

Introduction to OpenTravel 2.0

OpenTravel 2.0 Mechanics

NOTICE: The content in this presentation is **DRAFT** and is subject to change before the initial 2.0 Publication. It is **NOT** advisable to use any of these materials from an implementation planning perspective.

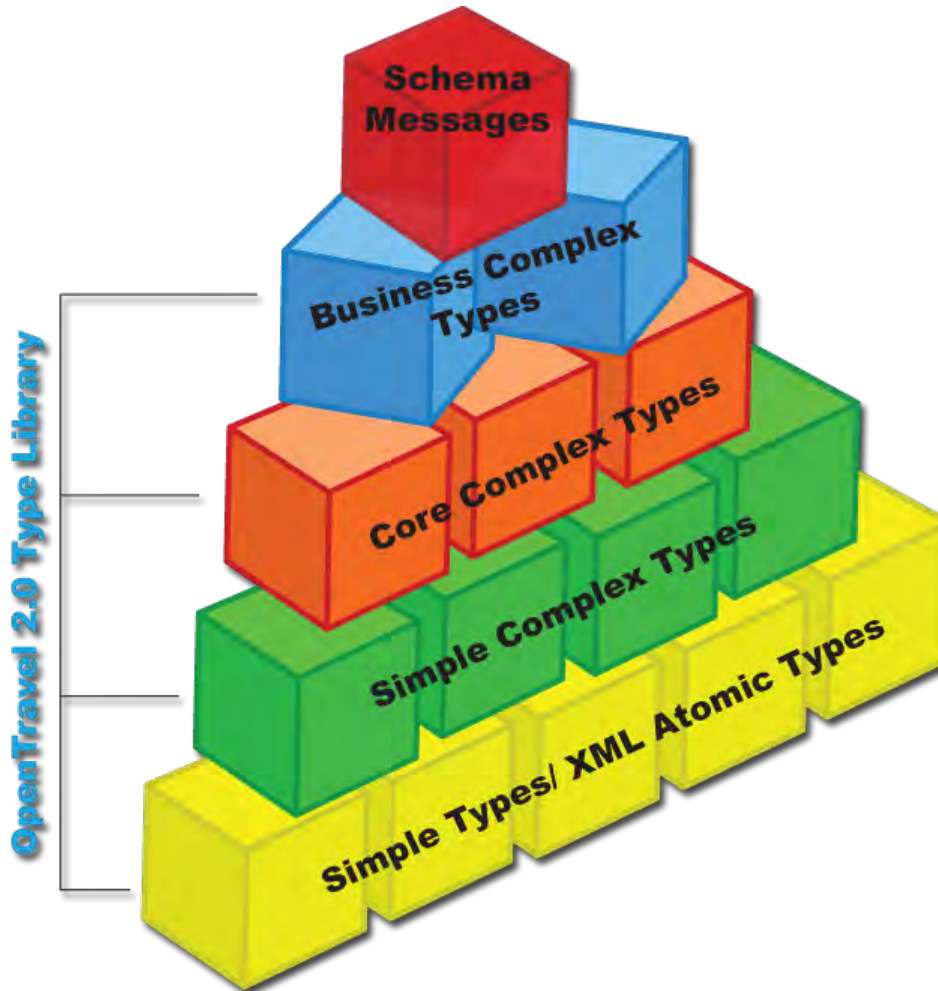
Let's Discuss...

2

- **OpenTravel 2.0 Mechanics**
 - ▣ The Big Picture - Physical Structure of Type Library
 - ▣ Naming Conventions
 - ▣ Type Library Construction
 - Simple Types/ XML Atomic Data Types
 - Code Lists & Enumerations
 - Simple Complex Types
 - Core Complex Types
 - Business Complex Types
 - ▣ Namespaces
 - ▣ Extensibility
- **Demonstrations**

The Big Picture

2.0 Type Library Construction

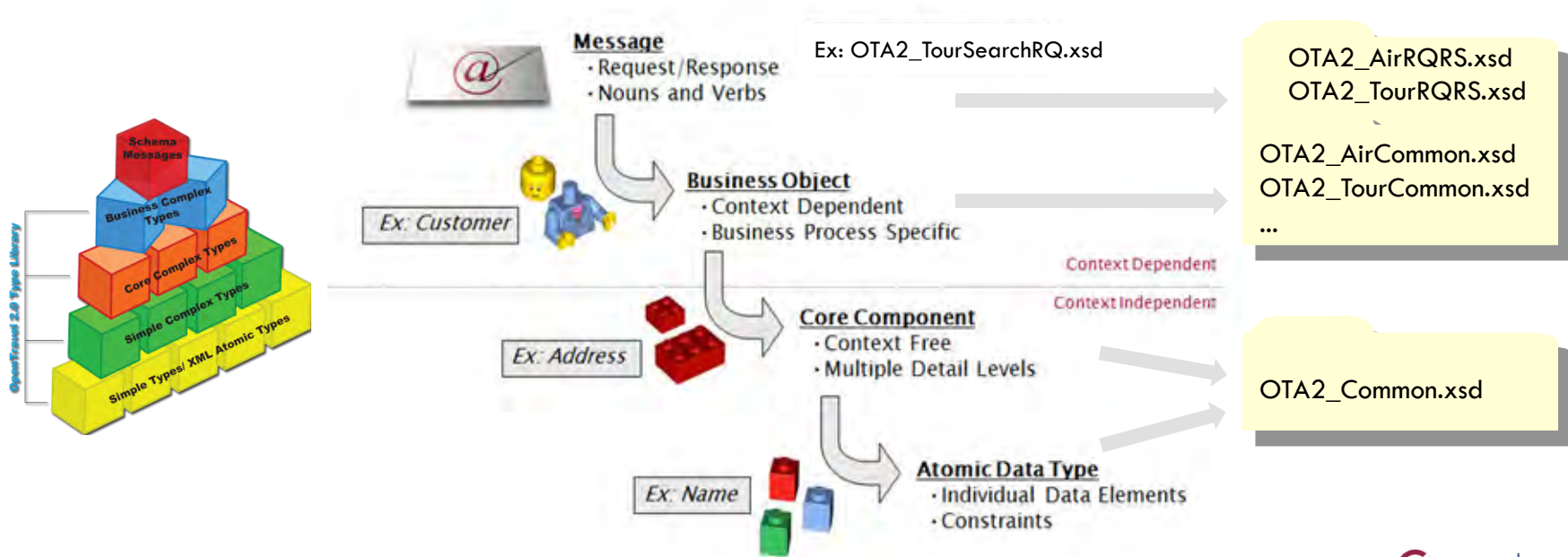


The Big Picture

2.0 Type Library Construction

5

- **Simple and Core types** in common xsd file
- **Business Objects** in segment common xsd files
- **Messages** in message group xsd file



Naming Conventions

- goal: standardized xml vocabulary
- role designation in element design
- mixed case
- compound words

2.0 Naming Conventions (1/4)

7

□ Goal: standardized XML vocabulary

□ Supports:

- Define **once** and **reuse** everywhere
- Improved design modularity and consistency
- 2.0 xml components have consistent naming and structural patterns
 - This improves learning and use

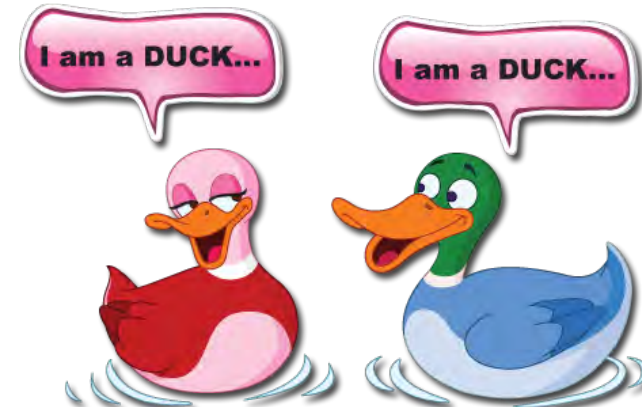
Differentiating Structural Components and Applying Best Practice Guidelines is Essential

A key factor in differentiating structural components, and creating guidelines that apply to them, is evident in how element declarations are different than type definitions. For example, elements describe where the data can occur in a message while type definitions define constraints on the data format in those messages. The distinction between elements and types is an important factor when creating a vocabulary designed to encourage reuse. One distinction, for example, is between the role versus the structure in that structurally, an airport is an airport regardless if the role of the airport is the origin airport, a connection airport or the final destination airport. Another example is a flight, regardless if it is incoming, onward, scheduled or departed--it is still a flight.

2.0 Naming Conventions (2/4)

8

- **Role designation in element design**
 - ▣ To describe the role of an element, use the element name when the element is a simple type. Add a "role" attribute by extension when the element is a complex type.
 - Provides a distinction between role and structure
 - An Airport code is an airport code whether it's representing an Origin or Destination



2.0 Naming Conventions (3/4)

9

□ Mixed case

- Use mixed case for element, attribute and type names.
 - For elements and types, use the leading character of each word in upper case and the remainder in lower case without the use of hyphens or spaces between words
 - For attributes, use Lower Camel Case (LCC) with the initial word all in lower case.

This format increases readability and is consistent with common industry practices. Lower case attributes improve readability and help differentiate elements from attributes.

Item	Case
Attribute	lowerCamelCase
Element	UpperCamelCase
Simple Type	UpperCamelCase
Simple Complex Type	UpperCamelCase
Core Complex Type	UpperCamelCase
Business Complex Type	UpperCamelCase

2.0 Naming Conventions (4/4)

10

□ Compound words

- Names should only be compound words when the real-world object they describe has a compound name
 - Or if needed to distinguish between differentiate between related properties.
- Names should never contain the name of the parent structure or physical structure.
- Element names may occasionally need to be compound to distinguish between elements with different roles but the same structure.

Types will be more reusable when they clearly describe specific real world objects and avoid synthetic types created for a single schema. The description of an item should not vary based on the role that item plays in the specific context, for example a person is a person regardless if they are the customer or contact.

Attribute	Permitted
Element	Permitted
Simple Type	Discouraged
Simple Complex Type	Discouraged
Core Complex Type	Discouraged
Business Complex Type	Discouraged

Component Architecture: Enumerations & Code Lists

2.0 Enumerations/ Code Lists (1/3)

12

□ Code Lists

- ▣ Used when there is a potential for 100+ choices
 - E.g. hotel amenities
- ▣ New 2.0 code lists/ code list items submitted online and incorporated into iterative releases

2.0 Enumerations/ Code Lists (2/3)

13

□ Closed Enumerations

- ▣ Used when a list of options are static
 - E.g. gender, day of week
- ▣ Used when an enumeration is actionable in a processing environment
 - E.g. target processing system (production/test)

2.0 Enumerations/ Code Lists (3/4)

14

□ Open Enumerations

- ▣ Contain a base of values that may be extended
- ▣ Used when a list of options are ≤ 100
- ▣ Used by implementers to exchange unique/ proprietary values with trading partners
- ▣ Do not require an iterative release

2.0 Enumerations/ Code Lists (4/4)

15

- Closed Enumeration
 - ▣ *“We want a version change before having a new ticket type.”*
 - ▣ *“Our programmers must change their code if ticket type changes”*
- Open Enumeration
 - ▣ Extendable
 - Base available
 - ▣ *“We know most seat types, but more are coming.”*

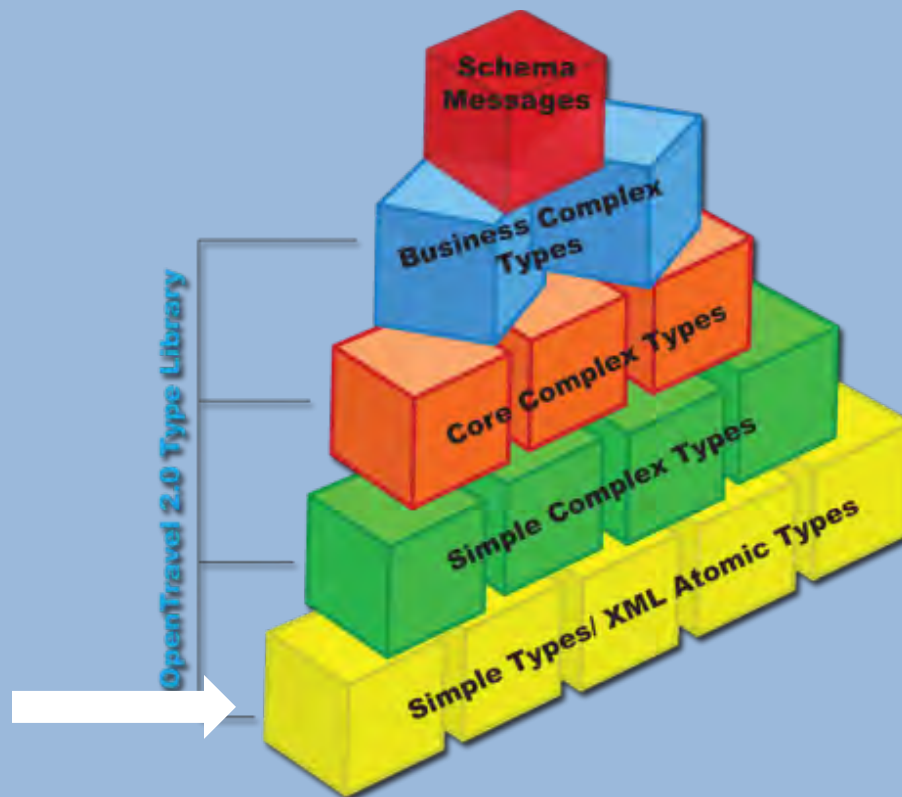
```
<xs:simpleType name="TicketType">
  <xs:restriction base="xs:string">
    <xs:enumeration value="eTicket"/>
    <xs:enumeration value="Paper"/>
    <xs:enumeration value="MCO"/>
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="SeatType_Base">
  <xs:restriction base="xs:string">
    <xs:enumeration value="Window"/>
    <xs:enumeration value="Aisle"/>
    <xs:enumeration value="Middle"/>
    <xs:enumeration value="Other_"/>
  </xs:restriction>
</xs:simpleType>

<xs:complexType name="SeatType">
  <xs:simpleContent>
    <xs:extension base="SeatType_Base">
      <xs:attribute type="xs:string"
        name="extension"/>
    </xs:extension>
  </xs:simpleContent>
</xs:complexType>
```

```
<SeatType extension="Sleeper">Other_</SeatType>
```

Component Architecture: Simple Types

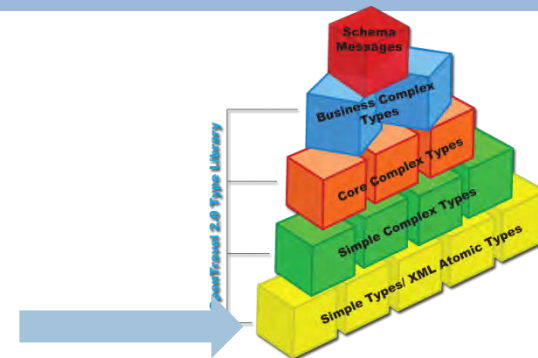


2.0 Simple Types (1/2)

17

- Constrain content of elements and attributes
- May have restrictions including patterns
- May be XML atomic (string, integer and date), list, or unions
- Named by role then variation to make them easier to find

- ☰ Code_Airport : xsd:string
- ☰ Code_Country : xsd:string
- ☰ Code_OTA : xsd:string
- ☰ Code_PaymentCard : Private_Paymer
- ☰ Code_Phone_Area : xsd:string
- ☰ Code_Phone_Country : xsd:string
- ☰ Code_Postal : xsd:string



```
<xsd:simpleType name="Code_Airport">  
  <xsd:restriction base="xsd:string">  
    <xsd:pattern value="[A-Z0-9]{3,5}"/>  
  </xsd:restriction>  
</xsd:simpleType>
```

```
<xsd:simpleType name="Code_Country">  
  <xsd:restriction base="xsd:string">  
    <xsd:pattern value="[a-zA-Z]{2}"/>  
  </xsd:restriction>  
</xsd:simpleType>
```

```
<xsd:simpleType name="Code_OTA">  
  <xsd:restriction base="xsd:string">  
    <xsd:pattern value="[0-9A-Z]{1,3}(\.[A-Z]{3}(\.X){0,1}){0,1}"/>  
  </xsd:restriction>  
</xsd:simpleType>
```

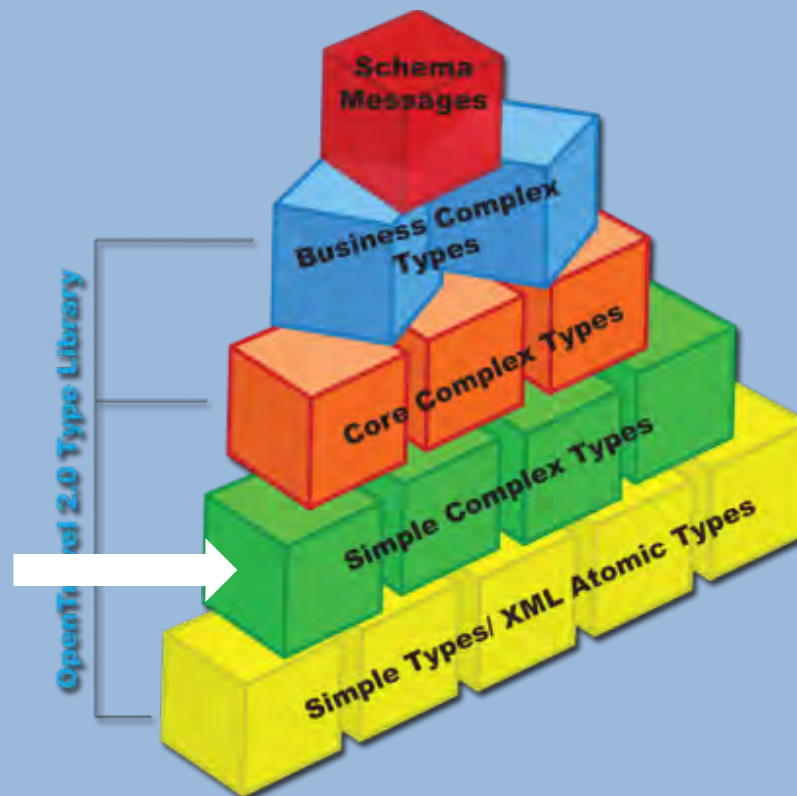
2.0 Simple Types (2/2)

18

- **OpenTravel 2.0 Simple Types are defined by role**
 - Money, Amount, Price
 - PhoneNumber, PaymentCardNumber
 - Code, Number
 - String (Tiny, Short, Medium, Long)
 - String_PersonName
 - Specific types including those defined by standards

- **OpenTravel 2.0 uses fewer string types**
 - Makes the library easier to use
 - Improves flexibility and interoperability
 - Truncate strings to fit your existing applications

Component Architecture: Simple Complex Types



2.0 Simple Complex Types

20

- Value is a single simple type
- Groups attributes related to the value

```
<Location alternateLocationInd="true" context="IATA"  
    multiAirportCityInd="false" name="Denver">DEN</Location>
```

```
<xsd:complexType name="Location">  
  <xsd:simpleContent>  
    <xsd:extension base="String.Token">  
      <xsd:attribute name="context" type="String_Short"/>  
      <xsd:attribute name="multiAirportCityInd" type="xsd:boolean"/>  
      <xsd:attribute name="alternateLocationInd" type="xsd:boolean"/>  
      <xsd:attribute name="name" type="String_Short"/>  
    </xsd:extension>  
  </xsd:simpleContent>  
</xsd:complexType>
```

Examples

Equipment

Fees

Locations

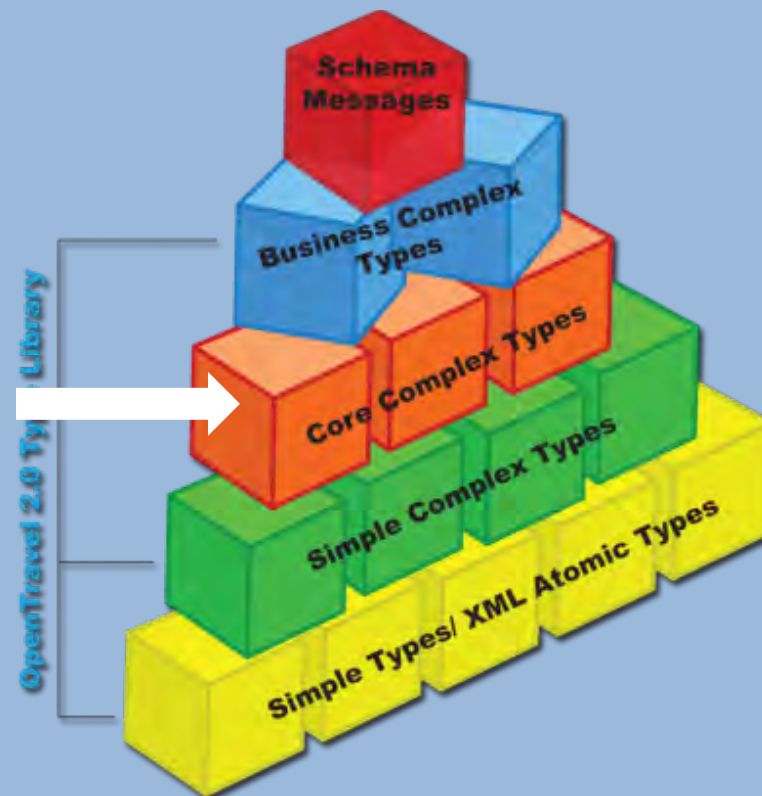
Loyalty Level

Operation Times

Seat

Tax

Component Architecture: Core Objects



2.0 Core Objects (1/5)

22

- **Core object types define a representation of common real world objects**
 - Same regardless of travel sector
 - Used *within* a business object
- **Core types have facets**
 - ***Simple*** – usable as attribute type
 - ***Summary*** – just enough to use the object (80/20)
 - ***Info*** – usage and associated information
 - ***Lists*** – more than one in a list with *Roles*

2.0 Core Objects (2/5)

23

- **Key characteristics**
 - ▣ Representation of *real world objects*
 - ▣ The same regardless of *context*
 - ▣ In 1.0 these types included related/associated information
 - ▣ In 2.0 we provide “facets” for the basic properties and extended information

Examples

Address

Company

Data Policy

Description

Email

Payment Card

Phone

Multimedia

2.0 Core Objects: Facets (3/5)

24

- 2.0 core object *facets* are defined for different purposes
 - {Core} – most common used properties
 - {Core}_{SimpleType} – attribute
 - {Core}_{SimpleType}_List
 - {Core}_Roles – how it is used
 - {Core}_Info – role and usage
 - {Core}_Info_List

Phone
Country 1
Area 555
345-9876

Phone_String
555-1010

Phone_String_List
555-122-0033,
555-543-5566

Phone_Roles
Home Work Cell

Phone_Info
555-122-2222
Home
Between 5 and 10pm
Do not SYNC

e Phone : Phone_Summary

e Phone_Info : Phone_Info

e Phone_Info_List : Phone_Info_List

e Phone_string : Phone_string

e Phone_string_List : Phone_string_List

Phone_Info
555-122-2222
Home
Between 5 and 10pm
Do not SYNC

2.0 Core Objects: Examples (4/5)

25

```
<xs:complexType name="Phone_Summary">
  <xs:simpleContent>
    <xs:extension base="Phone_string">
      <xs:attribute name="CountryCode" type="xs:string" />
      <xs:attribute name="AreaCode" type="xs:string" />
    </xs:extension>
  </xs:simpleContent>
</xs:complexType>
```

Phone

Country 1
Area 555
345-9876

```
<xs:complexType name="Phone_Info">
  <xs:sequence>
    <xs:element type="Phone_Summary" name="Phone"/>
    <xs:element type="Location" name="Location"/>
  </xs:sequence>
```

Phone_Info

555-122-2222
Home
Between 5 and 10pm
Do not SYNC

```
<xs:complexType name="Phone_Info_List">
  <xs:sequence>
    <xs:element type="Phone_Info" name="Phone_Info_List" maxOccurs="unbounded" />
  </xs:sequence>
</xs:complexType>
```

Phone_Info_List

555-122-2222
Home
Between 5 and 10pm
Do not SYNC

```
<xs:simpleType name="Phone_string">
  <xs:restriction base="xs:string"/>
</xs:simpleType>
type="xs:string" />
```

Phone_String

555-1010

```
<xs:simpleType name="Phone_string_List">
  <xs:list itemType="Phone_string"/>
</xs:simpleType>
```

Phone_String_List

555-122-0033,
555-345-5566

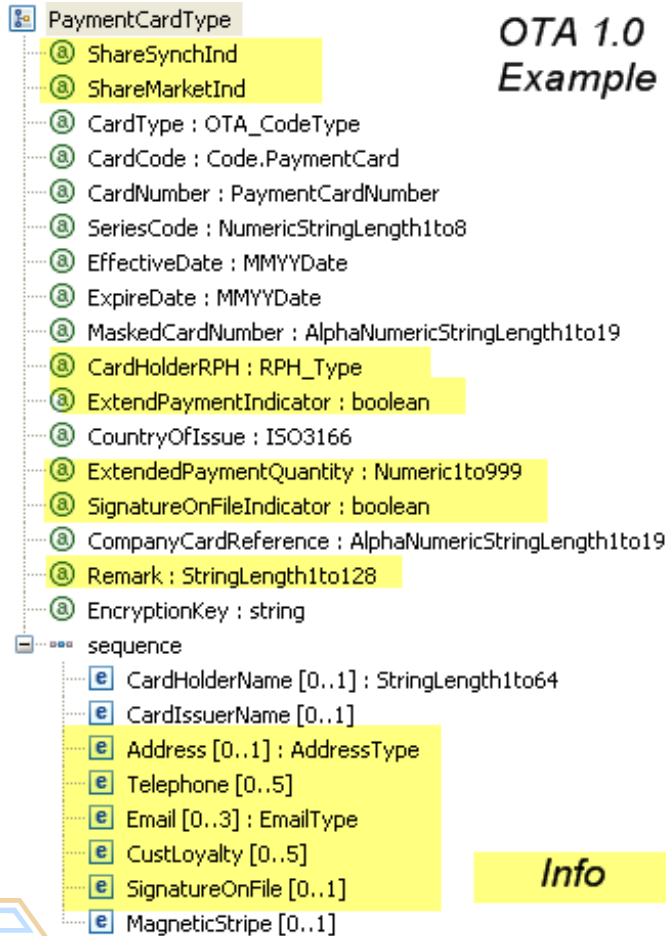
```
<xs:simpleType name="Phone_Roles">
  <xs:restriction base="xs:string">
    <xs:enumeration value="Home"/>
    <xs:enumeration value="Work"/>
    <xs:enumeration value="Cell"/>
  </xs:restriction>
</xs:simpleType>
```

Phone_Roles

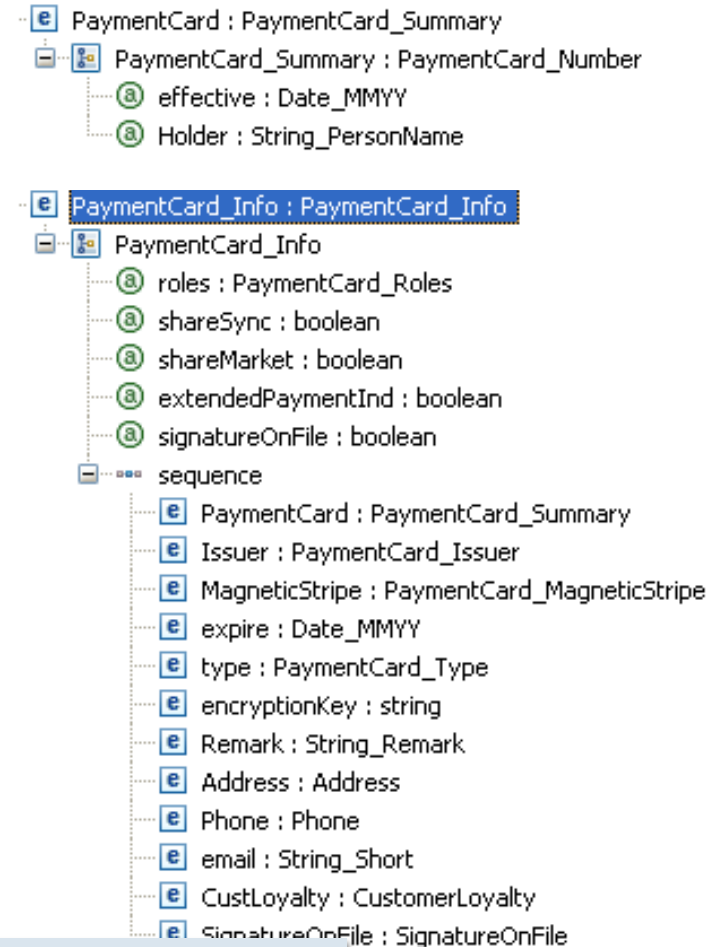
Home Work Cell

2.0 Core Objects: Examples (5/5)

26

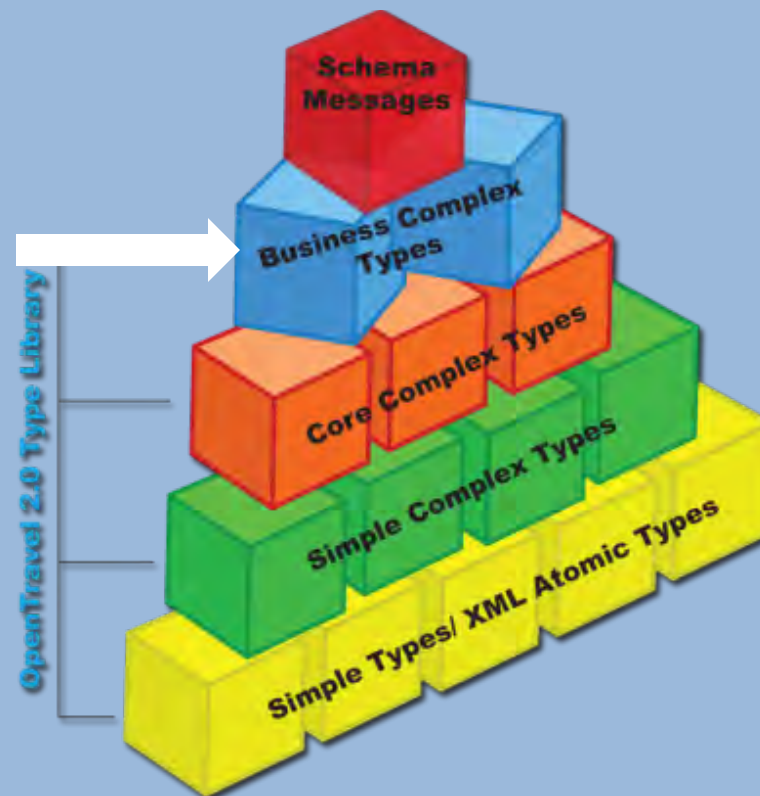


Info



Design Question – does Address, phone etc belong in the PaymentCard?
Should they be via summary or ID reference?

Component Architecture: Business Objects



2.0 Business Objects (1/x)

28

- **Single business object or concept represented in complex types**
 - Specific child elements and attributes varies with business context
 - In 1.0 one complex type represented business object
 - In 2.0 “facets” separate basic from detail
 - Allows less optional elements which improves interoperability

Examples

Preferences

O&D

Traveler Profiles

Products

Segments

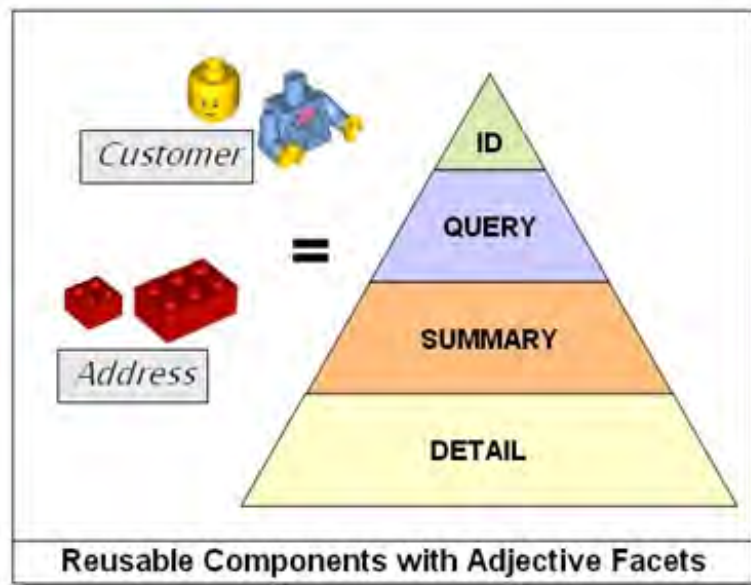
Itineraries

Etc.

2.0 Business Objects: Facets (2/x)

29

- Facets are designed for specific purposes
 - **ID** – identify a specific object
 - **Query** – used to search for matching objects
 - **Summary** – most commonly needed information
 - **Detail** – everything



- .. e Profile_Traveler : Profile_Traveler_Summary
- .. e Profile_Traveler_Detailed : Profile_Traveler_Detailed
- .. e Profile_Traveler_ID : Profile_Traveler_ID
- .. e Profile_Traveler_Query : Profile_Traveler_Query
- .. e Profile_Traveler_Summary : Profile_Traveler_Summary

- Additional facets can be used to capture data requirements for specific contexts

2.0 Business Objects: Example (3/x)

30

2.0 Business Object Construction

Definitions nest

- ID included in Summary
- Summary included in Detailed

Query

```
Profile_Traveler_Query  
ProfileID  
Authority  
Or  
Name  
Or  
Phone number
```

Profile_Traveler_ID

ProfileID
Authority

Profile_Traveler_Summary

Name
Address
Phone

Profile_Traveler_Detailed

Contact
Remarks

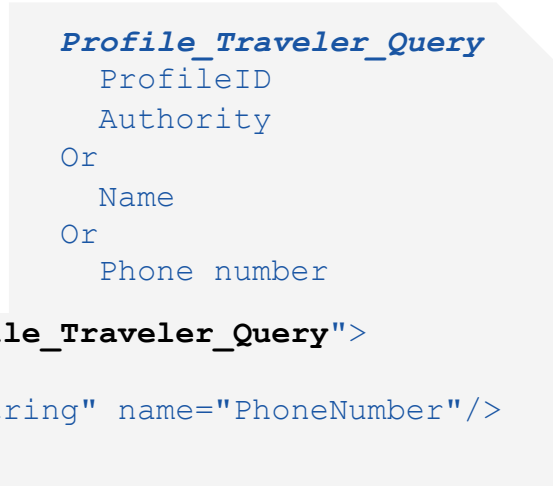
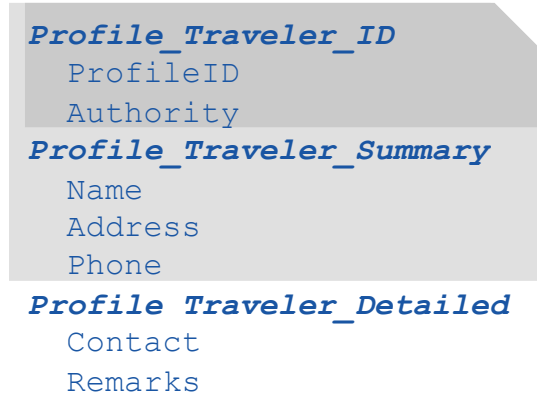
```
<Profile_Traveler_Detailed>  
  <ProfileID>R45682a</ProfileID>  
  <Authority>Acme</Authority>  
  <Name>Sue Traveler</Name>  
  <Address>3 Travers St,  
    Baldwinsville, NJ 87678</Address>  
  <Phone>555-666-7777</Phone>  
  <Contact>555-1212 555-445-3322</Contact>  
  <Remarks>Use first name</Remarks>  
</Profile_Traveler_Detailed>
```

2.0 Business Objects: Example (4/x)

```
<xs:complexType name="Profile_Traveler_ID">
  <xs:sequence>
    <xs:element type="xs:string" name="ProfileID"/>
    <xs:element type="xs:string" name="Authority"/>
  </xs:sequence>
</xs:complexType>
```

```
<xs:complexType name="Profile_Traveler_Summary">
  <xs:complexContent>
    <xs:extension base="Profile_Traveler_ID">
      <xs:sequence>
        <xs:element type="xs:string" name="Name"/>
        <xs:element type="xs:string" name="Address"/>
        <xs:element type="xs:string" name="Phone"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

```
<xs:complexType name="Profile_Traveler_Detailed">
  <xs:complexContent>
    <xs:extension base="Profile_Traveler_Summary">
      <xs:sequence>
        <xs:element type="xs:strin
        <xs:element type="xs:strin
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```



```
<xs:complexType name="Profile_Traveler_Query">
  <xs:sequence>
    <xs:element type="xs:string" name="PhoneNumber"/>
  </xs:sequence>
</xs:complexType>
```

2.0 Business Objects: Example (5/x)

32



Part of 1.0 OriginDestinationInformation

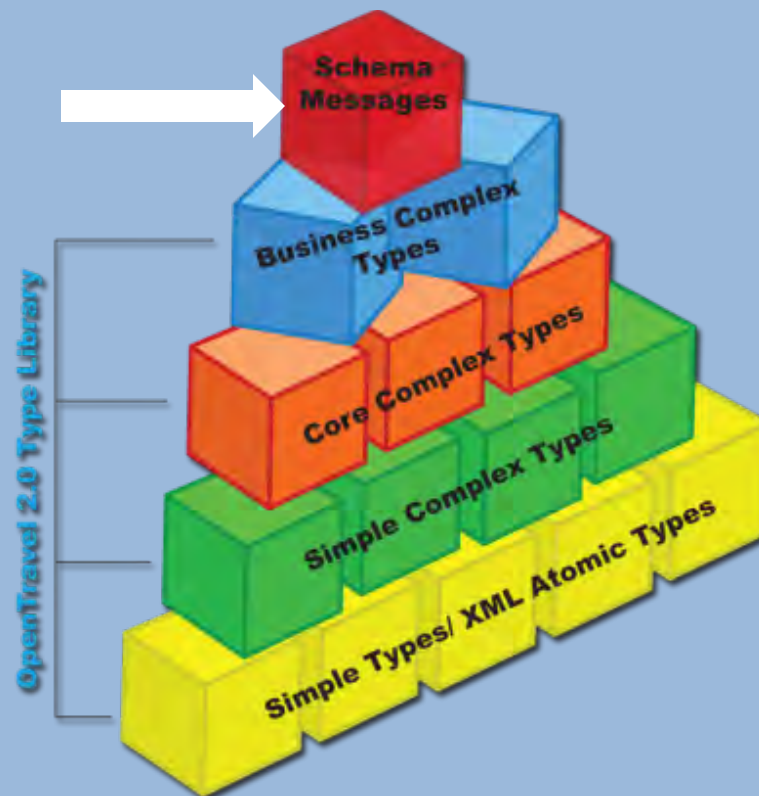
- "MultiAirportCityInd" If true, other airports within this city may be considered (i.e., EWR, JFK when origin location is LGA.)
- "AlternateLocationInd" If true, alternate locations may be considered
- "SameAirportInd" If true, the return departure must be from the same airport as the outbound arrival



Questions –

1. Is OandD a Core or Business Object?
 - Locations are different for hotels than airports.
2. Should it have travel preferences and flight info?

Component Architecture: Messages



2.0 Messages (1/4)

34

□ Common Envelope

- ▣ Allows common handling for Security, message identification, etc

```
<OTA2_Envelope xmlns="http://www.OpenTravel.com/ns/OTA2/Library_v01_01" ...
  <Service operation="RQ" version="1.01">Ping</Service>
  <MessageHeader>
    <Identification timeStamp="2011-02-01T12:33:00" messageID="a334999433"/>
    <PointOfSale requestorID="aa11bb22" travelSegment="Air"/>
  </MessageHeader>
  <Air:OTA2_AirPingRQ/>
</OTA2_Envelope>
```

□ Payload is substitution head

- ▣ Extended for messages
- ▣ Allows generic code to handle messages

```
<xsd:element name="AirAvailRQ" type="AirAvailRQ" substitutionGroup="lib:OTA2_Payload" />
<xsd:complexType name="AirAvailRQ">
  <xsd:complexContent>
    <xsd:extension base="lib:OTA2_Payload">
      <xsd:sequence>
```

2.0 Messages (2/4)

35

□ ID, Global-ID, IDREF

- 1.0 used RPH
- 2.0 uses ID facets and XML ID
 - XML ID supported by tools
 - Global ID part of all business objects
 - Provides unique name and who named it

2.0 Messages: Namespaces (3/4)

36

Namespaces in a Message

- Namespaces identify the source of the structure
 - OTA_PingRQ is from the message schema
 - EchoData is from the library schema.

Message

```
<?xml version="1.0" encoding="UTF-8"?>
<otm:OTA_PingRQ
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.ot2.org/msgs/ping/v1_1 OTA_Ping_v1.1.xsd"
  xmlns:otm="http://www.ot2.org/msgs/ping/v1_1"
  xmlns="http://www.ot2.org/library/v1_1">

  <EchoData>The quick brown fox jumps over the lazy dog</EchoData>
  <TimeStamp>2010-10-05T16:29:00Z</TimeStamp>
</otm:OTA_PingRQ>
```

Message Schema v1.1

Message NS

Library NS

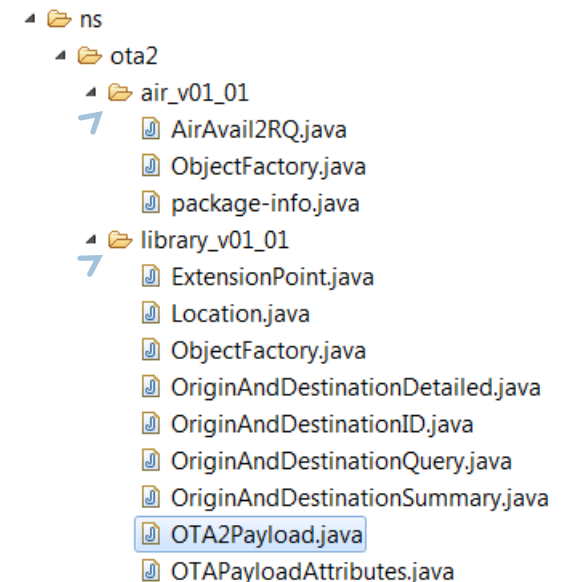
From Library

2.0 Messages: Namespaces (4/4)

37

- Namespaces and Java
 - Separate Packages for
 - Library
 - Message
 - Encourages type reuse

```
<xsd:schema  
  xmlns:lib="http://www.sabre.com/ns/OTA2/Library_v01_01"  
  xmlns="http://www.sabre.com/ns/OTA2/Air_v01_01"  
  xmlns:av="http://www.sabre.com/ns/OTA2/Air_v01_01"  
  targetNamespace="http://www.sabre.com/ns/OTA2/Air_v01_01"
```



Component Architecture: Extendable Objects

2.0 Business Object Extension (1/2)

39

- Adding extension points allows extension to be a minor update
- Extension point can be added to any business object

```
<xs:element name="Extension"
            type="Extension"/>
<!-- complex types -->
<xs:complexType name="Extension">
  <xs:sequence>
    <xs:any namespace="##other"
            processContents="lax"
            minOccurs="0"
            maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType>
  <xs:sequence>
    <xs:element ref="EchoData"/>
    <xs:element ref="TimeStamp"/>
    <xs:element ref="Extension"/>
  </xs:sequence>
</xs:complexType>
```

2.0 Business Object Extension (2/2)

40

- Extensions & namespaces in a message
 - ▣ Identify source and version of each type

Message

```
<?xml version="1.0" encoding="UTF-8"?>
<otm:OTA_PingRQ
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.ot2.org/msgs/ping/v1_1 OTA_Ping_v1.1.xsd"
  xmlns:otm="http://www.ot2.org/msgs/ping/v1_1"
  xmlns="http://www.ot2.org/library/v1_1"
  xmlns:o11="http://www.ot2.org/library/v1_1">
  <EchoData>The quick brown fox jumps over the lazy dog</EchoData>
  <TimeStamp>2010-10-05T16:29:00Z</TimeStamp>
  <Extension>
    <o11:HostName context="Sample">
      <o11:Pong>ABCD</o11:Pong>
    </o11:HostName>
  </Extension>
</otm:OTA_PingRQ>
```

Message Schema v1.1

Message NS

Library NS

Library Extension NS

From Extension

From Library

Versions and Namespaces

2.0 Versions & Namespaces (1/2)

42

Valid for both 1.0 and 1.1 XSD Schemas

```
<?xml version="1.0" encoding="UTF-8" >
<otm:OTA_PingRQ
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.ot2.org/messages/ping/v1_1_0/ping.xsd"
  xmlns:otm="http://www.ot2.org/msgs/ping/v1_1"
  xmlns="http://www.ot2.org/types/v1"
  xmlns:o11="http://www.ot2.org/types/v1_1">
  <EchoData>The quick brown fox jumps over the lazy dog</EchoData>
  <TimeStamp>2010-10-05T16:29:00Z</TimeStamp>
  <Extension>
    <o11:NewElement context="Sample">
      <o11:NewCode>ABCD</o11:NewCode>
    </o11:NewElement>
  </Extension>
</otm:OTA_PingRQ>
```

Message types (OTA_PingRQ) are always extendable.

Business Objects may be extendable.

Ignored when validated against 1.0 schema and by 1.0 systems

Understood when validated against 1.1 schema and by 1.1 systems

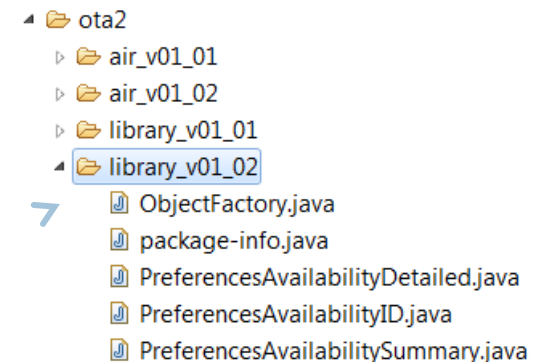
2.0 Versions & Namespaces (2/2)

43

□ Versions, Messages and Namespaces

▣ Separate Package for

- Message
- Message version changes
- Library
- Library version change



```
<xsd:schema
  xmlns:lib="http://www.sabre.com/ns/OTA2/Library_v01_01"
  xmlns:lib2="http://www.sabre.com/ns/OTA2/Library_v01_02"
  xmlns="http://www.sabre.com/ns/OTA2/Air_v01_02"
  xmlns:av="http://www.sabre.com/ns/OTA2/Air_v01_02"
  targetNamespace="http://www.sabre.com/ns/OTA2/Air_v01_02"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  attributeFormDefault="unqualified" elementFormDefault="qualified"
  version="1.0.0" xmlns:pref="http://www.sabre.com/ns/OTA2/Pref_v01_01"

  <xsd:import namespace="http://www.sabre.com/ns/OTA2/Library_v01_01"
  <xsd:import namespace="http://www.sabre.com/ns/OTA2/Library_v01_02"/>
```

```
private static final String schemaContext =
  ":com.opentravel.ns.ota2.library_v01_01"
  +":com.opentravel.ns.ota2.air_v01_01"
  +":com.opentravel.ns.sabre.air_v01_01"
  +":com.opentravel.ns.ota2.air_v01_02";

JAXBContext jc = JAXBContext.newInstance
(schemaContext);
```

2.0 XML Component Design & Construction

2.0 Component Design & Construction (1/2)

45

- **Right Now** - *how many people can*
 - Create 1.0 Schema
 - Create 2.0 Schema from Type Library
 - Create 2.0 Types
- **Lab**
 - Build a Business Object
 - Create an OTA 2.0 message
 - Add messages to dispatcher java code.
 - Extend messages
 - Implementation extensions in code

2.0 Component Design & Construction (2/2)

46

- **Creating Library Components**
 - Capturing the business requirements
 - Generate the XML/XSD

- **Support a new message**
 - Automating message handling

- **Implementing change - Add Extension**
 - Ex: adding pre-reserved seat to preferences
 - By trading partner extension
 - Added to OpenTravel schemas

2.0 Demo Tour Search

2.0 Demo: Tour Search

48

- **Constructing the Tour Library**
 - Leveraging 1.0 AND project team analysis
 - Business Objects
 - Core Objects
- **Lessons learned**
 - Business analysis
- **Constructing the TourSearchRQ and RS**
 - Using a prepared library

2.0 Demo: Generating Types

49

- **The Library Schema**
 - Compile Library XML into 2.0 types
 - No XSD knowledge needed (at least not much)
 - User Interface being considered

e Term_BusinessObject	
e Name	Profile_Traveler
e Aliases	PaxProfile
e Description	A profile of a traveler or other person or organization.
+ e Id	
+ e Query	
- e Summary	
- e Property	
e Name	Name
+ e Constraint	
e Description	Entities Name
- e Property	
e Name	Address
+ e Constraint	
e Description	Entities address
+ e Property	

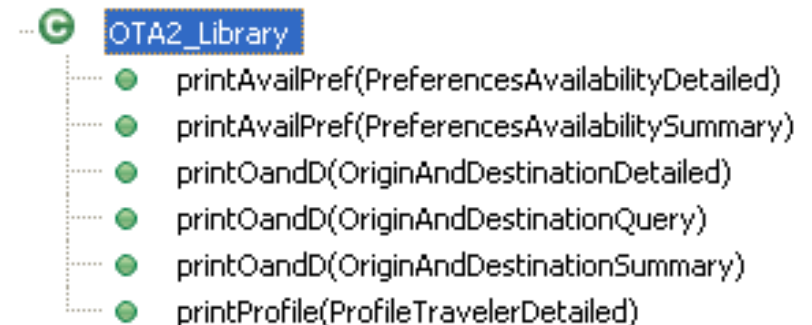
Library Compiler

```
name="Profile_Traveler_Summary">
Content>
ension base="Profile_Traveler_ID">
:sequence>
<xs:element type="xs:string" name="Name">
annotation>
<xs:documentation xml:lang="EN">Entities
</xs:annotation>
</xs:element>
<xs:element type="xs:string" name="Address">
<xs:annotation>
<xs:documentation xml:lang="EN">Entities
</xs:annotation>
</xs:element>
<xs:element type="xs:string" name="Phone">
```

2.0 Demo Code: Overview

50

- Library
 - Packages
 - Indicators (same Airport)
 - Detailed uses Summary



```
public void printOandD(OriginAndDestinationDetailed OandD) {  
    OriginAndDestinationSummary summary = (OriginAndDestinationSummary) OandD;  
    printOandD(summary);  
    System.out.println("Remarks:"+OandD.getRemarks());  
}
```

- What else we can do with O and D
 - Test and add Multi-Airport
 - test needed because it is optional

2.0 Demo: Overview

51

□ Dispatcher

▣ Standard properties (Service, POS, etc)

```
System.out.println("Service : " + envelope.getService().getValue());
```

▣ Payload substitution group

```
JAXBElement<? extends OTA2Payload> payload = envelope.getOTA2Payload();  
if (payload.getName().getLocalPart().equals("AirSearch2RQ"))  
    doSearch(payload);  
else if (payload.getName().getLocalPart().equals("AirAvailRQ"))  
    doAvail(payload);  
else  
    System.out.println("Unknown payload. " + payload.getName().getLocalPart());
```

```
private static void doAvail(JAXBElement<? extends OTA2Payload> payload) {  
    System.out.println("Availability with " + payload.getName().getLocalPart());  
    AirAvailRQ av = (AirAvailRQ) payload.getValue();  
    lib.printOandD(av.getOriginAndDestination());  
}
```

(TODO - fix schemas first,
create sample XML)

2.0 Demo: Add Message

52

- **Add dispatcher for Tour**
 1. xjc – bind xml to Java classes
 2. add context – tell Java about the classes
 3. add test and call
 4. Run
 - add message to parameter list

2.0 Demo: Add Extension

53

□ Add Private Extension to Preferences

▣ There is no extension on Preferences (good or bad?)

1. SabreSearch_v1.01.xsd
2. SabreSearch_v1.01.xml
3. Add to Air.xsd
4. xjc -- see new package for Sabre extension
5. add context
6. add test and call
7. add handler to do search
8. add parameter

2.0 Demo: Add Extension

54

- **Use an Extensible element**
 1. Avail_w_Ancillaries_v1.01-extended.xml
 - Uses *Profile_Traveler_Detailed* which is extensible.
 - Extensible = other namespace but not validated unless schema found.
 - I can find it -- but not process it.

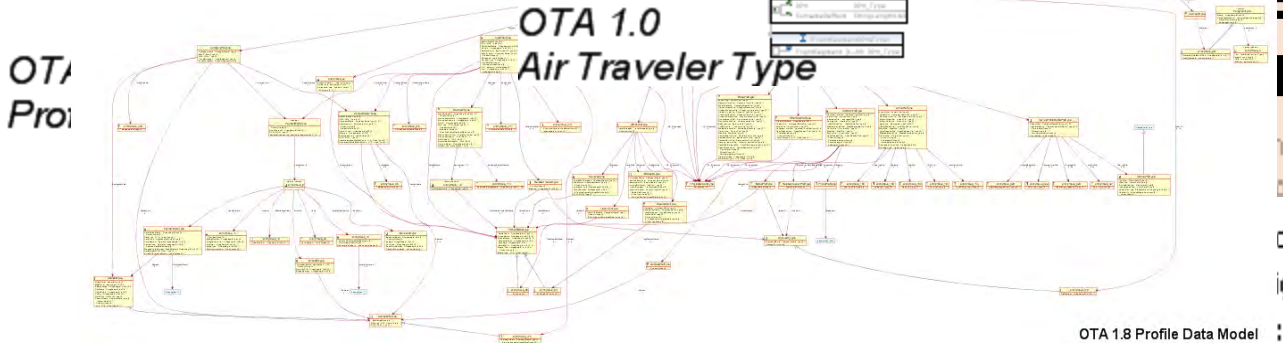
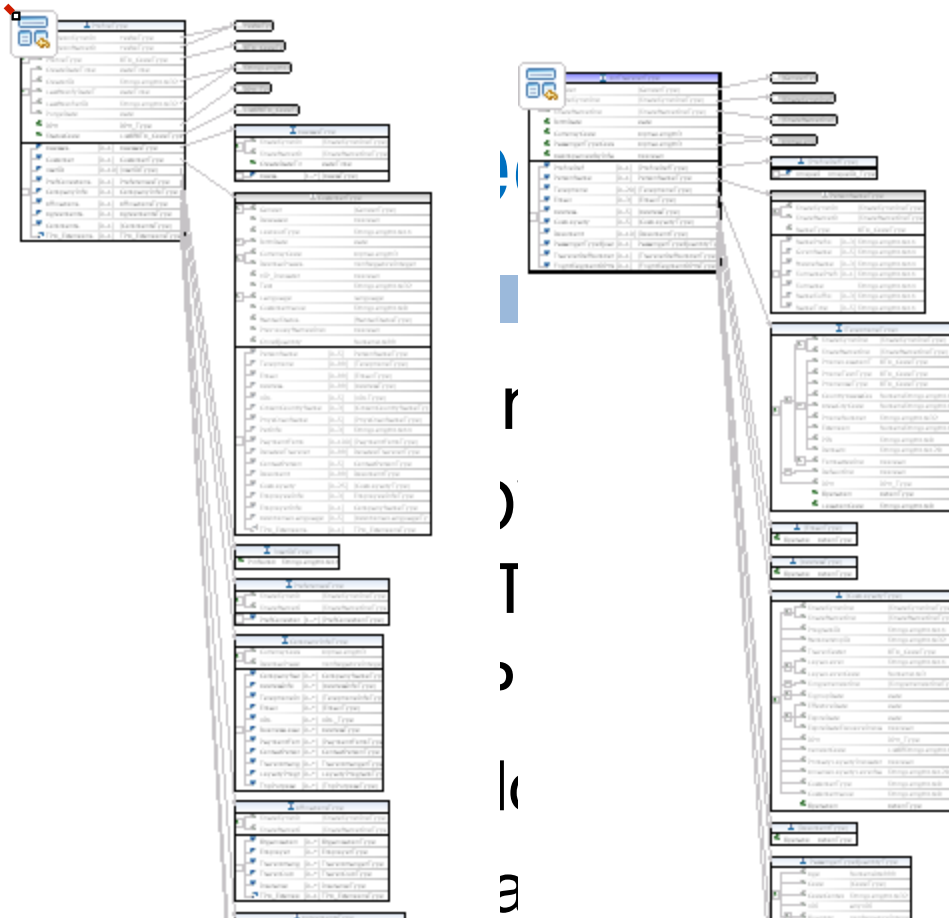
2.0 Demo: Add Extension

55

- **Backward compatible version using extensible element**
 1. Avail_w_Ancillaries_v1.02.xsd
 - simply define CVS type
 2. compile (xjc)
 - See the new namespace?
 3. add context
 4. add handler with cast

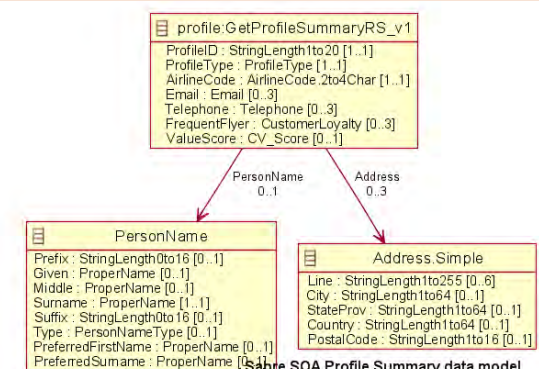
(inc)

to identify business objects
for existing ones (1.0/2.0 example
(tour/air preferences)
EN how to structure them
NIE/DAVE – WHEN MERGE TOURS/
LIBRARY LOOK FOR A SPECIFIC
EXAMPLE OF THIS
DISCUSS: REFACTORIZING ONLY



OTA
Pro

OTA 1.8 Profile Data Model



OriginAndDestination_Summary : OriginAndDestination_Summary

- TravelPreferences : AirSearchPrefsType
- SameAirportInd : xsd:boolean
- RPH : RPH_Type