OpenTravel™ Alliance
Transport Protocol Reference: HTTP

Version 1.0
August 5th 2005

© 2005 OpenTravel™ Alliance

Table of Contents

© 2005 OpenTravel™ Alliance

1 Introduction

1.1 Background

1.1.1 2001C OTA Infrastructure Guidelines

The **2001C OTA Infrastructure** guidelines were added to the OTA Specifications in the 2001C release.

These guidelines included design goals, XML best practices, the concept of using flexible request/response message pairs, and the use of ebXML as a transport protocol for passing these messages. The specification of the transport protocol was not formally part of the OTA Specifications because the OTA members wished to maintain flexibility with regard to which transport protocol they used.

1.1.2 Design Goals of OTA (2001C)

In 2001, the Design Goals were stated as follows:

OTA's basic goal is to design industry specifications capable of exploiting the communications systems that are available with Internet connectivity. To achieve these goals, OTA Specifications are designed to meet the following criteria:

- *Openness*- OTA Specifications are publicly available to all organizations seeking to develop new or enhanced systems. Membership in OTA is open to all organizations interested in developing these specifications.
- *Flexibility*- OTA Specifications provide travel service providers with the flexibility they need to develop, test, and deploy new services. The Specifications outline the minimum necessary functionality to provide reliable interactions between customers and the systems owned and maintained by the companies serving them.
- *Platform Independence*- OTA Specifications are developed to work with any equipment or software that can support the common standards used in the Specifications.
- *Security*- OTA places great importance on the need to protect information from unauthorized access and the need to give the customer control over the creation, update, and exchange of data with other parties.
- *Extensibility*- OTA plans to add more services and functionality to these Specifications in a way that minimizes incompatibility for those implementing this or other earlier versions. OTA work groups that develop the Specifications do so with future transitions in mind.
- *International scope*- The initial Specifications were written in English; however, OTA intends to extend later versions to provide representation in character sets supporting the Unicode standard. When possible, OTA has designed data elements to meet as many global elements as possible.

1.2 The Need for Interoperability

1.2.1 Emerging Trends

Since 2001, many new members have joined the OTA and want to know how to build systems that interoperate with other vendors who have implemented OTA messages. Within the travel industry, many more integrators and intermediaries have emerged that wish to work with many partners. Decreasingly, the emphasis is on enabling implementation of interactions between individual pairs of trading partners and increasingly on creating services that interoperate with all interested trading partners.

Furthermore, recent OTA message registrations have indicated that a simple transport protocol based on an HTTP POST is increasingly popular.

These two trends suggest that a new specification is appropriate. This document describes this specification. It provides the guidance necessary to create interoperable applications based on a reference transport protocol.

It should not be construed that companies are any less OTA conformant if they choose a different transport protocol. This specification serves simply as guidance for companies wishing to make OTA conformant applications that are highly interoperable.

1.2.2 New Design Goals

This HTTP transport protocol reference document addresses the following design goals in response to emerging trends:

- *Interoperable*- Companies developing OTA conformant systems should have sufficient guidance to build systems that are highly interoperable with systems built by other companies using the same OTA messages for the same use cases.
- *Simple*- To be able to reach the most partners, operating from the most platforms, the protocol would do well to be as simple as possible. This also reduces the hurdles for new members to implement OTA messages in their systems.
- *Optimal Performance*- The OTA HTTP transport protocol should not impede performance by burdening the processing with unnecessary overhead.
- *Readily Accepted*- The OTA HTTP transport protocol should reflect what member companies are already doing, thus assuring broad acceptance of the specification.

The following additional design goal is desirable, but is not directly addressed by this document:

- *Certifiable*- Companies should have some means whereby they can certify (for their own benefit) that they have done what it takes to give their system a good chance of interoperating with other companies' systems.

1.2.3 Requirements for Interoperability

True interoperability is a lofty goal. A reference transport specification is necessary but not sufficient to achieve it. The following are identified as important to achieving true interoperability.

- *Standard Usage Profiles*- The OTA messages are very flexible. However, if a set of systems are to be interoperable with each other, they must use the selected OTA messages in the same way for a specific use case. The OTA plans to support the definition by member companies of Usage Profiles as examples of the OTA messages, restricted to support a specific use case. Interoperability will require that a set of Usage Profiles emerge that are generally accepted.

© 2005 OpenTravel™ Alliance

- *Standard Transport Protocol*- This document describes the OTA Reference Transport Protocol. For purposes of interoperability, it may serve as the standard.
- *Standard Software Validation Suite*- To have any confidence that a system will interoperate with other systems, it must be exercised by a test suite to certify that its features comply with the standards. The production of software is currently beyond the scope of efforts sponsored by the OTA.

1.3 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.

In other words, "MUST," "MUST NOT," "REQUIRED," "SHALL," and "SHALL NOT" identify protocol features that are not optional.

Similarly, "SHOULD," "SHOULD NOT," "RECOMMENDED," "MAY," and "OPTIONAL" identify statements that require consideration with regard to the impact that inclusion or exclusion of the protocol feature will have on the interoperability of the system.

1.4 Status of this document

This document may be subsumed into a future release of the OTA Infrastructure document.

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 gwilson@opentravel.org.

OTA also welcomes comments to:

OpenTravel Alliance
1255 23rd Street, NW
Washington, DC 20037-1174 USA
Phone: (202) 521-6777 or (866) 682-1829
Fax: (202) 833-3636

EDITORS NOTE*: This document is a working draft publication of the OTA 2005B specification to be released for public review in December, 2005. It is anticipated that this document will undergo a period of commentary and resolution of comments may result in revisions prior to final publication.*

2 OTA Transport Protocol Reference: HTTP

2.1 Philosophy of Interoperability

For maximum interoperability, your system SHOULD be liberal in what it accepts and strict in what it emits.

2.2 Simple HTTP POST vs. ebXML

The prior recommendations regarding ebXML are still available for companies wishing to implement that transport protocol. However, the OTA acknowledges that adoption of ebXML among OTA member companies has been limited and that the following alternate protocol based on a simple HTTP POST represents a simpler method to create interoperable systems.

Other transport protocol references may be specified in future OTA releases. Such specifications may include (but are not limited to) SOAP (Simple Object Access Protocol).

2.3 Standard HTTP

Nothing in this specification should be understood to override or alter the standard definition of the HTTP protocol. The HTTP/1.0 and HTTP/1.1 versions of the HTTP protocol are described in the following documents.

- HTTP/1.0 – http://www.ietf.org/rfc/rfc1945.txt
- HTTP/1.1 – http://www.ietf.org/rfc/rfc2616.txt

However, the design of HTTP accommodates extensions through the use of custom headers. Custom headers that are defined for OTA transactions are described below.

2.4 HTTP Message Content

The OTA transport protocol reference is based on a simple HTTP POST transaction. An HTTP POST transaction allows for an arbitrarily large amount of "content" to be sent with both the request and the response.

An OTA Request message is sent as the "content" in an HTTP POST request, and an OTA Response message is received in response.

2.5 HTTP Message Headers

Besides the "content," an HTTP message (request or response) also carries HTTP "headers." The goal of the following description of headers is to describe maximum simplicity and interoperability using HTTP. If a system developer uses any standard HTTP-handling libraries or components, these details should be handled automatically.

Both clients and servers MUST support HTTP/1.0. They MAY support HTTP/1.1, but if they do, they MUST support the client/server negotiation that allows the server to downgrade to HTTP/1.0 if the client cannot support HTTP/1.1.

Different headers are allowed (or required) based on whether referring to the client (requestor) or server (responder) and whether using HTTP/1.0 or HTTP/1.1.

© 2005 OpenTravel™ Alliance

The only HTTP header REQUIRED for both request and response is the “Content-length” header. This allows systems that implement the HTTP protocol themselves to be written simply.

If HTTP/1.1 is used, the request requires the “Host” header (a requirement of HTTP/1.1 itself).

Other headers may be useful, but they are not required, and they may be safely ignored.

Header	Description	Example
Content-length	[REQUIRED] Tells how many bytes are in the “content” (after the “headers”).	Content-length: 1558
Host	[REQUIRED for HTTP/1.1 Client according to HTTP/1.1 spec] This allows a server that is hosting multiple virtual domains to know which domain the request was for. If the server is not hosting multiple virtual domains, the server MAY ignore this header.	Host: www.travelco.com
Connection	[OPTIONAL] Systems SHOULD utilize this header, but they MUST be robust when communicating with another server that does not recognize it. Using “Close” means that the network socket will be closed after the exchange of a single message (headers and content). “Keep-alive” means that the network socket will be kept open for a short time in case additional transactions are to be sent on the same socket. “Close” is the default for HTTP/1.0, and “Keep-alive” is the default for HTTP/1.1.	Connection: Close Connection: Keep-alive
Accept-charset	[OPTIONAL] This tells a server what character sets the client can handle. The server MAY ignore this if it will only be responding with ASCII text. However, if it might be responding with any non-ASCII text, it MUST encode the data in accordance with one of the acceptable character sets. “utf-8” is the preferred character set because XML is specified to support Unicode, and “utf-8” is widely supported (and is ASCII-compatible). If the header is supplied by the client, it MUST include “utf-8”. If the header is not supplied, the server should assume “utf-8”. In this way, both client and server should support “utf-8” and the content in both request and response can be understood to be “utf-8”.	Accept-charset: utf-8
Accept	[OPTIONAL] This tells a server what Content-types the client can handle. The server SHOULD ignore this, because it is going to respond with an XML OTA Response message regardless of what the client says can be accepted. The client MAY produce this header, but if so, it SHOULD include at least one of the “xml” content types.	Accept: text/xml Accept: application/xml Accept: application/xhtml+xml Accept: text/html, text/plain
Accept-language	[OPTIONAL] This tells a server what languages the client can handle.	Accept-language: en-us, en
Accept-encoding	[OPTIONAL] This tells a server whether the response content can be compressed or not. Clients and servers SHOULD implement a compressed encoding, but they MUST support the standard HTTP negotiation that allows them to interoperate with systems which do not implement it.	Accept-encoding: gzip, deflate
[other]	[OPTIONAL] Clients and servers MAY provide other headers, and they MAY be ignored by the other system.	n/a

Other headers MAY be supplied, and they MAY be ignored.

2.6 OTA Custom Header: OTA-Echo-Token

For performance reasons, in order to accommodate massive exchanges of OTA messages between two systems in a high-volume B2B relationship, an optional HTTP Header, "OTA-Echo-Token," may be used.

Without any extension to the HTTP protocol, the requesting system would send a request with a Keep-Alive header and wait for a response. Each request would have to wait for a response. This is not high enough performance for these applications.

When the "OTA-Echo-Token" header is used, the token value MUST be unique to that message and not shared by any other message sent by that system. The sending system does not necessarily expect an immediate response, but it does expect that whenever the response message is received, it will be accompanied by the same token value in an "OTA-Echo-Token" header.

Using the "OTA-Echo-Token," it is possible to multiplex several OTA message transactions asynchronously over the same TCP/IP connection. The token value acts as a correlator between the request and the response.

Header	Description	Example
OTA-Echo-Token	[OPTIONAL] A token sent with a request that is expected to be echoed in a response. This is a string token with a maximum size of 128 characters.	OTA-Echo-Token: 8335013 OTA-Echo-Token: A154/125/001

A requesting system that uses this header may be attached only to a responding system that uses the header. Otherwise it will not receive the information necessary for it to function.

A responding system that understands this header SHOULD be able to reply synchronously to clients that do not use the header.

2.7 Encryption

- Encrypted communication between systems is accomplished with SSL (i.e., HTTPS).
- If encryption is required, a system MUST support HTTPS.
- Systems MAY communicate via unencrypted HTTP for transactions that do not need to be secure. HTTP is also useful for transactions that are executed across a VPN or other networking channel that is already secure for other reasons.

2.8 Authentication

- Authentication of a client (requestor) to a server (responder) is achieved via HTTP Basic Authentication.
- Authentication of a server (responder) to a client (requestor) is achieved via SSL.
- If authentication is required, a system MUST support HTTP basic authentication.
- Systems MAY communicate without authentication for transactions that do not require it.

2.9 Other Features

2.9.1 Logging

Organizations offering OTA transactions MAY provide logging capability, regardless of the type of transaction in the business message (e.g., travel verbs, infrastructure verbs), and SHOULD provide logging capability for all transactions that involve implicit or explicit contracts or promises to pay. Trading partners MAY exchange event logs to provide audit trails.

© 2005 OpenTravel™ Alliance