Message Schemas

Describing what you're sending and receiving

Outline

- BizTalk and XML Schema
- The BizTalk Editor
- Designing a schema
- Schema migration
- Envelope schemas
- Property schemas
- Flat-file schemas

BizTalk messaging

- **BizTalk provides many features related to message processing**
  - Validation
  - Content-based routing
  - Translating between XML and non-XML formats
  - Transformations between message versions
  - Batched message processing
- **All of these features are built on message *schema definitions***
  - Understanding message schemas is fundamental in BTS

XSD language fundamentals

- **XSD provides a suite of *built-in datatypes***
  - Includes common types in use today (e.g., string, int, date, etc)
- **XSD also defines a vocabulary for defining new XML types**
  - You define simple types to restrict the value/lexical space of an existing datatype, thereby creating a custom datatype
  - You define complex types to describe element structures
- **XSD provides a way to map elements and attributes to types**

Note: download the Essential XML Quick Reference for a quick guide

A simple XSD definition

```
<xs:schema
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  targetNamespace="http://example.org/contacts"
 >
  <xs:element name="contact">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="name" type="xs:string" />
        <xs:element name="phone" type="xs:string" />
        <xs:element name="email" type="xs:string" />
        <xs:element name="birth" type="xs:date" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

built-in datatypes

schema vocabulary for defining new types

```
<c:contact
  xmlns:c="http://example.org/contacts">
  <name>Bob Smith</name>
  <phone>800-123-4567</phone>
  <email>bob@smith.com</email>
  <birth>1972-04-12</birth>
</c:contact>
```

BizTalk and XML Schema

- **BTS uses XSD as its official schema definition language**
  - Used to model traditional *document schemas*
  - BTS extends XSD to model other types of schemas
  - Extensions implemented with XSD *annotations*

Schema Type	Description
Document	Describes the structure of an XML element used as a message
Envelope	Describes the structure of an XML element used to wrap ("envelope") message elements (e.g., SOAP is an example of an envelope)
Property	Describes a mapping between logical property names and items found within an XML document using XPath expressions
Flat-file	Describes a mapping between an XML structure and a corresponding flat-file syntax (e.g., EDI, CSV, etc)

BizTalk Editor

- **BTS provides the *BizTalk Editor* for working with schemas**
  - Provides sophisticated support for the XSD language
  - Intuitive *instance-based* tree view design
  - Knows how to create BizTalk schema types via annotations
- **It's integrated with Visual Studio .NET 2005**
  - Registered as the default XSD editor by default
- **When you build, the schema is stored in a .NET assembly**
  - .NET assembly registered in the GAC

Designing a schema

- **Designing a schema consists of the following:**
  - Specifying the schema's target namespace
  - Defining elements and attributes used in messages
  - Defining custom simple and complex types for reusability
  - Importing external schemas to reuse existing types

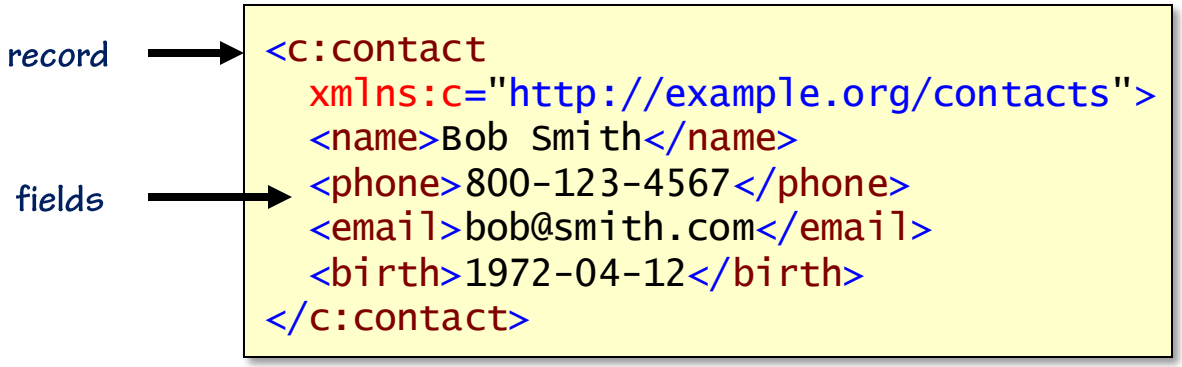
Schemas and namespaces

- **When designing a schema, first you set its *target namespace***
  - Children of the schema become part of the namespace
  - Similar to scoping classes with a .NET namespace
  - The target namespace can be empty in some cases (includes)
- **You can set the target namespace in the BizTalk Editor**
  - Select <Schema> and set the *Target Namespace* property
  - You'll need a heuristic for defining namespace names
  - Default name in BizTalk is *http://project_name/type_name*

Note: read Understanding XML Namespaces for more details

Defining elements

- **Elements are the basic building block of XML documents**
  - You define elements using
  - Elements can contain text and other elements
- **BizTalk Editor uses its own terminology**
  - A *record* is an element that contains other elements
  - A *field element* is a text-only element
  - You define records/fields in the tree view



Global vs. local elements

- **Elements can be defined at different scopes**
  - A *global element* is a child of <xsd:schema>
  - A *local element* is defined within another element
  - A local element can *reference* a global element
  - Local elements can repeat when desired
- **Local elements are not namespace qualified by default**
  - You can override this using *form* or *elementFormDefault*

Defining elements using the BizTalk Editor

- **You define elements using the tree view in BizTalk Editor**
  - Right click on a node and select *Insert Schema Node*
  - Insert *child records* to define elements containing other elements
  - Insert *child field elements* to define text-only elements
  - References defined by choosing element name for Data Type
  - *Min Occurs* and *Max Occurs* control repetition (default to 1)
  - Both *Form* and *Element Form Default* properties available

Defining elements

```
<xs:schema
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:tns="http://example.org/contacts"
  targetNamespace="http://example.org/contacts"
>
  <xs:element name="id" type="xs:string"/>
  <xs:element name="contact">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="tns:id" />
        <xs:element name="name" type="xs:string" />
        <xs:element name="phone" type="xs:string" />
        <xs:element name="email" type="xs:string" />
        <xs:element name="birth" type="xs:date" />
      </xs:sequence>
    </xs:complex
  </xs:element>
</xs:schema>
```

Global elements (*records*)

Reference to global element

Local elements (*fields*)

```
<c:contact
  xmlns:c="http://example.org/contacts">
  <c:id>333-22-4444</c:id>
  <name>Bob Smith</name>
  <phone>800-123-4567</phone>
  <email>bob@smith.com</email>
  <birth>1972-04-12</birth>
</c:contact>
```

Global element qualified

Local elements unqualified by default

Qualifying local elements

Local elements are namespace qualified

```
<xs:schema
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:tns="http://example.org/contacts"
  targetNamespace="http://example.org/contacts"
  elementFormDefault="qualified" >
  <xs:element name="id" type="xs:string"/>
  <xs:element name="contact">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="tns:id" />
        <xs:element name="name" type="xs:string" />
        <xs:element name="phone" type="xs:string" />
        <xs:element name="email" type="xs:string" />
        <xs:element name="birth" type="xs:date" />
      </xs:sequence>
    </xs:complex
  </xs:element>
</xs:schema>
```

```
<contact
  xmlns="http://example.org/contacts">
  <id>333-22-4444</id>
  <name>Bob Smith</name>
  <phone>800-123-4567</phone>
  <email>bob@smith.com</email>
  <birth>1972-04-12</birth>
</contact>
```

Specifying the "root" element

- **Many schemas have multiple global element declarations**
  - Sometimes it's necessary to know the "root" element
- **You can specify the "root" element using the BizTalk Editor**
  - Set the *Root Reference* property on the <Schema> node

root element →

```
<xs:schema
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  targetNamespace="http://example.org/contacts"
  elementFormDefault="qualified">
  <xs:annotation>
    <xs:appinfo>
      <schemaInfo root_reference="contact"
          xmlns="http://schemas.microsoft.com/BizTalk/2003"/>
    </xs:appinfo>
  </xs:annotation>
  <xs:element name="contact">
    ...
```

Defining attributes

- **Attributes are properties associated with an element**
  - You define attributes using *<xsd:attribute>*
  - Attributes can only contain text
  - Referred to as *field attributes* in BizTalk Editor
- **Attributes can also be defined at different scopes**
  - A *global attribute* is a child of <xsd:schema>
  - A *local attribute* is defined within an element
- **Attributes are not namespace qualified by default**
  - You can override this using *form* or *attributeFormDefault*

Defining attributes

```
<xs:schema
   xmlns:xs="http://www.w3.org/2001/XMLSchema"
   xmlns:tns="http://example.org/contacts"
   targetNamespace="http://example.org/contacts">
  <xs:element name="contact">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="name" type="xs:string" />
        <xs:element name="phone" type="xs:string" />
        <xs:element name="email" type="xs:string" />
        <xs:element name="birth" type="xs:date" />
      </xs:sequence>
      <xs:attribute name="source" type="xs:string" />
      <xs:attribute ref="tns:id"/>
    </xs:complexType>
  </xs:element>
  <xs:attribute name="id" type="xs:string" />
</xs:schema>
```

Local attributes

Reference to global attribute

Global attribute

```
<c:contact
  xmlns:c="http://example.org/contacts">
  source="MSOutlook"
  c:id="333-22-4444"
  ...
```

Defining simple types

- **When the built-in types aren't specific enough, you can create custom simple types**
  - Defined using *<xsd:simpleType>*
- **Simple types are based on an existing simple type**
  - You restrict *facets* of the existing type
  - You're basically restricting the value/lexical space

Custom simple type

Restricts the lexical space to valid SSN strings using the pattern facet

```
<xs:schema
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:tns="http://example.org/contacts"
  targetNamespace="http://example.org/contacts"
>
  <xs:attribute name="id" type="tns:SSN" />
  <xs:simpleType name="SSN">
    <xs:restriction base="xs:string">
      <xs:pattern value="\d{3}-\d{2}-\d{4}" />
    </xs:restriction>
  </xs:simpleType>  ...
```

Defining simple types with the BizTalk Editor

- **You can define custom simple types with the BizTalk Editor**
  - First define a field element or attribute
  - Set the *Derived By* property to Restriction
  - Set the *Base Data Type* property to the appropriate base type
  - Enter a name for your custom type in the *Data Type* property
  - Constrain facets in the *Restriction* property section
- **This gives you a named simple type that you can reuse**

Defining complex types

- **You define element structure with complex types**
  - Defined using *<xsd:complexType>*
- **Complex types contain a *compositor* (sequence, choice, all)**
  - Compositors contain elements or other compositors
  - The compositor describes how its particles are organized

Custom complex type

Compositor

```
...
<xs:element name="contact" type="tns:Contact" />
<xs:complexType name="Contact">
  <xs:sequence>
    <xs:element name="name" type="xs:string" />
    <xs:element name="phone" type="xs:string" />
    <xs:element name="email" type="xs:string" />
    <xs:element name="birth" type="xs:date" />
  </xs:sequence>
  <xs:attribute ref="id" />
</xs:complexType>
...
```

Defining complex types with the BizTalk Editor

- **You can define custom complex types with the BizTalk Editor**
  - First define a record that has the desired structure
  - Enter a name for your custom type in the *Data Type* property
  - Once you press <Enter>, it will factor out the complex type
  - You can control the compositor using *Group Order Type*
- **This gives you a named complex type that you can reuse**

Reusing schemas

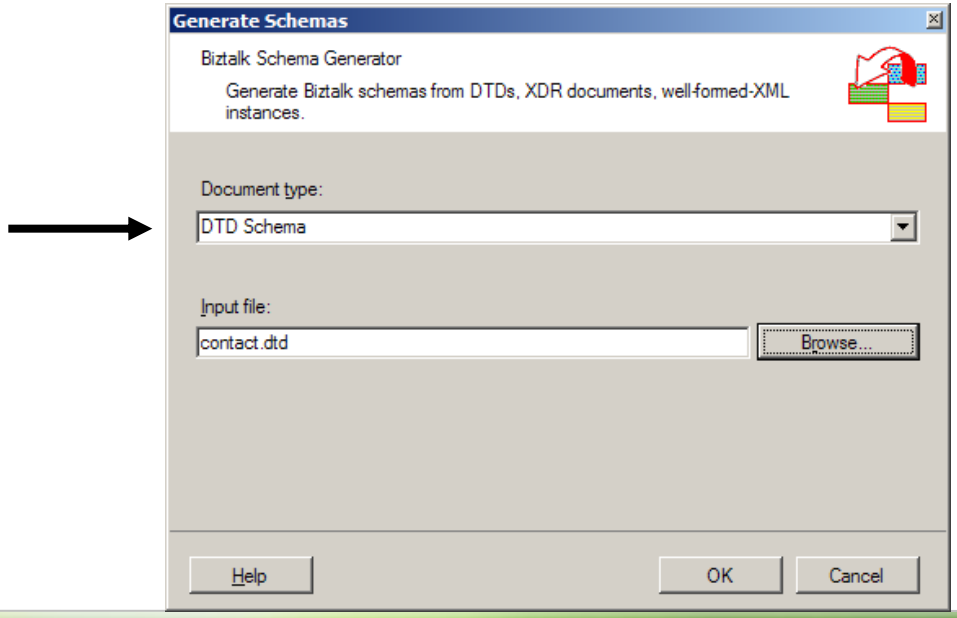
- **XSD makes it possible to import schemas a few different ways**
  - BizTalk Editor supports all three mechanisms
  - Accessible via *Imports* property on <Schema> node

Import Type	Description
Import	Imports an existing schema for use in this schema – the imported schema types already exist in a namespace, and do not become part of the new target namespace being defined
Include	Imports existing definitions and includes them in the target namespace – the included schema must be in the same namespace or no namespace
Redefine	Like include, but allows you to redefine existing definitions

Migrating existing schemas

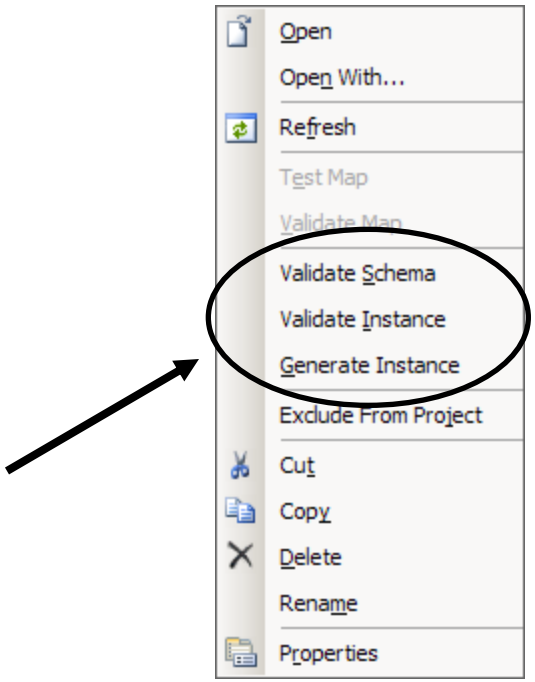
- **BizTalk Editor can work with XSD created with other tools**
  - Simply add the schema files to your project
- **It also supports migrating *legacy schemas* (XDR and DTD)**
  - Use *Add Generated Items* to launch the wizard
  - Wizard also generates XSD from XML instance

Choose the type of schema to migrate



Testing schemas

- **BizTalk Editor provides functionality for testing your schemas**
  - *Validate* the schema itself
  - *Generate* sample instances
  - *Validate* sample instances against the schema
- **You enable these features as follows**
  - Right click on the schema file, select Properties
  - Enter a file name for the output instance
  - Enter a file name for the input instance
- **Then right click on the schema and test**

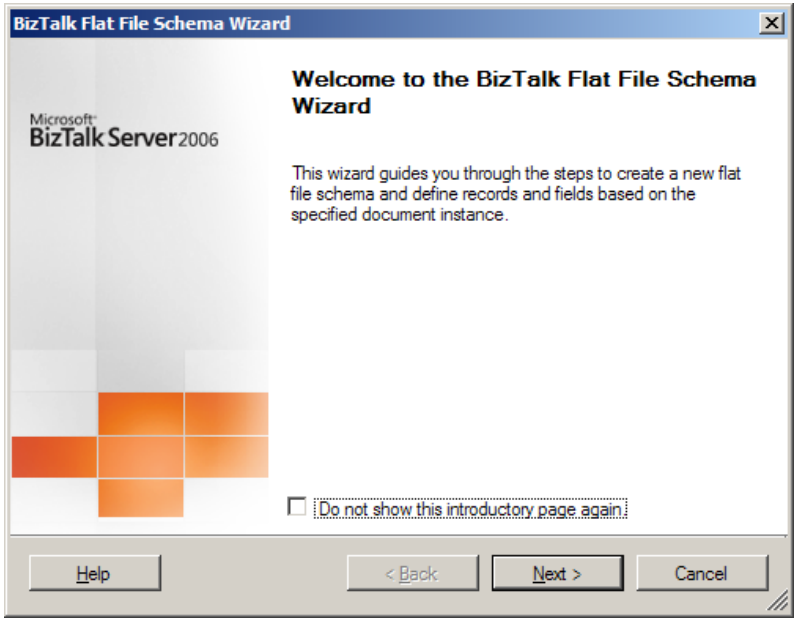


Flat-file schemas

- **BizTalk supports the concept of *flat-file schemas***
  - Define mappings between XML and text-based formats
  - Supports *delimited* or *positional* formats
- **Enable the Flat File *Schema Editor Extension***
  - Use the flat-file properties to define mapping
  - Mapping stored in XSD using annotations
- **Enables *translation* between XML and the flat-file format**

Flat-file schema wizard

- **BTS 2006 introduced the *Flat File Schema Wizard***
  - Takes the pain out of defining flat-file mappings
  - Add New Item | Schema Files | Flat Files Schema Wizard



Summary

- BizTalk relies on XML Schema for many valuable functions
- The BizTalk Editor simplifies working with XSD schemas
- BizTalk fully supports XSD, and provides for reuse
- BizTalk supports migrating legacy schemas forward to XSD
- BizTalk supports translating to & from flat-file schemas

References

- **Using XML Schemas in BTS**
  - http://msdn.microsoft.com/library/default.asp?url=/library/en-us/BTS_2004WP/html/8c17983d-90b1-4e0a-9cc3-3bda0b1e7f55.asp
- **BizTalk Server 2006 Developer Tools Improvements**
  - http://www.microsoft.com/biztalk/techinfo/whitepapers/bts2k6_devtoolswp.mspx
- **XML Schema Part 1: Structures**
  - http://www.w3.org/TR/xmlschema-1/
- **XML Schema Part 2: Datatypes**
  - http://www.w3.org/TR/xmlschema-2/
- **Essential XML Quick Reference (free download)**
  - http://www.theserverside.net/books/addisonwesley/EssentialXML/index.tss