatType, Quantity	Requirements for child seats in rented car

Attribute Name	Element	Data Type	Default	Values	Description
CarSpecialReq	Pref.Car.SpecialReq	String		Bicycle rack, Camcorder, Citizen band radio, Computer driving instructions, Child seat/infant, Child seat/toddler, FM radio, Hatchback car, Hand control/left, Hand control/right, Laser disc player, Left foot accelerator, Luggage rack, Navigational system plus phone, Navigational system, Mobile phone, Six	Special needs or requirements for car rental services

				passenger car, Ski rental, Ski equipped vehicle, Snow chains, Spinner knobs for disabled, Skierized vehicles, Cassette tape player, Trailer hitch, Television	
VehicleType	Pref.Car.VehicleType	String		Car, Van, SUV, Truck, Motorcycle, Limo, Station_Wagon, Pick_Up, Motor_Home, Other	Major category of vehicle

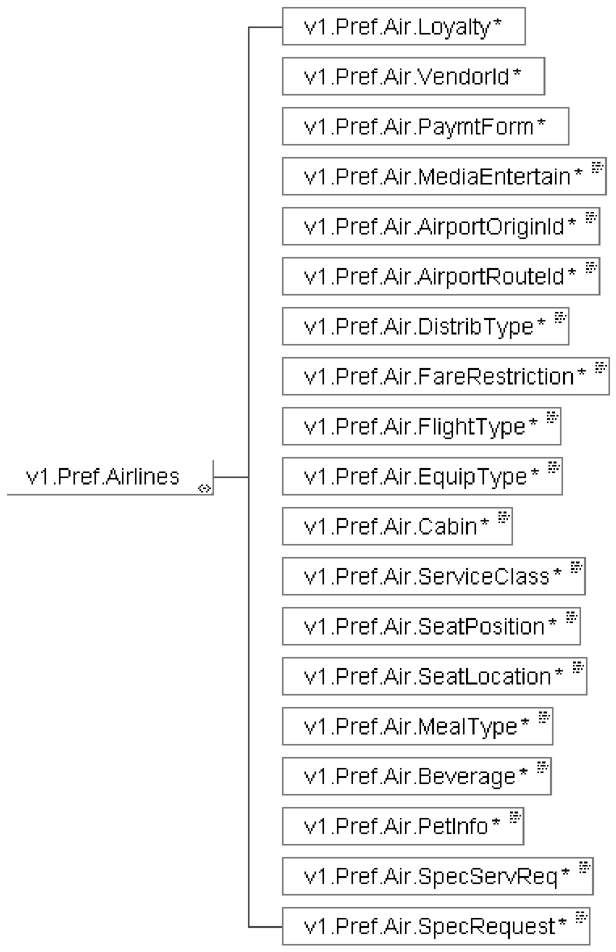
Attribute Name	**Element**	**Data Type**	**Default**	**Values**	**Description**
VehicleClass	Pref.Car.VehicleClass	String		Mini, Subcompact, Economy, Compact, Midsize, Intermediate, Standard, Full_Size, Luxury, Premium, Convertible, Minivan, 12_Passenger_Van, Moving_Van, 15_Passenger Van, Cargo_Van, 12_Foot_Truck, 15_Foot_Truck, 20_Foot_Truck, 24_Foot_Truck, 26_Foot_Truck, Moped, Stretch, Regular, Unique, Exotic, Other	Detailed type of vehicle
ChallEquip	Pref.Car.PhysChallEquip	String		Handbrakes, Spinner_Knob, Wheelchair_Ramp, Right_Hand_Control, Left_Hand_Control	Equipment needed for drivers with physical challenges
ChildSeatType	Pref.Car.ChildSeat	String		Infant, Child, Booster	Type of child seat needed in a rented car
Quantity	Pref.Car.ChildSeat	String			Number of child seats of the type indicated in ChildSeatType attribute

5.6.3 Air travel preferences

The OTA version 1 specification offers customers a number of features of air travel to include in their profiles. In this section, customers can indicate preferred vendors and loyalty programs for air travel, forms of payment used, media and entertainment choices, meal types required, beverages preferred, pet information, and special service requests (both free text fields and airline industry standard codes). 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. This section also indicates preferred aircraft equipment, choices for passenger cabin and service class, and preferences for position of seat in a row, and location of seat in the cabin.

Figure 12 shows the elements included in the air travel section of the specification. While each of these elements is OPTIONAL, if any of these elements appear in an OTA version 1 message, they MUST appear in the order given in Figure 12.

Figure 12. Air travel preference elements



Element	Occur	Content	Data type	Attributes	Description
Pref.Airlines	*	Element		AirPassengerType PreferLevel AirTicketType, SmokingInd, TicketTime, ShareSynchInd, ShareMarketInd, AllChildElements, DynSeq, BefDynSeq, AftDynSeq	Air travel preferences
Pref.Air.Loyalty	*	Element Entity: v1.NameOrCode	String	PreferLevel, DynSeq, BefDynSeq, AftDynSeq	Air travel loyalty program preferred for this collection. CompanyCode element refers back to loyalty programs defined under Customer
CompanyCode (Ent.v1.NameOrCode)	?	Text	String	CodeContext	Identifier for the company
FreeFormName (Ent.v1.NameOrCode)	?	Text	String		Name of company in free text

Pref.Air.VendorId	*	Element Entity: v1.NameOrCode	String	PreferLevel, DynSeq, BefDynSeq, AftDynSeq	Company identifiers for preferred airlines associated with this collection
CompanyCode (Ent.v1.NameOrCode)	?	Text	String	CodeContext	Identifier for the company
FreeFormName (Ent.v1.NameOrCode)	?	Text	String		Name of company in free text

Element	Occurc	Content	Data type	Attributes	Description
Pref.Air.PaymtForm	*	Text	String	PreferLevel, PayFormReferName, DynSeq, BefDynSeq, AftDynSeq	Means of paying for air travel associated with this collection. PayFormReferName refers back to forms of payment defined under Customer.

Attribute Name	Element	Data Type	Default	Values	Description
AirTicketType	Pref.Airlines	Enum	Paper	Electronic, Paper	Type of airline ticket preferred for this collection
AirPassengerType	Pref.Airlines	String			Category of airline passenger, using standard ATPCO codes
TicketTime	Shared	String			Ticket turnaround time desired, amount of time requested to deliver tickets

Element	Occurc	Content	Data type	Attributes	Description
Pref.Air.MediaEntertain	*	Text	String	DynSeq, BefDynSeq, AftDynSeq, PreferLevel	Media and entertainment preferences for this travel: books, magazines, newspapers, radio, television, cinema, games, online
Pref.Air.AirportOriginId	*	Text	String	DynSeq, BefDynSeq, AftDynSeq, PreferLevel,	Departure airport preferences, IATA airport codes
Pref.Air.AirportRouteId	*	Text	String	DynSeq, BefDynSeq, AftDynSeq, PreferLevel,	Preference for connecting airports, IATA airport codes
Pref.Air.DistribType	*	Text	String	DynSeq, BefDynSeq, AftDynSeq, PreferLevel, DistribType	Type of ticket distribution preferred , such as mail, courier, or airport pickup.
Pref.Air.FareRestriction	*	Text	String	DynSeq, BefDynSeq, AftDynSeq, PreferLevel, FareRestric	Type of fare restrictions acceptable for travel defined in this collection

Pref.Air.FlightType	*	Text	String	DynSeq, BefDynSeq, AftDynSeq, PreferLevel, FlightType	Type of flight preferred: non-stop, direct or connections
Pref.Air.EquipType	*	Text	String	DynSeq, BefDynSeq, AftDynSeq, PreferLevel	Type of make/model of aircraft preferred
Pref.Air.Cabin	*	Text	String	DynSeq, BefDynSeq, AftDynSeq, PreferLevel, CabinType	Cabin choice for this collection of preferences

Element	**Occurc**	**Content**	**Data type**	**Attributes**	**Description**
Pref.Air.ServiceClass	*	Text	String	DynSeq, BefDynSeq, AftDynSeq, PreferLevel	Air travel booking class choice for this collection of preferences

Attribute Name	**Element**	**Data Type**	**Default**	**Values**	**Description**
DistribType	Pref.Air.DistribType	String		Fax, Email, Courier, Mail, Airport_Pickup, City_Office	Ticket distribution method
FareRestric	Pref.Air.FareRestriction	String		None, Advance, SatStopOver, StandBy	Fare restrictions accepted
FlightType	Pref.Air.FlightType	String		NonStop, Direct, Connection	Preference for flights with connections or stop overs
CabinType	Pref.Air.Cabin	String		First, Business, Coach	Cabin location preferred

Element	**Occurc**	**Content**	**Data type**	**Attributes**	**Description**
Pref.Air.SeatPosition	*	Text	String	DynSeq, BefDynSeq, AftDynSeq, PreferLevel, SeatRow	Preferred position of passenger seat in row for this collection
Pref.Air.SeatLocation	*	Text	String	DynSeq, BefDynSeq, AftDynSeq, PreferLevel, SeatLocat	Preferred position of seat in cabin, such as bulkhead or exit row
Pref.Air.MealType	*	Text	String	DynSeq, BefDynSeq, AftDynSeq, PreferLevel	Dietary restrictions for air travel (e.g. Kosher, low-sodium)
Pref.Air.Beverage	*	Text	String	DynSeq, BefDynSeq, AftDynSeq, PreferLevel	Customer drink preferences for air travel
Pref.Air.PetInfo	?	Text	String		Describes the pets that accompany the customer during air travel

Pref.Air.SpecServReq	*	Text	String	SSR_OSI, DynSeq, BefDynSeq, AftDynSeq, PreferLevel	Special services required for air travel, using standard industry (SSR-OSI) code list
Pref.Air.SpecRequest	*	Text	String	DynSeq, BefDynSeq, AftDynSeq, PreferLevel	Special requests for other services needed in air travel for this collection

Attribute Name	Element	Data Type	Default	Values	Description
SeatRow	Pref.Air.SeatPosition	String		Aisle, Window , Center	Position of seat in row
SeatLocat	Pref.Air.SeatLocation	String		ExitRow, Bulkhead, Galley, Wing, Sun, Shade	Location of seat in cabin
SSR_OSI	Pref.Air.SpecServReq	String			Standard industry (SSR-OSI) code list

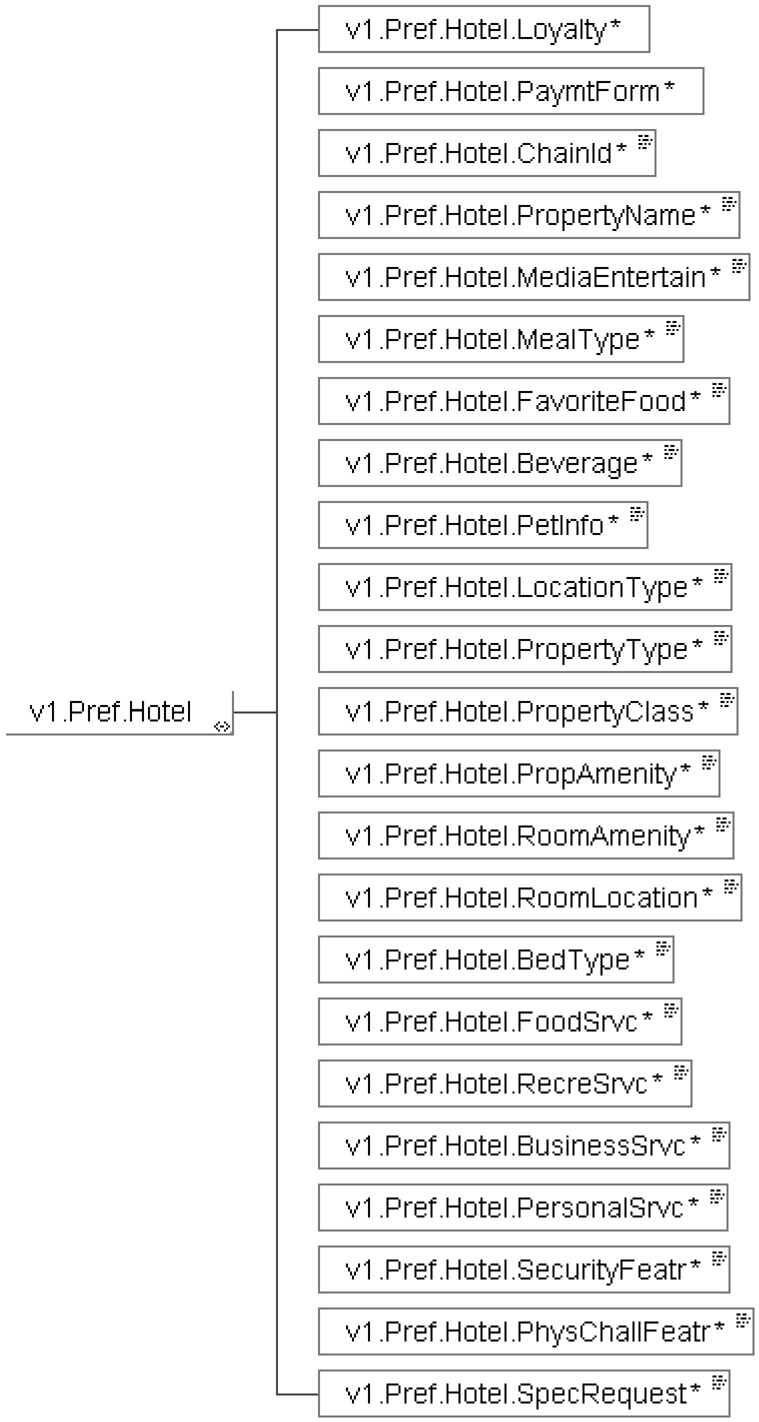
5.6.4 Hotel preferences

In OTA version 1, customers can select their preferences for various amenities of hotel properties and rooms. Travelers may identify preferred chains, properties, and loyalty programs, as well as the desired location, property type, and property class. Customers may indicate their preferences for services provided by the properties, such as recreational and business services, as well as in-room amenities and location 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 preferred media and entertainment 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.

In the tables below, some of the attribute lists are quite lengthy and are found in Appendix 3.

Figure 13 shows the elements included in the hotel section of the specification. While each of these elements is OPTIONAL, if any of these elements appear in an OTA version 1 message, they MUST appear in the order given in Figure 13.

Figure 13. Hotel preference elements


v1.Pref.Hotel
v1.Pref.Hotel.Loyalty*
v1.Pref.Hotel.PaymtForm*
v1.Pref.Hotel.ChainId*
v1.Pref.Hotel.PropertyName*
v1.Pref.Hotel.MediaEntertain*
v1.Pref.Hotel.MealType*
v1.Pref.Hotel.FavoriteFood*
v1.Pref.Hotel.Beverage*
v1.Pref.Hotel.PetInfo*
v1.Pref.Hotel.LocationType*
v1.Pref.Hotel.PropertyType*
v1.Pref.Hotel.PropertyClass*
v1.Pref.Hotel.PropAmenity*
v1.Pref.Hotel.RoomAmenity*
v1.Pref.Hotel.RoomLocation*
v1.Pref.Hotel.BedType*
v1.Pref.Hotel.FoodSrvc*
v1.Pref.Hotel.RecreSrvc*
v1.Pref.Hotel.BusinessSrvc*
v1.Pref.Hotel.PersonalSrvc*
v1.Pref.Hotel.SecurityFeatr*
v1.Pref.Hotel.PhysChallFeatr*
v1.Pref.Hotel.SpecRequest*

Element	Occur	Content	Data type	Attributes	Description
Pref.Hotel	*	Element		HotelGuestType, RatePlan, SmokingInd, TicketTime, PreferLevel, DynSeq, BefDynSeq, AftDynSeq, ShareSynchInd, ShareMarketInd	Hotel preferences
Pref.Hotel.Loyalty	*	Element Entity: v1.NameOrCode	String	PreferLevel, DynSeq, BefDynSeq, AftDynSeq	Hotel loyalty programs preferred for this collection. CompanyCode element refers back to loyalty programs defined under Customer
CompanyCode (Ent.v1.NameOrCode)	?	Text	String	CodeContext	Identifier for the company
FreeFormName (Ent.v1.NameOrCode)	?	Text	String		Name of company in free text
Pref.Hotel.PaymtForm	*	Text	String	PreferLevel, PayFormReferName, DynSeq, BefDynSeq, AftDynSeq	Means of paying for hotel accommodations associated with this collection. PayFormReferName refers back to forms of payment defined under Customer.
Pref.Hotel.ChainId	*	Element Entity: v1.NameOrCode	String	PreferLevel, DynSeq, BefDynSeq, AftDynSeq	Company identifiers for hotel chains associated with this collection
CompanyCode (Ent.v1.NameOrCode)	?	Text	String	CodeContext	Identifier for the company
FreeFormName (Ent.v1.NameOrCode)	?	Text	String		Name of company in free text
Pref.Hotel.PropertyName	*	Text	String	DynSeq, BefDynSeq, AftDynSeq, PreferLevel	Name of preferred hotel properties associated with this collection

Attribute Name	Element	Data Type	Default	Values	Description
RatePlan	Pref.Hotel	String			WHIS: Code identifying preferred hotel rate plan for this collection
HotelGuestType	Pref.Hotel	String			For future use -- Categories of hotel guests
TicketTime	Shared	String			Ticket turnaround time desired, amount of time requested to deliver tickets

Element	Occurc	Content	Data type	Attributes	Description
Pref.Hotel.MediaEntertain	*	Text	String	DynSeq, BefDynSeq, AftDynSeq, PreferLevel	Media and entertainment preferences for hotel stays in this collection: books, magazines, newspapers, radio, television, cinema, games, online
Pref.Hotel.MealType	*	Text	String	DynSeq, BefDynSeq, AftDynSeq, PreferLevel	Dietary restrictions at hotels
Pref.Hotel.FavoriteFood	*	Text	String	DynSeq, BefDynSeq, AftDynSeq, PreferLevel	Dining preferences when a guest at hotels
Pref.Hotel.Beverage	*	Text	String	DynSeq, BefDynSeq, AftDynSeq, PreferLevel	Customer drink preferences at hotels
Pref.Hotel.PetInfo	?	Text	String		Describes the pets that accompany the customer during hotel stays
Pref.Hotel.LocationType	*	Text	String	DynSeq, BefDynSeq, AftDynSeq, PreferLevel, LocationType	HEDNA: property location preferences

Attribute Name	Element	Data Type	Default	Values	Description
LocationType	Pref.Hotel.LocationType	String		City, Rural, Suburban, Airport, Beach, Mountain, Lake	Location of property in region

Element	Occurc	Content	Data type	Attributes	Description
Pref.Hotel.PropertyType	*	Text	String	DynSeq, BefDynSeq, AftDynSeq, PreferLevel, PropertyType	HEDNA: property type preferences
Pref.Hotel.PropertyClass	*	Text	String	DynSeq, BefDynSeq, AftDynSeq, PreferLevel, PropertyClass	HEDNA: property class preferences
Pref.Hotel.PropAmenity	*	Text	String	DynSeq, BefDynSeq, AftDynSeq, PreferLevel, PropAmenity	HEDNA: property features and services
Pref.Hotel.RoomAmenity	*	Text	String	DynSeq, BefDynSeq, AftDynSeq, PreferLevel, RoomAmenity	HEDNA: room features and services

Element	Occurc	Content	Data type	Attributes	Description
Pref.Hotel.RoomLocation	*	Text	String	DynSeq, BefDynSeq, AftDynSeq, PreferLevel, RoomLocation	Physical location of room in property
Pref.Hotel.BedType	*	Text	String	DynSeq, BefDynSeq, AftDynSeq, PreferLevel, BedType	Size, features of bed preferred
Pref.Hotel.FoodSrvc	*	Text	String	DynSeq, BefDynSeq, AftDynSeq, PreferLevel, FoodSrvc	HEDNA: type of restaurant facilities preferred

Attribute Name	Element	Data Type	Values	Description
PropertyType	Pref.Hotel.PropertyType	String	All-Suite, All-Inclusive Resort, Apartment, Bed&Breakfast, Cabin/Bungalow, Corporate, Hostel, Inn, Health Spa, Holiday Resort, Hotel, Lodge, Condominium, Conference Center, Chalet, Campground, Guesthouselimited Service, Guest Farm, Meeting Resort, Monastery, Motel,Ranch, Self-Catering Accommodation, Vacation Home, Villas, Wildlife Reserve	Category of hotel property preferred
PropertyClass	Pref.Hotel.PropertyClass	String	Luxury, Moderate, Deluxe, Economy, 1st Class, Budget, Tourist	General quality level preferred
PropAmenity	Pref.Hotel.PropAmenity	String	See Appendix 3	Special features of hotel properties
RoomAmenity	Pref.Hotel.RoomAmenity	String	See Appendix 3	Special features of hotel rooms
RoomLocation	Pref.Hotel.RoomLocation	String	Higher Floor, Lower Floor, Quiet Room, Near Elevator	Location or type of room in property
BedType	Pref.Hotel.BedType	String	King, Queen, Double, Double/Twin	Size of bed preferred
FoodSrvc	Pref.Hotel.FoodSrvc	String	Full Service Restaurant, Coffee Shop, Buffet, Bar/Lounge, Snack Bar, Ice Cream/Dessert Shop, Fast Food	Type of food services required

Element	Occurc	Content	Data type	Attributes	Description
Pref.Hotel.RecreSrvc	*	Text	String	DynSeq, BefDynSeq, AftDynSeq, PreferLevel, RecreSrvc	HEDNA: recreational services at property preferred
Pref.Hotel.BusinessSrvc	*	Text	String	DynSeq, BefDynSeq, AftDynSeq, PreferLevel, BusinessSrvc	HEDNA: preferred business services at property
Pref.Hotel.PersonalSrvc	*	Text	String	DynSeq, BefDynSeq, AftDynSeq, PreferLevel, PersonalSrvc	HEDNA: preferred personal services at property
Pref.Hotel.SecurityFeatr	*	Text	String	DynSeq, BefDynSeq, AftDynSeq, PreferLevel, SecurityFeatr	HEDNA: preferred security features at property
Pref.Hotel.PhysChallFeatr	*	Text	String	DynSeq, BefDynSeq, AftDynSeq, PreferLevel, PhysChallFeatr	Features to meet needs of persons with physical challenges, such as disabilities and allergies
Pref.Hotel.SpecRequest	*	Text	String	DynSeq, BefDynSeq, AftDynSeq, PreferLevel	Special requests for other services needed for hotel stays

Attribute Name	Element	Data Type	Values	Description
RecreSrvc	Pref.Hotel.RecreSrvc	String	See Appendix 3	Recreational services offered at properties
BusinessSrvc	Pref.Hotel.BusinessSrvc	String	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	Type of business-related services required

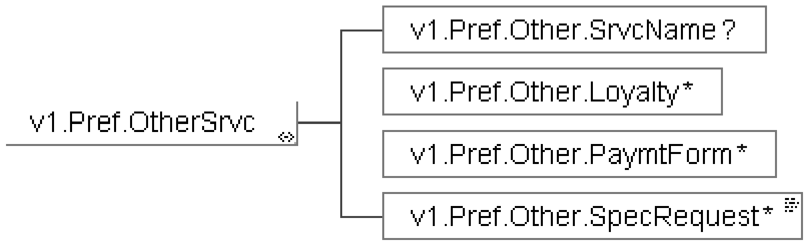
Attribute Name	Element	Data Type	Values	Description
PersonalSrvc	Pref.Hotel.PersonalSrvc	String	Express Checkin, Express Checkout, Concierge Service, Laundry/Valet Service, Childcare, Cribs, Currency Exchange /Banking, Safety Deposit Boxes, Language Translation, Beauty Shop/ Barber/Hairdresser available, Beauty treatments, Massage Service, Washers/Dryers available	Type of personal comfort services offered by properties
SecurityFeatr	Pref.Hotel.SecurityFeatr	String	See Appendix 3	Features related to health, safety, and security
PhysChallFeatr	Pref.Hotel.PhysChallFeatr	String	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	Features needed by guests with physical challenges

5.6.5 Preferences for other travel services

The specification allows customers to identify services for other, as yet undefined travel services. The Pref.OtherSrvc element 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.

Figure 14 shows the elements contained in the Pref.OtherSrvc hierarchy. While each of these elements is OPTIONAL, if any of these elements appear in an OTA version 1 message, they MUST appear in the order given in Figure 14.

Figure 14. Other travel service preference elements



Element	Occurc	Content	Data type	Attributes	Description
Pref.OtherSrvc	*	Element		OtherServiceType, TicketTime, DynSeq, BefDynSeq, AftDynSeq, PreferLevel, ShareSynchInd, ShareMarketInd	Preferences for travel services other than the services listed above
Pref.Other.SrvcName	?	Text	String		
Pref.Other.Loyalty	*	Element, Entity: v1.NameOrCode	String	PreferLevel, DynSeq, BefDynSeq, AftDynSeq	Loyalty programs for other travel services associated with this collection. CompanyCode element refers back to loyalty programs defined under Customer
CompanyCode (Ent.v1.NameOrCode)	?	Text	String	CodeContext	Identifier for the company
FreeFormName (Ent.v1.NameOrCode)	?	Text	String		Name of company in free text
Pref.Other.PaymtForm	*	Text	String	PreferLevel, PayFormReferName, DynSeq, BefDynSeq, AftDynSeq	Means of paying for other travel services associated with this collection. PayFormReferName refers back to forms of payment defined under Customer.
Pref.Other.SpecRequest	*	Text	String	DynSeq, BefDynSeq, AftDynSeq, PreferLevel	Special requests for other services needed for this collection

Attribute Name	Element	Data Type	Values	Description
OtherServiceType	Pref.OtherSrvc	String		Future use: Category of other travel service user
TicketTime	Shared	String		Ticket turnaround time desired, amount of time requested to deliver tickets

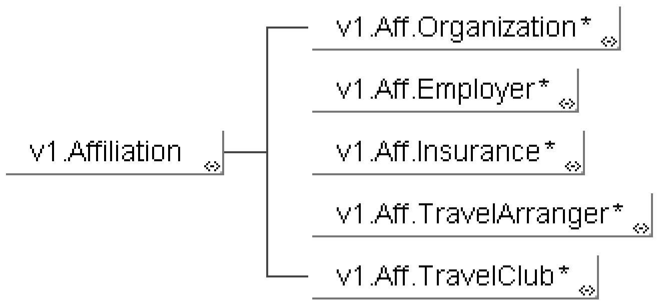
5.7 Affiliations

OTA version 1 allows for the capture of data on Affiliations (tag prefix v1.Aff.), defined as organizations with which the customer has contact and offer travel services or benefits. These organizations include employers and membership organizations, such as AAA, AARP, and alumni associations that provide travel programs, discounts, and benefits. This section also covers vital travel services including travel arrangers and insurance companies, as well as vendor-sponsored travel clubs offering special lounges and related services. These clubs include those provided by airlines at airports or Amtrak for its first class and Metroliner passengers.

Insurance in this case indicates insurance policies carried by the individual. In the section on car rental preferences, insurance refers to insurance needed for car rentals.

Each of the five types of affiliations -- organizations, employers, insurance, travel arrangers, and travel clubs -- is a child element of the Affiliation element, tag <v1.Affiliation>, as seen in Figure 15. While each of these elements is OPTIONAL, if any of these elements appear in an OTA version 1 message, they MUST appear in the order given in Figure 15.

Figure 15. Affiliation elements



Element	Occurc.	Content	Data type	Attributes	Description
Affiliation	*	Element		DynSeq, BefDynSeq, AftDynSeq, AllChildElements, ShareSynchInd, ShareMarketInd	Companies or organizations connected with the customer that can affect travel service decisions

Aff.Organization	*	Element		DefaultInd, PersonId, OfficeType, ExpiryDate, DynSeq, BefDynSeq, AftDynSeq, ShareSynchInd, ShareMarketInd	Membership organization that has travel benefits, programs, or discounts
Aff.Organiz.Name	?	Text	String		Name of organization
Aff.Organiz.RelatedName	*	Text	String	DynSeq, BefDynSeq, AftDynSeq	Entity related to the organization, such as division or subsidiary
Aff.Organiz.Level	?	Text	String		Level in organization (e.g. seniority) that conveys privileges
Aff.Organiz.MemberTitle	?	Text	String		Rank in organization that conveys privileges
Aff.Organiz.TravelSrvcVendorName	*	Element, Entity: v1.NameOrCode	String	BenefitId, DynSeq, BefDynSeq, AftDynSeq	Names of travel service suppliers to organization providing benefits, privileges, or discounts
CompanyCode (Ent.v1.NameOrCode)	?	Text	String	CodeContext	Identifier for the company
FreeFormName (Ent.v1.NameOrCode)	?	Text	String		Name of company in free text

Attribute Name	**Element**	**Data Type**	**Default**	**Values**	**Description**
OfficeType	Shared	String		Headquarters, Parent, Branch	Main or field office of the organization
PersonId	Shared	String			Identifier assigned to the person by the organization
ExpiryDate	Shared	String			Date when membership in organization ends, in ISO 8601 format
BenefitId	Shared	String			Identifier for program that conveys benefits, privileges, or discounts

Element	Occurc	Content	Data type	Attributes	Description
Aff.Employer	*	Element		DefaultInd, EmployeeStatus, PersonId, OfficeType, ExpiryDate, AllChildElements, DynSeq, BefDynSeq, AftDynSeq, ShareSynchInd, ShareMarketInd	Company or organization that employs the customer
Aff.Empl.CompName	?	Text	String		Name of employer organization
Aff.Empl.RelatedName	*	Text	String	DynSeq, BefDynSeq, AftDynSeq	Entity related to the employer company such as division or subsidiary, for which the customer works
Aff.Empl.Level	?	Text	String		Level in employer organization (e.g. seniority) that conveys privileges
Aff.Empl.Title	?	Text	String		Rank in employer company that conveys privileges
Aff.Empl.InternalRefNmbr	*	Text	String	DynSeq, BefDynSeq, AftDynSeq	Accounting code assigned to travel for employer
Aff.Empl.TravelSrvcVendorName	*	Text	String	BenefitId, DynSeq, BefDynSeq, AftDynSeq	Names of travel service suppliers to employer providing benefits, privileges, or discounts
Aff.Empl.Loyalty	*	Element, Entity: v1.NameOrCode	String	PreferLevel, DynSeq, BefDynSeq, AftDynSeq	Loyalty programs with travel service vendors to which the employer subscribes. CompanyCode element refers back to loyalty programs defined under Customer
CompanyCode (Ent.v1.NameOrCode)	?	Text	String	CodeContext	Identifier for the company
FreeFormName (Ent.v1.NameOrCode)	?	Text	String		Name of company in free text

Attribute Name	Element	Data Type	Default	Values	Description
EmployeeStatus	Aff.Employer	String		Active, Retired, Leave, Terminated	Type of employment with organization

Element	Occurc.	Content	Data type	Attributes	Description
Aff.Insurance	*	Element		PreferLevel, InsuranceType, PolicyNumber, DynSeq, BefDynSeq, AftDynSeq, ShareSynchInd, ShareMarketInd	Insurance for travel carried by customer
Aff.Insur.CompName	R	Text	String		Insurance company providing coverage
Aff.Insur.Underwriter	?	Text	String		Underwriting company for insurance

Attribute Name	Element	Data Type	Values	Description
InsuranceType	Aff.Insurance	String		Type of insurance policy carried by the individual
PolicyNumber	Aff.Insurance	String		Identifier assigned by insurance company to the policy held by the customer
EffectDate	Aff.Insurance	String		Date insurance coverage begins in ISO 8601 format
ExpiryDate	Aff.Insurance	String		Date insurance coverage ends in ISO 8601 format

Element	Occurc	Content	Data type	Attributes	Description
Aff.TravelArranger	*	Element		DefaultInd, TravelArrangerType, DynSeq, BefDynSeq, AftDynSeq, ShareSynchInd, ShareMarketInd, AllChildElements	Companies or individuals responsible for making travel plans or transactions either for the customer or organizations affiliated with the customer (e.g., employer)
Aff.TrvlArr.CompName	?	Element, Entity: v1.NameOrCode	String		Name of organization making travel plans or transactions
CompanyCode (Ent.v1.NameOrCode)	?	Text	String	CodeContext	Identifier for the company
FreeFormName (Ent.v1.NameOrCode)	?	Text	String		Name of company in free text
Aff.TrvlArr.PersonName	*	Element Entity = PersonName		DefaultInd, DynSeq, BefDynSeq, AftDynSeq	Names of persons assigned to make travel plans or transactions
Aff.TrvlArr.Address	*	Element Entity = Address		DefaultInd, DynSeq, BefDynSeq, AftDynSeq	Locations of entity making travel plans or transactions

Element	Occurc	Content	Data type	Attributes	Description
Aff.TrvlArr.Phone	*	Element Entity = Phone		DefaultInd, DynSeq, BefDynSeq, AftDynSeq	Telephones of entity making travel plans or transactions
Aff.TrvlArr.Email	*	Text	String	DefaultInd, DynSeq, BefDynSeq, AftDynSeq	Electronic mail addresses for travel arranger, IETF specified format
Aff.TrvlArr.URL	*	Text	String	DefaultInd, DynSeq. BefDynSeq, AftDynSeq	Web site addresses for travel arranger, IETF specified format

Attribute Name	Element	Data Type	Default	Values	Description
TravelArrangerType	Aff.TravelArranger	String			Type of service making travel plans or transactions

Element	Occur	Content	Data type	Attributes	Description
Aff.TravelClub	*	Element		ExpiryDate, DynSeq, BefDynSeq, AftDynSeq, ShareSynchInd, ShareMarketInd	Special-privilege club room and related services offered by travel vendors
Aff.Club.ProgramName	R	Element, Entity: v1.NameorCode	String		Name of travel club
CompanyCode (Ent.v1.NameOrCode)	?	Text	String	CodeContext	Identifier for the company
FreeFormName (Ent.v1.NameOrCode)	?	Text	String		Name of company in free text
Aff.Club.PersonName	?	Element Entity = PersonName		PersonId	Name of individual registered with the travel club

Attribute Name	Element	Data Type	Default	Values	Description
ExpiryDate	Shared	String			Date club membership ends, in ISO 8601 format
PersonId	Shared	String			Identifier assigned to person registered with the travel club

5.8 Reusable content (entities)

Several groups of elements and attributes repeat at least several times in the content model, which enables them to be stored as separated entities and called up by the version 1 DTD as needed. The name of each entity is given below with its content model. As with the elements in the main DTD, elements stored in reusable entities are prefixed with the version 1 (v1.) notation.

Entity: Ent.v1.PersonName	**Occurc.**	**Content**	**Data type**	**Attributes**	**Description**
PersonName		Element			Name of persons on record
Person.NameTitle	*	Text	String	DynSeq, BefDynSeq, AftDynSeq	Salutation, honorific, title
Person.GivenName	?	Text	String		Given name, first name
Person.MiddleName	*	Text	String	DynSeq, BefDynSeq, AftDynSeq	Middle name
Person.MiddleInitial	*	Text	String	DynSeq, BefDynSeq, AftDynSeq	Middle initials
Person.Surname	R	Text	String		Family name, last name
Person.NameSuffix	*	Text	String	DynSeq, BefDynSeq, AftDynSeq	Jr/Sr, degree, Ret, honors (such as OBE)

Entity Ent.v1. Address	**Occurc.**	**Content**	**Data type**	**Attributes**	**Description**
Address		Element			Physical address of parties to the profile
Address.StreetNmbr	?	Text	String		Street name; number on street
Address.BldgRoom	*	Text	String		Building name; room, apartment, or suite number
Addr.CityName	?	Text	String		Name of city or town
Addr.StateProv	?	Text	String	PostalCode	State, province, or region name or code needed to identify location
Addr.CountryName	?	Text	String	CountryCode	Country name if needed for delivery
Addr.Line	*	Text	String	DynSeq, BefDynSeq, AftDynSeq	Additional lines needed in the address

Attribute Name	**Element**	**Data Type**	**Default**	**Values**	**Description**
POBox	Addr.StreetNmbr	String			Box, drawer, or route location reserved for postal delivery
PostalCode	Addr.StateProv	String			Identifier for location assigned by postal authorities
CountryCode	Addr.CountryName	String			ISO 3166 code for country in address

Entity Ent.v1.Telephone	Occurc.	Content	Data type	Attributes	Description
Telephone		Element		CountryAccess, Extension, PIN	Telephone numbers of the parties in the profile
Phone.AreaCityCd	R	Text	String		Code assigned for telephones in a specific region, city, or area
Phone.Number	R	Text	String		Specific number assigned to single location

Attribute Name	Element	Data Type	Default	Values	Description
CountryAccess	Telephone	String			Code assigned by international telecommunications authorities for country
Extension	Telephone	String			Extension to reach a specific party
PIN	Telephone	String			Additional codes used for pager

Entity Ent.v1.UniqueId	Occurc.	Content	Data type	Attributes	Description
UniqueId		Element			Identifier, different from all others, assigned to the profile
TradingPartner	*	Element			Contains an identifier for the company exchanging data and qualifier for that identifier that describes the type of identifier
CompanyCode	R	Text	String	CodeContext	Identifier for the company exchanging data
ResourceId	R	Text	String	ResourceContext	Unique code assigned by system that created the customer profile

Attribute Name	Element	Data Type	Default	Values	Description
ResourceContext	ResourceId	String			Qualifier for the code displayed in ResourceId that describes the type of code
CodeContext	Shared	String			Describes the type of identifier used in the CompanyCode, such as DUNS or IATA

Entity Ent.v1.CompanyId	Occurc.	Content	Data type	Attributes	Description
CompanyCode		Element		CodeContext	Identifier for the company exchanging data

Entity Ent.v1.NameOrCode	Occurc.	Content	Data type	Attributes	Description
CompanyCode	Choice	Text	String	CodeContext	Identifier for the party exchanging data
FreeFormName	Choice	Text	String		Name of the party in free form text

Entity: Ent.v1.Privacy	Attribute Name	Data Type	Default	Values	Description
	ShareSynchInd	Enum	Inherit	Yes, No, Inherit	Permission for sharing data in element for marketing information
	ShareMarketInd	Enum	Inherit	Yes, No, Inherit	Permission for sharing data in element for synchronization of profiles held by other travel service providers

Entity: Ent.v1.Repeating	Attribute Name	Data Type	Default	Values	Description
	DynSeq	String		0, 1, 2, etc.	Used in the Update process to indicate the appearance sequence of repeating elements. Starts with 0, increments by 1.
	BefDynSeq	String		0, 1, 2, etc.	
	AftDynSeq	String		0, 1, 2, etc.	

Entity: Ent.v1.HasRepeating	Attribute Name	Data Type	Default	Values	Description
	AllChildElements	Enum		Yes, No	Used in the Update process to indicate all subordinate elements in the structure are included.

5.9 Custom fields

Trading partners may have business needs to create fields in the profile record not yet covered in OTA version 1. The elements under Pref.OtherSrvc (see section 5.6.5) can help cover many of these contingencies, since they allow for definition of travel services not covered under common, car, rental, air, or hotel. For other circumstances, data items added to the message are not covered by the Version 1 specification and are the responsibility of the trading partners. By adding private fields to a message, that message will NOT conform to the specification. Trading partners may need to write their own schemas to cover items not included in the version 1 specification, so there is no confusion between conforming and non-conforming messages.

6 Error and non-versioned message operations[6]

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 operates independently of the versioned content. Because the messages do not have a version they must be structurally stable yet have operational flexibility.

6.1 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 a single DTD for all of the non-versioned OTA messages (OTA_Nv.dtd).

This DTD is designed to be used individually (not combined), and as such the element and attribute names (or symbols) need only be unique within this DTD and the non-versioned OTA messages. Unlike versioned OTA messages (OTA_v1), non-versioned OTA messages do not use Content and Control elements. As such the specific action element is directly within (or under) the non-versioned OTA root element.

6.1.1 Standard error messages

The OTA XML infrastructure defines a non-versioned standard error message. The set of errors that can use this non-versioned standard error (for operational flexibility) is constrained by its limited structure (for structural stability). The non-versioned standard error consists of a StandardError element (immediately under the OTA root) with four “payloads”. These “payloads” are three attributes:

• DocURL	- OPTIONAL
• Status	- OPTIONAL
• Type	- REQUIRED

The fourth payload consists of the additional OPTIONAL information provided by parsed character data (PCDATA) returned by the trading partner's system.

The StandardError element MUST contain the Type attribute.

The StandardError element MAY contain an optional element data (PCDATA) for human readability to convey responses from legacy systems.

[6] We thank Scott Hinkelman of IBM and George Smith of ConXtra for developing this part of the specification.

The StandardError element MAY contain an optional DocURL (Uniform Resource Locator) attribute to refer to an online description of the error type through a URL.

The StandardError element MAY contain an optional StandardError Status attribute. This attribute uses an enumeration, which indicates the outcome of the invoked request.

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 responses:

- NotProcessed. The error occurred prior to processing of the request (or if during the processing of the request, before any intentions of the request where 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.

The StandardError element MUST contain the StandardError Type attribute that uses an evolving enumeration to indicate the error type. An evolving enumeration is one where the validating DTD’s enumeration list evolves (by adding, and only adding, new acceptable values), as such, the receiving application can expect to accept values that it has NOT been explicitly coded for and process them in an acceptable way. This is facilitated by already accepting a StandardError Type of “Unknown”. The initial enumeration list 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 the PCDATA.
- UnrecognizedRoot. Indicates an error in the root tag. Additional information may be provided within the PCDATA.

StandardError messages SHOULD use the following syntax:

```
<StandardError Status ="…" Type="…" DocURL="…">
Additional Information</StandardError>
```

All OTA message requests MAY result in a response message that consists of a StandardError construct alone, directly within the OTA root. Depending on the parser used by a particular implementation, this error could be returned in any of the following ways:

- StandardError Type=”Validation”
- StandardError Type=”UnrecognizedRoot”
- Hi-level context related error

Examples of non-versioned StandardError OTA messages (returned from some xxxRQ) follow:

```
<OTA>
      <StandardError Type="Malformed">Missing close tag: Address</StandardError>
</OTA>

<OTA>
      <StandardError Type="Validation">Not Date, tag:DepDT</StandardError>
</OTA>

<OTA>
      <StandardError Status="Incomplete" Type="Unknown">
           ERROR: FORM OF PAYMENT NOT RECOGNIZED
      </StandardError>
</OTA>
```

6.2 Error messages in versioned OTA messages (OTA_v1)

Section 4.3 gives the specifications for versioned error messages, coming under the <OTA_v1> root tag.

The versioned specification also provides a facility to help trading partners identify the outcome of a message. Typically, if a business message, such as updating a customer profile, fails for a business level reason, the business message itself should declare a failure response (xxxRS) that may be 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. When a StandardError or v1.StandardError is not returned, for efficiency reasons, trading partners should be able to quickly determine if the request succeeded.

Therefore, every v1.*xxx*RS element MUST have an optional Success attribute. The attribute values MUST include the enumerated list of: Yes, No, Part (Part means a partial success, and "Yes" is the default). This attribute provides a consistent approach for determining the result; successful or not. Beyond that, it is up to the upper level model and business operation to specify the contents of the <v1.*xxx*RS> tags. Depending on the specification of the Success indicator, the <v1.*xxx*RS> tag MAY contain error information, resulting output information, or nothing at all. This information is best determined within each upper working group.

A simple example:

```
   <OTA_v1>
      <v1.Content>
            <v1.CustomerProfileUpdateRS Success="Y"/>
      </v1.Content>
   </OTA_v1>
```

6.3 Version discovery messages

The OTA XML infrastructure supports version discovery message protocols in order for OTA businesses to transition through time with the standard, 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. This version discovery process consists of two questions:

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

To answer the first question, to determine the versions supported, trading partners MAYexchange a non-versioned OTA message, using the following syntax for the request:

```
<OTA>
   <VersionsSupportedRQ />
</OTA>
```

In response, the VersionSupportedRS element MUST contain a repeating Version element. The Version element will have five attributes:

- XML_Root - required
- Protocol - required
- Access - required
- Params - optional
- DocURL - optional

And an optional element data (PCDATA) for additional information.

The returned VersionSupportedRS element MUST contain a Version Root attribute to indicate both the context and version number by specifying the expected root element name. See example below.

The returned VersionSupportedRS element MUST contain a Version Protocol attribute to indicate the access protocol. See example below.

The returned VersionSupportedRS element MUST contain a Version Access attribute to indicate access path (or ID or address). For a protocol that is based on HTTP, this will be a URL. See example below.

The returned VersionSupportedRS element MAY contain a Version Params attribute to communicate small additional information needed to access the version. See example below.

The returned VersionSupportedRS element MAY contain a Version DocURL (Uniform Resource Locator) attribute to refer to an online description of the support context and version.

Note: 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 a VersionsSupportedRS message follows:

```
<OTA>
   <VersionsSupportedRS>
```

```
            <Version XML_Root="OTA"        Protocol ="HTTP 1.1 Post"
                                           Access="OTA.XYZ.com"
                                           DocURL="http://www.XYZ.com/OTA" />
            <Version XML_Root="OTA_v1"     Protocol ="HTTP 1.1 Post"
                                           Access="www.XYZ.com/ota/impl" />
            <Version XML_Root="OTA_v2"     Protocol ="HTTPS 1.1 Post"
                                           Params="XML="
                                           Access="www.XYZ.com/ota/impl2" />
            <Version XML_Root="OTA_v5"     Protocol ="RMI over HTTP"
                                           Access="RMI-OTA.XYZ.com:8000" />
         </VersionsSupportedRS>
      </OTA>
```

With a response like the one above, a trading partner can choose the most appropriate OTA version to use to interact with the target trading partner.

NOTE: It is assumed that the protocol and access information for the non-versioned OTA message is already known between the two trading partners.

6.4 Non-versioned message DTD

Non-versioned messages MUST use a document type definition (DTD) separate from the DTD used for versioned messages. OTA_Nv.dtd provides a DTD for non-versioned OTA messages, listed in Appendix 1.

7 Security and Privacy

Earlier chapters on message structure and profile content addressed specific implementations of security and privacy in the OTA version 1 specification. This chapter discusses the requirements for security and privacy, and provides guidance in addition to the features discussed earlier in this document.

7.1 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.

7.2 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 customer profiles, 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 version 1 address the following factors:

7.2.1 Authentication

Is the party engaged in this transaction really the party they claim to be or an imposter seeking to engage in fraud? This concern raises the need to identify and credential the parties in the transaction.

7.2.2 Confidentiality

Can the parties conduct this transaction with the assurance this information remains known only to the parties sharing this sensitive information? Adequate confidentiality limits the possibility of eavesdropping.

7.2.3 Integrity

Do the parties engaged in the transaction have assurance that the data received by one party are the same as the data sent by the other party? Can the parties limit the possibility of distortion of the message by sabotage even by or non-malicious intent, such as network errors?

7.2.4 Non-repudiation

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

7.2.5 Privacy

Do the individuals providing data about themselves and family members have control over the disposition of data after collection by the travel companies? Can the parties provide assurance to the individuals providing data that the data will be used only for the purposes originally indicated on the collection forms? Will individuals have the ability to remove their names from any sharing arrangements? Authorities in many jurisdictions around the world have regulations on the collection and dissemination of personal data.

7.3 Security and privacy recommendations

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

7.3.1 Authentication

See section 4.2.1 that presents a recommended syntax for authenticating trading partners using the OTA specifications. Section 4.2.3.1 gives examples of OTA message syntax, showing the relationship between authentication and encryption.

7.3.2 Confidentiality

To meet the needs of confidentiality OTA version 1 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 partners MAY encrypt the body portion of the message. See section 4.2.3 for a discussion of the EncryptedContent element. Section 4.2.3.1 offers examples of various scenarios and the recommended message syntax for those scenarios.

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 (section 6.3), 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.

7.3.3 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.

Encryption (see 7.3.2 above) provides a message integrity check, since the decryption routine performs a defacto hash function to correctly reassemble the content. Also, the OTA version 1 specification now 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.

7.3.4 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.

7.3.5 Privacy

As with security, privacy connotes control over the information collected, and the OTA version 1 specification builds in ways for the customer to determine the data they want shared with other parties. To address these concerns, the specification provides customers with choices to determine the sharing of data in their profiles overall, and for components of the profiles. OTA version 1 assumes the customer does NOT want the data shared for any reason unless specifically authorized. As a result, when the customer provides data for the profile, he or she MUST give authorization to share that data with other parties.

For the profiles as a whole, the specification provides customers with two attributes on CustProfile, the highest-level content element:

- *ShareAllSynchInd*, to permit sharing data for synchronization, and

- *ShareAllMarketInd,* to permit sharing data for marketing purposes

Synchronization means updating profiles that may reside on other servers than the one that created it. Customers may want, for example, to have their profiles reside with vendors in which they take part in loyalty programs (e.g., frequent flyers) as well as the company that created it. Marketing purposes include providing the data to other vendors who can send offers related to the interests of the customer.

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

Customers can determine the data shared within a profile as well. The major elements (those with sub-elements) in the specification have two attributes for sharing the data they represent:

- *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 over-ride that choice for individual elements.

Another feature of the OTA specification that can help meet privacy requirements is the optional GeoCode attribute on the root (OTA_v1) element, see section 4.1.1. This attribute gives the latitude and longitude of the server generating the message. Companies implementing the specification MAY wish to identify geographic areas with special privacy requirements and use the GeoCode to alert trading partners to those requirements.

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 Appendix 4.

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 a last call for comments on the latest working draft of the P3P text.