OTA

The OpenTravel™ Alliance

OpenTravel™ Alliance Implementation Guide: WSDL

Version 0.2
7 June 2006

TABLE OF CONTENTS

1. Introduction

The WSDL Implementation Guide serves as a guideline to proper WSDL creation for services based on OTA XML Schema specifications. Many industry experts consider the "Contract-First" style of service design to be the correct choice for maximum service interoperability. Contract-First design can be considered akin to "Interface-First" design, whereby the service interface is designed *before* development begins rather than generated from code. Since the OTA is only responsible for defining message interfaces, it is naturally aligned with the concepts behind Contract-First service design. Unfortunately, many of the developer tools currently on the market attempt to simplify service creation by allowing the Web service interface to be generated from code. While convenient, this style of interface definition can hamper interoperability and ultimately shields developers from learning proper XML Schema and WSDL design.

OTA schemas already define individual message payloads; however, a WSDL file is responsible for defining the interface of a given service. This document attempts to serve as a guideline for proper WSDL creation, while helping OTA implementers create highly interoperable services through the use of Contract-First design principles.

Readers of this document may want to consider it in conjunction with the OTA SOAP and/or HTTP transport protocol reference documents.

1.1. Terminology

The Web Services Description Language (WSDL) is an XML format for describing service interfaces (i.e. data types, messages, ports and bindings). A WSDL file can consist of both abstract interface definitions as well as concrete implementation details.

The WS-I is the Web Services Interoperability organization which is an open industry organization chartered to promote Web services interoperability across platforms, operating systems and programming languages.

A usage profile schema is a modified schema which further restricts the original OTA options for a given implementation. An instance document based on a Usage Profile schema should still validate to the original OTA schema that it is based on. A usage profile service, therefore, is a service that implements a set of usage profile schemas that satisfy a given use-case.

1.2. Purpose

Although document style services, whose message payloads are defined by XML schemas, are common, it has always been challenging to create an interface definition based on the Web service Description Language for this type of service. Many developers creating client applications that will utilize Web Services are now demanding WSDL files, which greatly reduce the time and complexity of consuming Web Services. In order to be truly useful, the WSDL needs to reference the schema in a way that allows toolkits to bind the message data types and understand how to marshal and un-marshal the XML message payload. OTA based services fall into this category and many service providers have been unable to create WSDL files that can be used by their partners to automatically generate a client application.

During the 2005B publication period, a WSDL Publication Feasibility Study Committee was tasked with determining if the OTA could define WSDL creation for their messages, and if so, what the project scope would be. The committee concluded that a thorough summary of WSDL best practices, as well as a set of working samples, would be a tremendous aid to the OTA implementer. Since Web Services technologies are still evolving open ended specifications, there is much confusion regarding the proper implementation techniques. WSDL creation is particularly challenging as most developers rely on toolkits to generate WSDL files from their code. This is a dangerous practice, however, because it results in service interfaces that are more tightly coupled with their implementation environment and hampers interoperability. Without the ability to rely on tools to generate proper WSDL files, many implementers will be challenged to create a functional highly interoperable service definition document.

The WS-I has been instrumental in creating best practice guidelines that can help clarify these issues and guide the industry towards the goal of highly interoperable Web Services. Along with the WS-I guidelines, the OTA WSDL Implementation Guide provides further guidance and examples, which will aid in the adoption and implementation of OTA messages.

1.3. Scope

The primary intent of this document is to define the correct method of WSDL creation for OTA messages as well as to show sample interface definitions (WSDL) for selected OTA schemas (request/response message pairs). Secondarily, this document will provide WSDL Best Practices documentation, which will aid OTA Implementers by providing real world implementation insights into XML Schema design and WSDL creation.

As previously mentioned, the OTA conducted a WSDL Publication Feasibility Study as a precursor to creating the WSDL Implementation Guide, with the goal of outlining the scope of WSDL creation for OTA based services. The study considered defining WSDL files for OTA usage profile based services and ultimately defined the scope of WSDL creation for this document as follows:

> "While usage profile based WSDL files may provide the greatest benefit to OTA implementers, it may be overly ambitious to create these usage profile WSDL files at this point. Currently there is a parallel OTA study project underway which will further define a usage profile. Once this study is complete and sample usage profiles are defined, the OTA may choose to publish usage profile WSDL files as well. In the meantime, the OTA's goal should be to publish sample message based WSDL files which can either be used as is, or serve as an example to help OTA implementers.
>
> These message based Interface Definition WSDL files can be created for each request/response message pair. These interface definitions will not use usage profile schemas at this point, and will instead refer to the flattened[1] root message schemas published by the OTA. In the future, usage profile based interface definitions can also be published. However, until the concept of usage profiles are more fully defined and common usage patterns determined, these message root based WSDL files will serve as valuable guidelines."

[1] The WSDL Implementation Guide project team has since decided against the use of flattened schemas within a WSDL. Please see sections 2.3.2 and section 4.3.1 for additional information.

1.4. 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.

2. WSDL Best Practices

2.1. Overview

The WSDL Implementation Usage Guide will aid in the adoption of OTA messages by providing real world guidance for developers and implementers.

2.2. WSDL Definition[2]

A WSDL file defines **services** as collections of network endpoints, or **ports**. In WSDL, the abstract definition of endpoints and messages is separated from their concrete network deployment or data format bindings. This allows the reuse of abstract definitions: **messages**, which are abstract descriptions of the data being exchanged, and **port types** which are abstract collections of **operations**. The concrete protocol and data format specifications for a particular port type constitutes a reusable **binding**. A port is defined by associating a network address with a reusable binding, and a collection of ports define a service. Hence, a WSDL file uses the following elements in the definition of network services:

- **Types**– a container for data type definitions using some type system (such as XSD).
- **Message**– an abstract, typed definition of the data being communicated.
- **Operation**– an abstract description of an action supported by the service.
- **Port Type**–an abstract set of operations supported by one or more endpoints.
- **Binding**– a concrete protocol and data format specification for a particular port type.
- **Port**– a single endpoint defined as a combination of a binding and a network address.
- **Service**– a collection of related endpoints.

2.3. OTA WSDL Creation

A WSDL file consists of two logical sections, namely the interface definition section and the implementation binding section. The interface definition section is an abstract interface definition that describes data types and messages supported by the service. The implementation binding section defines the concrete implementation specifics such as the transport and binding information for a given service. While the OTA cannot provide the implementation specific second portion of the WSDL file, it is possible to create the interface definition portion of the WSDL file.

It is an acceptable practice in WSDL design to have separate WSDL files for each of the aforementioned sections of a WSDL. In this way, an implementation specific WSDL file can refer to a common Interface Definition WSDL file, such as that published by the OTA.

[2] Taken from the W3C WSDL 1.1 specification.

2.3.2 Building Modular WSDL

For the same reasons that the OTA chose to utilize a modular approach to schema design, the Architecture sub-committee recommends a similar modular approach to WSDL design. Below is a summary of the components of modular WSDL design:

WSDL includes a robust import mechanism to enable web services to be built in a modular fashion.

- Three important considerations:
  - Define the XML Schemas independent from the WSDL
  - Separate your WSDL into modular components
  - Leverage import to tie everything together
- Key benefits of this approach:
  - Easier to manage and maintain
  - Fits model where XSD/WSDL are created independently
  - Addresses situations where XML schemas already exist
  - Increases reusability of schemas across projects
- Use import to separate WSDL into modular components:

Figure 1: Schema\WSDL Modular Design

- WSDL files are organized into the following three components:
  - XML schemas are placed in the 1st file
  - WSDL message abstractions are placed in a 2nd file
  - Service bindings are placed in a 3rd file
- How import is leveraged here:
  - Service bindings would import the message definitions
    - Uses <**wsdl:import**> to import definitions
  - Message definitions would import the schemas
    - Uses **<xs:import>** to import definitions

- Overall, this approach can greatly improve component reusability
    - e.g., could provide multiple service bindings or URL locations for the same WSDL operations

2.3.3 Creating an Interface Definition WSDL

The abstract service interface portion of the WSDL can be defined by the OTA, since it refers only to the data types and message operations already defined by OTA schemas. This portion of the WSDL can generically define the interface for multiple service implementations. It should be noted that some partners may choose to define their own Interface Definition WSDL if they are using customized usage profile schemas and\or including a unique combination of OTA message schemas within the same WSDL file.

The sample below shows an Interface Definition WSDL for a Vehicle Reservation message. While the OTA schemas themselves can be included in the <types> section of the WSDL, the schemas have been imported (referenced) in order to reduce the size and complexity of the WSDL, and to adhere to modular WSDL design. In the example below, the import element is qualified as “xs:import” where the xs: prefix is associated with the W3C Schema namespace. As stated earlier, the WSDL specification allows for importing of both schema files and other WSDL files. Since in this case the imported file is a schema, the WS-I recommends that the “import” element be qualified with a namespace prefix indicating that the import statement refers to the W3C Schema xs:import rather than a WSDL wsdl:import.


```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
                  xmlns:ota="http://www.opentravel.org/OTA/2003/05"
                  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
                  xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
                  xmlns:xs="http://www.w3.org/2001/XMLSchema"
                  targetNamespace="http://www.opentravel.org/OTA/2003/05"
                  name="VehReservationService">

        <!-- Define data types (import OTA schemas) -->
        <wsdl:types>
                <xs:schema>
                        <xs:import namespace="http://www.opentravel.org/OTA/2003/05"
                        schemaLocation="OTA_VehResRQ.xsd"/>
                </xs:schema>
                <xs:schema>
                        <xs:import namespace="http://www.opentravel.org/OTA/2003/05"
                        schemaLocation="OTA_VehResRS.xsd"/>
                </xs:schema>
        </wsdl:types>

        <!-- Define request and response messages-->
        <wsdl:message name="VehicleReservationRequest">
                <wsdl:part name="OTA_VehResRQ" element="ota:OTA_VehResRQ"/>
        </wsdl:message>
        <wsdl:message name="VehicleReservationResponse">
                <wsdl:part name="OTA_VehResRS" element="ota:OTA_VehResRS"/>
        </wsdl:message>

        <!-- Define operation and reference messages-->
        <wsdl:portType name="VehicleReservationPortType">
                <wsdl:operation name="OTA_VehResRQ">
                        <wsdl:input message="ota:VehicleReservationRequest"/>
                        <wsdl:output message="ota:VehicleReservationResponse"/>
                </wsdl:operation>
        </wsdl:portType>
</wsdl:definitions>
```

Figure 2: Sample OTA Interface Definition WSDL

2.3.4 Creating an Implementation Binding WSDL

The concrete implementation portion of the WSDL will be specific to every service implementer, and therefore, every implementation will have a unique Implementation Binding WSDL. The Implementation Binding WSDL can import the Interface Definition WSDL, as shown in the example below. It should be noted that the "import" element in this WSDL refers to the "WSDL" namespace rather than the "Schema" namespace. In the following example, the default namespace is the WSDL namespace, therefore it is not necessary to explicitly qualify the element with a pre-defined 'wsdl:' prefix. While each WSDL Implementation file is unique, and therefore not published by the OTA, the following sample can serve as an example for OTA implementers.

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions xmlns="http://schemas.xmlsoap.org/wsdl/"
             xmlns:ota="http://www.opentravel.org/OTA/2003/05"
             xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
             xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
             xmlns:xs="http://www.w3.org/2001/XMLSchema"
             targetNamespace="http://www.opentravel.org/OTA/2003/05">

        <!-- Import Interface Definition WSDL-->
        <import namespace="http://www.opentravel.org/OTA/2003/05" location="OTA_VehResInterface.wsdl"/>

        <!-- Define SOAP binding-->
        <binding name="VehicleReservationBinding" type="ota:VehicleReservationPortType">
                <!-- Use document style and not rpc-->
                <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
                <operation name="OTA_VehResRQ">
                        <!-- Use 'literal' to include OTA XML as-is-->
                        <soap:operation soapAction="CreateReservation" style="document"/>
                        <input>
                                <soap:body use="literal" namespace="http://www.opentravel.org/OTA/2003/05"/>
                        </input>
                        <output>
                                <soap:body use="literal" namespace="http://www.opentravel.org/OTA/2003/05"/>
                        </output>
                </operation>
        </binding>

        <!-- Define SOAP interface with previously declared binding-->
        <service name="OTAVehicleReservationService">
                <port name="VehicleReservationPort" binding="ota:VehicleReservationBinding">
                        <!-- Replace "http://mydomain/myservicename" with actual service endpoint-->
                        <soap:address location="http://mydomain/myservicename"/>
                </port>
        </service>
</definitions>
```

Figure 3: Sample Implementation Binding WSDL

2.4 Implications to Toolkit Use\Test Results

Please refer to the OTA Implementers Forum hosted on "Google Groups" for WSDL\Schema issues related to specific toolkits. The forum can be found at:

http://groups.google.com/group/OTA-Impl-Forum?lnk=oa

3. OTA WSDL Reference

The following section of this document provides additional detail on the individual sections of a WSDL file with special attention paid to usage rules. Section 3.1 provides general guidance regarding the use of the WSDL recommendation as issued by the W3C. Section 3.2 builds on that guidance by describing the best practices for applying the WSDL recommendation within OTA implementations.

3.1. W3C WSDL Usage Breakdown

Section 3.1 of this document expands on section 2.2 (<<section title>>) by further specifying each section of a WSDL file and its usage in relation to the other WSDL sections. The information contained in this section of the document describes the generic makeup of a WSDL file and is not necessarily based on the OTA best practices outlined in this document. Section 3.2 provides specific examples for how these WSDL sections appear within an OTA implementation.

3.1.1. Interface Definition WSDL Section

The interface definition section of a WSDL file is comprised of the following parts.

- *definitions*—The wsdl:definitions element MUST be the root element of a WSDL file and defines the name of the web service in addition to serving as a container for the other WSDL sections. Additionally, all namespaces used within the WSDL file SHOULD be declared here.

- *types*—The wsdl:types element describes all the data types used in the message request and response payload. By default, the W3C Schema language is used to define these types, although this is not mandatory. If the service uses only XML Schema built-in simple types, such as strings and integers, the types element is not required. For document literal services, however, this section of a WSDL file SHOULD be used to define the request and response message payloads by referencing the appropriate XML Schemas.

- *message*—The wsdl:message element describes a single request or response message. It defines the name of the message and contains zero or more message part elements, which can refer to message parameters or message return values. For document literal services, both request and response messages SHOULD be defined and reference the appropriate XML Schema(s) from the wsdl:types section of the WSDL file.

- *portType*—The wsdl:portType element combines multiple wsdl:message elements to form a complete one-way or round-trip operation. For example, a portType can combine one request and one response message into a single request/response operation, most commonly used in SOAP services. Each wsdl:portType element can (and frequently does) define multiple wsdl:operation elements.

3.1.2. Implementation Binding WSDL Sections

The implementation binding section of a WSDL file is comprised of the following parts.

- *binding*—The wsdl:binding element specifies how the service will be implemented on the wire. The WSDL recommendation includes built-in extensions for defining SOAP services. Therefore, SOAP-specific information MUST be defined within the wsdl:binding element.
- *service*—The wsdl:service element defines the Internet Protocol (IP) address for invoking the specified service. Most commonly, this includes a resolvable URL for invoking the service.

3.2. OTA WSDL Usage Breakdown

Section 3.2 of this document expands on each section of a WSDL file by specifying its usage within OTA-based implementations.

3.2.1. Interface Definition WSDL Rules

This section describes the rules for the interface definition section of a WSDL file. Generic Interface Definition WSDL files can be created to define a reusable service interface. These Interface Definition WSDL files can be imported by other Implementation Binding WSDL files, which define the implementation specific details (e.g., SOAP, HTTP).

Definitions

§ 1. The wsdl:definitions element MUST define a namespace prefix for the OTA namespace.

```
xmlns:ota="http://www.opentravel.org/OTA/2003/05"
```

§ 2. The namespace declaration MAY be in the form of a default WSDL file namespace or an explicitly defined namespace prefix.

```
xmlns="http://schemas.xmlsoap.org/wsdl/"
                    OR
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
```

§ 3. The targetNamespace attribute MAY reference the OTA namespace.

types

§ 4. *The wsdl:types element MUST reference the relevant OTA XML Schemas (e.g., request, response, acknowledgement).*

§ 5. *The wsdl:types element MUST reference the relevant OTA XML Schemas (e.g., request, response, acknowledgement).*

§ 6. *XML Schemas MUST be included using the xs:import element rather than the wsdl:import element (prefixed using the XML Schema namespace rather than the WSDL namespace).*

§ 7. *The xs:import element SHOULD be referenced from within an xs:schema element.*

§ 8. *The xs:import element SHOULD include a namespace attribute referencing the OTA namespace.*

§ 9. The schemaLocation attribute of the xs:import element MUST reference the applicable OTA XML Schema by name and MAY include the fully qualified file path or URL to the WSDL file (If a URL reference is used it MUST NOT reference the OTA Online Schemas which are available for reference only)[3].

```
<wsdl:types>
        <xs:schema>
                <xs:import namespace="http://www.opentravel.org/OTA/2003/05"
                schemaLocation="OTA_VehResRQ.xsd"/>
        </xs:schema>
        <xs:schema>
                <xs:import namespace="http://www.opentravel.org/OTA/2003/05"
                schemaLocation="OTA_VehResRS.xsd"/>
        </xs:schema>
</wsdl:types>
```

[3] http://www.opentravel.org/online_schema.cfm

message

§ 10. Each message (e.g., request, response, acknowledgement) SHOULD be defined within a wsdl:message element.

§ 11. The attribute named element within the wsdl:part element MUST reference the root node of the appropriate OTA Schema, which is defined by a wsdl:message element, and be prefixed with the OTA namespace.

```
<wsdl:message name="VehicleReservationRequest">
        <wsdl:part name="OTA_VehResRQ" element="ota:OTA_VehResRQ"/>
</wsdl:message>
<wsdl:message name="VehicleReservationResponse">
        <wsdl:part name="OTA_VehResRS" element="ota:OTA_VehResRS"/>
</wsdl:message>
```

portType

§ 12. The wsdl:input and wsdl:output elements of the wsdl:operation element MUST contain a message attribute which references the OTA request and response messages defined in the name attribute of a wsdl:message element.

§ 13. Multiple wsdl:operation elements MAY be used for services that implement multiple operations.

```
<wsdl:portType name="VehicleReservationPortType">
        <wsdl:operation name="OTA_VehResRQ">
                <wsdl:input message="ota:VehicleReservationRequest"/>
                <wsdl:output message="ota:VehicleReservationResponse"/>
        </wsdl:operation>
</wsdl:portType>
```

3.2.2 Implementation Binding WSDL Rules

This section describes the rules for the Implementation Binding section of a WSDL file, which is specific to each implementation. The OTA provides the following information for creating a SOAP based Implementation Binding WSDL merely as a guideline.

If the Implementation Binding WSDL is a separate document referencing the Interface Definition WSDL, the referenced WSDL file should be imported as follows:

§ 14. Other WSDL files MUST be included using the WSDL wsdl:import element rather than the Schema xs:import element (Prefixed using the WSDL namespace rather than the Schema namespace).

§ 15. The namespace attribute of the wsdl:import element SHOULD reference the namespace of the imported Interface Definition WSDL.

§ 16. The location attribute of the wsdl:import element MUST reference the Interface Definition WSDL by name, and MAY include the fully qualified file path or URL to the WSDL.

```
<wsdl:import namespace="http://www.opentravel.org/OTA/2003/05" location="OTA_VehResInterface.wsdl"/>
```

binding

§ 17. The type attribute of the wsdl:binding element MUST reference the name attribute of the wsdl:portType.

§ 18. The style attribute of the wsdl:binding element SHOULD be "document" and not "RPC".

§ 19. For SOAP based services, the use attribute of the soap:body element SHOULD be "literal" and not "encoded".

```
<wsdl:binding name="VehicleReservationBinding" type="ota:VehicleReservationPortType">

        <!-- Use document style and not rpc-->
        <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
        <wsdl:operation name="OTA_VehResRQ">

        <!-- Use 'literal' to include OTA XML as-is-->
        <soap:operation soapAction="CreateReservation" style="document"/>
                <wsdl:input>
                        <soap:body use="literal" namespace="http://www.opentravel.org/OTA/2003/05"/>
                </wsdl:input>
                <wsdl:output>
                        <soap:body use="literal" namespace="http://www.opentravel.org/OTA/2003/05"/>
                </wsdl:output>
        </wsdl:operation>
</wsdl:binding>
```


service

§ 20. The binding attribute of the wsdl:port element MUST reference the name attribute of the wsdl:binding element.

§ 21. For SOAP based services, the location attribute of the soap:address element MUST be a resolvable URL to the actual service endpoint.

```
<wsdl:service name="OTAVehicleReservationService">
        <wsdl:port name="VehicleReservationPort" binding="ota:VehicleReservationBinding">

                <!-- Replace "http://mydomain/myservicename" with actual service endpoint-->
                <soap:address location="http://mydomain/myservicename"/>
        </wsdl:port>
</wsdl:service>
```


4. Examples

Following are additional examples of WSDL files for OTA services, which will illustrate different WSDL techniques. The examples shown in figures 2 and 3 are examples of properly formatted Interface Definition and Implementation Binding WSDL files.

4.1. Consolidated Interface Definition and Implementation Binding WSDL

The following example shows a complete WSDL where both the interface definition and the implementation binding portions are contained within the same WSDL file. While this is an acceptable WSDL practice, the OTA recommends the aforementioned modular WSDL design when a generic Interface Definition WSDL file exists. This practice allows the Interface Definition WSDL file to be separated from the implementation specific binding WSDL and thereby remain reusable.


```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
                  xmlns:ota="http://www.opentravel.org/OTA/2003/05"
                  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
                  xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
                  xmlns:xs="http://www.w3.org/2001/XMLSchema"
                  targetNamespace="http://www.opentravel.org/OTA/2003/05"
                  name="VehCancelService">

        <!-- Define data types (import OTA schemas) -->
        <wsdl:types>
                <xs:schema>
                        <xs:import namespace="http://www.opentravel.org/OTA/2003/05"
                  schemaLocation="OTA_VehCancelRQ.xsd"/>
                  </xs:schema>
                  <xs:schema>
                        <xs:import namespace="http://www.opentravel.org/OTA/2003/05"
                  schemaLocation="OTA_VehCancelRS.xsd"/>
                  </xs:schema>
        </wsdl:types>

        <!-- Define request and response messages-->
        <wsdl:message name="VehicleCancelRequest">
                <wsdl:part name="OTA_VehCancelRQ" element="ota:OTA_VehCancelRQ"/>
        </wsdl:message>
        <wsdl:message name="VehicleCancelResponse">
                <wsdl:part name="OTA_VehCancelRS" element="ota:OTA_VehCancelRS"/>
        </wsdl:message>

        <!-- Define operation and reference messages-->
        <wsdl:portType name="VehicleCancelPortType">
                <wsdl:operation name="OTA_VehCancelRQ">
                        <wsdl:input message="ota:VehicleCancelRequest"/>
                        <wsdl:output message="ota:VehicleCancelResponse"/>
                </wsdl:operation>
        </wsdl:portType>

        <!-- Define SOAP binding-->
        <wsdl:binding name="VehicleCancelBinding" type="ota:VehicleCancelPortType">

                <!-- Use document style and not rpc-->
                <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
                <wsdl:operation name="OTA_VehCancelRQ">
                        <!-- Use 'literal' to include OTA XML as-is-->
                        <soap:operation soapAction="CancelReservation" style="document"/>
                        <wsdl:input>
                                <soap:body use="literal" namespace="http://www.opentravel.org/OTA/2003/05"/>
                        </wsdl:input>
                        <wsdl:output>
                                <soap:body use="literal" namespace="http://www.opentravel.org/OTA/2003/05"/>
                        </wsdl:output>
                </wsdl:operation>
        </wsdl:binding>

        <!-- Define SOAP interface with previously declared binding-->
        <wsdl:service name="OTAVehicleCancelService">
                <wsdl:port name="VehicleCancelPort" binding="ota:VehicleCancelBinding">

                        <!-- Replace "http://mydomain/myservicename" with actual service endpoint-->

                        <soap:address location="http://mydomain/myservicename"/>
                </wsdl:port>
        </wsdl:service>
</wsdl:definitions>
```

Figure 4: Consolidated WSDL (Interface and Binding)

4.2. Incorrect Schema Import

The following example shows an incorrect method for importing schemas into a WSDL file. Unlike importing a schema from within another schema, an XML Schema import within a WSDL file should be contained within the “Types\Schema” section of the WSDL file (see figure 2 for the correct method).

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
                  xmlns:ota="http://www.opentravel.org/OTA/2003/05"
                  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
                  xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
                  xmlns:xs="http://www.w3.org/2001/XMLSchema"
                  targetNamespace="http://www.opentravel.org/OTA/2003/05"
                  name="VehReservationService">

        <!-- This is a Schema style import not a WSDL style import. -->
        <xs:import schemaLocation="OTA_VehResRQ.xsd"/>
        <xs:import schemaLocation="OTA_VehResRS.xsd"/>

        <!-- Schemas should be imported within the unused Types section below. -->
        <wsdl:types/>

        <!-- Define request and response messages-->
        <wsdl:message name="VehicleReservationRequest">
                <wsdl:part name="OTA_VehResRQ" element="ota:OTA_VehResRQ"/>
        </wsdl:message>
        <wsdl:message name="VehicleReservationResponse">
                <wsdl:part name="OTA_VehResRS" element="ota:OTA_VehResRS"/>
        </wsdl:message>

        <!-- Define operation and reference messages-->
        <wsdl:portType name="VehicleReservationPortType">
                <wsdl:operation name="OTA_VehResRQ">
                        <wsdl:input message="ota:VehicleReservationRequest"/>
                        <wsdl:output message="ota:VehicleReservationResponse"/>
                </wsdl:operation>
        </wsdl:portType>
```

Figure 5: Incorrect Schema Import Example

4.3. Flattened Schema Contained Within WSDL

The example shown in figure six illustrates two different issues that are not recommended as best practices. Please note that the example in figure six contains partial OTA schemas due to space limitations.

4.3.1. Use of Flattened Schemas

Although the OTA now publishes flattened[4] versions of their schemas which are designed to counteract some performance issues related to the OTA's modular schema design, it is not suggested that these flattened schemas be used within a WSDL. The use of flattened schemas in WSDL (either imported or contained within the WSDL file) can cause problems for some data binding technologies. Data binding tools may return a duplicate data type error due to the fact that multiple WSDL schemas (request and response schemas for example) now contain repeated common data types due to the flattening process. Since the OTA uses a single namespace for all messages, the common data types copied in each self contained, flattened, schema are, therefore, considered to be duplicates. By referencing a single common type schema from within the main message schema, this problem can usually be avoided.

4.3.2. Schemas Contained Within the WSDL File

Aside from creating an unwieldy WSDL, this practice will usually lead to the use of flattened schemas in order to avoid the creation of a huge WSDL file that contains unused OTA common types. For the sake of clarity and to avoid the issues inherent to the use of flattened schemas, it is recommended that OTA WSDL files utilize “import” to reference all schemas.

[4] A flattened schema is a schema where all “included” schema common types that are referenced within the parent message schema, are moved into the parent schema file to create a self contained schema without references or unused types.


```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
                  xmlns:ota="http://www.opentravel.org/OTA/2003/05"
                  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
                  xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
                  xmlns:xs="http://www.w3.org/2001/XMLSchema"
                  targetNamespace="http://www.opentravel.org/OTA/2003/05"
                  name="VehReservationService">

        <!-- Define data types (import OTA schemas) -->
        <wsdl:types>
                <!-- Flattened request schema-->
                <xs:schema xmlns="http://www.opentravel.org/OTA/2003/05"
                           xmlns:xs= "http://www.w3.org/2001/XMLSchema"
                           targetNamespace="http://www.opentravel.org/OTA/2003/05"
                           elementFormDefault="qualified" version="1.000" id="OTA2003A">
                        <xs:element name="OTA_VehRetResRQ">
                                <xs:annotation>
                                        <xs:documentation>The root tag of OTA_VehRetResRQ
contains…</xs:documentation>
                                </xs:annotation>

                                <!-- Contents of flattened schema truncated for this example-->

                </xs:schema>

                <!-- Flattened response schema-->
                <xs:schema xmlns="http://www.opentravel.org/OTA/2003/05"
                           xmlns:xs="http://www.w3.org/2001/XMLSchema"
                           targetNamespace="http://www.opentravel.org/OTA/2003/05"
                           elementFormDefault="qualified" version="1.000" id="OTA2003A">
                        <xs:element name="OTA_VehRetResRS">
                                <xs:annotation>
                                        <xs:documentation>The root tag of OTA_VehRetResRS
contains…</xs:documentation>
                                </xs:annotation>

                                <!-- Contents of flattened schema truncated for this example-->

                </xs:schema>
        </wsdl:types>

        <!-- Define request and response messages-->
        <wsdl:message name="VehicleReservationRequest">
                <wsdl:part name="OTA_VehResRQ" element="ota:OTA_VehResRQ"/>
        </wsdl:message>
        <wsdl:message name="VehicleReservationResponse">
                <wsdl:part name="OTA_VehResRS" element="ota:OTA_VehResRS"/>
        </wsdl:message>

        <!-- Define operation and reference messages-->
        <wsdl:portType name="VehicleReservationPortType">
                <wsdl:operation name="OTA_VehResRQ">
                        <wsdl:input message="ota:VehicleReservationRequest"/>
                        <wsdl:output message="ota:VehicleReservationResponse"/>
                </wsdl:operation>
        </wsdl:portType>
        .
        .
```

Figure 6: Flattened Schema Example