

Microsoft BizTalk Server 2006 Part-III

Table of Contents

[Module 1: Ways to Use Complex Global Types](#)

Complex Global Type Re-use

[Module 2: Complex Global Type Derivation](#)

Complex Type Derivation Using the Extension Mechanism
Complex Type Derivation Using the Restriction Mechanism

[Module 3: Simple Type Derivation](#)

Simple Type Derivation Using the Restriction Mechanism
Simple Type Derivation Using the List Mechanism
Simple Type Derivation Using the Union Mechanism

[Module 4: Ways to Use Message Content to Control Message Processing](#)

About BizTalk Message Context Properties
Instance Message Processing Using Distinguished Fields
Instance Message Processing Using Property Promotion

[Module 5: Using BizTalk Editor](#)

How to Manage the Schema Tree View
How to Manage the XSD View
How to Manage the Properties Window
How to Manage Other Visual Studio 2005 Windows
Using BizTalk Editor Commands
Working with Large Schemas

[Module 6: Creating Schemas](#)

Managing Schemas Within Projects
Managing the Nodes Within a Schema
Promoting Properties

[Module 7: Promoting Properties](#)

How to Create Schemas for XML Messages
How to Create Schemas for Flat File Messages
How to Create Schemas for Envelopes
How to Create Property Schemas
How to Add Existing Schemas
How to Save, Rename, and Close Schemas
How to Include and Exclude Schemas
How to Delete Schemas
How to Set Schema File and Schema Item Properties
How to Migrate XDR Schemas to XSD Schemas
How to Create Schemas That Use Other Schemas

[Module 8: Managing the Nodes Within a Schema](#)

Inserting Nodes into a Schema
Working with Existing Nodes

[Module 9: Inserting Nodes into a Schema](#)

How to Insert a Record, Field Element, or Field Attribute Node

- How to Create a New Occurrence of an Existing Record, Field Element, or Field Attribute Node
- How to Insert a Sequence Group, Choice Group, or All Group Node
- How to Insert an Any Element or Any Attribute Node
- How to Create References to Another Node or Type

Module 10: Working with Existing Nodes

- How to Work with Multiple Nodes Simultaneously
- How to Move and Copy Nodes
- How to Rename Nodes
- How to Delete Nodes
- How to Set Node Properties

Module 11: Promoting Properties

- How to Open the Promote Properties Dialog Box
- How to Copy Data to the Message Context as Distinguished Fields
- How to Copy Data to the Message Context as Property Fields

Module 12: Creating Schemas Using BizTalk Flat File Schema Wizard

- How to Use BizTalk Flat File Schema Wizard
- BizTalk Flat File Schema Wizard Walkthrough

Module 13: BizTalk Flat File Schema Wizard Walkthrough

- Walkthrough: Creating a Flat File Schema From a Document Instance

Module 14: Testing Schemas

- How to Validate Schemas in Visual Studio
- How to Validate Instance Messages
- How to Generate Instance Messages

Module 15: Extending BizTalk Editor

- Property Managers
- Custom Views
- Schema Validation
- Instance Validation
- Instance Generation

Module 16: Considerations When Creating Schemas

- Namespace Management
- Code List Management
- Considerations When Creating Flat File Message Schemas

Module 17: Considerations When Creating Flat File Message Schemas

- Code Page Specification for Flat File Schemas
- Case Handling in Flat File Schemas
- Restricted Character Ranges
- Nested Positional and Delimited Records
- Positional Record Considerations
- Delimited Record Considerations
- Field Considerations
- Ways to Interpret Special Characters as Part of a Field Value

Module 18: Positional Record Considerations

- Tag Handling in Positional Records
- Nested Positional Records
- Position Counting in Bytes
- Field Position Calculation

Module 19: Delimited Record Considerations

- Tag Handling in Delimited Records
- Child Order Considerations
- Delimiter Preservation and Suppression

Module 20: Field Considerations

- Field Element Nodes vs. Field Attribute Nodes
- Custom Date/Time Formats
- Field Padding
- Field Justification
- Specification of Field Positions within Positional Records
- Minimum Field Lengths Within Delimited Records

Module 21: Ways to Interpret Special Characters as Part of a Field Value

- Escape Characters
- Wrap Characters
- Known Issues with Schema Generation and Validation

Module 22: Managing Business Processes Using BizTalk Explorer

- Considerations When Using BizTalk Explorer
- Permissions Required for Deploying and Managing a BizTalk Application
- About BizTalk Explorer Artifacts
- Planning for BizTalk Explorer
- Determining the Right Tool For Your Task
- Managing Assemblies Using BizTalk Explorer
- Managing Send Ports Using BizTalk Explorer
- Managing Send Port Groups Using BizTalk Explorer
- Managing Receive Ports and Receive Locations Using BizTalk Explorer
- Managing Orchestrations Using BizTalk Explorer
- Managing Parties Using BizTalk Explorer
- Enlisting a Party Using BizTalk Explorer
- Automating Business Process Management

Module 23: About BizTalk Explorer Artifacts

- About Enlisting, Starting, Stopping, and Unenlisting BizTalk Server Artifacts
- Partner Management in BizTalk Explorer
- Content-Based Routing in BizTalk Explorer
- Document Normalization in BizTalk Explorer
- Orchestration Management in BizTalk Explorer
- BizTalk Explorer Automation
- How to Open BizTalk Explorer
- How to Add or Remove BizTalk Management Databases Using BizTalk Explorer

Module 24: Managing Assemblies Using BizTalk Explorer

- How to Browse Assemblies Using BizTalk Explorer
- How to Undeploy an Assembly Using BizTalk Explorer

Module 25: Managing Send Ports Using BizTalk Explorer

- How to Add a Static Send Port Using BizTalk Explorer
- How to Add a Dynamic Send Port Using BizTalk Explorer
- How to Edit Send Port Properties Using BizTalk Explorer
- How to Edit Send Port Adapter Properties Using BizTalk Explorer
- How to Manage Send Port Encryption Certificates Using BizTalk Explorer
- How to Add a Filter to a Send Port for Content-Based Routing Using BizTalk Explorer
- How to Add a Filter to a Send Port for Passthrough Using BizTalk Explorer
- How to Add Maps to Send Ports for Outbound Normalization Using BizTalk Explorer
- How to Add Maps to Send Ports for Inbound Normalization Using BizTalk Explorer
- How to Enlist and Start a Send Port Using BizTalk Explorer
- How to Stop and Unenlist a Send Port Using BizTalk Explorer
- How to Delete a Send Port Using BizTalk Explorer

Module 26: Managing Send Port Groups Using BizTalk Explorer

- How to Add a Send Port Group Using BizTalk Explorer
- How to Add and Delete Send Ports from a Send Port Group Using BizTalk Explorer
- How to Add a Filter to a Send Port Group Using BizTalk Explorer
- How to Enlist and Start a Send Port Group Using BizTalk Explorer
- How to Stop and Unenlist a Send Port Group Using BizTalk Explorer
- How to Delete a Send Port Group Using BizTalk Explorer

Module 27: Managing Receive Ports and Receive Locations Using BizTalk Explorer

- How to Add a Receive Port Using BizTalk Explorer
- How to Edit Receive Port Properties Using BizTalk Explorer
- How to Add a Receive Location Using BizTalk Explorer
- How to Edit Receive Location Adapter Properties Using BizTalk Explorer
- How to Add Maps to Receive Ports for Inbound Normalization Using BizTalk Explorer
- How to Add Maps to Receive Ports for Outbound Normalization Using BizTalk Explorer
- How to Enable and Disable a Receive Location Using BizTalk Explorer
- How to Specify a Primary Receive Location Using BizTalk Explorer
- How to Delete a Receive Location Using BizTalk Explorer
- How to Delete a Receive Port Using BizTalk Explorer

Module 28: Managing Orchestrations Using BizTalk Explorer

- How to Browse Orchestrations Using BizTalk Explorer
- How to Browse Used Roles of an Orchestration Using BizTalk Explorer
- How to Browse Implemented Roles of an Orchestration Using BizTalk Explorer
- How to Browse Invoked Orchestrations Using BizTalk Explorer
- How to Bind an Orchestration Using BizTalk Explorer
- How to Enlist and Start an Orchestration Using BizTalk Explorer
- How to Stop and Unenlist an Orchestration Using BizTalk Explorer

Module 29: Managing Parties Using BizTalk Explorer

- How to Add a Party Using BizTalk Explorer
- How to Manage Party Send Ports Using BizTalk Explorer
- How to Add an Alias to a Party Using BizTalk Explorer
- How to Manage Party Signature Certificates Using BizTalk Explorer

How to Delete a Party Using BizTalk Explorer

Module 30: Enlisting a Party Using BizTalk Explorer

How to Browse Roles Using BizTalk Explorer
How to Enlist Parties Using BizTalk Explorer
How to Unenlist Parties Using BizTalk Explorer
How to Refresh BizTalk Explorer

Module 31: Automating Business Process Management

Using the BizTalk Explorer Object Model from Managed Code
Common Programmatic Tasks Using the BizTalk Explorer Object Model

Module 32: Common Programmatic Tasks Using the BizTalk Explorer Object Model

Browsing BizTalk Server Assembly Artifacts Using the BizTalk Explorer Object Model
Orchestration Binding Using the BizTalk Explorer Object Model
Partner Management Using the BizTalk Explorer Object Model
Content-Based Routing Using the BizTalk Explorer Object Model
Managing Artifacts Using the BizTalk Explorer Object Model

Module 33: Creating Maps Using BizTalk Mapper

Planning to Create Maps
About Maps
Using BizTalk Mapper
Creating Maps
Compiling and Testing Maps

Module 34: About Maps

Overview of Maps
Links in Maps
Functoids in Maps
Map Compilation and Testing

Module 35: Overview of Maps

Map Components
Transformation vs. Translation
Source and Destination Schemas
Types of Mapping Operations
Specific vs. Generic Maps
Valid BizTalk Mapper XSLT Encoding Types
Order of Records and Fields
XSD Element Groups
Map Migration from Previous Versions of BizTalk Server
The Map Grid and Its Pages
Custom XSLT

Module 36: Links in Maps

Types of Links
Link Creation
Link Configuration
Record-to-Record Linking
Links To and From the Any Element and anyAttribute Nodes
Compiler Directives and Links

Module 37: Compiler Directives and Links

Data Transformation Configuration
Node-Hierarchy Level Matching

Module 38: Functoids in Maps

Functoid Categories
Functoid Input Parameters
Basic Functoids
Advanced Functoids
Cascading Functoids
Functoid Properties
Migrating Functoids

Module 39: Basic Functoids

Conversion Functoids
Cumulative Functoids
Database Functoids
Date and Time Functoids
Logical Functoids
Mathematical Functoids
Scientific Functoids
String Functoids

Module 40: Advanced Functoids

Index Functoid
Iteration Functoid
Looping Functoid
Mass Copy Functoid
Record Count Functoid
Table Looping and Table Extractor Functoids
Value Mapping Functoid
Value Mapping (Flattening) Functoid
Scripting Functoid

Module 41: Looping Functoid

Conditional Looping
Flat Schema to Catalog
Loop Paths

Module 42: Table Looping and Table Extractor Functoids

Table Looping Functoid
Table Extractor Functoid
Table-Driven Looping Configuration
Table-Driven Looping Example

Module 43: Scripting Functoid

Scripting Using External Assemblies
Scripting Using Inline C#, JScript .NET, and Visual Basic .NET
Scripting Using Inline XSLT and XSLT Call Templates

Module 44: Map Compilation and Testing

Map Compilation
Map Testing

Module 45: Map Testing

Using Map File Properties for Testing
Validating Instance Data
Empty Node Values in Source Instance Messages
Optional Nodes

Module 46: Using BizTalk Mapper

Using BizTalk Mapper Commands
Working with Grid Pages
How to Change Views Sizes
How to Customize Colors and Font in BizTalk Mapper
How to Expand and Collapse the Schema Trees

Module 47: Working with Grid Pages

How to Manage Grid Pages
How to Move Between and Within Grid Pages
How to Move Functoids and Links Between Grid Pages

Module 48: Creating Maps

Managing Maps Within Projects
Using Links to Specify Record and Field Mappings
Using Functoids to Create More Complex Mappings
How to Create a Map without Maps

Module 49: Managing Maps Within Projects

How to Create New Maps
How to Add Existing Maps
How to Open, Save, Close, and Rename Maps
How to Replace Schemas

Module 50: Using Links to Specify Record and Field Mappings

How to Create Links
How to Edit Link Properties
How to Link Records Automatically
How to Manage Existing Links
How to Select Multiple Links
How to Configure Node Hierarchy Matching
How to Set the Source Links Compiler Value
How to Show and Hide Compiler Links

Module 51: Using Functoids to Create More Complex Mappings

How to Open the Functoid Toolbox
How to Add Basic Functoids to a Map
Adding Advanced Functoids to a Map
Editing Functoid Properties and Input Parameters
How to Select Multiple Functoids
How to Replace Functoids
How to Replace Functoids

Module 52: Adding Advanced Functoids to a Map

- How to Add Index Functoids to a Map
- How to Add Iteration Functoids to a Map
- How to Add Looping Functoids to a Map
- How to Add Mass Copy Functoids to a Map
- How to Add Record Count Functoids to a Map
- How to Add Scripting Functoids to a Map
- How to Add Table Looping and Table Extractor Functoids to a Map
- How to Add Value Mapping Functoids to a Map
- How to Add Value Mapping (Flattening) Functoids to a Map

Module 53: Editing Functoid Properties and Input Parameters

- How to Label a Functoid
- How to Configure Functoid Input Parameters]
- How to Configure the Scripting Functoid
- How to Configure the Table Looping and Table Extractor Functoids

Module 54: Compiling and Testing Maps

- How to Resolve Map Warnings and Errors
- How to Add Constant Values
- How to Configure Map Validation and Test Parameters
- How to Validate Maps
- How to Test Maps
- How to Create Multi-Part Mappings
- How to Specify XSLT Output Settings

Complex Global Type Definition and Naming

Within BizTalk Editor, you begin to define a complex global type by defining the first occurrence of the complex type in one of the locations where the global type will be used, after it has been converted to a global type. Continuing with the address example, you might define the complex address type in the course of defining a shipping address within the schema.

After the complex type is defined, you can convert it to a global complex type by giving it a type name. You do this by selecting the node that corresponds to the complex type, which will generally be a **Record** node, and then typing a new type name into the **Data Structure Type** property of that node. Although no visible changes occur in the schema tree when you give this property a name (such as, **GlobalAddrType**, as in the following example), if you examine what happens within the underlying XSD representation of the schema, you would see the following (abbreviated) change.

Before, with the address structure first defined within the context of the **ShippingAddress** element, the following occurred.

After the **ShippingAddress** node has been given a unique name in its **Data Structure Type** property, causing it to become available as a complex global type and subject to reuse in multiple places within the schema, the following occurs.

Ways to Use Complex Global Types

After you have converted a complex type to a global complex type, it becomes available for reuse in other locations within your schema. For more information about defining a complex type and then converting it to a global complex type, see Complex Global Type Definition and Naming.

First, you insert a new **Record** node. Then you select the inserted node and set one of the following two node properties in the Properties window, each for a different effect:

- **Data Structure Type property.** If you want to use the complex global type without modifying it in any way, set this property to the type name you gave to the complex global type, which is available as a choice in the drop-down list. In the schema tree, the chosen global node structure will be graphically duplicated in the new location, and any subsequent changes to the node structure in any of its locations in the schema tree are automatically made to all locations that use that complex global type.
- **Base Data Type property.** If you want to use a variation on the complex global type, either extending it or restricting it in some way, set this property to the type name you gave to the complex global type, which is available as a choice in the drop-down list. When you set this property, the **Derived By** node property changes to **Extension** (and the **Content Type** property changes to **ComplexContent**), indicating that extending the complex global type is the default derivation type. You can change it to **Restriction** if your modifications are of that nature. Changes to the base complex global type from which you are deriving are automatically reflected in the derived type, but changes in the derived type are never reflected in the base type.

This section describes using complex global types, both as is, and by extending and restricting them, as controlled by the settings of the properties described in this topic.

In This Section

- Complex Global Type Re-use
- Complex Global Type Derivation

Complex Global Type Re-use

To use a complex global type as is, in another location in the schema tree, begin by inserting a new **Record** node at the desired location. Then set its **Data Structure Type** property to the name of a complex global type.

In the following example, **BillingAddress** is the name of the newly inserted **Record** node, and **GlobalAddrType** is the name of the complex global type that it adopts. In the schema tree view, a duplicate node structure would be displayed below the node named **BillingAddress**, identical to the adjacent node structure under the node named **ShippingAddress**.

Complex Global Type Derivation

There are two types of inheritance in XSD: extension and restriction. BizTalk Editor provides access to this XSD functionality by using the following two **Record** node properties:

- **Base Data Type.** This property provides the list of all global complex types and simple types available for use as a base data type. The complex global types can be in the same schema or in any imported schema.
- **Derived By.** This property is used to choose between deriving by extension or by restriction. This property is automatically set to **Extension** when you set the **Base Data Type** property to any type in the list. If you set this property to **(Default)**, any type in the **Base Data Type** property is removed, disabling inheritance for the node.

This section explains complex type derivation by using the extension and restriction mechanisms in greater detail.

In This Section

- Complex Type Derivation Using the Extension Mechanism
- Complex Type Derivation Using the Restriction Mechanism

Complex Type Derivation Using the Extension Mechanism

A complex type derived by extension is a functional superset of its base data type. As the name implies, its base data type is the basis for the type being defined, where the differences from the base type are additive. This topic provides an example in which the two elements **ShippingAddress** and **BillingAddress** are based on the complex global type **GlobalAddrType**. **ShippingAddress** is simply defined to be of type **GlobalAddrType**, whereas **BillingAddress** is defined to extend the type **GlobalAddrType**. At the end of the example, an additional element is added to **BillingAddress**, named **Department**, with a type of string and a default value of Accounts Payable.

For comprehensive information about deriving new complex types by using the extension mechanism, refer to the W3C website. For various links to this and other websites, see XSD Resources on the Web.

To derive from a complex global type by extension, in another location in the schema tree, begin by inserting a new **Record** node at the desired location. Then set its **Base Data Type** property to the name of a complex global type.

In the following example, **BillingAddress** is the name of the newly inserted **Record** node, and **GlobalAddrType** is the name of the complex global type from which it derives, and intends to extend. In the schema tree view, after **Base Data Type** has been set to **GlobalAddrType**, a duplicate node structure would be displayed below the node named **BillingAddress**, identical to the adjacent node structure under the node named **ShippingAddress**. The difference between them is that the **BillingAddress** node structure will be extensible beyond the base data type **GlobalAddrType**, and the **ShippingAddress** structure will remain identical to the base data type **GlobalAddrType**.

. You specify extensions to the base data type by inserting nodes into the **BillingAddress** node in the schema tree. The following XSD fragment shows how the empty extension element is expanded when a **Sequence Group** node is inserted as a new child of the **BillingAddress** node and then a **Field Element** node, named **PaymentType**, is inserted as a child node of the **Sequence Group** node.

Complex Type Derivation Using the Restriction Mechanism

Derivation by restriction is similar to derivation by extension, in terms of BizTalk Editor functionality. A complex type derived by restriction is similar to its base data type, except that its declarations are more limited than the corresponding declarations in the base data type. In fact, the values represented by the new type are a subset of the values represented by the base data type (as is the case with restriction of simple types). An application prepared for the values of the base data type ought to be able to successfully process any of the values of the restricted type.

For comprehensive information about deriving new complex types by using the restriction mechanism, refer to the W3C website. For various links to this and other websites, see XSD Resources on the Web.

To derive from a complex global type by restriction, in another location in the schema tree, begin by inserting a new **Record** node at the desired location. Then set its **Base Data Type** property to the name of a complex global type. Finally, change the setting of the **Derived By** property from its default value of **Extension** (at least when a base data type is set) to **Restriction**.

In the following example, **BillingAddress** is the name of the newly inserted **Record** node, and **GlobalAddrType** is the name of the complex global type from which it derives, and intends to restrict. In the schema tree view, a duplicate node structure would be displayed below the node named **BillingAddress**, identical to the adjacent node structure under the node named **ShippingAddress**. The difference between them is that the **BillingAddress** node structure will be subject to possible restrictions to the base data type **GlobalAddrType**, and the **ShippingAddress** structure will remain identical to the base data type **GlobalAddrType**.

Because you have chosen to restrict the base data type, you are not allowed to insert any new nodes, but you can change the properties of the existing nodes to further restrict their possible values or behavior.

Simple Type Derivation

XML Schema definition (XSD) language provides several mechanisms for defining variations of simple types, based on derivations of existing simple types, as follows:

- **Restriction.** Simple type derivation by using the restriction mechanism involves the introduction of restrictions to the possible values for that type. Examples are minimum and/or maximum values for numeric types, or a minimum and maximum length for string types.
- **List.** Simple type derivation by using the list mechanism allows a single attribute or element value in an instance message to be defined as consisting of a list of space-separated values of a particular type.
- **Union.** Simple type derivation by using the union mechanism allows a single attribute or element value in an instance message to be defined as a single value of one of several specified types.

This section describes these simple type derivations in more detail, including how to define these derivations by setting the appropriate node properties in the Properties window, as well as how the corresponding XSD is constructed.

In This Section

- Simple Type Derivation Using the Restriction Mechanism
- Simple Type Derivation Using the List Mechanism
- Simple Type Derivation Using the Union Mechanism

Simple Type Derivation Using the Restriction Mechanism

When you derive a new simple type from an existing simple type by using the restriction mechanism, you are typically restricting the values allowed in an instance message for that attribute or element value to a subset of those values allowed by the base simple type. For example, you can restrict a string type to be one of several enumerated strings.

For comprehensive information about deriving new simple types by using the restriction mechanism, refer to the W3C website. For various links to this and other websites, see XSD Resources on the Web.

To derive a simple type by using restriction, select the relevant **Field Element** node or **Field Attribute** node in the schema tree and then, in the Properties window, select a simple type from the drop-down list for the **Base Data Type** property. As soon as you have selected a value for this property, the **Derived By** property automatically changes from its default value to **Restriction**, which serves as the default value for type derivation. Also, a whole new category of properties, called **Restriction**, becomes available in the Properties window.

Depending on the base data type you select, different properties are available to be set in this new category. For example, if the base data type is numeric, the properties **MaxFacet Type** (when **MaxFacet Value** is set), **MaxFacet Value**, **MinFacet Type** (when **MinFacet Value** is set), and **MinFacet Value**

are available for defining an inclusive or exclusive range of allowed values. If the base data type is a string type, the **Length**, **Maximum Length**, and **Minimum Length** properties are available for constraining the length of the string.

For more information about the various restriction properties of field nodes, see **Field Element Node Properties**.

When you first change a **Field Element** node or **Field Attribute** node from having a data type to having a base data type (thereby starting the process of simple type derivation), leave the **Derived By** property set to **Restriction**, and provide an enumeration-based restriction to the allowed string values, you can observe the following changes in the corresponding fragment in the XSD view:

Simple Type Derivation Using the List Mechanism

When you derive a new simple type from an existing simple type by using the list mechanism, you are specifying that the value for this attribute or element can be a space-separated list of values of the specified type. For example, you can specify that an attribute or element value is a space-separated list of integers.

For comprehensive information about deriving new simple types by using the list mechanism, refer to the W3C website. For various links to this and other websites, see XSD Resources on the Web.

To derive a simple type as a list of that type, select the relevant **Field Element** node or **Field Attribute** node in the schema tree and then, in the Properties window, select a simple type from the drop-down list for the **Base Data Type** property. As soon as you select a value for this property, the **Derived By** property automatically changes from its default value to **Restriction**, which serves as the default value for type derivation. You must change the **Derived By** property from **Restriction** to **List**, which causes the **Base Data Type** property to be renamed as the **Item Type** property (incidentally, the renamed property moves to a different position in the property list due to the alphabetical sorting of the properties).

When you first change a **Field Element** node or **Field Attribute** node from having a data type to having a base data type (thereby starting the process of simple type derivation), and then set the **Derived By** property to **List**, you can observe the following change in the corresponding fragment in the XSD view:

Simple Type Derivation Using the Union Mechanism

When you derive a new simple type from an existing simple type by using the union mechanism, you are specifying that the value for this attribute or element can be of more than one type, according to a list of types that you specify. For example, you can specify that an attribute or element value is either a date, a time, or a date/time value.

For comprehensive information about deriving new simple types by using the union mechanism, refer to the W3C website. For various links to this and other websites, see XSD Resources on the Web.

To derive a simple type as a union of several possible types, select the relevant **Field Element** node or **Field Attribute** node in the schema tree and then, in the Properties window, select a simple type from the drop-down list for the **Base Data Type** property. As soon as you have selected a value for this property, the **Derived By** property automatically changes from its default value to **Restriction**, which serves as the default value for type derivation. You must change the **Derived By** property from

Restriction to Union, which causes the **Base Data Type** property to be renamed as the **Member Types** property (incidentally, the renamed property moves to a different position in the property list due to the alphabetical sorting of the properties).

Finally, you can use the check boxes in the **Member Types** drop-down checklist to select additional types to allow for corresponding values in instance messages.

When you first change a **Field Element** node or **Field Attribute** node from having a data type to having a base data type (thereby starting the process of simple type derivation), and then set the **Derived By** property to **Union**, you can observe the following change in the corresponding fragment in the XSD view.

Schema Migration from Previous Versions of BizTalk Server

This version of BizTalk Server uses XML Schema definition (XSD) language to represent message schemas, while previous versions used the XML-Data Reduced (XDR) syntax to represent message schemas. If you are migrating from a previous version of BizTalk Server, you must convert your schemas to use XSD rather than XDR.

You need to perform the following tasks to convert a BizTalk schema from XDR syntax to XSD syntax:

- Change the extension of the schema file from .xdr to .xsd.
- Add the renamed schema to the appropriate BizTalk project.
- Convert the renamed schema from XDR to XSD by opening the schema in BizTalk Editor.
- Make the conversion permanent by saving the schema.

For detailed information about how to perform these steps, see *Migrating XDR Schemas to XSD Schemas*.

Ways to Use Message Content to Control Message Processing

There are two types of property promotion: **Distinguished Fields** and **Property Fields**, the latter of which uses property schemas. In BizTalk Editor, you manage both of these types of property promotion by using the **Promote Properties** dialog box, which is accessible by using the **Promote Properties** property of the **Schema** node.

You must decide which type of property promotion to use based on the details of your scenario. Consider the following distinctions between **Distinguished Field** property promotion and **Property Field** property promotion:

- **Distinguished Fields** cannot participate in message routing and therefore they do not appear in the filter expressions. They are only available within the context of an orchestration but they have less overhead when sending and receiving messages.
- **Distinguished Fields** do not have any size limitations. **Property Fields** are limited to 255 characters.

- **Distinguished Fields** do not require any separate artifacts to be created whereas **Property Fields** require the creation and maintenance of a property schema.

Drawing upon these considerations, you should promote nodes as **Property Fields** only when you intend to use the promoted properties for message routing or tracking purposes. Otherwise, if you are only going to access the promoted properties from within your orchestrations, you should take advantage of the fact that **Distinguished Fields** are much cheaper, more lightweight, and more easy-to-use than **Property Fields**.

Properties promoted by using the **Distinguished Field** mechanism are only accessible from within orchestrations and cannot be accessed from pipelines, ports, and so on. On the other hand, properties promoted by using the **Property Fields** and property schema mechanism are accessible from all of these components. Another important difference is that property field values are limited to 255 characters and distinguished field values have no such limit.

This section provides information about these two types of property promotion, including how information about how to establish such promoted properties for a message schema by using BizTalk Editor.

In This Section

- About BizTalk Message Context Properties
- Instance Message Processing Using Distinguished Fields
- Instance Message Processing Using Property Promotion

About BizTalk Message Context Properties

When a document is received by a BizTalk Server adapter, the adapter creates a BizTalk message for the document. The BizTalk message contains the document that was received as well as a message context. The message context is a container for various properties that are used by BizTalk Server when processing the document. Each property in the Message Context is composed of three things, a name, a namespace, and a value. For example, the following message context property describes the Interchange ID for a document:

Message context properties are added to the message context throughout the lifetime of the message as it passes through BizTalk Server.

There are two different types of message context properties used by BizTalk as described below:

Property Fields

Property fields are message context properties that are used by the BizTalk Messaging Engine for purposes of document routing, for message tracking, and for evaluation in Orchestrations. You can explicitly elevate a field in a document to the message context as a Property field by editing the schema for the document in the BizTalk Server Schema Editor that is available in Visual Studio 2005. In order to write a field in a document to the message context as a property field, the document schema must have an associated property schema. Property fields are limited to 255 characters. The **IsPromoted** property of Property fields in the message context is set to **True**.

Distinguished Fields

Distinguished fields are message context properties that do not require a separate property schema and that are only accessible from Orchestrations. Distinguished fields cannot be used for routing or tracking. Since Distinguished fields do not require a separate property schema, the evaluation of Distinguished fields by the Orchestration engine consumes less overhead than the evaluation of Property fields by the Orchestration engine. The evaluation of Property fields requires an XPath query, the evaluation of Distinguished fields does not require an XPath query. Distinguished fields do not have a size limitation. The **IsPromoted** property of Distinguished fields in the Message context is set to **False**.

Summary of differences between Property Fields and Distinguished Fields

The table below summarizes the differences and similarities between Property fields and Distinguished fields:

Attribute	Property Field	Distinguished Field
IsPromoted Property	True	False
Size Limitation	255 Characters	No limit
Used for Routing	Yes	No
Used for Tracking	Yes	No
Used in Orchestration	Yes	Yes
Requires Property Schema	Yes	No
Accessible by Pipelines and Ports	Yes	No

Instance Message Processing Using Distinguished Fields

Promoting properties by using the **Distinguished Field** mechanism does not require the creation of a property schema. As with all property promotion, you use the **Promote Properties** dialog box, which is accessible by using the **Promote Properties** property of the **Schema** node in message schemas, or by using the **Promote | Show Promotions** command on the **BizTalk** or shortcut menus.

In the **Promote Properties** dialog box, ensure the **Distinguished Fields** tab is selected in the right side of the dialog box. Then you expand the nodes in the schema tree on the left side of the dialog box to find and select the **Field Element** node or **Field Attribute** node that you want to promote as a distinguished field, and then click **Add**. For step-by-step instructions about promoting properties to **Distinguished Fields** using the **Promote Properties** dialog, see Copying Data to the Message Context as Distinguished Fields.

To remove a node from the set of properties being promoted as distinguished fields, select the promoted property on the **Distinguished Fields** tab, and click **Remove**.

When you promote properties by using the distinguished field mechanism, an XML Schema definition (XSD) language fragment is added within the annotation subelement of the root element. In the following example, the fragment shows two properties promoted by using the distinguished field mechanism.

Instance Message Processing Using Property Promotion

Promoting properties by using the **Property Field** method requires the creation of a property schema. For more information about creating a property schema, see [Creating Property Schemas](#). As with all property promotion, you use the **Promote Properties** dialog box, which is accessible by using the **Promote Properties** property of the **Schema** node in message schemas.

In the **Promote Properties** dialog box, make sure the **Property Fields** tab is selected in the right side of the dialog box. Next, ensure the appropriate property schema is included in the Property Schemas List at the top of the **Property Fields** tab. If necessary, use the folder button to select the appropriate property schema by using the **BizTalk Type Picker** dialog box. Next, expand the nodes in the schema tree on the left side of the dialog box to find and select the **Field Element** node or **Field Attribute** node that you want to promote as a property field, and then click **Add**. Finally, use the drop-down list in the **Property** column of the **Property-Fields dictionary** table to select a **Field Element** node in a property schema with which to associate the promoted property. For step-by-step instructions about promoting properties to **Property Fields** using the **Promote Properties** dialog, see [Copying Data to the Message Context as Property Fields](#).

To remove a **Field Element** node or **Field Attribute** node from the set of properties being promoted as property fields, select the promoted property in the **Property-Fields dictionary** table on the **Property Fields** tab, and click **Remove**.

The **Node Path** column in the **Property-Fields dictionary** table shows the **XPath** to the **schema** node corresponding to the promoted property. You can edit this value directly by using the **Edit Instance XPath** dialog box. You can open this dialog box by clicking the ellipsis (...) button that appears at the right end of the corresponding cell when you select that cell. Care must be taken when editing **XPaths** values directly because **XPaths** that cannot be resolved by BizTalk Editor will prevent proper validation operations.

BizTalk Editor also provides a streamlined command for promoting properties using the **Property Field** mechanism. This command is called **Quick Promotion**, and it is available using the **Promote | Quick Promotion** command on the BizTalk and shortcut menus. This command will promote the selected **Field** node (or **Record** node) to a **Property Field** that automatically created in the property schema specified by the **Default Property Schema Name** property in the **Property Pages** dialog box for the containing schema. For step-by-step instructions about promoting properties to **Property Fields** using the **Quick Promotion** command, see [Copying Data to the Message Context as Property Fields](#).

When you promote a property by using the property field mechanism, two XML Schema definition (XSD) language fragments are added to the XSD representation of the message schema. The first XSD fragment is an annotation fragment associated with the **schema** element that identifies the corresponding property schema, as in the following example.

The second XSD fragment is an annotation fragment associated with the **Root** element (regardless of whether it has been renamed) that identifies the **Field Element** node or **Field Attribute** node values that have been promoted by using the property field mechanism, as in the following example.

Schema Validation in Visual Studio

After you have finished creating a schema, or at different times during its creation, you can validate it to determine whether it contains any internal inconsistencies, or has other issues that might prevent it from being used effectively for process instance messages.

To validate a schema, use the **Validate Schema** command on the shortcut menu associated with the schema in Solution Explorer. The results of the validation are reported in the Visual Studio 2005 Output window.

For detailed step-by-step instructions regarding how to validate a schema, see [Validating Schemas](#).

Instance Message Generation and Validation

After you have validated a schema, you can use it to generate a sample instance message. The sample instance message that is generated contains the element and attribute structure specified by the schema, and generate fake data where required.

To generate a sample instance message from a schema, use the **Generate Instance** command on the shortcut menu associated with the schema in Solution Explorer. The results of the instance message generation operation are reported in the Visual Studio .NET Output window.

For detailed step-by-step instructions regarding how to generate an instance message from a schema, including how to configure an output file to contain the generated instance message, see [Generating Instance Messages](#).

If you have validated your schema, you can use BizTalk Editor to determine whether an instance message conforms to that schema.

To validate an instance message against a schema, use the **Validate Instance** command on the shortcut menu associated with the schema in Solution Explorer. The results of the validation are reported in the Visual Studio .NET Output window.

For detailed step-by-step instructions regarding how to validate an instance message, including how to specify the instance message to be validated, see [Validating Schemas](#).

Using BizTalk Editor

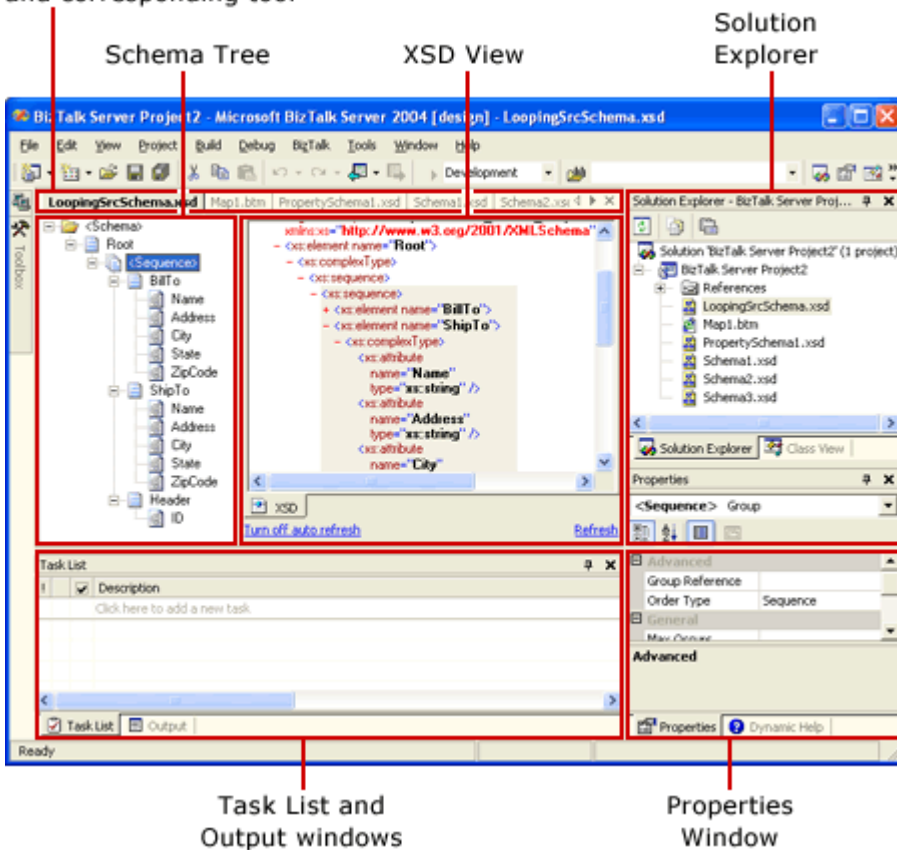
BizTalk® Editor resides within the Microsoft® Visual Studio® 2005 shell. Some of the functionality within BizTalk Editor relies upon existing user interface elements within the Visual Studio 2005 shell. For example, you use the **File**, **Edit**, and **View** menus just as you would for other development within Visual Studio 2005. Information about this common functionality is available from the **Help** menu.

BizTalk Editor becomes active when you add a new schema to a BizTalk project, when you open an existing schema (an .xsd file) within a BizTalk project, or when you reactivate a schema by clicking its tab in the main Visual Studio 2005 editing window.

The following figure shows the various views within the Microsoft Visual Studio 2005 shell that are part of BizTalk Editor.

BizTalk Editor Views

Tabs for choosing item and corresponding tool



BizTalk Editor consists of the following views within the Microsoft Visual Studio 2005 shell and their associated dialog boxes:

- **Tabs for choosing item and corresponding tool.** You choose which tool is active within Visual Studio 2005 by selecting a file of the type that corresponds to the tool you want to use. For example, select a tab that is displaying schema (a file with an .xsd extension) to activate BizTalk Editor. When you first open an item, such as a schema, the corresponding tab and tool are activated.
- **Schema Tree.** This view is on the left side of the main Visual Studio 2005 editing window. You actively construct your schema in this view by building up the tree structure that describes the structure of the message that the schema defines. For more information about how BizTalk schemas are represented in the schema tree view, see BizTalk Representation of Schemas.
- **XSD View.** This view is on the right side of the main Visual Studio 2005 editing window. It shows the XML Schema definition (XSD) language structure that represents the schema you are constructing in the schema tree view. This view is read-only and is provided to help you become accustomed to the XSD syntax of the schema you are creating. If you prefer, you can adjust the view splitter so that the XSD view only barely visible.
- **Solution Explorer.** This view is on the right side of the Visual Studio 2005 shell. It shows the BizTalk project and the various items it contains. Use Solution Explorer to add new and existing

schemas to the project, and to open schemas that are already part of the project. For example, to create a new schema, right-click the BizTalk project in the Solution Explorer window, click **Add**, click **Add New Item**, and then use the **Add New Item** dialog box to name and create a new schema.

- **Visual Studio 2005 Properties window.** You use this view to examine and set most of the schema and node properties. When you select a node in the schema tree view, or select a schema in the Solution Explorer window, the corresponding properties of that node or schema are displayed in the Properties window using the standard Visual Studio paradigms. For example, the properties are grouped into categories, and can be displayed according to these categories or alphabetically. For detailed information about the different sets of properties that are available when different types of nodes, or the schema, are selected, see **Schema Property Reference**.
- **Visual Studio 2005 Task List and Output windows.** You use these views to examine the resulting of validating your schemas, generating instance messages from your schemas, and validating instance messages against your schemas in much the same way that these views are used when compiling source code and building other types of projects.

In addition to these views, you can interact with several dialog boxes. You usually open these dialog boxes when you are editing a complex property such as a collection.

This section provides information about the views, commands, and shortcut keys available in BizTalk Editor.

In This Section

- Managing the Schema Tree View
- Managing the XSD View
- Managing the Properties Window
- Managing Other Visual Studio 2005 Windows
- Using BizTalk Editor Commands
- Working with Large Schemas

How to Manage the Schema Tree View

Management tasks with respect to the schema tree view can be divided into four categories: changing its size, changing its background color and font, changing its use of warning dialogs, and expanding and collapsing its tree structure. This topic provides step-by-step instructions for these various tasks.

To make the schema tree view taller or shorter

1. Move the mouse pointer to the bottom edge of the Microsoft® Visual Studio® 2005 main editing window, which displays the schema tree view side-by-side with the XSD view, until the cursor changes to the standard window vertical resizing icon.
2. Click and hold the left mouse button and drag the window edge either up or down.

You have vertically resized the schema tree view by vertically resizing the entire main editing window.

To make the schema tree view wider or more narrow

1. Move the mouse pointer to the pane divider in the Visual Studio 2005 main editing window, which divides the schema tree view from the XSD view, until the cursor changes to the standard window horizontal resizing icon.
2. Click and hold the left mouse button and drag the pane edge either left (narrower) or right (wider).

You have horizontally resized the schema tree view by changing the amount of the main editing window dedicated to the schema tree view, relative to the XSD view.

You can also make the schema tree view wider or narrower by horizontally resizing the entire main editing window.

To change the background color and/or font used by the schema tree view

1. In Visual Studio 2005, on the **Tools** menu, click **Options**.
2. In the **Options** dialog box, click the BizTalk Editor folder, and if necessary, expand the **Schema Display** category by clicking the plus (+) icon.
3. Change the background color and/or font by using the drop-down color picker and/or **Font** dialog box associated with the **Schema Tree Background Color** and **Schema Tree Font** properties, respectively.

Access the **Font** dialog box by using the ellipsis () button located at the right end of the **Schema Tree Font** property value box.

To change the warning dialogs used when working in the schema tree view

1. In Visual Studio 2005, on the **Tools** menu, click **Options**.
2. In the **Options** dialog box, click the **BizTalk Editor** folder, and if necessary, expand the **Editing Options** section by clicking the plus (+) icon.
3. Set any of the following properties to **True** or **False** by using the drop-down list accessed at the right edge of the respective property value box.

Property	Description
Show Structure Dialog Destroy Warning	When set to True , displays a warning dialog box before the schema structure is destroyed, and allows you to cancel the destructive operation.
Show Warning Dialog Encode	When set to True , displays a dialog box when the node name you have typed will not be valid in XML unless encoded, allowing you to cancel the naming operation,

	or proceed with encoding the name.
Show Invalid Insert Dialog	<p>When set to True, displays a warning dialog box for certain node insertion errors, and offers options about how to proceed. The possible node insertion errors include:</p> <ul style="list-style-type: none"> You have created duplicate Field Attribute nodes with the same name and the same parent node. You have created duplicate Record nodes with the same name and the same parent node, but with different underlying types.

To completely expand all or part of the schema tree

1. Select the node under which you want to completely expand the schema tree.

The selected node must be a node with a plus (+) or minus (-) icon next to it.

2. On the **BizTalk** menu, or on the shortcut menu for that node, click **Expand Schema Node**.

All nodes below the selected node are completely expanded.

If the schema tree below the selected node is already completely expanded, the **Expand Schema Node** menu item will be dimmed.

To completely collapse all or part of the schema tree

1. Select the node under which you want to completely collapse the schema tree.

The selected node must be a node with a plus (+) or minus (-) icon next to it.

2. On the **BizTalk** menu, or on the shortcut menu for that node, click **Collapse Schema Node**.

All nodes below the selected node are completely collapsed.

To expand a node of the schema tree

1. Click the plus (+) icon next to the node you want to expand.

To collapse a node of the schema tree

1. Click the minus (-) icon next to the node you want to collapse.

How to Manage the XSD View

Management tasks with respect to the XSD view can be divided into three categories: changing its size, changing its background color and font, and changing its refresh characteristics.

The latter category, concerning refresh characteristics, is most useful when working with large schemas, for which using automatic refresh might cause undesirable delays. In such cases, you can disable automatic (continuous) refresh, and instead refresh the XSD view manually as needed.

This topic provides step-by-step instructions for these tasks.

To make the XSD view taller or shorter

1. Move the mouse pointer to the bottom edge of the Microsoft® Visual Studio® 2005 main editing window, which displays the XSD view side-by-side with the schema tree view, until the cursor changes to the standard window vertical resizing icon.
2. Click and hold the left mouse button and drag the window edge either up or down.

You have vertically resized the XSD view by vertically resizing the entire main editing window.

To make the XSD view wider or more narrow

1. Move the mouse pointer to the pane divider in the Visual Studio 2005 main editing window, which divides the XSD view from the schema tree view, until the cursor changes to the standard window horizontal resizing icon.
2. Click and hold the left mouse button and drag the pane edge either left (wider) or right (narrower).

You have horizontally resized the XSD view by changing the amount of the main editing window dedicated to the XSD view, relative to the schema tree view.

You can also make the XSD view wider or narrower by horizontally resizing the entire main editing window.

To change the background color and/or font used by the XSD view

1. In Visual Studio 2005, on the **Tools** menu, click **Options**.
2. In the **Options** dialog box, click the BizTalk Editor folder, and if necessary, expand the **Schema Display** category by clicking the plus (+) icon.
3. Change the background color and/or font by using the drop-down color picker and/or **Font** dialog box associated with the **XSD View Background Color** and **XSD View Font** properties, respectively.

Access the **Font** dialog box by using the ellipsis () button located at the right end of the **XSD View Font** property value box.

To turn automatic refresh of the XSD view on and off

1. In BizTalk Editor, below the **XSD** tab, click **Turn off auto refresh** or **Turn on auto refresh**.

When auto-refresh is off, the XSD view is blank.

To manually refresh the XSD view

1. On the **BizTalk** menu, click **Refresh XSD**.

The XSD view updates to reflect any changes you have made to the schema tree.

How to Manage the Properties Window

BizTalk Editor uses the Microsoft® Visual Studio® .NET Properties window to display the values of the properties associated with nodes in the schema tree view, and provides an interface for changing those values. This topic provides step-by-step instructions for managing certain characteristics of the Properties window.

To turn property descriptions on and off

1. In the Properties window, right-click any property name, and then click **Descriptions**.

To arrange properties alphabetically within their categories

1. Click the **Categorized** toolbar button above the list of properties in the Properties window.

To arrange properties alphabetically without regard to their categories

1. Click the **Alphabetic** toolbar button above the list of properties in the Properties window.

How to Manage Other Visual Studio 2005 Windows

Certain operations in BizTalk Editor use other Microsoft® Visual Studio® 2005 windows, namely the Output window and the Task List window. These windows are used to display output from operations such as generating sample instance messages and validating schemas and instances. This topic provides step-by-step instructions about managing these windows.

To open the Output window

1. In Visual Studio 2005, on the **View** menu, point to **Other Windows**, and then click **Output**.

To open the Task List window

1. In Visual Studio 2005, on the **View** menu, point to **Other Windows**, and then click **Task List**.

By default, the Output window and the Task List window share the same section of the Visual Studio 2005 shell. If both windows are open at the same time, they are displayed with named tabs that allow you to switch back and forth between them.

Using BizTalk Editor Commands

When BizTalk Editor becomes active, it adds a menu called **BizTalk** to the Visual Studio 2005 shell. This menu provides access to the BizTalk Editor commands and their functionality. When BizTalk Editor is active, the **BizTalk** menu provides the commands that are specific to editing BizTalk schemas.

In addition and where applicable, BizTalk Editor uses existing Visual Studio 2005 menu items for commands that have obvious parallels to standard application functionality. For example, when BizTalk Editor is active, the **Save** command on the **File** menu saves changes to the schema that is currently being edited.

The following table describes the commands within BizTalk Editor that can be used in the process of developing schemas.

Command (Menu Locations)	Description
Open Schema (File Open File...)	Opens a BizTalk schema for editing in BizTalk Editor. Also available on the shortcut menu in Solution Explorer when a schema is selected, and by double-clicking a schema in Solution Explorer.
Close Schema (File Close)	Closes BizTalk Editor for the current schema, prompting to save any unsaved changes. Also available by using the standard Close button in the upper-right corner of the main editing window in Visual Studio 2005.
New Schema (File Add New Item... Schema) (Project Add New Item... Schema)	Creates a new BizTalk schema for editing and opens BizTalk Editor. Also available on the shortcut menu for the BizTalk project in Solution Explorer (Add Add New Item).
Save Schema (File Save schema.xsd) (File Save schema.xsd As...) (File Save All)	Saves the BizTalk schema currently being edited under its own name, under a new name, or as part of saving all unsaved changes, respectively.
Cut, Copy, Paste Node	Allows Schema nodes to be moved and duplicated within the schema tree view. When applicable, these commands are also available on the shortcut menu for the

(Edit Cut, Edit Copy, Edit Paste)	selected node, and by using the standard cut, copy, and paste shortcut keys: CTRL+X, CTRL+C, and CTRL+V.
Delete Node (Edit Delete) (BizTalk Delete)	Deletes the currently selected node in the schema tree view, with confirmation prompting. When applicable, this command is also available on the shortcut menu for the selected node.
Find Node (Edit Find and Replace Find)	Provides search functionality for node names in the schema tree view, which can be useful with large schemas.
Properties (View Properties Window) (BizTalk Properties)	Provides access to the properties of schema tree view nodes, and to higher-level objects like the schemas themselves. Also available on the shortcut menu for the selected node.
Insert Schema Nodes (BizTalk Insert Schema Node *)	Provides a way to insert a variety of different types of nodes into the schema tree view. For more information about the types of nodes that can be added, see BizTalk Representation of Schemas. When applicable, these commands are also available on the shortcut menu for the selected node.
Promote (BizTalk Promote *)	Provides different ways to promote properties. Also available on the shortcut menu for the selected node.
Expand Schema Node (BizTalk Expand Schema Node)	Completely expands the currently selected (and at least partially collapsed) node in the schema tree view. When applicable, this command is also available on the shortcut menu for the selected node.
Collapse Schema Node (BizTalk Collapse Schema Node)	Collapses the currently selected (and at least partially expanded) node in the schema tree view. When applicable, this command is also available on the shortcut menu for the selected node.
Refresh XSD	Refreshes the XSD view, which may be set to not refresh automatically by selecting

(BizTalk XSD) Refresh	the Turn Off Auto Refresh check box below the XSD view. This command is also available on the shortcut menu for the selected node, and by clicking Refresh below the XSD view.
Rename (BizTalk Rename)	Allows Record , Field Attribute , and Field Element nodes to be renamed, in place, within the schema tree view. This command is equivalent to changing the Node Name property of the selected node. When applicable, this command is also available on the shortcut menu for the selected node.
BizTalk Options (Tools Options BizTalk Editor)	Editor Allows configuration of confirmation dialogs (on or off), as well as font and background color adjustment.

The following table describes additional schema commands that are available on the shortcut menu for the selected schema in Solution Explorer.

Schema command	Description
Open	Opens the selected schema in BizTalk Editor.
Open With	Allows the selected schema to be opened in a variety of XSD editors, including BizTalk Editor.
Validate Schema	Validates the selected schema.
Validate Instance	Validates the instance specified by the Input Instance Filename property in the Property Pages dialog box against the selected schema. The Property Pages dialog box can be opened by clicking Properties at the bottom of the shortcut menu being described here.
Generate Instance	Generates an instance of the selected schema using the file name specified by the Output Instance Filename property in the Property Pages dialog box against the selected schema. If no name is specified for the generated output instance, a default file name will be used. The default file name is the schema file name prepended to the string "_output.xml" in the folder _SchemaData in a temp folder in your Documents and Settings folder.

Exclude From Project	Removes the currently selected schema from the current BizTalk project. Use the Add Existing Item command to re-add a schema that was previously excluded from the current BizTalk project.
Cut, Copy, Paste	Use these commands to perform the standard Visual Studio 2005 behavior of cutting, copying, and pasting entire schemas within a BizTalk project.
Delete	Permanently deletes the currently selected schema, with a confirmation prompting.
Rename	Allows the currently selected schema to be renamed, in place.
Properties	Opens the Property Pages dialog box for the currently selected schema, in which some of the properties of the schema can be examined and set. For more information about setting schema properties and the properties of schemas, see Setting Schema File and Schema Item Properties and Schema File Properties , respectively.

Working with Large Schemas

When your schemas become very large, you may find that refreshing the XSD view is increasingly slow and that the response of other aspects of the user interface can be impacted as well. The solution to this issue is to turn off the auto-refresh feature and instead use the manual **Refresh** link in the XSD view to periodically refresh the XSD view. For step-by-step instructions about how turn the auto-refresh feature on an off, see the procedure "To turn automatic refresh of the XSD view on and off" in Managing the XSD View.

Performance may also be slow in large schemas with multiple roots attached to the **schema** node. Setting the **Root Reference** property may increase performance in the user interface. Setting **Root Reference** improves performance because the compiler creates C# classes for all root schemas. A single root means the creation of fewer classes.

Validate Instance may appear slow in the user interface because a validation is always done on the schema before validating the instance. Schema validation occurs only in the user interface. Setting the **Root Reference** property also improves user interface performance in this case.

Creating Schemas

You can use BizTalk® Editor to create two types of schemas: message schemas and property schemas.

Message schemas define the structure and constrain the content of messages that you expect to send to, and receive from, your trading partners or applications. Message schemas are the most common types of schemas that you will use with Microsoft® BizTalk® Server 2006. BizTalk Server 2006 can create XML message schemas for both business documents and the envelopes used to contain them, and through the use of the Flat File Extension to BizTalk Editor, it can create flat file message schemas, including schemas for headers, bodies and trailers.

Property schemas are a special type of schema. Property schemas provide a validation template for field and/or record data that is promoted from within an instance message into what is known as the message context. The purpose of property schemas is to provide a formal, strongly typed definition of the data to be promoted at run time.

Property promotion provides a centralized mechanism for pulling key pieces of information, defined by you, from within an instance message and making it more easily accessible to BizTalk Server components that are handling the message as it passes through BizTalk Server. One common use of promoted properties is in matching a message instance to a subscription, allowing the message to be properly routed for processing.

This section describes the ways in which you can create different types of schemas within BizTalk Server, and presents related subjects that pertain to multiple types of schemas.

In This Section

- Managing Schemas Within Projects
- Managing the Nodes Within a Schema
- Promoting Properties

Managing Schemas Within Projects

This section provides step-by-step instructions for working with entire schemas; for example, the steps involved in creating a schema, using multiple schemas together, and saving schemas. For information about working with nodes within a schema, see [Managing the Nodes Within a Schema](#).

In This Section

- Creating Schemas for XML Messages
- Creating Schemas for Flat File Messages
- Creating Schemas for Envelopes
- Creating Property Schemas
- Adding Existing Schemas
- Saving, Renaming, and Closing Schemas
- Including and Excluding Schemas
- Deleting Schemas
- Setting Schema File and Schema Item Properties
- Migrating XDR Schemas to XSD Schemas

- Creating Schemas That Use Other Schemas

How to Create Schemas for XML Messages

There are several methods for creating BizTalk message schemas. This topic provides step-by-step instructions for some of those methods.

To create a new schema

1. In **Solution Explorer**, select the BizTalk project to which you want to add a schema.
2. On the **Project** menu, click **Add New Item**.
3. In the **Add New Item - <BizTalk ProjectName>** dialog box, in the **Templates** section, click **Schema**.
4. In the **Add New Item - <BizTalk ProjectName>** dialog box, in the **Name** box, type a name for the schema, and then click **Open**.
5. If necessary, press F4 to open the Visual Studio 2005 Properties window.
6. In the schema tree view, select the **Schema** node, and then in the Properties window, select the **Target Namespace** property and type a name for the target namespace. It is important that you set this property in this initial phase of schema creation; avoid using the default **Target Namespace** property value.

To generate a schema from a non-XSD source

1. In **Solution Explorer**, right-click a BizTalk project, point to **Add**, and then click **Add Generated Items**.
2. In the **Add Generated Items - <BizTalk ProjectName>** dialog box, in the **Templates** section, click **Generate Schemas**, and then click **Open**.
3. In the **Generate Schemas** dialog box, in the **Document type** drop-down list, select **XDR Schema**, **DTD Schema**, or **Well-Formed XML**.

If you see either **DTD (Not Loaded)** or **Well-Formed XML (Not Loaded)** in the drop-down list, select the appropriate document type anyway, and you will be guided through the process of installing the missing DLL. Then repeat these steps.

4. In the **Generate Schemas** dialog box, click **Browse**, locate the file you want to import, and then click **OK**. The file you locate must match the document type you selected in the previous step.

A new schema is generated from the specified file, using the same name as that file with the .xsd extension, and opened in BizTalk Editor.

How to Create Schemas for Flat File Messages

After creating an XML message schema as described in *Creating Schemas for XML Messages*, use the **Schema Editor Extensions** property of the **Schema** node to enable the flat file extension. Enabling the flat file extension adds a considerable number of properties to many of the node types within a schema. These properties are generally used to control how a flat file business document is translated to and from its equivalent XML business document, and their values are stored as XML Schema definition language (XSD) annotations within the schema file. Using these properties properly takes some practice and a thorough understanding of which aspects of the flat file format they each govern. For conceptual and reference information about the flat file properties, see *Considerations When Creating Flat File Message Schemas* and **Supplemental Node Properties for Flat File Schemas**, respectively.

How to Create Schemas for Envelopes

After creating an XML message schema as described in *Creating Schemas for XML Messages*, set the **Envelope** property of the **Schema** node to **Yes**. Depending on certain characteristics of your envelope schema, such as whether there are multiple root nodes, you will need to set several other envelope-specific properties. For more information, see *Envelope Schemas*.

The body XPath property of the envelope points to the element that contains the document elements. The actual element where the XPath points to does not belong to the document.

How to Create Property Schemas

If you choose to promote fields as property fields, you will need to define a property schema first. This property schema specifies an unstructured collection of fields into which you can promote fields from within an instance message defined by a schema associated with your property schema.

To create a property schema

1. In **Solution Explorer**, right-click a BizTalk project, point to **Add**, and then click **Add New Item**.
2. In the **Add New Item** dialog box, in the **Templates** section, click **Property Schema**.
3. Select the text in the **Name** box, type a name for the schema, and then click **Open**.

The new property schema opens, and it already contains a **Field Element** node named Property1.

4. In the schema tree, right-click that **Field Element** node, click **Rename**, type a descriptive name for the first property in the schema, and then press ENTER.
5. Set the **Data Type** and other relevant properties, as appropriate, for the **Field Element** node in the Properties window.
6. If you want to insert **Field Element** nodes for additional properties, right-click the <Schema> node, click **Insert Schema Node**, and then click **Child Field Element**, and then repeat steps 4 and 5. Repeat as necessary to create the required set of **Field Element** nodes into which you intend to promote values from instance messages.

How to Add Existing Schemas

Sometimes you will want to add an existing schema to an existing BizTalk project. This topic describes the required steps.

To add an existing schema

1. In **Solution Explorer**, select the BizTalk project to which you want to add an existing schema.
2. On the **Project** menu, click **Add Existing Item**.
3. In the **Add Existing Item - <BizTalk ProjectName>** dialog box, browse to the location of the schema, select the schema, and then click **Open**.

The existing schema is added to the BizTalk project but is not automatically opened in BizTalk Editor.

How to Save, Rename, and Close Schemas

In BizTalk Server 2006, schemas are XML Schema definition (XSD) language files and reside on the file system with .xsd extensions. When you use BizTalk Editor to develop schemas, you will routinely need to save and close schema files, and occasionally you may need to rename them. This topic describes the steps required to perform these basic operations.

To save a schema

1. If necessary, activate BizTalk Editor for the schema to be saved by clicking the appropriate tab at the top of the main editing window in Microsoft® Visual Studio® 2005.
2. On the **File** menu, click **Save <Name of Schema>**.

If the schema had unsaved changes, its name as displayed on the tab at the top of the main editing window will no longer end with an asterisk (*), which is used to indicate unsaved changes.

To rename a schema

1. In Solution Explorer, right-click the schema that you want to rename, and then click **Rename**.
2. In the editing box that appears in the location of the schema in Solution Explorer, type the new name for the schema, or alter its existing name, and then press ENTER.

To close a schema

1. If necessary, activate BizTalk Editor for the schema to be closed by clicking the appropriate tab at the top of the main editing window in Microsoft® Visual Studio® 2005.
2. On the **File** menu, click **Close**.

BizTalk Editor closes for the schema that has been closed.

How to Include and Exclude Schemas

A schema file can exist in a BizTalk project folder and not be included in that project. Such a schema is said to be excluded from the project. Excluded schemas are not compiled when you build the BizTalk project. This topic describes the steps required to include an excluded schema in a BizTalk project, and to exclude a schema from a BizTalk project.

To include a schema in a BizTalk project

1. In Solution Explorer, open the BizTalk project that contains the schema to be included in its project folder.
2. If all files are not already showing, on the **Project** menu, click **Show All Files**.
3. In Solution Explorer, right-click the excluded schema that you want to include, and then click **Include In Project**.

The icon for the previously excluded schema in Solution Explorer changes from an empty outline to the normal schema icon.

4. Optionally, on the **Project** menu, click **Show All Files** to hide all files in the project folder that are not included in the project.

To exclude a schema from a BizTalk project

1. In Solution Explorer, right-click the schema that you want to exclude, and then click **Exclude From Project**.

The schema is now excluded from the BizTalk project.

How to Delete Schemas

Sometimes you might want to not only exclude a schema from its BizTalk project, but also completely remove the schema from your hard disk. This is known as deleting, as opposed to excluding, a schema. This topic describes the steps required to delete a schema.

To delete a schema

1. In Solution Explorer, right-click the schema that you want to delete, and then click **Delete**.
2. In the confirmation dialog box, click **OK**.

How to Set Schema File and Schema Item Properties

Schema properties are examined and set in two different ways. Some properties are set in a **Property Pages** dialog box, while others are set in the Visual Studio .NET Properties window. The former set of properties is used for schema generation, schema validation, and the **Quick Promotion** feature.

The latter set of properties is those which schemas share with other types of BizTalk items, such as maps and orchestrations.

This topic describes the steps required to examine and/or set properties in both ways.

To set schema file properties in the Property Pages dialog box

1. In Solution Explorer, right-click the schema for which you want to examine and/or set the relevant properties, and then click **Properties**.
2. In the **<Schema File Name> Property Pages** dialog box, on the **General** tab, examine and/or set the properties of interest. For information about the properties available in the **Property Pages** dialog box for a BizTalk schema, see **Schema File Properties**.
3. When you have finished examining and/or setting schema file properties, click **OK** or **Cancel** as appropriate.

To set schema item properties in the Properties window

1. In Solution Explorer, select the schema for which you want to examine and/or set the relevant properties.
2. If necessary, press F4 to open the Visual Studio .NET Properties window.
3. In the Properties window, examine and/or set the properties of interest. For information about the properties available in the Properties window for a BizTalk schema, see **Schema File Properties**.

Any new property values you set are effective immediately without any requirement to approve or cancel the changes you have made.

How to Migrate XDR Schemas to XSD Schemas

If you are migrating schemas from a previous version of BizTalk Server, you will need to convert your XML-Data Reduced (XDR) schemas into XML Schema definition (XSD) language schemas. This topic describes the necessary steps.

To generate an XSD schema from an XDR schema

1. In **Solution Explorer**, right-click the relevant BizTalk project, point to **Add**, and then click **Add Generated Items**.
2. In the **Add Generated Item - <BizTalk ProjectName>** dialog box, in the **Templates** section, click **Generate Schemas**, and then click **Open**.
3. In the **Generate Schemas** dialog box, in the **Document type** list, select **XDR Schema**.
4. In the **Generate Schemas** dialog box, click **Browse**, locate the XDR schema file you want to migrate, and then click **OK**.

A new schema is generated from the specified XDR schema file, using the same name as that file with the .xsd extension, and then opened in BizTalk Editor.

How to Create Schemas That Use Other Schemas

XML Schema definition (XSD) language provides three different but related mechanisms for using one schema within another schema. These mechanisms are importing a schema, including a schema, and redefining a schema. For a brief summary of these mechanisms and how they differ, see *Schemas That Use Other Schemas*. For detailed information, see *XSD Resources on the Web* for links to the XSD primer and specifications.

This topic describes the steps required to import, include, and redefine other schemas within the schema you are developing.

To import, include, or redefine one schema within another schema

1. In BizTalk Editor, open the schema to which you want to import, include, or redefine another schema. You can open a schema by double-clicking it in Solution Explorer.
2. Select the **Schema** node at the top of the schema tree view.
3. If necessary, press F4 to open the Visual Studio 2005 Properties window.
4. In the Properties window, in the **Advanced** category, in the value portion of the **Imports** property, click the ellipsis (...) button.
5. In the **Imports** dialog box, in the **Import New Schema as** list, select **XSD Import**, **XSD Include**, or **XSD Redefine**, as appropriate, and then click **Add**.
6. In the **BizTalk Type Picker** dialog box, expand the **Schema** node in the project tree, select the schema that you want to import, include, or redefine, and then click **OK**.
7. In the **Imports** dialog box, click **OK**.

The appropriate XSD directives to implement the import, include, or redefine operation are added to the **schema** element in the XSD view, including a new **import**, **include**, or **redefine** element, as appropriate.

Managing the Nodes Within a Schema

After you have a schema open for editing within BizTalk Editor, you will perform tasks like adding new nodes to the schema tree, setting properties on those nodes, examining and changing the properties of existing nodes, and so on. This section provides step-by-step instructions for performing a variety of such tasks.

In This Section

- Inserting Nodes into a Schema
- Creating References to Another Node or Type
- Working with Existing Nodes

Inserting Nodes into a Schema

One of the main types of tasks you will perform in the course of developing a schema is inserting nodes of various types into the schema tree at different locations and depths, thereby creating the structure that will represent the corresponding instance messages. This section describes several ways to insert a node.

In This Section

- Inserting a Record, Field Element, or Field Attribute Node
- Creating a New Occurrence of an Existing Record, Field Element, or Field Attribute Node
- Inserting a Sequence Group, Choice Group, or All Group Node
- Inserting an Any Element or Any Attribute Node

How to Insert a Record, Field Element, or Field Attribute Node

Record nodes (including the **Root** node), **Field Attribute** nodes, and **Field Element** nodes are unique in that they can be renamed so that their names represent the names of the actual, custom-named elements in a corresponding instance message. For example, if you name a **Record** node **FullName**, at the corresponding location in an instance message an XML element named **FullName** is expected. If that **Record** node named **FullName** has a child **Field Attribute** node named **RequireFullName** (with its **Min Occurs** and **Max Occurs** properties set to **1**), the **FullName** element in a corresponding instance message will need to have an attribute named **RequireFullName** associated with it.

All **Record** nodes, when initially inserted, are represented in the XSD as with a **complexType** element, but not with a subsequent **sequence**, **choice**, or **all** element. For this reason, the **Group Order Type**, **Group Max Occurs**, and **Group Min Occurs** properties of the **Record** node are not available for modification.

As soon as you add a child **Record** or **Field Element** node to the **Record** node, a **sequence** element is added to the XSD representation within the **complexType** element to contain this first child node, and the **Group Order Type** property of the **Record** node shows a value of **Sequence**. In most circumstances, you can change the **Group Order Type** property from **Sequence** to **Choice**, and in more limited circumstances, from **Sequence** to **All**, thereby changing the element pair that contains the corresponding child nodes to either **complexType/choice** or **complexType/all**, respectively.

Field Attribute nodes cannot have the same node names while in the same scope.

Record and **Field Element** nodes can have the same node names while in the same scope only if the following conditions are met:

- They have the same data type.
- They are not within an **All Group** node.

To insert a new child **Record** node, **Field Element** node, or **Field Attribute** node within the **Schema** node or an existing **Record** node

1. Select the **Schema** node or an existing **Record** node.
2. On the **BizTalk** menu, point to **Insert Schema Node**, and then click **Child Record**, **Child Field Element**, or **Child Field Attribute**, as appropriate.

A child node of the chosen type is added after either the last node in the schema tree (when inserting within the **Schema** node) or after the last existing child node of the selected **Record** node (when inserting within an existing **Record** node).

3. Type a name for the newly inserted **Record**, **Field Element**, or **Field Attribute** node, and then press ENTER.

To insert a sibling **Record** node, **Field Attribute** node, or **Field Element** node within an existing **Record** node

1. Select any child node of the **Record** node into which you want to insert the sibling **Record**, **Field Attribute**, or **Field Element** node.
2. On the **BizTalk** menu, point to **Insert Schema Node**, and then click **Sibling Record**, **Sibling Field Attribute**, or **Sibling Field Element**, as appropriate.

A sibling node of the chosen type is inserted at the end of the siblings of the selected node.

3. Type a name for the newly inserted **Record**, **Field Attribute**, or **Field Element** node, and then press ENTER.

How to Create a New Occurrence of an Existing **Record**, **Field Element**, or **Field Attribute** Node

You can create new instances of an existing **Record**, **Field Element**, or **Field Attribute** node such that subsequent modifications to any instance are reflected in all instances.

To create a new instance of an existing **Record**, **Field Element**, or **Field Attribute** node

1. Select the original **Record**, **Field Element**, or **Field Attribute** node, make sure its **Base Data Type** property is set correctly, and then in its **Data Structure Type** (**Record** nodes) or its **Data Type** (**Field Element** and **Field Attribute** nodes) property, type a custom data type name.
2. Insert the new **Record**, **Field Element**, or **Field Attribute** node and set one of the following properties to the custom data type name you typed in step 1.
 - Data Structure Type (**Record** nodes)

- Data Type (**Field Element** and **Field Attribute** nodes)

3. You have created a new instance of the existing **Record**, **Field Element**, or **Field Attribute** node.

How to Insert a Sequence Group, Choice Group, or All Group Node

BizTalk Editor supports three types of group nodes for elements: **Sequence Group**, **Choice Group**, and **All Group**. These different types of group nodes establish different constraints on the child nodes of the group in corresponding instance messages. For information about the constraints of these different types of groups, you should refer directly to information about the XML Schema definition (XSD) language on the Web. For links to this information, see XSD Resources on the Web.

BizTalk Editor also supports a group node for attributes, the **Attribute Group** node. This type of node allows a set of attributes to be defined once, and then be associated with more than one **Record** node within the schema.

When you build structure into your schema, **Record** nodes are assumed to contain ordered sequences of child nodes by default, and are represented by using **sequence** elements in the XSD representation of the schema. You can change the type of a group node by changing its **Order Type** property.

To insert a Sequence Group node or a Choice Group node

1. In the schema tree view, select the **Record** node in which you want to insert the **Sequence Group** node or the **Choice Group** node.
2. On the **BizTalk** menu, point to **Insert Schema Node**, and then click **Sequence Group** or **Choice Group**, as appropriate.

To insert an All Group node

1. In the schema tree view, select the new **Record** node in which you want to insert the **All Group** node.
2. On the **BizTalk** menu, point to **Insert Schema Node**, and then click **All Group**.

The **All Group** choice is only available on the BizTalk (or shortcut) menu when the relevant parent **Record** node meets the constraints that XSD imposes on the use of the **all** element.

The **All Group** choice requires that the **Record** node have a base data type. A quick way to give the node a data type is to set its **Content Type** to **Complex**.

To insert an Attribute Group node

1. In the schema tree view, select the **Record** node in which you want to insert the **Attribute Group** node.
2. On the **BizTalk** menu, point to **Insert Schema Node**, and then click **Attribute Group**.

If you want to change the name of the newly inserted **Attribute Group** node, type the new name in its **Group Reference** property.

How to Insert an Any Element or Any Attribute Node

BizTalk Editor supports two types of any nodes: **Any Element** and **Any Attribute**. These nodes allow you to create locations within your schema that correspond to locations within the corresponding instance messages where you do not know what elements or attributes will appear. For detailed information about how these nodes are interpreted when processing instance messages, refer directly to information about the XML Schema definition (XSD) language on the Web. For links to this information, see XSD Resources on the Web.

To insert an Any Element node or an Any Attribute node

1. In the schema tree view, select the **Record** node into which you want to insert the **Any Element** node or the **Any Attribute** node.
2. On the **BizTalk** menu, point to **Insert Schema Node**, and then click **Any Element** or **Any Attribute**, as appropriate.

How to Create References to Another Node or Type

You can use global nodes to create reusable data types—fragments of structure—that you can use throughout the schema wherever that structure is appropriate. You can only use nodes that are direct children of the **Schema** node to create global types.

You can also create cyclical references using the data types of nodes that are not direct descendants of the **Schema** node. This is useful for representing recursive structures in schemas.

This topic provides step-by-step instructions for various types of global nodes and how to refer to them to use them.

Creating Global Declarations

You can create global types using records, fields or attributes. Global types created from records may only be used in records, types created from fields only in fields, attribute types only in attributes. The following procedures describe how to define and use global declarations.

To create a global declaration from a node

1. Select the **Record**, **Field Attribute**, or **Field Element** node whose type you want to make globally available.
2. In the **Properties** window, type a name in the **Data Structure Type** list that will be used as the global name for the complex type, and then press ENTER.

To create a globally defined Sequence Group node, Choice Group node, or All Group node

1. Select the **Record** node into which you want to insert the globally defined group node.

2. On the **BizTalk** menu, point to **Insert Schema Node**, and then click **Sequence Group**, **Choice Group**, or **All Group**, as appropriate.
3. Create a structure in the newly inserted group. For example, insert **Record** or **Field Element** nodes to express the structure of the data within the group node.
4. Select the group node inserted in step 2.
5. In the Properties window, click **Group Reference**, type a name into the value field, and then press ENTER.

By providing a name in the **Group Reference** property, you have globally defined group node, after which you can associate other group nodes with this globally defined type (structure).

To create a globally defined Attribute Group node

1. Select the **Record** node into which you want to insert the globally defined **Attribute Group** node.
2. On the **BizTalk** menu, point to **Insert Schema Node**, and then click **Attribute Group**.

This adds an **Attribute Group** node to the end of the child nodes in the selected **Record** node.

3. Add the appropriate **Field Attribute** or **Attribute Group** nodes to your **Attribute Group**.
4. Optionally, if you want to rename the **Attribute Group** node, select the **Attribute Group** node and change its **Group Reference** property to a new name of your choosing.

Attribute groups are always global and referenced from their point of use.

To use a type or group that has been globally defined

1. Select the node for which you want to use a globally defined type.
2. In the Properties window, select the globally defined type from the drop-down list for the **Data Structure Type** property (**Record** nodes), **Data Type** property (**Field Element** and **Field Attribute** nodes), or **Group Reference** property (**Sequence Group**, **Choice Group**, **All Group**, and **Attribute Group** nodes).

After you have created a global declaration, you cannot delete it in a single step. However, you can delete it by using the **Cleanup Global DataTypes** dialog box when the schema is saved, using the following procedure.

To delete a global declaration

1. Delete all of the nodes where this global type or group is used, or specify a different type or group to be used in all of those nodes, or some combination thereof. For step-by-step instructions for deleting a node, see *Deleting Nodes*.

2. Upon saving your specification, the **Cleanup Global DataTypes** dialog box appears. Select the global declaration you want to completely delete from your specification, and then click **OK**.

Creating Cyclical References to Another Node

You can create cyclical references to a node to represent recursive schema elements. You do this by creating a node whose type is defined by an enclosing record. For example, consider an instance message that is wrapped in an arbitrary number of envelopes having the same structure. Using cyclical references, you can create a schema that defines such instance messages.

To create a cyclical reference

1. Select a **Record** node for which you want to create a recursive reference. This is the node representing the top of the recursive structure.
2. In the Properties window, verify that the **Data Structure Type** has a value.

Verifying that the **Record** node has a named type associated with it is necessary because recursive structures are defined when a type contains itself. Types can only contain themselves through nested use of named global types.

3. Select a child **Record** node or insert a child **Record** node.
4. For the child **Record** node, in the Properties window, in the **Data Structure Type** list, select the data structure identified in step 2.

Working with Existing Nodes

After nodes have been placed into the schema tree, you will probably find it necessary to move nodes to different locations, rename them, and so on. This section provides step-by-step instructions for the most common such tasks.

In This Section

- Working with Multiple Nodes Simultaneously
- Moving and Copying Nodes
- Renaming Nodes
- Deleting Nodes
- Setting Node Properties

How to Work with Multiple Nodes Simultaneously

Some operations can be performed on multiple nodes simultaneously, but the relevant nodes must all be selected. This topic provides instructions about how to work with multiple nodes simultaneously.

To select multiple nodes that are not adjacent

1. Select the first node in the set of nonadjacent nodes to be selected at the same time.
2. Press and hold CTRL and then select the remaining nodes in the set of nonadjacent nodes to be selected at the same time.

The nonadjacent nodes are all selected at the same time.

To select a range of adjacent nodes

1. Select the node at one extreme of the set of adjacent nodes to be selected at the same time.
2. Press and hold SHIFT and then select the node at the other extreme of the set of nodes to be selected at the same time.

The range of adjacent nodes defined by the nodes selected in steps 1 and 2 are all selected at the same time.

To modify property values of simultaneously selected nodes

1. Select multiple nodes, as described in the procedure "To select multiple nodes that are not adjacent". earlier in this document.
2. Modify the relevant property values in the **Properties** window.

How to Move and Copy Nodes

You will probably encounter situations where you want to move an existing node to a different location in the schema tree. You might also be able to save time by making a copy of an existing node and then pasting it into a different location in the schema tree. This topic provides step-by-step instructions for performing these tasks.

To move a node within a schema

1. If necessary, expand the schema tree to show both the node you are moving and the location to which you want to move it. For more information about expanding and collapsing the schema tree view, see *Managing the Schema Tree View*.
2. Click and hold the node that you want to move and drag it to another location in the tree.

To copy a node within a schema

1. Right-click the node that you want to copy, and then click **Copy**.
2. Right-click the **Record** node or group node into which you want to insert the copied node, and then click **Paste**.

The copy of the copied node is placed at the end of the child nodes of the **Record** node or group node you selected in step 2.

How to Rename Nodes

There will probably be times when you want to rename existing nodes. This topic describes two variations on this task.

To rename a single node

1. Select the **Record** node, **Field Element** node, or **Field Attribute** node that you want to rename.
2. On the **BizTalk** menu, click **Rename**.
3. Type a new name for the node, and then press ENTER.

To rename multiple nodes to have the same name

1. Select the first node in the set of nodes to be given the same name.
2. Press and hold CTRL, and then select the remaining nodes to be given the same name.
3. In the Properties window, change the value of the **Node Name** property, and finish by pressing ENTER.

All selected nodes are updated in the schema tree view with the new name.

How to Delete Nodes

There will be times when you want to delete a node in the schema tree. This topic provides step-by-step instructions for this task.

To delete a node

1. Right-click the node that you want to delete, and then click **Delete**.
2. Click **Yes** in the confirmation dialog box.

How to Set Node Properties

After inserting a node into a BizTalk schema, you will often need to change the properties of that node from their default values. Each type of node has a distinct set of properties, and within one of those sets, the settings of one property can affect the availability of other properties. For example, before setting the **Default Wrap Character** property of the **Schema** node, you must set the **Default Wrap Character Type** property to either **Character** or **Hexadecimal**, thereby establishing the format in which you intend to represent the former property. Further, neither of these properties is available, nor is the entire **Parse**

property category to which they belong, unless **Flat File Extension** is enabled by using the **Schema Editor Extensions** property.

Node properties are extensive and can be complex, as is the XML Schema definition (XSD) language that they support. This topic only briefly describes the general steps involved in examining and setting node properties. For detailed information about these properties, including, for example, information about their default and allowed values, see **Schema Property Reference**. For even more detailed information about the XSD concepts and elements that underlie most of these node properties, refer directly to information about XSD on the Web. For links to this information, see XSD Resources on the Web.

To examine a node property value

1. In BizTalk Editor, open the schema that contains the property you want to examine, and then select the node that contains that property.
2. If necessary, press F4 to open the Visual Studio .NET Properties window.
3. If necessary, scroll the Properties window to find the property of interest, and note its value.

You can use buttons at the top of the Properties window to change the way that the properties are sorted, either alphabetically within their categories, or alphabetically without regard for (or display of) their categories.

To set a node property value

1. In BizTalk Editor, open the schema that contains the property you want to set, and then select the node that contains that property.
2. If necessary, press F4 to open the Visual Studio .NET Properties window.
3. If necessary, scroll the Properties window to find the property of interest.

You can use buttons at the top of the Properties window to change the way that the properties are sorted, either alphabetically within their categories, or alphabetically without regard for (or display of) their categories.

4. Set the value of the property in the value field, which is to the right of the property name. Depending on the type of the property, you might type a value or choose a value from the drop-down list that is displayed when the value field is selected. Some properties allow either operation. Some properties display an ellipsis (...) button that when clicked opens a dialog box in which more complicated values, such as collections, can be set.
5. If you have typed a new value for the property, finish by pressing ENTER.

To clear a node property value

1. Select the node that contains the property of interest.

2. In the Properties window, double-click the property value that you want to clear, right-click the property value, and then click **Delete**.

To restore a node property to its default value

1. Select the node that contains the property of interest.
2. In the Properties window, right-click the property name that you want to reset to its default value, and then click **Reset**.

Promoting Properties

Promotion of properties involves either promoting **Field Element** or **Field Attribute** nodes in a schema to be **Distinguished Fields** or **Property Fields**. You can also promote **Record** nodes as **Property Fields** if they have simple content (**Content Type** property of the **Record** node set to **SimpleContent**).

To promote a **Record** (with simple content), **Field Element**, or **Field Attribute** node as a **Property Field**, you may first define a special type of schema called a property schema. Property schemas define an unstructured set of **Field Element** nodes into which you promote **Record** (with simple content), **Field Element**, or **Field Attribute** nodes. For step-by-step instructions for creating a property schema, see [Creating Property Schemas](#).

Alternatively, you can use the **Quick Promotion** feature, which will automatically create and update a single property schema whenever you promote a new **Field Element**, **Field Attribute**, or **Record** (with simple content) node.

XSD and CLR Data Types

In some places, such as in property promotion, XSD data types are promoted to Common Language Runtime (CLR) data types. The following table shows the XSD data types that can be promoted and the corresponding CLR data types.

XSD Data Type	CLR Data Type
anyURI	String
Boolean	Boolean
byte	sbyte
date	DateTime
dateTime	DateTime
decimal	Decimal
double	Double

ENTITY	String
float	Single
gDay	DateTime
gMonth	DateTime
gMonthDay	DateTime
gYear	DateTime
gYearMonth	DateTime
ID	String
IDREF	String
int	Int32
integer	Decimal
language	String
Name	String
NCName	String
negativeInteger	Decimal
NMTOKEN	String
nonNegativeInteger	Decimal
nonPositiveInteger	Decimal
normalizedString	String
NOTATION	String
positiveInteger	Decimal
QName	String

short	Int16
string	String
time	DateTime
token	String
unsignedByte	Byte
unsignedInt	UInt32
unsignedShort	UInt16

This section provides step-by-step instructions for promoting nodes as either **Distinguished Fields** or as **Property Fields**.

In This Section

- Opening the Promote Properties Dialog Box
- Copying Data to the Message Context as Distinguished Fields
- Copying Data to the Message Context as Property Fields

How to Open the Promote Properties Dialog Box

Property promotion is managed in the **Promote Properties** dialog box. This topic provides step-by-step instructions for opening this dialog box to manage your promoted properties.

There are two ways to open the Promote Properties dialog box.

To open the Promote Properties dialog box from the Schema node

1. In the schema tree, select the **<Schema>** node.
2. In the Properties window, click the ellipsis (...) button in the value field of the **Promote Properties** property.

The **Promote Properties** dialog box opens.

To open the Promote Properties dialog box from any node

1. In the schema tree, select any node.
2. Right-click the selected node, click **Promote**, and then click **Show Promotions**

The **Promote Properties** dialog box opens.

How to Copy Data to the Message Context as Distinguished Fields

This topic provides step-by-step instructions for promoting a property as a **Distinguished Field**.

You might choose **Distinguished Field** promotion over **Property Field** promotion for the following reasons:

- **Property Field** values are limited to 255 characters in length.
- **Property Field** values are stored in a database, and therefore have more overhead than **Distinguished Fields**. **Distinguished Fields** do not require any additional storage; they are essentially an alias for an **XPath**, thereby allowing orchestrations to more easily access the relevant values directly from the message.

To promote a property as a Distinguished Field

1. In BizTalk Editor, open the schema for which you want to promote one or more properties, and then select the (first) **Field Element**, **Field Attribute**, or **Record** node that you want to promote as a **Distinguished Field**.
2. Right-click the selected node, click **Promote**, and then click **Show Promotions**.

The **Promote Properties** dialog box opens with the selected node showing as selected in the schema tree on the left side of the dialog box.

3. In the **Promote Properties** dialog box, select the **Distinguished Fields** tab.
4. With the node to be promoted still selected in the schema tree on the left side of the **Promote Properties** dialog box, click **Add**.

If allowed, the selected node is added to the list on the **Distinguished Fields** tab. If not allowed, a message box provides an explanation.

5. You can select additional nodes for promotion in the schema tree on the left side of the dialog box, clicking **Add** after each selection.
6. When complete, click **OK**.

The nodes that you selected to promote are now **Distinguished Fields**.

How to Copy Data to the Message Context as Property Fields

You can promote a property as a **Property Field** in much the same way as promoting a property as a **Distinguished Field**, and you can also use the **Quick Promotion** feature to streamline the process.

You might choose **Property Field** promotion over **Distinguished Field** promotion for the following reasons:

- The values you want to promote are shorter than the 255 character limitation that applies to **Property Fields**.
- You need the values that you promote to be accessible outside of orchestrations, such as within pipelines or ports.

This topic provides step-by-step instructions for promoting a property as a **Property Field** in both of these ways.

To promote a property as a **Property Field** using the **Promote Properties** dialog box

1. If necessary, create an appropriate property schema into which you will promote a property. For step-by-step instructions for creating property schemas, see [Creating Property Schemas](#).
2. In BizTalk Editor, open the schema for which you want to promote one or more properties, and then select the (first) **Field Element**, **Field Attribute**, or **Record** node that you want to promote as a **Property Field**.
3. Right-click the selected node, click **Promote**, and then click **Show Promotions**.

The **Promote Properties** dialog box opens with the selected node showing as selected in the schema tree on the left side of the dialog box.

4. In the **Promote Properties** dialog box, select the **Property Fields** tab.
5. Confirm that the property schema into which you want to promote a property is present in the **Property Schemas List** in the **Property Fields** tab. If it is present, skip to step 8.
6. In the **Property Schemas List** section, click the **Folder** icon. The **BizTalk Type Picker** dialog box appears.
7. In the **BizTalk Type Picker** dialog box, navigate to the appropriate property schema (that you may have created in step 1), select that schema, and then click **OK**.
8. With the node to be promoted still selected in the schema tree on the left side of the **Promote Properties** dialog box, click **Add**.

If allowed, the selected node is added to the end of the **Property-Fields dictionary** list on the **Property Fields** tab. If not allowed, a message box provides an explanation. If not allowed, the **Add** button is not enabled.

9. Double-click **Property** column cell for the row you just added to the **Property-Fields dictionary** list, and then in the drop-down list, select the **Property Schema** and corresponding **Field Element** node into which you want to promote the selected node. Drop-down list values have the form X:Y, where X is the namespace prefix of a property schema in the **Property Schemas List**, and Y is the node name of a **Field Element** node in that property schema.

The default value in the drop-down list is the first property schema (**Field Element**) node that has not yet been promoted, sorted alphabetically across all relevant property schemas. This will rarely be the property schema node into which you intend to promote a given schema node.

10. You can select additional nodes for promotion in the schema tree on the left side of the dialog box, clicking **Add** and then performing step 9 after each selection.
11. When complete, click **OK**.

The nodes that you selected to promote are now **Property Fields** and are associated with a particular **Field Element** node in a property schema.

To promote a property as a Property Field using the Quick Promotion command

1. In BizTalk Editor, open the schema for which you want to promote one or more properties, and then select the (first) **Field Element**, **Field Attribute**, or **Record** node that you want to promote as a **Property Field**.
2. Right-click the selected node, click **Promote**, and then click **Quick Promotion**.

If the default property schema, as defined by the **Default Property Schema Name** property on the **Property Pages** for the relevant schema, does not exist, you must click **OK** in the confirmation dialog to create the default property schema and configure it with an appropriate **Field Element** node to accommodate your property promotion.

Creating Schemas Using BizTalk Flat File Schema Wizard

The BizTalk Flat File Schema Wizard is designed to simplify the process of creating flat file schemas by providing the following:

- The ability to use flat file instances as input
- A visual design surface for working with delimited and positional flat file schemas
- An interactive wizard-based process for adding elements to the schema and defining flat file related annotations.

This section provides an overview of the BizTalk Flat File Schema Wizard, steps for launching the wizard, considerations for working with the wizard, and a walkthrough that shows you how to create flat file schemas from a document instance.

In This Section

- How to Use BizTalk Flat File Schema Wizard
- BizTalk Flat File Schema Wizard Walkthrough

How to Use BizTalk Flat File Schema Wizard

In previous release of BizTalk Server, you need to manually add the annotations to XSD schemas in BizTalk Schema Editor for these schemas to be understandable to the Flat File Pipeline components such as Flat File Disassembler and Flat File Assembler. You can still achieve this today using BizTalk Server 2006 Schema Editor. Yet, to reduce the number of manual steps and time to solutions, BizTalk Flat File Schema Wizard can help you to achieve that by providing the below functionalities:

- Using real flat file instances as input.
- Visual design surface for working with delimited and positional flat file schemas.
- Re-entrant wizard based process for adding elements to the schema and defining flat file related annotations interactively.
- Support for tag identifiers and character and hexadecimal delimiters.
- The wizard will automatically determine tag identifier offsets and child delimiting order.
- Preview capabilities for the file format.
- The ability to select preferred sub-data within the interchange instance to use for definition of the flat file schemas.

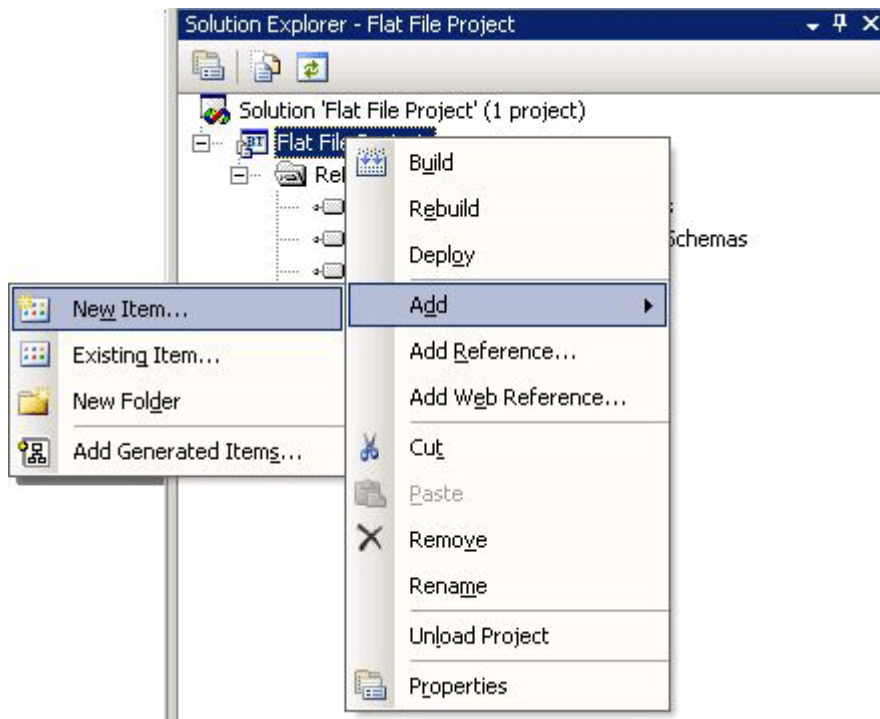
Launching the BizTalk Flat File Schema Wizard

The wizard can be launched in two ways:

- From Visual Studio Solution Explorer.
- From BizTalk Schema Editor.

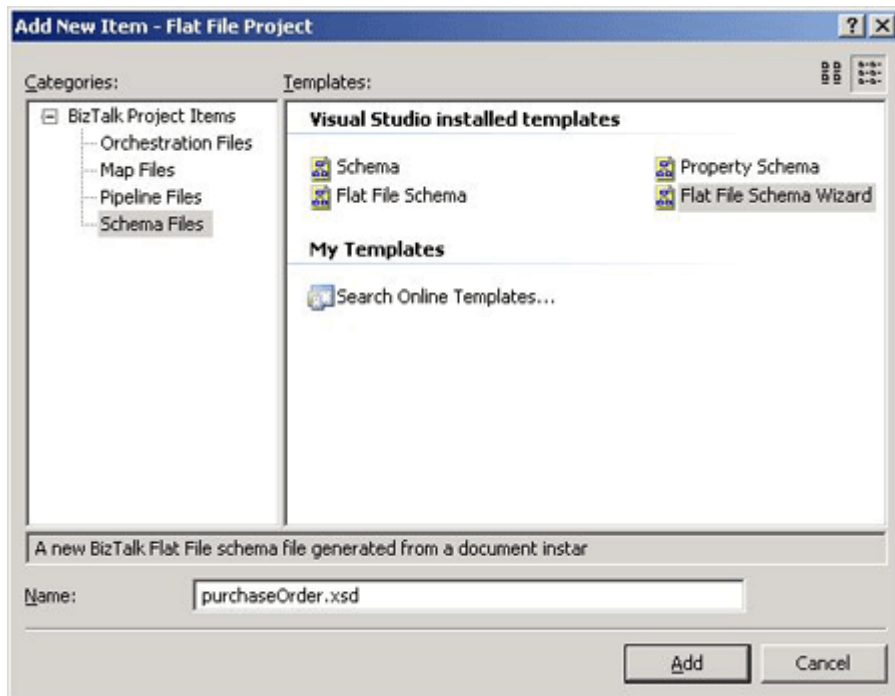
To launch the Wizard from Solution Explorer

1. In Visual Studio, open the Solution Explorer.
2. Right-click the project to add the new flat-file schema, select **Add**. Point to **New Item** and click on it.



3. In the **Add New Item** window, do the following:
 - a. In the **Categories** section, select **Schema Files**.
 - b. In the **Templates** section, select **Flat File Schema Wizard**.
 - c. In the **Name** field, enter a name for the new schema.
 - d. Click **Add**.

The BizTalk Flat File Schema Wizard now launches.



To launch the Wizard from Schema Editor

1. In Visual Studio, open the Solution Explorer.
2. Right-click the project to add the new flat-file schema, then point to **New Item**, then select **Add**.
3. In the **Add New Item** window, do the following:
 - a. In the **Categories** section, select **Schema Files**.
 - b. In the **Templates** section, select **Flat File Schema**.
 - c. In the **Name** field, enter a name for the new schema.
 - d. Click **Add**.
 - e. Right click on the **Root** and select **Define Record from Flat File Instance**.

The BizTalk Flat File Schema Wizard now launches.

Considerations for Working with the Wizard

This section describes the considerations when working with the BizTalk Flat File Schema Wizard.

Working with MBCS

By default, **Count positions in bytes** check box is unchecked. The wizard will count the positions by characters which mean that if you have MBCS characters in your positional flat file instance, the positions will not get counted correctly. By checking the **Count positions in bytes** check box, the wizard displays

MBCS characters with additional indication of their positional length. The character takes one position on the display and the remaining length is filled out with "•" symbol or symbols depending on the files were saved in DBCS, Unicode or UTF-8 format.

Working with Code Page

The following is a list of supported code page for the wizard:

- Arabic (1256)
- ASCII (20127)
- Baltic (1257)
- Big-Endian-UTF16 (1201)
- Central-European (1250)
- Cyrillic (1251)
- Greek (1253)
- Hebrew (1255)
- Japanese-Shift-JIS (932)
- Korean (949)
- Little-Endian-UTF16 (1200)
- Simplified-Chinese-GBK (936)
- Simplified-Chinese-GB18030 (54936)
- Thai (874)
- Traditional-Chinese-BIG5 (950)
- Turkish (1254)
- UTF-7 (65000)
- UTF-8 (65001)
- Vietnamese (1258)
- Western-European (1252)

Working with Record Type

The structure of flat file does not allow to have delimited records within positional records. Since the wizard is re-entrant based, the radio buttons that define the type of the **By delimiter symbol** and **By relative positions** will be enabled or disabled based on the format of the parent records for the child which you will be going to define in the wizard. For example,

- If the wizard is run for a child of a delimited record, both radio buttons are enabled.
- If the wizard is run for a child of a positional record, **By delimiter symbol** radio button is disabled.

Working with Delimiter Symbols

The Child delimiter property drop down selection list is populated with the following common seen delimiters:

- {CR}{LF}
- {CR}
- {LF}
- {TAB}
- {SPACE}
- {0x1A}
- |
- ,
- ;

The default value for this property is {CR}{LF}. The property is also an editable box which means that you can specify the child delimiter as sequence of characters or as hexadecimal values of the characters. An example of using characters for child delimiter would be "a" or "street". Hexadecimal delimiters are specified using the following format:

- {0xn timer}. For example, {0x0D}{0x0A}, {0x09} or {0x20}.
- Example of using sequence of hexadecimal values, {0x0D}{0x0A}, {0x09}{0x20}.

If using \, { or } as the delimiter, you must place an additional backslash symbol in front of the delimiter otherwise you will receive an error message from the wizard. For example,

- \\, \{ or \}.

Working with Relative Positions

Setting Position Markers

Use right or left mouse button to click on empty space to set the new position marker. The position marker will be represented as a solid line. By default, there will be a position marker at the beginning of the record at position zero. To remove an existing position marker, use right or left mouse button to click on it and the line will be removed.

Working with Escape Characters

An escape character is a single character that suppresses any special meaning of the character that follows it. See [Escape Characters](#) for more details. Escape character property in the wizard allows you to specify which escape character to use when parsing the flat file instance. The default is empty meaning that the default escape character defined at the schema level will be used.

The escape character can be specified as a character or as a hexadecimal value. The hexadecimal value is specified using the following format:

- `{0xn timer }`. For example, `{0x0D}{0x0A}`, `{0x09}` or `{0x20}`.

If using `\`, `{` or `}` as the escape character, you must place an additional backslash symbol in front of the delimiter otherwise you will receive an error message from the wizard. For example,

- `\\, \{, \}`.

Working with Child Elements

Each child element contains **Element Name**, **Element Type**, **Data Type** and **Content**. The **Element Name** is an editable box for you to input a meaningful name for the node. The **Element Type** is a drop down selection box with the following value:

- **Field element**: Specifies that this node will be created in the schema as element.
- **Field attribute**: Specifies that this node will be created in the schema as attribute.
- **Record**: Specifies that a record without children will be created in the schema.
- **Repeating record**: Specifies that a record without children will be created in the schema and its max occurrence is set to **Unbounded**.
- **Ignore**: Nothing will be created in the schema for this node.

By default all elements are of type **Field element** and their data type is **string**.

- The **Field attribute** is defined after **Field element**, **Record** or **Repeating record**.
- The child element does not have a name.
- The child element has a duplicate name.

- The selected data type for the child element is not suitable for the content

BizTalk Flat File Schema Wizard Walkthrough

The following walkthrough shows you how to create flat file schema from document instance using the BizTalk Flat File Schema Wizard.

To create flat file schema from interchange instance, you will first create the document schema by following the steps described in the below walkthrough with the exception of that selecting in the part which describes the individual document structure in the wizard. Then, you will proceed to define the envelope schema. For more information about defining an envelope schema and configuring pipeline component for using with document schema and envelope schema, see Envelope Schemas and How to Configure the XML Assembler Pipeline Component .

In This Section

- Walkthrough: Creating a Flat File Schema From a Document Instance

Walkthrough: Creating a Flat File Schema From a Document Instance

This walkthrough shows you how to create a flat file schema from a document instance using the BizTalk Flat File Schema Wizard based on the below sample purchase order. For an introduction to the BizTalk Flat File Schema Wizard, see How to Use BizTalk Flat File Schema Wizard.

```
PO1999-10-20␣  
US      Alice Smith      123 Maple Street  Mill Valley  CA 90952␣  
US      Robert Smith     8 Oak Avenue    Old Town    PA 95819␣  
ITEMS,ITEM872-AA|Lawnmower|1|148.95|Confirm this is electric,ITEM926-AA|Baby  
Monitor|1|39.98|Confirm this is electric␣
```

Each line of the purchase order ends with a carriage return line feed (CRLF) which marked in green color. The purchase order starts with a PO tag marked in red color and followed by a date. Two repeating fixed positional fields for customer information which marked in purple color. After the comment field, there is a repeating field starting with ITEMS tag which contains multiple records delimited by commas (,) and data values separated by a pipes (|) which marked in blue color.

Prerequisites

Before exercising this walkthrough, make sure the following software is installed and configured on your server:

- Visual Studio 2005
- BizTalk Server 2006 with the following components installed
 - Developer tools

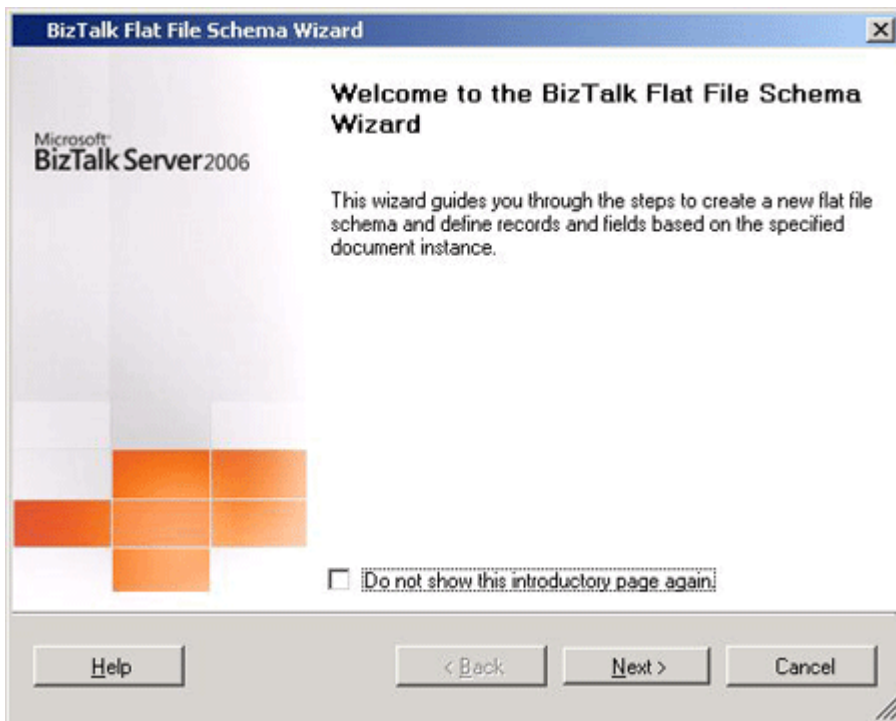
Prepare the document instance for the walkthrough by copying the below to **Notepad** and save it as a text file.

To launch the Wizard

1. In Visual Studio, open the Solution Explorer.
2. Right-click the project to add the new flat-file schema, select **Add**. Point to **New Item** and then click on it.
3. In the **Add New Item** window, do the following:
 - a. In the **Categories** section, select **Schema Files**.
 - b. In the **Templates** section, select **Flat File Schema Wizard**.
 - c. In the **Name** field, enter **PurchaseOrder.xsd** for the new schema.
 - d. Click **Add**.

To begin using the wizard

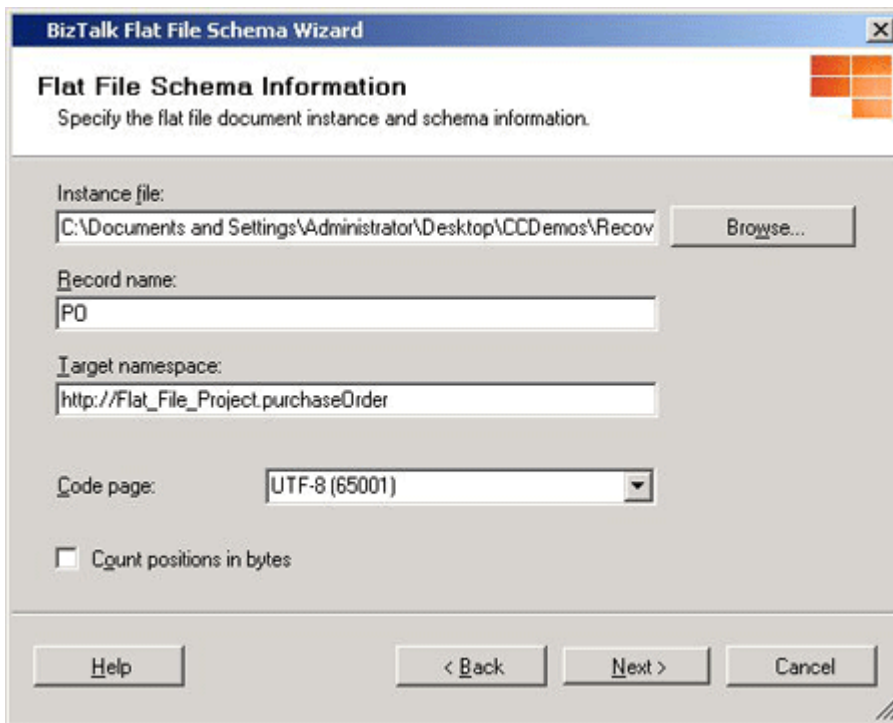
- When the BizTalk Flat File Schema Wizard is launched, the welcome screen appears. If you want to skip this screen when running the wizard in the future, check the **Do not show this introduction page again** box. Click **Next** to continue.



Selecting Document Instance

To select purchase order instance

- On the **Flat File Schema Information** screen, select your instance and enter the following information. When you are done, click **Next** to continue.
 - Instance file:** Click the **Browse** button to locate the flat file from which the schema will be generated. Browse to the folder contains the text file you prepare in the Prerequisites section.
 - Record name:** Type **PO** as it will be the schema root name.
 - Target namespace:** Type `http://Flat_File_Project.PurchaseOrder` for schema target namespace.
 - Code page:** Select **UTF-8 (65001)** from the drop down selection list.
 - Count positions in bytes:** Check this box if you wish to count the positional data fields by bytes. By default, the positional data fields are counted in characters. In this walkthrough, leave the **Count positions in bytes** check box unchecked.

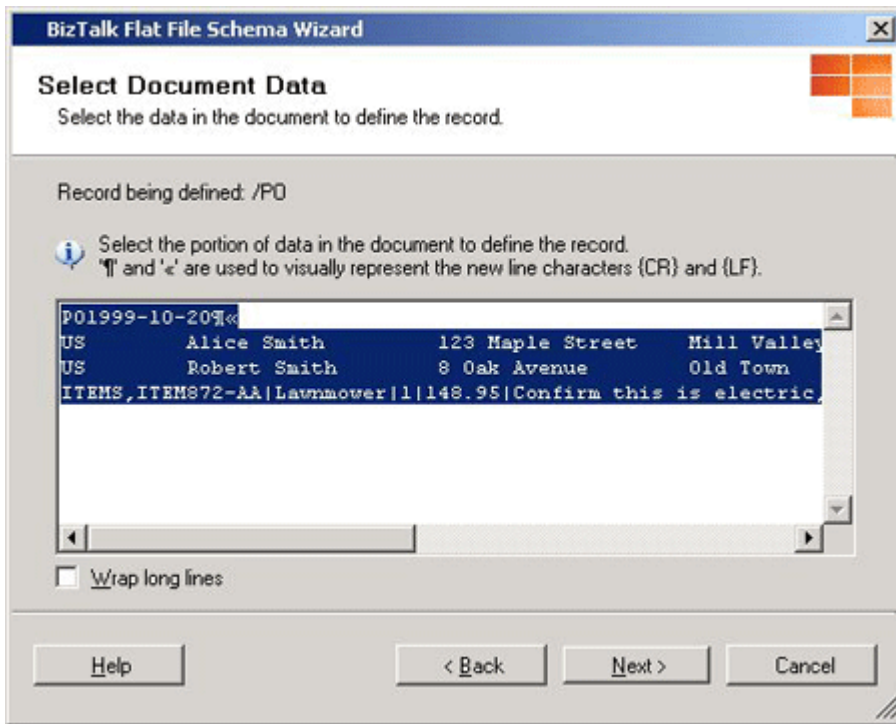


The screenshot shows the 'BizTalk Flat File Schema Wizard' window, specifically the 'Flat File Schema Information' tab. The window has a title bar with the text 'BizTalk Flat File Schema Wizard' and a close button. Below the title bar is a subtitle 'Flat File Schema Information' and a description 'Specify the flat file document instance and schema information.' The main area contains several input fields and a checkbox. The 'Instance file:' field has a text box with the path 'C:\Documents and Settings\Administrator\Desktop\CCDemos\Recov' and a 'Browse...' button to its right. The 'Record name:' field has a text box with the value 'PO'. The 'Target namespace:' field has a text box with the value 'http://Flat_File_Project.purchaseOrder'. The 'Code page:' field has a dropdown menu with 'UTF-8 (65001)' selected. Below these fields is a checkbox labeled 'Count positions in bytes' which is currently unchecked. At the bottom of the window are four buttons: 'Help', '< Back', 'Next >', and 'Cancel'.

Selecting Document Instance Data

To select purchase order data

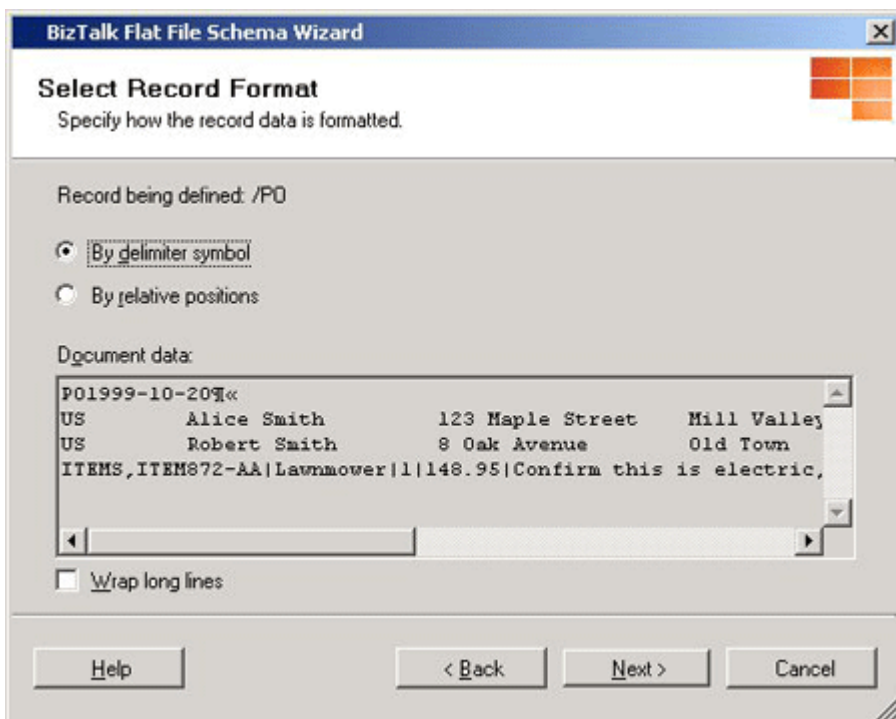
- On the **Select Document Data** screen, the contents of the flat file are displayed. Select the data needed for creating the schema, and then click **Next**.



Delimiting Record Data

To delimit purchase order record

- As each line of the purchase order ends with a carriage return line feed (CRLF), select **By delimiter symbol** and then click **Next** on **Select Record Format** screen.



Specifying Record Data Property

To specify purchase order record property

- On the **Delimited Record** screen, enter the following to define the first level of the schema and when you are done, click **Next**.
 - Child delimiter:** Select **{CR}{LF}**.
 - Escape character:** An escape character is a single character that suppresses any special meaning of the character that follows it. See **Escape Characters** for more information. Leave it blank for the walkthrough.
 - Check **Record has a tag identifier** box and type **PO** in **Tag**. In a multiple records file, **PO** will be used to identify each individual record. Click **Next** to continue.

BizTalk Flat File Schema Wizard

Delimited Record
Specify properties of the delimited record.

Record being defined: /PO

Child delimiter: **{CR}{LF}** Escape character:

Tag identifier
☒ Record has a tag identifier Tag: **PO**

Document data:

PO1999-10-20	US	Alice Smith	123 Maple Street	Mill Valley
	US	Robert Smith	8 Oak Avenue	Old Town

☐ Wrap long lines

Help < Back Next > Cancel

Defining Element Property in Record Data

To define element in purchase order record

- The wizard has identified four elements in the purchase order record and it is time to define the element property. On the first row, do the following:
 - Enter **date** for **Element Name**.
 - Select **Field element** for **Element Type**.
 - Select **date** from drop down selection list for **Data Type**.

2. Repeat Step a to Step c for the following elements. When you are done, click **Next**.

Element Name	Element Type	Data Type
customer	Repeating record	
	Ignore	
items	Record	

BizTalk Flat File Schema Wizard

Child Elements
Specify properties of the child elements.

Record being defined: /PO

Child nodes:

Element Name	Element Type	Data Type	Contents
date	Field element	date	1999-10-20
customer	Repeating record		US Alice Smith
	Ignore		US Robert Smith
items	Record		ITEMS,ITEM872-AA Law

Help < Back Next > Cancel

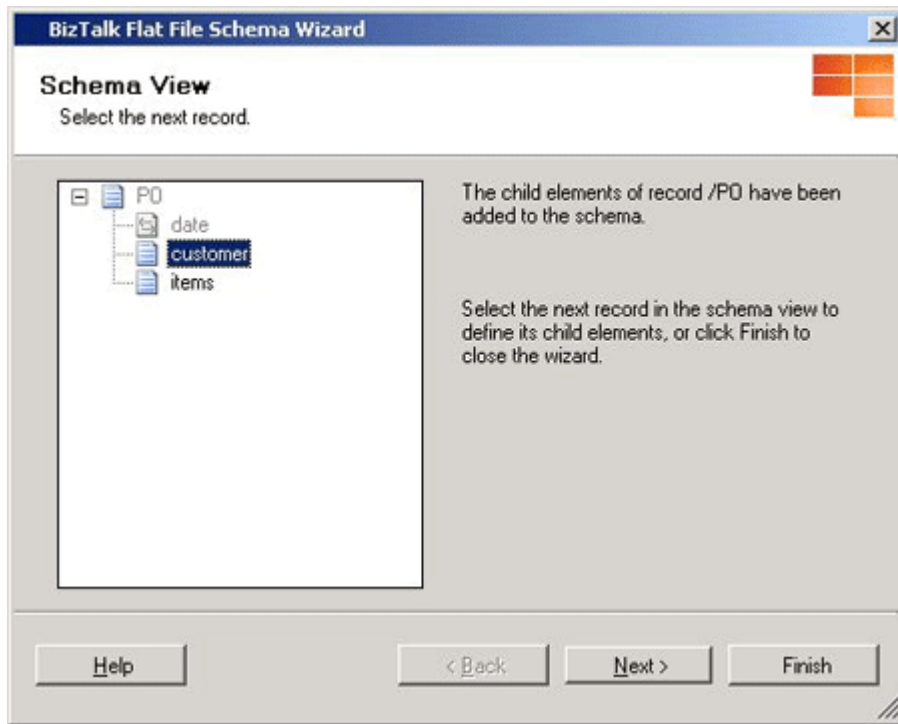
3.

Note After you click on **Next** on the **Child Elements** screen, you will not be able to click **Back** to redefine or make any changes to the child elements. You may need to close the wizard and launch the wizard again to define the flat file schema.

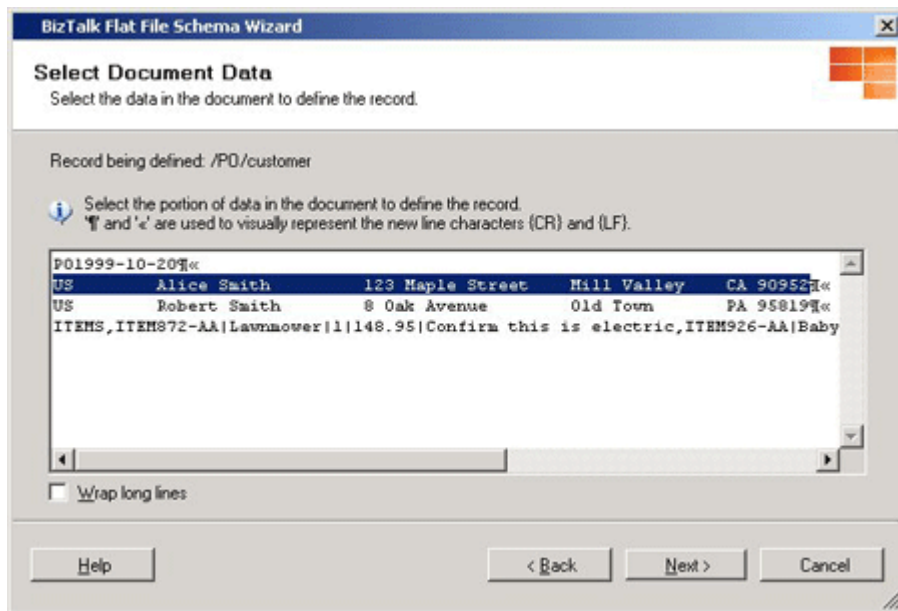
Defining Element Type of Record

To define customer record

1. Because we defined the customer element as **Repeating record** and items element as **Record**, the BizTalk Flat File Schema Wizard now continues to further define these two elements. On the **Schema View** screen, select customer and click **Next** to continue.



2. In order to work with the customer record, we need to select the data that represents that element. Since it is a repeating record, either of the lines will help to define the record. Select the first line of customer data and click **Next** to continue.

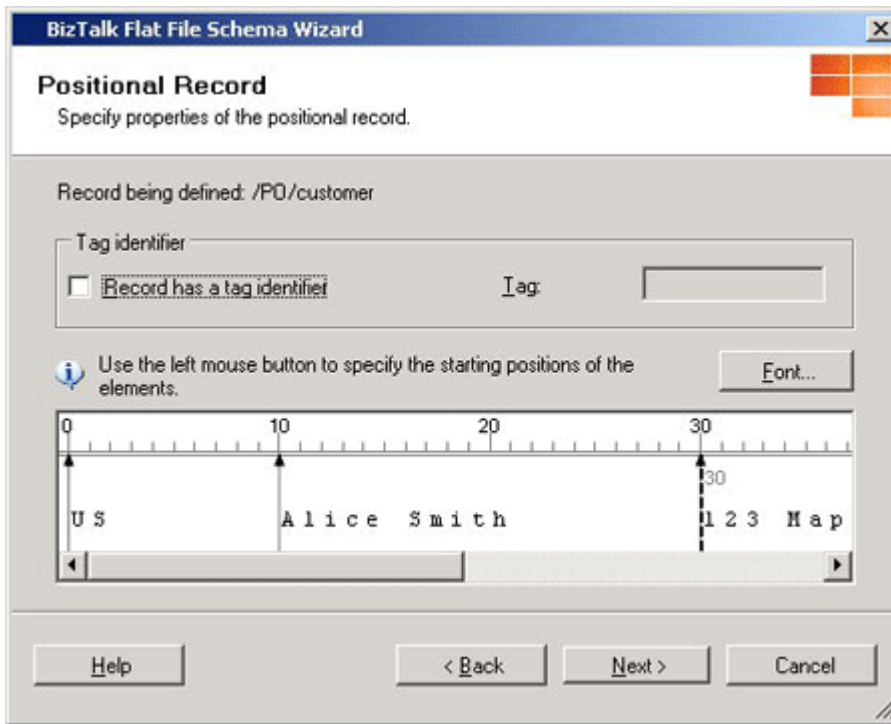


3. On the **Select Record Format** screen, select **By relative positions**, and then click **Next**.

4. The wizard provides a visual tool for showing and calculating the distance between the fields. On the **Positional Record** screen, use the mouse to click on 10 to represent where the name field begins. Click on the following points to represents the rest of data fields:

Field Name	Position Marker
street	30
city	50
state	65
postalcode	68

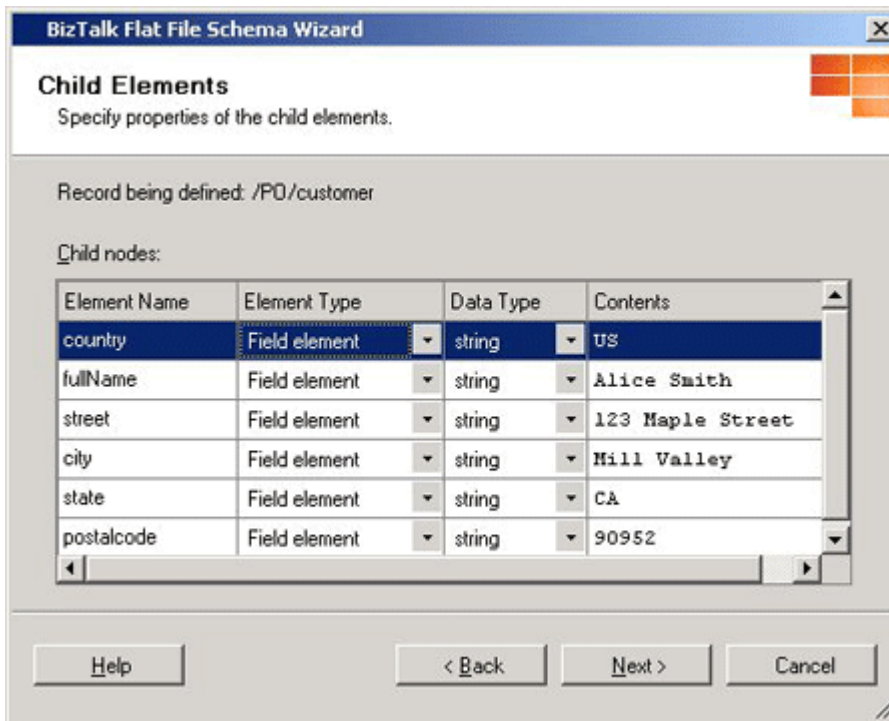
5. Click **Next** to continue.



- 6.
7. On the **Child Elements** screen, you will specify the properties of the child elements. Select the first row and enter **country** as **Element Name**. Leave the rest of columns as default. Change the rest of the **Element Name** as following:

Element Name	Change To
customer_Child2	fullName
customer_Child3	street
customer_Child4	city
customer_Child5	state
customer_Child6	postalcode

8. Click **Next** to continue.



BizTalk Flat File Schema Wizard

Child Elements
Specify properties of the child elements.

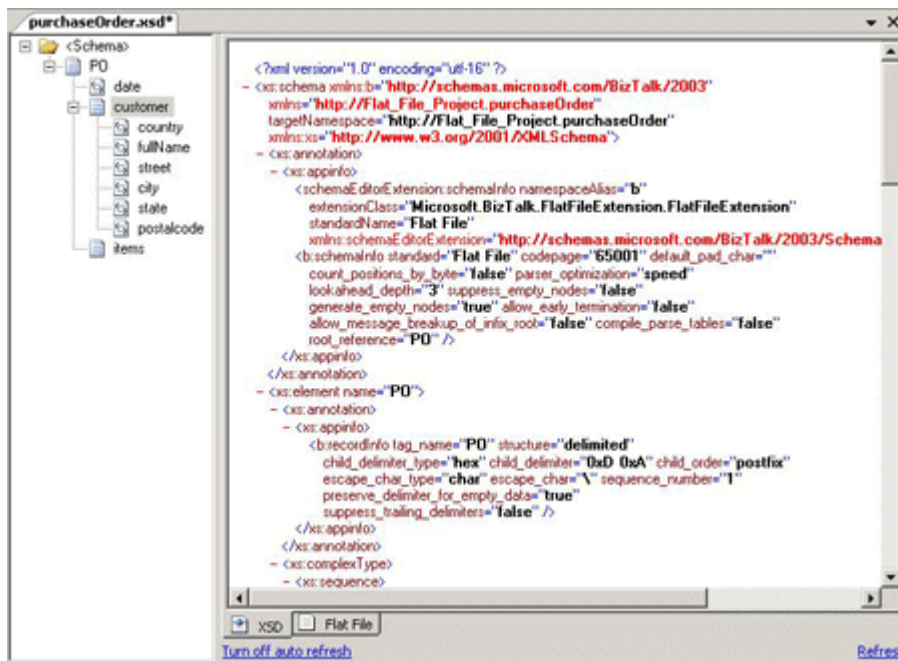
Record being defined: /PO/customer

Child nodes:

Element Name	Element Type	Data Type	Contents
country	Field element	string	US
fullName	Field element	string	Alice Smith
street	Field element	string	123 Maple Street
city	Field element	string	Hill Valley
state	Field element	string	CA
postalcode	Field element	string	90952

Buttons: Help, < Back, Next >, Cancel

- 9.
10. The child elements of customer record are created as shown in the following screenshot. Click **Next** to define the child elements for items record.



purchaseOrder.xsd*

Schema View showing the XML Schema Definition (XSD) for the purchaseOrder file. The schema is defined in the main pane, and the left pane shows the tree structure of the schema elements.

Left pane tree structure:

- PO
 - date
 - customer
 - country
 - fullName
 - street
 - city
 - state
 - postalcode
 - items

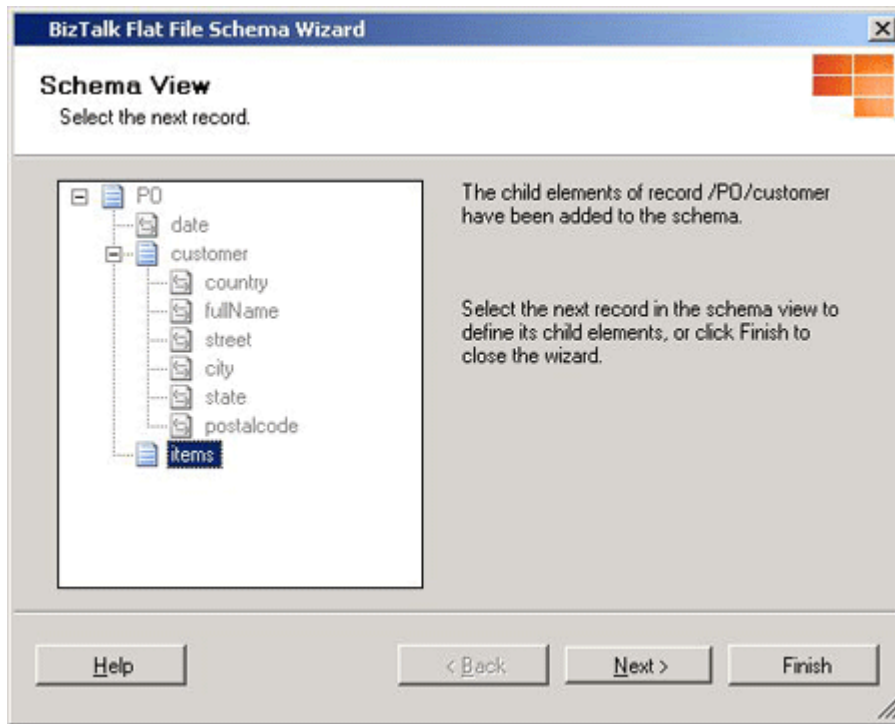
Main pane XSD content:

```
<?xml version="1.0" encoding="utf-16" ?>
<xs:schema xmlns:b="http://schemas.microsoft.com/BizTalk/2003"
  xmlns="http://Flat_File_Project.purchaseOrder"
  targetNamespace="http://Flat_File_Project.purchaseOrder"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  <xs:annotation>
    <xs:appinfo>
      <schemaEditorExtension:schemaInfo namespaceAlias="b"
        extensionClass="Microsoft.BizTalk.FlatFileExtension.FlatFileExtension"
        standardName="Flat File"
        xmlns:schemaEditorExtension="http://schemas.microsoft.com/BizTalk/2003/Schema"
        <b:schemaInfo standard="Flat File" codepage="65001" default_pad_char=" "
          count_positions_by_byte="false" parser_optimization="speed"
          lookahead_depth="3" suppress_empty_nodes="false"
          generate_empty_nodes="true" allow_early_termination="false"
          allow_message_breakup_of_infix_root="false" compile_parse_tables="false"
          root_reference="PO" />
      </xs:appinfo>
    </xs:annotation>
    <xs:element name="PO">
      <xs:annotation>
        <xs:appinfo>
          <b:recordInfo tag_name="PO" structure="delimited"
            child_delimiter_type="hex" child_delimiter="0xD 0xA" child_order="postfix"
            escape_char_type="char" escape_char="\\" sequence_number="1"
            preserve_delimiter_for_empty_data="true"
            suppress_trailing_delimiters="false" />
          </xs:appinfo>
        </xs:annotation>
        <xs:complexType>
          <xs:sequence>
```

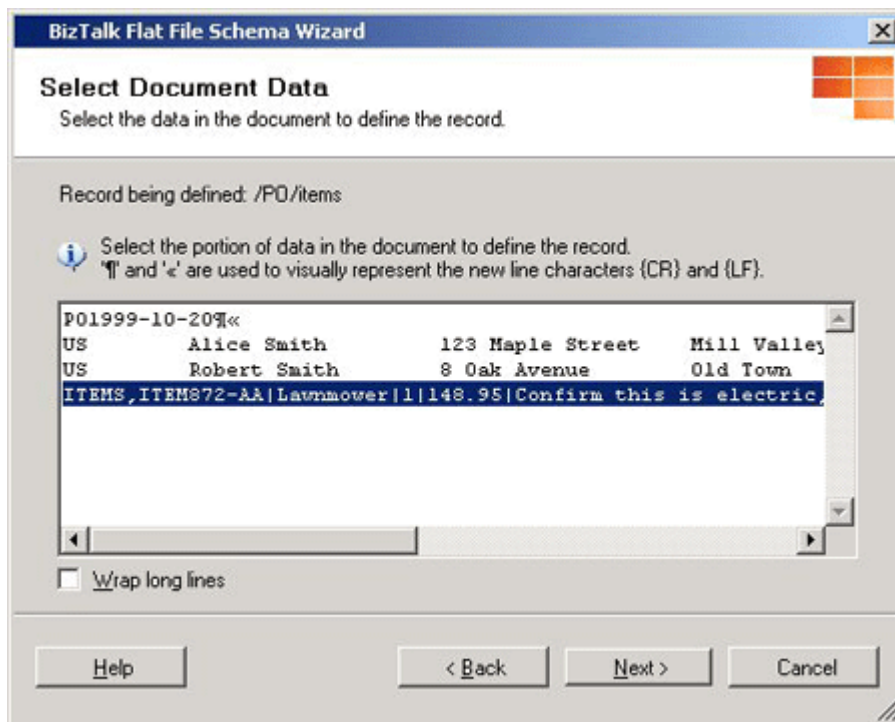
Buttons: XSD, Flat File, Turn off auto refresh, Refresh

To define items record

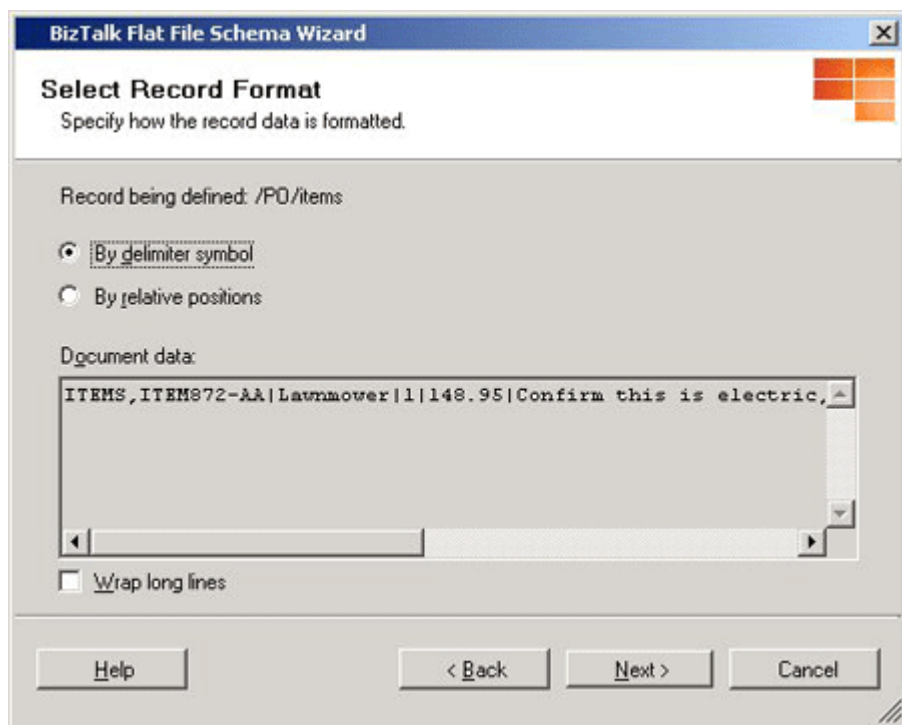
- On the **Schema View** screen, select items and clicking **Next**.



2. On the **Select Document Data** screen, select the ITEMS line and click on **Next** to start to define its child elements.



3. Select **By delimiter symbol** on the **Select Record Format** screen as the items record is delimited by commas (,) and pipes (|).



4. In the items record, comma is used to delimit the individual items. Therefore, on the **Delimited Record** screen, enter the following to define the items record. When you are done, click **Next**.
- **Child delimiter:** Select ,.
 - **Escape character:** Leave it blank.
 - Check **Record has a tag identifier** box and type **ITEMS** in **Tag**. In a multiple items record, **ITEMS** will be used to identify each individual record.

BizTalk Flat File Schema Wizard

Delimited Record
Specify properties of the delimited record.

Record being defined: /PO/items

Child delimiter: Escape character:

Tag identifier
☒ Record has a tag identifier Tag:

Document data:

☐ Wrap long lines

Help < Back Next > Cancel

5. Using the values from the **Delimited Record** screen, the wizard identifies 2 child elements. As one of them is a repeating record, select the first element and enter item for Element Name and select Repeating Record from the drop down selection list for **Element Type**. Leave the rest of the columns as default. Select the second row and select the **Ignore** from **Element Type** list. By clicking **Next**, the next level for the items record will be created in the schema. You will continue to define the final part of the purchase order schema.

BizTalk Flat File Schema Wizard

Child Elements
Specify properties of the child elements.

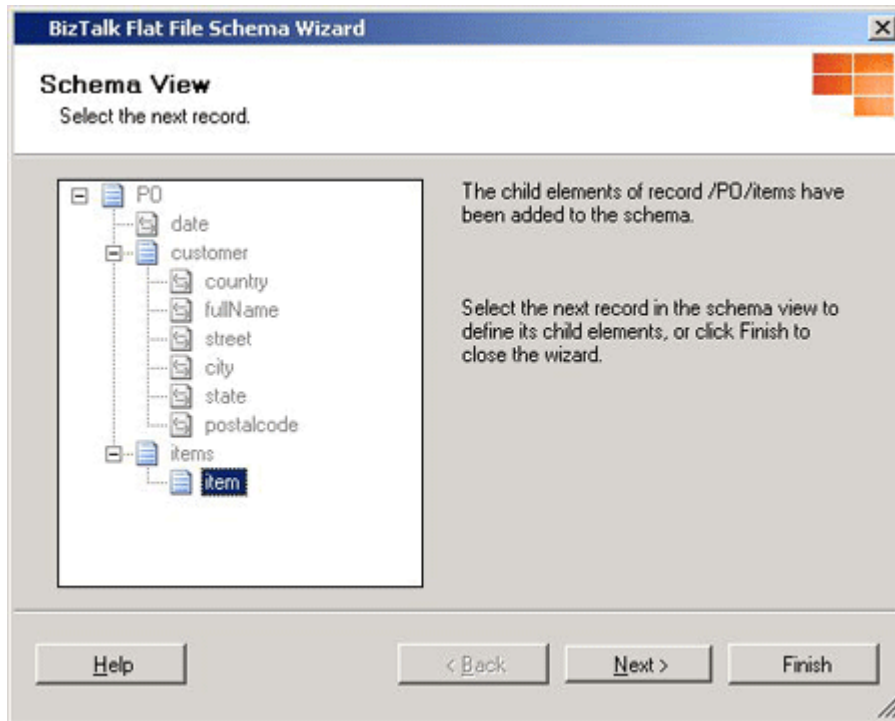
Record being defined: /PO/items

Child nodes:

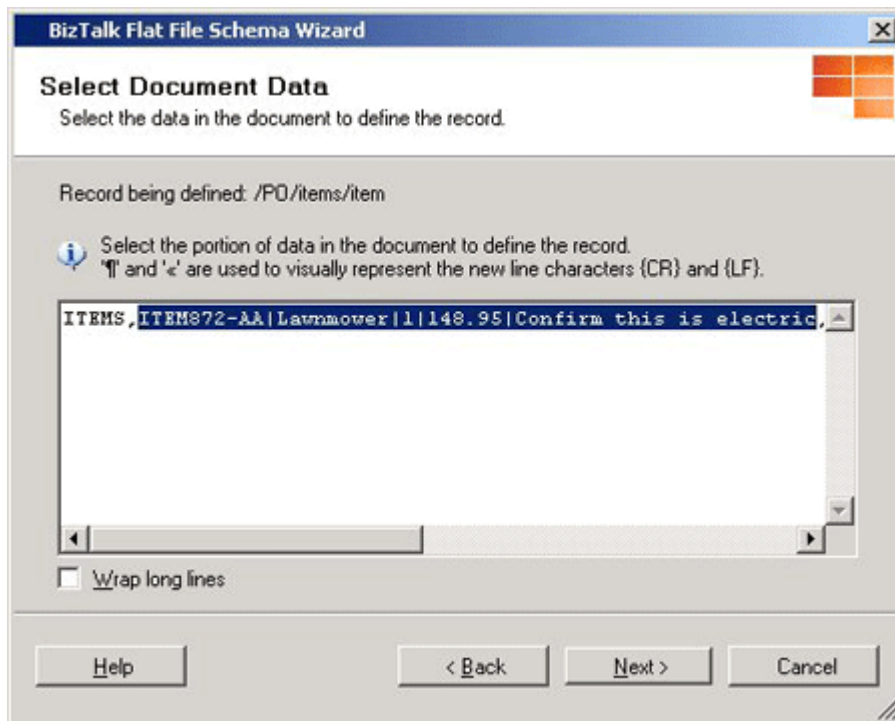
Element Name	Element Type	Data Type	Contents
item	Repeating record		ITEM872-AA Lawnmower
	Ignore		ITEM926-AA Baby Monit

Help < Back Next > Cancel

6. Select item and click **Next** to continue on the **Schema View** screen.



7. On the **Select Document Data** screen, select the data right of the ITEMS tag and left of the second commas in the line. Click on **Next** to continue.



8. Select **By delimiter symbol** option on the **Select Record Format** screen as the individual item is delimited by a pipe (|).

9. As the individual elements in the item record are delimited by a pipe (|). On the **Delimited Record** screen, enter the following to define the item record. When you are done, click **Next**.

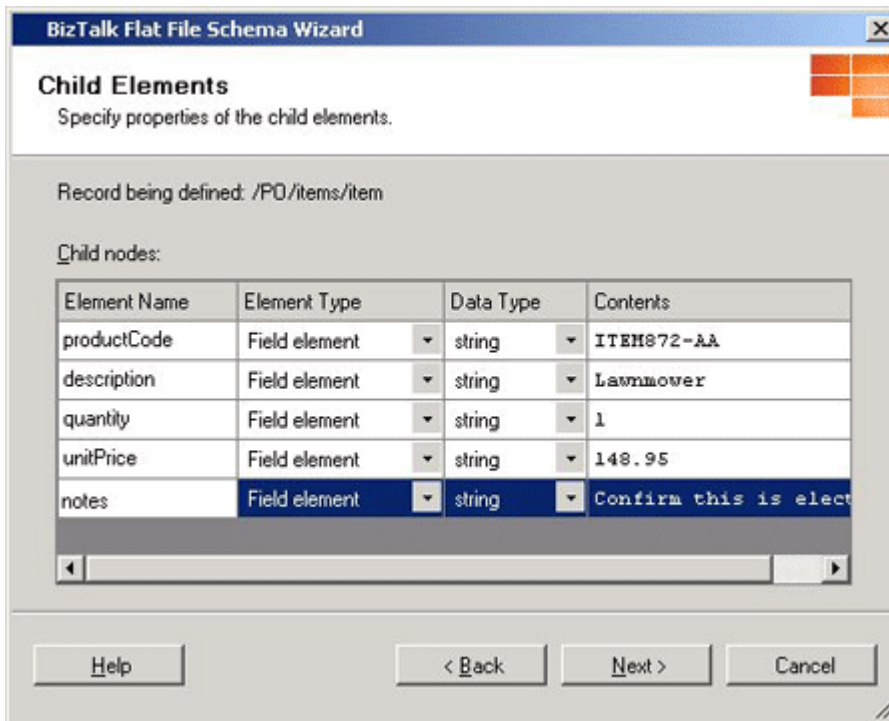
- **Child delimiter:** Select |.
- **Escape character:** Leave it blank.

and

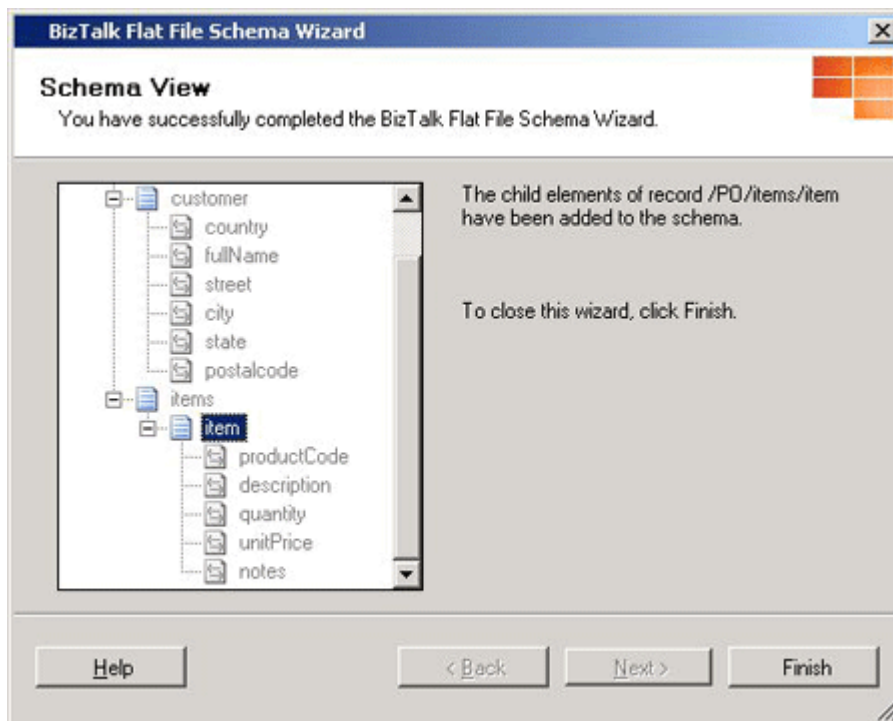
10. On the **Child Elements** screen, the wizard has identified 5 child elements. Select the first row and enter **productCode** for **Element name**. Leave the rest of the columns as default. For the rest of the **Element Name**, enter the following:

Element Name	Change To
item_Child2	description
item_Child3	quantity
item_Child4	unitPrice
item_Child5	notes

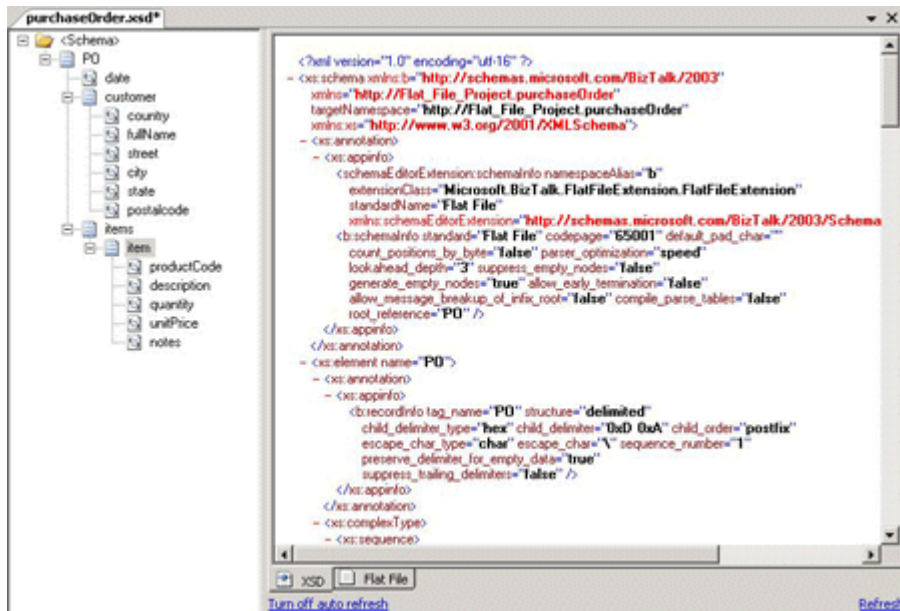
11. Click **Next** to continue.



- 12.
13. At this step, you have defined all the nodes for the purchase order schema. Click **Finish** on the **Schema View** screen.



14. In the following screenshot, you can view the final purchase order schema. You may also refine the schema by using the BizTalk Schema Editor. See Flat file property name and Property tables in **Schema Node Properties**.



Summary

In this walkthrough you have seen how to create a flat file schema from a document instance using the BizTalk Flat File Schema Wizard.

Next Steps

Validating PO Instance

To ensure that the PurchaseOrder.xsd schema can correctly parse the PO instance, do the following:

1. Right click on the **PurchaseOrder.xsd** in Solution Explorer and then select **Properties**.
2. In the **Property Pages**, make sure that the **Input Instance Filename** is pointing to the location of the PO instance you created in Prerequisites section. If not, browse to select the PO instance by clicking on the ellipse button. Click on OK.
3. Right click on **PurchaseOrder.xsd** in Solution Explorer and select **Validate Instance**. The validation component will be invoked.
4. After the validation is completed, a link will be provided to you for viewing the XML output based on parsing result on PO instance using PurchaseOrder.xsd schema. To view the XML output, press Ctrl and click on the link.

Creating Pipelines for the Schema

Now you can create a receive or send pipeline based on the PurchaseOrder.xsd schema to use it with your BizTalk application. To create a new pipeline, see [How to Create a New Pipeline](#) . To configure the Flat File Pipeline components, see [How to Configure the Flat File Assembler Pipeline Component](#) and [How to Configure the Flat File Disassembler Pipeline Component](#) .

Testing Schemas

After you have created your schema, you may want to validate that it describes the XML structure you intend it to describe. You can perform the following three operations on your schema to validate it:

- **Instance generation.** This operation generates an instance message from a schema, creating test data to accompany the XML elements and attributes specified by the schema.
- **Schema validation.** This operation validates the internal consistency of a schema, assuring that it conforms to the XML Schema definition (XSD) language schema standard.
- **Instance validation.** This operation validates a given instance message against a schema.

All three of these operations are useful for testing your schemas before putting them into use in a production environment.

This section describes these operations in greater detail.

In This Section

- Validating Schemas in Visual Studio
- Validating Instance Messages
- Generating Instance Messages

How to Validate Schemas in Visual Studio

After you have constructed a schema, you can check your work by validating the schema itself in isolation.

This topic provides step-by-step instructions for this validation task.

To validate a schema in isolation

1. In **Solution Explorer**, right-click the schema that you want to validate, and then click **Validate Schema**.
2. In the Output window, view the results. Success and error messages are displayed in this window.

How to Validate Instance Messages

After you have constructed a schema, you can check your work by validating a pre-existing instance message that you know is a good representation of such instance messages against the schema.

This topic provides step-by-step instructions for this validation task.  To validate an instance message against a schema

1. In Solution Explorer, right-click the schema, and then click **Properties**.

The **<Schema file name.xsd> Property Pages** dialog box opens.

2. If necessary, in the **<Schema file name.xsd> Property Pages** dialog box, expand the **General** section of the **General** tab by clicking its plus (+) icon.
3. In the **Input Instance Filename** property value field, either type the name of a file or use the ellipsis (...) button at the right end of the value field to browse for a file that contains the instance message to be validated against the schema, and then click **OK**.
4. In **Solution Explorer**, right-click the schema name, and then click **Validate Instance**.
5. In the Output window, view the results. Success and error messages are displayed in this window.

How to Generate Instance Messages

After you have constructed a schema, one way to check your work is to generate a sample instance message from the schema. In many ways, looking at an instance message is much more straightforward than looking at either the schema tree or the XML Schema definition (XSD) language representation of the schema. This is because the schema needs to describe all of the possible variations of the corresponding instance messages, and a specific instance message just needs to convey some data by using the format specified by the schema. The generated instance message is a sample and may not show all of the structures defined by the corresponding schema.

To explicitly specify a file to contain the generated instance message

1. In Solution Explorer, right-click the schema for which you want to generate an instance message, and then click **Properties**.

The **<Schema file name.xsd> Property Pages** dialog box opens.

2. If necessary, in the **<Schema file name.xsd> Property Pages** dialog box, expand the **General** section of the **General** tab by clicking its plus (+) icon.
3. In the **Output Instance Filename** property value field, either type the name of a file or use the ellipsis (...) button at the right end of the value field to browse for a file into which generated instance messages will be placed, and then click **OK**.

To specify the type of the generated instance message

1. In Solution Explorer, right-click the schema for which you want to generate an instance message, and then click **Properties**.

The **<Schema file name.xsd> Property Pages** dialog box opens.

2. If necessary, in the **<Schema file name.xsd> Property Pages** dialog box, expand the **General** section of the **General** tab by clicking its plus (+) icon.
3. In the **Create Output Instance Type** property value field, use the drop-down list to select either **XML** or **Native** as the type of the instance message to be generated.

XML is the default value.

To generate a sample instance message for a schema

1. In Solution Explorer, right-click the schema for which you want to generate an instance message, and then click **Generate Instance**.
2. In the Output window, view the results. Success and error messages are displayed in this window.

Extending BizTalk Editor

BizTalk® Editor is designed to allow extensions that support alternative instance message formats. In fact, the XML format is the only format that is built into BizTalk Editor. Even support for flat file formats, which is included in Microsoft® BizTalk® Server 2006, is implemented as a BizTalk Editor extension, thereby serving as a good example of the type of functionality that can be added by such extensions.

In general, BizTalk Editor extensions persist their custom data as XML Schema definition (XSD) language annotations associated with the XSD elements that correspond to the nodes in the schema tree. Again, the extensive set of annotations added by the Flat File Extension to BizTalk Editor serves as a good example of the way in which BizTalk Editor extensions can persist their custom data in the schema.

BizTalk Editor extensions are .NET assemblies that extend the functionality of BizTalk Editor. To be identified as an extension, an assembly must have one class that implements the **IExtension** interface, and must be located under the Developer Tools\Schema Editor Extensions folder in the product installation directory.


The developer of an extension must have its assembly reference the Microsoft.BizTalk.SchemaEditor.Extensibility.dll, which contains the definition of all the interfaces needed for exposing extended functionality to BizTalk Editor. Those interfaces are defined under the **Microsoft.BizTalk.SchemaEditor.Extensibility** namespace.

The **IExtension** interface is the entry point for the extension, from which BizTalk Editor accesses the extended functionality, such as property managers, custom views, schema validation, native instance generation, and native instance validation.

A given schema can have multiple extensions associated with it, but only one can be set as the standard at a given time; this is set in the **Standard** property of the **Schema** node. The extension currently set as the standard is the one used for native instance generation and validation, and for schema validation.

Extensions can be associated with a given schema by editing the **Schema Editor Extensions** property of the **Schema** node. The information about the extensions associated with a schema is stored in the schema itself, within the **annotation** element of the **schema** element, as illustrated in the following XSD fragment.

After instantiating the extension object, the framework invokes the **Initialize** method of the **IExtension** interface, passing an **ITree** object so that the extension can access information about the schema tree. For example, the extension could traverse all child nodes by accessing the **ITree.RootNode** property.

This section describes the ways in which a BizTalk Editor extension can integrate into the BizTalk Editor environment and hook itself into existing BizTalk Editor commands.  In This Section

- Property Managers
- Custom Views
- Schema Validation
- Instance Validation
- Instance Generation

Property Managers

Property Managers allow an extension to add custom properties (generally as XSD annotations) to elements and attributes in the XSD representation of the schema, as well as extending the Properties window to include the custom properties associated with the extension.

A Property Manager is an object that implements the **IPropertyManager** interface, a reference to which is obtained by calling **IExtension.GetPropertyManager**, and passing an **ITreeNode** object as the input parameter. Typically the extension provides one **IPropertyManager** object for each **ITreeNode** object. The Property Manager is responsible for the collection of custom properties for that **ITreeNode** object.

A custom property is represented by a **System.ComponentModel.PropertyDescriptor** object, which can be obtained from the collection returned by the **IPropertyManager.GetProperties** method.

Using **PropertyDescriptor** objects to represent the custom properties associated with the extension facilitates integration with the Microsoft® Visual Studio® .NET Properties window. When **PropertyDescriptor** objects are used, it is easy for BizTalk Editor to integrate the custom properties of the extension into the set of standard node properties already being integrated into the Properties window. Custom property information such as the display name, the display value, the type of property control, the property description, and the property category is obtained from the **PropertyDescriptor** object.

Custom properties are stored in the XSD representation of the schema as attributes of an element within the annotation element within the element corresponding to the relevant node in the schema tree. Each custom property of a schema tree node can be an attribute of a common element, or alternatively, each can have its own associated element.

Custom Views

A custom view is typically a read-only window control object (derived from **System.Windows.Forms.Control**), and is provided by an extension to represent the schema in a display format customized for the type of file or files supported by the BizTalk Editor extension. An extension can implement multiple custom views, though it need not have any custom view.

A custom view is displayed as an additional view in the same BizTalk Editor pane as the XSD view, accessible by using tabs at the bottom of the views. For example, the Flat File Extension adds a new view with a tab labeled Flat File.

Schema Validation

If a BizTalk Editor extension provides an **ISchemaValidator** object, the framework will invoke **ISchemaValidator.ValidateSchema** when the user invokes the **Validate Schema** command, or during compilation when the user builds the BizTalk project containing the schema. The extension usually validates the custom properties, and can return error messages as an array of **IValidationInfo** objects. The error messages are displayed in the Visual Studio .NET Task List window, along with errors returned from BizTalk Editor compiler.

Instance Validation

BizTalk Editor invokes the **IInstanceValidator.ValidateInstance** method of an extension when the following conditions are met:

- The extension is set as standard by using the **Standard** property of the **Schema** node.
- On the **Property Pages** dialog box associated with the schema, the **Create Instance Output Type** property is set to **Native**.
- On the **Property Pages** dialog box associated with the schema, the **Input Instance Filename** property is set to the name of a file containing the instance message to be validated.

The file specified in the **Input Instance Filename** property is passed as a parameter to the **IInstanceValidator.ValidateInstance** method.

If errors occur, error messages are returned as an array of **IValidationInfo** objects, and are displayed in the Visual Studio .NET Task List window.

Instance Generation

BizTalk Editor invokes the **IInstanceGenerator.GenerateInstance** method of an extension when the following conditions are met:

- The extension is set as standard by using the **Standard** property of the **Schema** node.
- On the **Property Pages** dialog box associated with the schema, the **Create Instance Output Type** property is set to **Native**.
- On the **Property Pages** dialog box associated with the schema, the **Output Instance Filename** property is set to the name of the file that the output instance will be saved to.

Before the **IInstanceValidator.ValidateInstance** method is called, BizTalk Editor generates a sample XML instance message, and then passes it and the file specified in the **Output Instance Filename** property, or a default file name, to that method.

If errors occur, error messages are returned as an array of **IValidationInfo** objects, and are displayed in the Visual Studio .NET Task List window.

Considerations When Creating Schemas

This section provides information about a variety of considerations related to schema creation. Some of these considerations apply to all types of schemas, and are addressed first. Other considerations are specific to flat file message schemas and envelope schemas. These latter considerations are organized into their own subsections.

For information about properly formed names for XML elements and attributes, see Node Name Property.

In This Section

- Namespace Management
- Code List Management
- Considerations When Creating Flat File Message Schemas

Namespace Management

BizTalk® Editor provides support for namespaces. An XML namespace is a collection of names that can be used as element or attribute names in an XML message. The namespace qualifies element and attribute names to avoid conflicts between the same element and attribute names that might be defined elsewhere within the same schema.

Namespaces are identified by a Universal Resource Identifier (URI), either as a Uniform Resource Locator (URL) or a Uniform Resource Name (URN). They are also given a typically short prefix alias that is prepended with a separating colon (:) from the element or attribute name itself. For example, it is common to see the following namespace declaration within the **schema** element in the XSD representation of the schema.

The prefix is **xs**, which you see throughout the XSD representation, qualifying such elements as the **element** element (xs:element) and the **attribute** element (xs:attribute).

When you first create a new schema, regardless of whether it is a message schema or a property schema, it is important to set the **Target Namespace** property of the **Schema** node properly. You need to establish the target namespace before the schema is used by another schema with the import/include/redefine mechanisms, and before any property promotions are defined.

The following two namespaces are automatically added as namespace declarations to the schema element in the XML Schema definition (XSD) language representation of the schema:

- xmlns:b="http://schemas.microsoft.com/BizTalk/2003"
- xmlns:xs="http://www.w3.org/2001/XMLSchema"

In the course of using other schemas within the schema you are creating, other namespaces will be declared. You can examine these namespaces, as well as the automatically included namespaces, in the **Imports** dialog box that you can access by using the **Imports** property of the **Schema** node. For more information about using other data types declared in other schemas within the schema you are creating, see Schemas That Use Other Schemas and Creating Schemas That Use Other Schemas.

Namespaces associated with property schemas can be examined in the **Promote Properties** dialog box.

Code List Management

You use XSD to specify a specific set of values that are valid for an element or attribute. This functionality is available using the **enumeration** element. When you derive a data type for a **Field Element** or **Field Attribute** node by restriction, one of the properties that becomes available to you in the **Restriction** category is the **Enumeration** property. Using this property, you can open the **Enumeration Editor** dialog box in which you can enter the values that should be considered valid for the corresponding element or attribute in instance messages.

Microsoft BizTalk Server provides an alternative, richer way to manage enumerations in your schemas, known as code lists. Code lists use a Microsoft Access database to store the choices for your various enumerations, which allows you to manage them in a more centralized way. Further, if the enumeration values you need to use consist of non-intuitive numeric codes, which would need to be entered in that form using the **Enumeration** property, the tables you create in an Access database to use with the code list functionality include textual descriptions of these numeric values. The textual descriptions are used in the **CodeList** dialog box rather than their more obscure numeric equivalents.

You must perform several different steps to use the code list feature, including:

- You must create an Access database with an appropriately named table, with the expected columns, and populate it with values.
 - The name of the table is a combination of the **Standard** and **Standard Version** properties of the **Schema** node, separated by an underscore (_) character. For example, if you have set the **Standard** property of the **Schema** node to XML and the **Standard Version** property to MyVersion1, the Access database specified by the **CodeList Database** property must have a table named XML_MyVersion1.
 - This table must have three columns, typically named Code, Value, and Desc. The first column identifies rows that are related to one another, where each such row provides one of the enumeration choices that may potentially be allowed for the data that corresponds to the selected **Field Element** or **Field Attribute** node. All rows with the same value in the first column form a group. These values are typically integers, but can be any string that does not contain spaces.

The second and third columns of each row in the table must be configured to contain the corresponding value and textual representation of each possible enumeration value, respectively.

For example, the following representation of an Access database table for use with the Code List feature contains two sets of three related enumeration values. The specific values in the first column are arbitrary and are used to associate related rows.

Code	Value	Desc
1	13	Red
1	16	Green
1	19	Blue

2	1	Small
2	2	Medium
2	3	Large

- You must properly configure three properties of the **Schema** node:
 - The **CodeList Database** property must be set to the name of the Access database you have created.
 - The **Standard** and **Standard Version** properties must be set such that when they are combined with a separating underscore (_) character, they form the name of the appropriate table within the specified Access database.
- To actually make use of the values in the Access database for a particular selected **Field Element** or **Field Attribute** node, you must configure two of its properties:
 - You must set its **Derived By** property to **Restriction**. The other property you need to configure, **CodeList**, will not be enabled until you perform this step.
 - You must type a value into the **CodeList** property that corresponds to the value in the first column (the **Code** column) of one or more rows in the specified Access database. This action identifies the set of enumeration values that you intend to have correspond to the selected **Field Element** or **Field Attribute** node.

Then you must click the ellipsis (...) button located to the right of the **CodeList** property value field to open the **CodeList** dialog box. Using the check boxes in this dialog box, select the values that you want to allow as legal values for the instance message data that corresponds to the selected **Field Element** or **Field Attribute** node. You are allowed to select only a subset of the available values. For example, using the preceding table example, if you type the value 1 into the **CodeList** property, the **CodeList** dialog box will contain the choices Red, Green, and Blue. If you select the check boxes for Red and Green, and do not select the check box for Blue, only the former colors will appear in the XSD as valid values for the selected **Field Element** or **Field Attribute** node.

Considerations When Creating Flat File Message Schemas

There are a number of considerations when working with flat file message schemas. This includes considerations that apply to all flat file schemas, as well as considerations that apply specifically to positional records, delimited records, positional fields, or delimited fields. There are also considerations about how to interpret otherwise special characters as regular data. This section provides information about these considerations.

In This Section

- Code Page Specification for Flat File Schemas
- Case Handling in Flat File Schemas

- Restricted Character Ranges
- Nested Positional and Delimited Records
- Positional Record Considerations
- Delimited Record Considerations
- Field Considerations
- Ways to Interpret Special Characters as Part of a Field Value

Code Page Specification for Flat File Schemas

The value in the **Code Page** property is used to create an encoding object that is used during the disassembly and assembly of flat file documents. This encoding object allows the flat file parser to convert the native encoding of an inbound flat file document into the normalized UTF-8 encoding that is used internally by Microsoft BizTalk Server 2006. The encoding object also allows the flat file serializer to convert the internal UTF-8 encoding back into the native encoding of the flat file document.

The setting of the **Code Page** property plays an important, but not exclusive, role in determining the character encoding scheme used by your flat file business documents. You must consider how inbound flat file messages are interpreted by the flat file disassembler as well as how the flat file assembler will encode characters as outbound messages are translated into flat file format.

There are multiple factors that play a role in determining how character encoding for a given instance message is handled, as follows:

- When disassembling a flat file instance message, the following algorithm is used to determine and preserve encoding information:
 1. If the **Charset** in the Message body part is set, its value is used.
 2. Otherwise, if the envelope (or document) schema specifies a code page using the **Code Page** property, its value is used.
 3. Otherwise, if a byte order mark is present, its value is used.
 4. Otherwise, assume UTF-8.
- When assembling a flat file instance message, the following algorithm is used to determine the character set to use for decoding:
 - If the **XMLNorm.TargetCharset** message context property is set, its value is used.
 - Otherwise, if the **TargetCharset** assembler (design-time) property is set, its value is used.
 - Otherwise, if the envelope (or document) schema specifies a code page using the **Code Page** property, its value is used.

1. Otherwise, if the **SourceCharset** message context property is set, its value is used.
2. Otherwise, use UTF-8.

Case Handling in Flat File Schemas

You can use the **Case** property to perform case conversion of flat file data when being translated from its equivalent XML format. When the flat file assembler translates an XML message into its equivalent flat file format, and the **Case** property is set to either **Uppercase** or **Lowercase**, all data governed by the corresponding schema will be converted to uppercase or lowercase, respectively, during the translation.

When the **Case** property is set to **(Default)**, no case conversion is performed.

Restricted Character Ranges

When creating a schema for flat file messages, you can direct BizTalk Server 2006 to block a particular character or a range of characters from being included in any outgoing message. To do this, in BizTalk Editor, open a schema that is to be used as a destination schema. Select the **Schema** node and use the **Restricted Characters** property to open the **Restricted Characters** dialog box. Enter the character range(s) that you want to prevent being included in any message sent by BizTalk Server, and then click **OK**. Whenever BizTalk Server attempts to process a character specified in the **Restricted Characters** dialog box of a destination schema, processing stops and an error message is generated.

Nested Positional and DelimIn the flat file formats supported by Microsoft BizTalk Server 2006, some combinations of positional and delimited records are allowed and others are disallowed. The following combinations are allowed:

- Flat files in which delimiters are used to determine the boundaries between all records and their subordinate records from each other, and in which (possibly different) delimiters are used to separate the fields within those records.
- Flat files in which the boundaries between all records, their subordinate records, and their fields are determined based on their position within the file according to predefined record and field lengths.
- Flat files in which delimiters are used to determine the boundaries between at least the outermost set of records in the file, and in which a mix of delimited and positional subordinate records are used. Boundaries between the fields within a delimited or positional subordinate record are determined using either delimiters or fixed field lengths, respectively. The subordinate records of a positional (subordinate) record must also be positional; in other words, once a portion of the file switches from delimited to positional records, that entire subordinate portion of the file must be positional.

Due to the parsing ambiguities that would result, positional records, wherever they occur, are prohibited from containing delimited subordinate records.

Positional Record Considerations

There are a number of considerations that you should keep in mind when working with positional **Record** nodes within your schemas. This includes the considerations about tag handling, positional record nesting,

how positions are counted, and how field positions are calculated. This section provides information about these considerations.

In This Section

- Tag Handling in Positional Records
- Nested Positional Records
- Position Counting in Bytes
- Field Position Calculation

Tag Handling in Positional Records

Positional records can include a well-known sequence of characters, known as a tag, which can be used to disambiguate one type of record from another. This allows the flat file disassembler to properly identify the appropriate **Record** node in the schema that contains the information required to correctly parse the flat file record.

You can use the **Tag Identifier** and **Tag Offset** properties together to specify the tag and its position within a positional record. Note that this allows the tag to occur anywhere within the positional record and that depending on your settings for the **Positional Length** and **Positional Offset** properties of the various fields within the record, the tag can be interpreted as part of data associated with one or more of these fields.

The **Positional Offset** property also allows you to treat the tag separately from the data in the record. For example, if the tag occurs at the beginning of the record and is four characters in length, you could set the value of the **Positional Offset** property of the first field in the record to four, thereby effectively skipping over the positional record tag when the value of the first field is translated in the equivalent XML format of the record.

Nested Positional Records

Nested positional records are allowed if the **Max Occurs** property of child records is set to a positive integer. Field autocalculation should be able to handle the new depth. However, there is a modification to the way this behaves. Specifically, because of the possibility for null delimiters, autocalculation of field positions will function only if one of the following conditions is met:

- The selected node has a parent that is infix delimited.
- The selected node has a specified starting position.

Note that there is a distinction between nested positional records and positional records whose parent is a delimited container where the delimiter is null. For structures to be truly nested positionally, there must not be any ambiguity in determining their length. For example, a delimited loop node can contain a repeating positional record that occurs 0 to N times. However, for that loop node itself to be positional, and possibly also contain fields as peers to the repeating positional record, the occurrence of the repeating positional record must be deterministic (a positive integer).

Position Counting in Bytes

You can use the **Count Positions In Bytes** property of the **Schema** node to specify how the values that you provide for the **Positional Length** and **Positional Offset** properties of the various fields within positional records as well as the values you provide for the **Tag Offset** property of the positional records themselves will be interpreted. By default, these values are interpreted as a number of characters, but when the **Count Positions In Bytes** property is set to **True**, these values are interpreted as a number of bytes.

Setting the **Count Positions In Bytes** property to **True** might be necessary when dealing with multibyte character set (MBCS or DBCS) data, or when your flat file messages originate in SAP, mainframes, or other systems that may count positions in bytes.

Counting field lengths in bytes can be complicated when the number of bytes used to encode characters is variable, and can result in some issues with respect to determining field boundaries. When the flat file disassembler parses a flat file in such situations, it attempts to make appropriate parsing decisions based on its knowledge of the character encoding in use.

An example of this type of parsing decision concerns lead bytes in MBCS character encodings. Lead bytes are well-known byte values that are used to begin multibyte character encodings, and which should never occur on their own. When specifying the length of the fields using bytes rather than character, situations may arise in which the last byte in a field is found to be a lead byte, which cannot constitute an entire character on its own. In such cases, the flat file disassembler will treat the character occurring just prior to the lead byte as the last character in the previous field, and begin parsing the next field starting with the lead byte.

Field Position Calculation

When you use the **Positional Length** and **Positional Offset** properties of the **Field Element** and **Field Attribute** nodes in your schema to define the layout of the positional records in your flat file message, the **Start Position** and **Length** columns of the **Flat File** tab in BizTalk Editor show the calculated starting positions and lengths, respectively, of the relevant fields and records.

In general, the starting position of a particular field *N* is the starting position of the previous field, plus the length of the previous field, plus the (positional) offset you have specified for field *N*.

All field position calculation in Microsoft BizTalk Server 2006 is performed automatically, on-the-fly, without any need to execute a command (as was required in previous versions of BizTalk Server).

Delimited Record Considerations

There are a number of considerations that you should keep in mind when working with delimited **Record** nodes within your schemas. This includes the considerations about the order of subordinate nodes relative to their delimiters, cases in which you can choose to omit delimiters when assembling the flat file representation of a message, and restrictions regarding the intermixing of delimited and positional records. This section provides information about these considerations.

In This Section

- Tag Handling in Delimited Records
- Child Order Considerations
- Delimiter Preservation and Suppression

Tag Handling in Delimited Records

Delimited records can include a well-known sequence of characters, known as a tag, which can be used to disambiguate one type of record from another. This will allow the flat file disassembler to properly identify the appropriate **Record** node in the schema that contains the information required to correctly parse the flat file record.

You can use the **Tag Identifier** property to specify the tag within a delimited record. Unlike tags in positional records, tags in delimited records must occur at the beginning of the delimited record and are automatically never included in the data when the record is translated to its equivalent XML format.

Child Order Considerations

There are two scenarios related to delimited flat files for which special considerations apply when setting the **Child Order** property. The first such scenario concerns situations in which the flat file document has a header, a body, and optionally, a trailer. In these scenarios, you must observe the following requirements:

- You must set the **Child Order** property of the (delimited) root record of the header to **Postfix**.
- If a trailer is present, you must set the **Child Order** property of the (delimited) root record of the body to **Postfix**.
- If a trailer is not present, you may set the **Child Order** property of the (delimited) root record of the body to **Prefix**, **InFix**, or **Postfix**.
- If a trailer is present, you may set the **Child Order** property of the (delimited) root record of that trailer to **Prefix**, **InFix**, or **Postfix**.
- You may set the **Child Order** property of delimited subordinate records of the header, body, and trailer to **Prefix**, **InFix**, or **Postfix**.

The second scenario related to delimited flat files and the **Child Order** property is that this property must be set according to what the runtime components expect for the nodes. The correct setting for the **Child Order** property may not be apparent for root and group nodes, as illustrated in the following scenarios:

- **Root node.** Consider a typical flat file whose structure consists of records followed by a CR/LF combination. The delimiter separates records in the file, and the sequence is typically record, delimiter, record, delimiter, and so on. In this situation, the delimiter always follows the data, which corresponds to a **Child Order** property setting of **Postfix**.
- **Group nodes.** The group nodes shown in the BizTalk Server and XSD representation of the schema are not explicitly present in the flat file representation of the instance message. Consider a

purchase order (PO) that contains a collection of records for each line item, and those records repeat numerous times to represent multiple line items in a single PO. The schema for such a message would likely include a node named `LineItems` to serve as the (sometimes conceptual) container for the repeating set: in the flat file representation of the instance message, the `LineItems` container is conceptual in nature, represented by the appropriate sequence of data and delimiters; in the XML representation of the instance message, the `LineItems` container is explicitly present in the form of a **LineItems** element in XML.

Consider a message containing a root node and only one group node. It is easy to see where the last delimiter in the input stream would belong to the root node. Therefore, the data/delimiter sequence in the conceptual loop would merely be one or more line item records. Only in the case where there are more than one line item records would there be a delimiter to separate them. In that case, the number of delimiters is one less than the sets of things being delimited, and the delimiters are located between the delimited items in a structure known as Infix.

Delimiter Preservation and Suppression

There are two properties that apply to delimited records, **Preserve Delimiter For Empty Data** and **Suppress Trailing Delimiters**, that you can use to control how the flat file assembler handles delimiters associated with nonexistent data and trailing delimiters. When you set the **Preserve Delimiter For Empty Data** property to **Yes** (which is the default setting), delimiters are included in the translated flat file message for:

- Fields without data.
- Immediately subordinate records without data that do not have a tag associated with them.

When you set the **Preserve Delimiter For Empty Data** property to **No**, delimiters are not included in the translated flat file for records and fields without data. Further, regardless of the setting of the **Preserve Delimiter For Empty Data** property, delimiters will not be included in the translated flat file message for immediately subordinate records without data for which a tag is defined.

When you set the **Suppress Trailing Delimiters** property to **No** (which is the default setting), one or more trailing delimiters may be included in the translated flat file message. When you set the **Suppress Trailing Delimiters** property to **Yes**, trailing delimiters are not included in the translated flat file message.

There are some special cases where the behaviors caused by the settings of the **Preserve Delimiter For Empty Data** and **Suppress Trailing Delimiters** properties can conflict. In such cases, the behaviors associated with the latter property, **Suppress Trailing Delimiters**, will take precedence. Further, there are some special cases where you will be warned about potential conflicts between the settings you have chosen for these two properties.

For example, consider a **Record** node defined with the following property values:

- Node Name is MyRec
- Tag Identifier is Rec
- Child Delimiter is ,

- Child Order is Infix

And defined to contain five **Field Element** nodes with the following names (they could also be **Field Attribute** nodes or subordinate **Record** nodes):

- FieldElem1
- FieldElem2
- FieldElem3
- FieldElem4
- FieldElem5

Next, assume that the following mainly empty XML fragment, representing this **Record** node, is passed to the flat file assembler.

The following table shows the output produced, and the associated additional property setting requirements for the relevant schema nodes, based on different settings for the **Preserve Delimiter For Empty Data** (PDFED) and **Suppress Trailing Delimiters** (STD) properties.

PDFED setting	STD setting	Output	Additional node requirements
Yes	No	Rec,,,Val,,	None.
No	Yes	Rec,Val	All Field Element nodes must be configured as optional.
Yes	Yes	Rec,,,Val	Nodes named FieldElem4 and FieldElem5 must be configured as optional.
No	No	Rec,Val,,	All Field Element nodes must be configured as optional.

When these property settings specify that delimiters should either not be preserved or should be suppressed, a message warning that it might not be possible to parse the serialized flat file data using the same schema will be issued in the following two cases:

- When the **Record** node for which the **Preserve Delimiter For Empty Data** property is set to **No** and/or the **Suppress Trailing Delimiters** property is set to **Yes**, respectively, contains subordinate **Field Element** nodes, **Field Attribute** nodes, or **Record** nodes for which no tag is specified.
- When the subordinate **Field Element** nodes, **Field Attribute** nodes, and **Record** nodes for which no tag is specified are not configured to be optional (by setting the **Min Occurs** property set to zero) in the schema. When the **Suppress Trailing Delimiters** property is set to **Yes**, only the last such subordinate nodes need to be configured as optional. When the **Preserve Delimiter For Empty Data** property is set to **No**, all trailing subordinate nodes need to be configured as optional.

Field Considerations

There are a number of considerations that you should keep in mind when working with **Field Element** and **Field Attribute** nodes within your schemas. This includes the basic distinctions regarding when to use each of these nodes types, as well as more specific considerations related to the specification of field lengths, field justification, and field padding. This section provides information about these considerations.

In This Section

- Field Element Nodes vs. Field Attribute Nodes
- Custom Date/Time Formats
- Field Padding
- Field Justification
- Specification of Field Positions within Positional Records
- Minimum Field Lengths Within Delimited Records

Field Element Nodes vs. Field Attribute Nodes

Flat file schemas are used by the flat file disassembler to control how inbound flat file instance messages are translated into their equivalent XML form, and are used by the flat file assembler to control how outbound XML messages are translated into their equivalent flat file instance messages. When constructing such schemas, you use either a **Field Element** node or a **Field Attribute** node in particular positions within the schema to control whether a particular field in the flat file instance message corresponds to an XML element or to an XML attribute in the equivalent XML form of the message.

For example, the left-aligned, asterisk-padded field value "red*****" in a flat file instance message can be translated into its equivalent XML representation in two different ways depending upon whether that field in the schema is a **Field Element** node or a **Field Attribute** node. When that field is represented in the schema by a **Field Element** node with its **Node Name** property set to "color", and the containing **Record** node has its **Node Name** property set to "shirt", the XML equivalent of the flat file field is (shown in bold type).

When that same flat file field is represented in the schema by a **Field Attribute** node with its **Node Name** property set to color, and the containing **Record** node has its **Node Name** property set to shirt, the XML equivalent of the flat file field is (shown in bold type):

Custom Date/Time Formats

Due to their legacy origins, the flat file formats for which you create flat file schemas are bound to use date and time formats that do not conform to ISO 8601 formats. Therefore, when you are creating a flat file schema and you set the **Data Type** property of a **Field Element** or **Field Attribute** node to one of the XML Schema definition (XSD) language primitive data types, **xs:dateTime**, **xs:time**, or **xs:date**, you can use the **Custom Date/Time Format** property to specify an alternative format for date or time values.

When the flat file disassembler translates such a field to its equivalent XML format, the value of the **Custom Date/Time Format** property will be used to allow the flat file date/time format to be converted to its ISO 8601 compliant equivalent. Likewise, when the flat file assembler translates an ISO 8601 compliant date/time value to its flat file equivalent, the format string specified in the **Custom Date/Time Format** property will be used to construct the appropriate date/time format expected in the flat file.

You can configure the **Custom Date/Time Format** property with almost any time and date format, except for Julian dates. The drop-down list provides various choices, but you can also type a different format of your choosing. The date and time formats use the Common Language Runtime (CLR) **DateTime** facilities. The exception is that a single character d, m, or M is automatically prepended with a percent sign (%) to yield the corresponding single element of the **DateTime** value. The allowable separators for custom date/time formats are dash (-), slash (/), and period (.). For more information about **DateTime** formats, search on "DateTimeFormatInfo" in the Visual Studio document collection.

Field Padding

Pad characters are used in fields within both delimited and positional records when the data contained within the field is smaller than the number of characters or bytes reserved for the field. These characters occupy the portion of the field not required by the data, if any. Pad characters are specified on a field-by-field basis using the **Pad Character** and **Pad Character Type** properties of the corresponding **Field Element** and **Field Attribute** nodes. If no pad character is specified for a particular field, the default pad character, space (" "), is used for that field.

For inbound instance messages, regardless of whether a particular record is positional or delimited, the flat file disassembler discards leading or trailing instances for the specified or default pad character for a particular field as the instance message is translated into its equivalent XML form. Whether it is leading or trailing instances of the relevant pad character that are discarded depends on whether the **Justification** property of corresponding **Field Element** and **Field Attribute** node is set to **Right** or **Left**, respectively.

For outbound instance messages, the flat file assembler will insert the appropriate number of the specified or default pad character into fields so that the length of the field is correct. The pad characters will be inserted before or after the data characters based on whether the **Justification** property of the corresponding **Field Element** and **Field Attribute** node is set to **Right** or **Left**, respectively.

When the field to be padded in an outbound instance message is contained within a positional record, the **Positional Offset** and **Positional Length** properties of the corresponding **Field Element** or **Field Attribute** node, combined with the number of data characters that the field must contain, determine whether any pad characters are required, and if so, how many. When the field to be padded in an outbound instance message is contained within a delimited record, pad characters are only inserted when the value of the **Minimum Length with Pad Character** property of the corresponding **Field Element** or **Field Attribute** node exceeds the number of data characters.

Field Justification

Field justification concerns whether the data characters in a field occur before (left-aligned) or after (right-aligned) any accompanying pad characters.

Sometimes the data characters contained within a field do not require all of the space dedicated to that field. This is true most frequently in positional records, where the number of bytes or characters dedicated to a field is fixed, as determined by the **Positional Length** and **Positional Offset** properties. It is

common in such scenarios that the item of data is smaller than the field length, with the unused portion of the field being filled with pad characters.

Such padding can also occur in delimited records when the value of the **Minimum Length with Pad Character** property exceeds the space required to store the relevant item of data.

In both such cases, the value of the **Justification** property (**Left** or **Right**) of the relevant **Field Element** or **Field Attribute** node determines whether the pad characters will follow the data characters (left-aligned) or whether the pad characters will precede the data characters (right-aligned).

When the flat file disassembler is converting a flat file instance message into an equivalent XML instance message, the **Justification** property is used when parsing the corresponding field. When the flat file assembler is converting an XML instance message into an equivalent flat file instance message, the **Justification** property is used to determine when the pad characters associated with a particular field, if any, should be added to the data stream: either before or after the corresponding data characters.

Specification of Field Positions within Positional Records

To define a positional record, you must provide information about the positions and lengths of the fields within that record. If the record contains subrecords, the positions and lengths of the fields in the subrecord are rolled up to contribute to the information about the containing record.

The sum of the values you specify for the **Positional Offset** and **Positional Length** properties for a particular **Field Element** or **Field Attribute** node determines the number of characters dedicated to the corresponding field. The series of these sums across all of the fields in the record and any of its subrecords determine the boundaries of the fields in the records.

Positional Offset property

When the flat file disassembler is converting a flat file instance message into an equivalent XML instance message, the value you specify for the **Positional Offset** property defines a number of characters (or bytes) that are ignored and skipped over at that position in the instance message. In other words, any information occurring at that starting position and length (the latter as specified by the **Positional Offset** property) in the flat file instance message will not be copied into the XML version of the message.

When the flat file assembler is converting an XML instance message into an equivalent flat file instance message, the value you specify for the **Positional Offset** property defines a number of characters (or bytes) that are filled with space characters at that starting position within the flat file instance message being created. The space character is always used to fill offset positions; the character used is not configurable.

The **Positional Offset** property provides flexibility for interpreting the contents of positional records. Essentially, this property allows you to ignore any fixed length data that precedes the field for which it is set to a nonzero value. That fixed length data might be one or more entire fields of data or some type of constant data, such as a tag associated with the field, that does not need to be included in the XML equivalent of the flat file instance message. For more information, see the example that follows.

Positional Length property

When the flat file disassembler is converting a flat file instance message into an equivalent XML instance message, the value you specify for the **Positional Length** property defines the number of characters (or bytes) that are associated with the field at that position in the instance message. The information occurring at that starting position and length in the flat file instance message constitutes the data in the field, subject to the additional information provided by the associated **Justification** and **Pad Character** properties. For more conceptual information about justification and field padding, see Field Justification and Field Padding.

When the flat file assembler is converting an XML instance message into an equivalent flat file instance message, the value you specify for the **Positional Length** property defines a number of characters (or bytes) available for writing the data associated with that field. If there are fewer data characters than the specified length of the field, the relevant pad character is used to fill up the difference. If there are more data characters than the specified length of the field, the beginning or end of the data is truncated based on the setting of the **Justification** property and not included in the flat file instance message being constructed.

Example

Consider the following field definitions for a record.

Field node name	Offset	Length	Pad Character	Justification
Field1	0	6	Default (space)	Left
Field2	0	4	*	Right
Field3	2	6	*	Left
Field4	4	6	Default (space)	Right

And the following stream of characters is encountered at the starting point of a record with these field definitions (the first line is for counting character positions).

When these field definitions are applied to this sample record data, the flat file disassembler produces the following XML equivalent (with the data shown in bold type).

The following observations are relevant to how this data is parsed:

- The characters associated with Field1 (length 6 with no offset) are "abc ", but the spaces are not included in the XML because the space character is the (default) pad character for Field1 and Field1 is defined as left-aligned.
- The characters associated with Field2 (length 4 with no offset) are "**12", but the asterisks are not included in the XML because the asterisk character is the pad character defined for Field2 and Field2 is defined as right-aligned.
- The characters associated with Field3 (length 6 plus an offset of 2) are "345678**", but the 3 and 4 are not included in the XML because of the offset. The asterisks are also not included in the XML because the asterisk character is the pad character defined for Field2 and Field2 is defined as left-aligned.
- The characters associated with Field4 (length 6 plus an offset of 4) are "skip here", but the character sequence "skip" is not included in the XML because of the offset. The two space characters are also not included in the XML because the space character is the (default) pad character for Field4 and Field4 is defined as right-aligned.

If the XML produced by the flat file disassembler in this example is passed to the flat file assembler using the same field definitions, the same flat file data is produced, with two exceptions: the discarded offset sequences 34 and skip are filled with space characters (indicated with the ^ character in the line following the data).

Minimum Field Lengths Within Delimited Records

By definition, the fields in positional records are all defined to have specific exact lengths. Fields in delimited records can also be defined to have a minimum length. This characteristic is defined by the **Minimum Length with Pad Character** property of **Field Element** and **Field Attribute** nodes.

When you provide a nonzero value for the **Minimum Length with Pad Character** property, the flat file assembler will determine whether the number of data characters associated with the field is smaller than the setting of the **Minimum Length with Pad Character** property, the relevant pad character will be used to make up the difference.

The pad characters will be added before or after the data characters based on the setting of the **Justification** property for the field. When the **Justification** property is set to **Left**, any pad characters required to meet the minimum length will be added after the data characters. When the **Justification** property is set to **Right**, any pad characters required to meet the minimum length will be added before the data characters.

When you provide a nonzero value for the **Minimum Length with Pad Character** property, the flat file disassembler will examine the beginning or end (based on the setting of the **Justification** property) of the field value for the presence of the relevant pad character, and if present, the pad characters will be discarded and not appear in the equivalent XML message being constructed.

Ways to Interpret Special Characters as Part of a Field Value

Fields in both positional and delimited records can contain special characters of different types, such as pad, wrap, and escape characters. Delimited records can also contain one or more different delimiter characters that are used to separate the fields within a record and one record from another record. Sometimes these special characters are part of the data itself, and are not meant to be interpreted as special in those cases.

The flat file schemas used by Microsoft BizTalk Server 2006 support two different techniques for flagging occurrences of otherwise special characters as simply part of the data contained by a field. These two techniques employ escape characters and wrap characters, respectively.

In This Section

- Escape Characters
- Wrap Characters

Escape Characters

An escape character is a single character that suppresses any special meaning of the character that follows it. For example, if you define a flat file record as having the following characteristics:

- Name = Record1
- Delimited
- Child delimiter = comma character (,)
- Child order = prefix
- Escape character = backslash character (\)
- Tag = RECORD1
- Two fields named Field1 and Field2

Then the following flat file data applies for the record.

The data will be disassembled into the following fragment of XML.

Note that the escape character sequence \, indicated on the line following the flat file record, has been converted to a single comma character without the escape character in the data for Field1 in the equivalent XML record. Furthermore, that comma character was not interpreted as a field delimiter like the other two commas are.

When the flat file assembler performs the reverse operation, converting the XML version of the record to its equivalent flat file record, the escape character will be inserted before the comma in the middle of Field1, thereby indicating that it should be interpreted as data rather than as a field delimiter.

When creating a flat file schema using BizTalk Editor, you can define a default escape character for the entire schema using the **Default Escape Character** and **Default Escape Character Type** properties of the **Schema** node. Then, you can configure each individual record in the schema to either use this default escape character or a custom, record-specific escape character using the **Escape Character** and **Escape Character Type** properties of the **Record** node.

Wrap Characters

A wrap character is a single character that is used to wrap the data characters in a field for the purpose of suppressing any special meaning that any of those data characters would otherwise have. For example, if you define a flat file record as having the following characteristics:

- Name = Record1
- Delimited
- Child delimiter = comma character (,)
- Child order = infix
- Escape character = backslash character (\)
- Tag = RECORD1
- Three fields named Field1, Field2, and Field3, each defined to use the number sign character (#) as their wrap character.

Then the following flat file data applies for the record.

The data will be disassembled into the following fragment of XML.

Note that the wrap characters (#) surrounding the bolded data characters field1, field2, and field3 have been removed.

When the flat file assembler performs the reverse operation, converting the XML version of the record to its equivalent flat file record, the wrap characters will be inserted before and after the data characters of each of the fields, yielding the original sequence of flat file characters.

The defined escape character can be used in conjunction with the defined wrap character. For example, suppose the value of Field1 is changed as follows (shown in bold type).

When this XML fragment is assembled, using the record and field definitions provided, the following sequence of flat file characters is produced (the escaped number sign character sequence is shown in bold type).

When creating a flat file schema using BizTalk Editor, you can define a default wrap character for the entire schema using the **Default Wrap Character** and **Default Wrap Character Type** properties of the **Schema** node. Then, you can configure each individual field in the schema to either use this default wrap character or a custom, field-specific wrap character using the **Wrap Character** and **Wrap Character Type** properties of the **Field Element** or **Field Attribute** nodes.

Known Issues with Schema Generation and Validation

This topic provides information about known issues with schema generation and validation.

An instance message generated for a positional record with tags could be incorrect

For positional records, the tag can be within a field or can span between fields. In either case, the instance generated will not be valid and will cause a failure in the Parsing Engine during the parsing stage.

If the tag is not part of any children (child records or child fields), this problem will not occur.

To work around this issue, include the actual value of the tag as the default in the schema. In the flat file extension of the BizTalk Editor, you can set the **Fixed Value** or **Default Value** property of the appropriate positional field with the value of the tag.

An instance message generated for a field with some restrictions may not pass validation

When you generate an instance message from a schema that contains one or more **Field Element** and **Field Attribute** nodes that have data types that have been derived using the restriction mechanism, such as when the **Pattern** property is used, the sample data generated for such fields may not conform to the requirements of the restriction, thereby preventing successful validation of that instance message using the same schema from which it was generated.

An instance message generated for a schema that contains an infinite loop may not be valid.

Your schema can contain an infinite loop when it contains a circular reference to a node with a **Min Occurs** property value greater than or equal to one, essentially preventing a termination condition. Instance message generation will artificially terminate so that the generation operation can complete, but the produced instance message will thereby not conform to the schema from which it was generated. Such schemas are usually suspect.

Schemas that contain GB18030 characters are not supported

If your schema contains GB18030 characters it will fail compilation. The only workaround is to avoid using the GB18030 character set in your schema.

Managing Business Processes Using BizTalk Explorer

BizTalk® Explorer is a Microsoft® Visual Studio® .NET tool window that displays the contents of a BizTalk Management database. It displays items such as assemblies, ports, and parties in a hierarchical tree. You can use BizTalk Explorer to configure and manage BizTalk projects, parties, and orchestrations.

The primary tool for managing the artifacts in your BizTalk Management database is BizTalk Explorer. BizTalk Explorer enables you to explore and manage artifacts such as receive locations and send ports through a graphical interface in the Visual Studio .NET development environment. This same functionality is also available within the BizTalk Explorer Object Model, which enables you to generate and manage your artifacts programmatically using C#, Microsoft® Visual Basic® .NET, or scripts.

The benefit of using BizTalk Explorer is that it decouples the business logic involved in creating a solution from the implementation details of the solution. A developer can create an orchestration without knowledge of the specific send ports or receive locations; these can be configured later using BizTalk Explorer. This not only enables the reuse of orchestrations, it also enables increased flexibility to make changes without having to change the actual applications.

BizTalk Explorer is a tool that enables you to view and manage the configuration details of your project. Included in these management tasks are:

- Viewing databases and assemblies
- Deploying and undeploying business processes
- Creating and editing ports, roles, and parties
- Enlisting parties into roles

In This Section

- Permissions Required for Deploying and Managing a BizTalk Application
- About BizTalk Explorer Artifacts
- Planning for BizTalk Explorer

- Using BizTalk Explorer
- Determining the Right Tool For Your Task
- Managing BizTalk Management Databases Using BizTalk Explorer
- Managing Assemblies Using BizTalk Explorer
- Managing Send Ports Using BizTalk Explorer
- Managing Send Port Groups Using BizTalk Explorer
- Managing Receive Ports and Receive Locations Using BizTalk Explorer
- Managing Orchestrations Using BizTalk Explorer
- Managing Parties Using BizTalk Explorer
- Enlisting a Party Using BizTalk Explorer
- Refreshing BizTalk Explorer
- Automating Business Process Management

Considerations When Using BizTalk Explorer

While BizTalk Explorer is a powerful tool for controlling the different configuration options for a project, there are some considerations that may affect your ability to use it in certain circumstances.

Managing artifacts across multiple BizTalk applications is not supported

If you need to manage artifacts for different BizTalk applications you should use the BizTalk Server Management Console. This is because the Explorer UI in Visual Studio is not application-aware -- it does not properly "know" about other application configurations that may be deployed on the server.

Permissions Required for Deploying and Managing a BizTalk Application

Application deployment includes deploying BizTalk assemblies from Visual Studio as well as importing, exporting, and installing BizTalk applications. The basic permissions you need to perform these tasks are as follows:

- As a member of the BizTalk Server Administrators group, you are granted the permissions required to deploy BizTalk assemblies from Visual Studio.
- As a member of the BizTalk Server Administrators group, you are granted the permissions required to import BizTalk applications into a BizTalk group. If the option to add an assembly included in the application to the global assembly cache (GAC) on import has been specified, you must also have

Write permissions on the assembly folder. As a member of the local Administrators group, you have this permission.

- As a member of the BizTalk Server Administrators or BizTalk Server Operators group, you are granted the permissions required to:
 - Export BizTalk applications
 - Start and stop send ports, send port groups, and orchestrations
 - Enable and disable receive locations
 - Suspend, resume, and terminate instances
 - Start and stop applications
- As a member of the local Administrators group you are granted permissions to install BizTalk applications on the local computer.

You may want to provide the most restrictive permissions for users to perform these tasks. The remainder of this topic provides more details on the required permissions, as follows.

- Permissions for deploying BizTalk assemblies from Visual Studio
- Permissions for importing an application
- Permissions for exporting an application
- Permissions for installing an application
- Permissions for the Base EDI Adapter

Permissions for deploying BizTalk assemblies from Visual Studio

To deploy BizTalk assemblies from within Visual Studio, you must have Write permission on the BizTalk Management database, at a minimum. You are granted this permission as a member of the BizTalk Server Administrators group.

Permissions for importing an application

To import a BizTalk application, you must have the following permissions, at a minimum. You are granted all of the required permissions as a member of the BizTalk Server Administrator's group, except that if you want to install any assemblies to the GAC, you must also have Write permissions on the assembly folder.

Item	Permissions	When Required
BizTalk Management database	Read/Write	Always required.
BizTalk Rule Engine database	Read/Write	Required only if the application includes rules resources.
BAS database	Read/Write	Required only if the application includes BAS resources.
BAM database	Read/Write	Required only if the application includes BAM resources
Global assembly cache (GAC)	Read/Write	Required only if the application includes assembly resources, and you specify that the assemblies are added to the GAC on import. (See Note.)

Permissions for exporting an application

To export a BizTalk application, you must have the following permissions, at a minimum. You are granted the required permissions as a member of the BizTalk Operators group.

Item	Permissions	When Required
BizTalk Management database	Read	Always required.
BizTalk Rule Engine database	Read	Required only if the application includes rules resources.
BAS database	Read	Required only if the application includes BAS resources.
Certificate store	Read	Required only if the application includes certificate resources.
Internet Information Services	Read	Required only if the application includes virtual directory resources.

Permissions for installing an application

By default, members of the local Administrators group have the permissions required to install BizTalk applications on the local computer. If you want to provide more restricted permissions to users who need to install applications, the following table provides the minimum permissions that you must configure. In addition to these permissions, if your application has resources that require additional permissions to install, such as to create a new database or database table, you must also have these permissions.

Item	Permissions	When Required
Certificate store	Read/Write	Required only if the application includes certificate resources.
Internet Information Services	Read/Write	Required only if the application includes virtual directory resources.
GAC	Read/Write	Required only if the application includes assembly resources, and you specify that the assemblies are added to the GAC on install. (See Note, below.)
File system	Read/Write	Required only if a destination property has been set for a resource.
Registry	Read/Write	Required if the regsvcs or regasm property is set to True for an assembly resource containing managed COM or COM+ components.
Registry	Read/Write	Required if the application includes unmanaged COM resources

Permissions for the Base EDI Adapter

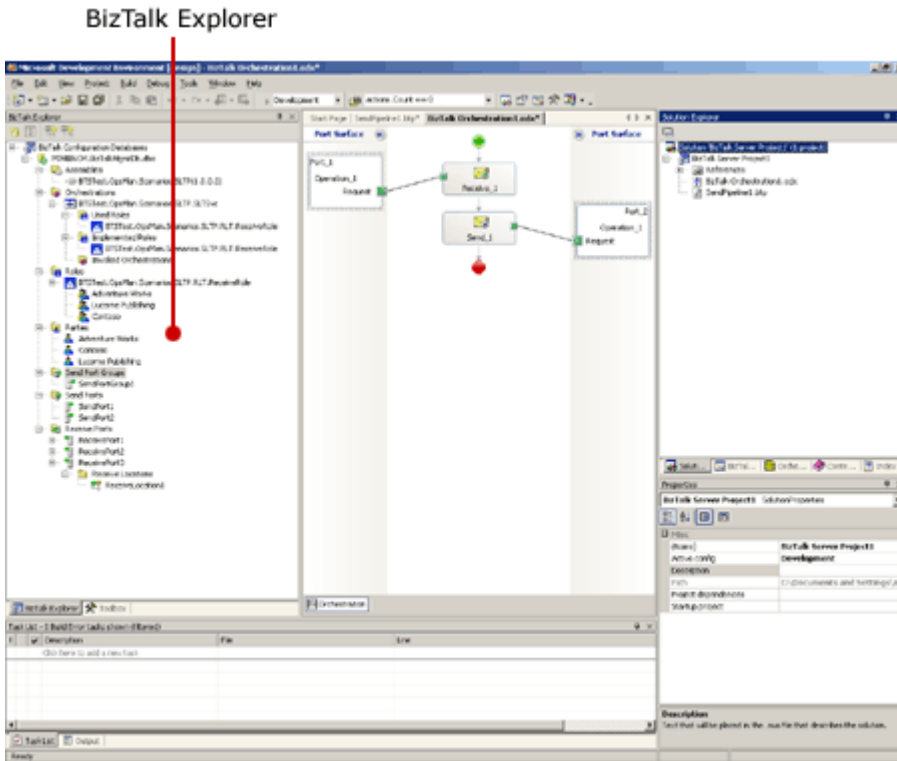
The EDI subsystem service account must be a member of the BizTalk Host Users group ("BizTalk Application Users" by default) in order for it to be able to access the BizTalk Management database. If the EDI subsystem service account is not able to access the BizTalk Management database, the error message "The EDI Subsystem Service was not started because user [domain\service account user name] does not have sufficient authorization" will appear in the event log and in an error dialog box when you start the EDI subsystem service manually.

About BizTalk Explorer Artifacts

BizTalk® Explorer is a tool that enables you to view and manage the configuration details of your project. Included in these management tasks are:

- Viewing, managing, and configuring your BizTalk Management database
- Working with assemblies
- Deploying and undeploying orchestrations
- Creating and editing roles, parties, send port groups, send ports, receive ports, and receive locations

BizTalk Explorer displays items in a hierarchical tree.



Each of the items in the BizTalk Explorer tree is described in this section.

BizTalk Management databases

This is the root node in BizTalk Explorer. It acts as a container for the BizTalk Management databases that you add to BizTalk Explorer. The BizTalk Management database, also referred to as the Configuration database, contains all configuration information specific to a solution.

Assemblies

BizTalk Server 2006 projects are deployed as assemblies. This node contains the assemblies that have been deployed into the parent BizTalk Management database.

You can use BizTalk Explorer to undeploy a specific version of an assembly if it has changed. The assembly can then be redeployed using the BizTalk Deployment Wizard. Other than this, you use BizTalk Explorer to work with the objects inside an assembly rather than the assembly as a whole.

The assembly name is specified as a property of the project. For more information about deploying assemblies to the BizTalk Management database, see [How to Deploy a BizTalk Assembly from Visual Studio](#).

For more information about assemblies, see the [.NET Framework Developer's Guide](http://go.microsoft.com/fwlink/?LinkId=24808) at <http://go.microsoft.com/fwlink/?LinkId=24808>.

Orchestrations

This node contains the orchestrations and dependencies of the orchestrations that have been deployed into the parent BizTalk Management database.

An orchestration is a representation of a business process. When you create an orchestration, the orchestration provides abstract send and receive ports, but no implementation details for those ports.

By using BizTalk Explorer, you can change the configuration of an orchestration as your requirements change. This is one of the advantages of decoupling implementation details from abstract design-time concepts.

You can use BizTalk Explorer to perform the following actions on an orchestration:

- Bind
- Enlist
- Unenlist
- Start
- Stop

For more information about orchestration management, see *Orchestration Management in BizTalk Explorer* and *Managing Orchestrations Using BizTalk Explorer*.

Roles

This node contains the roles that are deployed in the parent BizTalk Management database and created in an orchestration. A role represents the type of interaction that a party, or organizational unit, can have with one or many orchestrations. Roles provide flexibility and ease of management as the number of parties increases.

For more information about managing roles, see *Partner Management in BizTalk Explorer* and *Enlisting a Party Using BizTalk Explorer*.

Parties

This node contains the parties that are created in the parent BizTalk Management database. A party represents an entity outside of BizTalk Server that interacts with an orchestration. In BizTalk, each organization with which you exchange messages is represented by a party. You can define how the party will interact by enlisting it in a role.

For more information about managing parties, see *Partner Management in BizTalk Explorer* and *Managing Parties Using BizTalk Explorer*.

Send port groups

This node contains the send port groups in the parent BizTalk Management database. A send port group is a named collection of send ports that you can use to send the same message to multiple destinations. You can create a group through the **Send Port Group Properties** dialog box by selecting send ports that exist in the database. Only one-way static send ports can be added to a send port group.

For more information about managing send port groups, see *Managing Send Port Groups Using BizTalk Explorer*.

Send ports

This node contains the send ports in the parent BizTalk Management database. There are two types of send ports:

- **Static send ports.** You can create static send ports within Orchestration Designer or by using BizTalk Explorer. You specify the adapter type, destination address (URI), and pipeline to use when you create a static send port.
- **Dynamic send ports.** Dynamic send ports do not contain a fixed destination address, only a pipeline. The destination address is determined at run time from a specified property in the message.

A BizTalk Explorer port is a BizTalk element that receives or sends messages. Binding a BizTalk Explorer port to an orchestration links a physical address on a server to a pipeline and a port on the orchestration.

A send port can function as a one-way port or as a two-way port. A one-way send port sends messages. A solicit-response (two-way) port can both receive and send messages. Not all adapters support two-way ports. For example, the File, MSMQT, and SMTP adapters only support one-way ports, while HTTP and SOAP adapters support two-way ports.

A send port can either bind to the send port of an orchestration, contain filters and maps to enable pass-through and content based routing scenarios, or both at the same time.

For more information about managing send ports, see *Managing Send Ports Using BizTalk Explorer*.

Receive ports

This node contains the receive ports in the parent BizTalk Management database. A receive port can function as a one-way port or as a two-way port. A one-way receive port only receives messages. A request-response (two-way) port can both receive and send messages. A receive port is a logical grouping of receive locations. For example, you create multiple receive locations, and you want the messages received at these locations to be processed by a specific orchestration. You would use BizTalk Explorer to create a receive port and group these two receive locations under that port.

For more information about managing receive ports, see *Managing Receive Ports and Receive Locations Using BizTalk Explorer*.

Receive locations

Receive locations are displayed as child nodes of the receive port that contains them. A receive location connects a receive handler that is configured for a specific address (URI) to a receive pipeline.

For more information about managing receive locations, see [Managing Receive Ports and Receive Locations Using BizTalk Explorer](#).

In This Section

- [About Enlisting, Starting, Stopping, and Unenlisting BizTalk Server Artifacts](#)
- [Partner Management in BizTalk Explorer](#)
- [Content-Based Routing in BizTalk Explorer](#)
- [Document Normalization in BizTalk Explorer](#)
- [Orchestration Management in BizTalk Explorer](#)
- [BizTalk Explorer Automation](#)

About Enlisting, Starting, Stopping, and Unenlisting BizTalk Server Artifacts

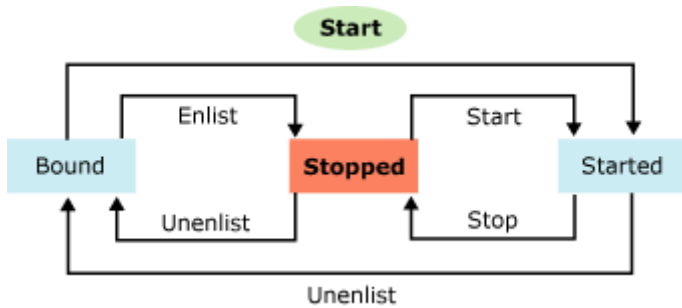
BizTalk Server has many artifacts, such as send ports, send port groups, receive locations, parties, and orchestrations. Of the artifacts, send ports, send port groups, and orchestrations have a state, which determines whether the artifact can process messages. To change the state of the artifact, you must invoke an action on the artifact.

The following table lists all of the states and actions available for send ports, send port groups, and orchestrations.

States	Actions
Bound	Enlist
Started	Start
Stopped	Unenlist
Unbound (Orchestrations only)	Stop
	Bind (Orchestrations only)

The following illustration shows how the states and the actions of the artifacts can be changed.

Illustrates the states and actions of send ports, send port groups, and orchestrations.



Partner Management in BizTalk Explorer

Partner Management involves managing the following entities:

- Parties
- Roles and Party Enlistment
- Service Link Type
- Aliases

Parties

Parties are all the entities outside of BizTalk Server that interact with an orchestration. All of the partners that your organization deals with are considered parties, and your organization may have several thousand partners. BizTalk Explorer has a user interface that can be used to manage parties, but managing thousands of parties can be unreasonable in a user interface, in which case you can use the BizTalk Explorer Object Model to write code to add parties, remove parties, add a party and automatically enlist to a role, or copy a party from a legacy system.

Roles and party enlistment

After you add a party to BizTalk Server 2006, it is likely that you want to interact with that party using an orchestration. Parties interact with orchestrations through roles. During development, you do not need to specify the actual party that the orchestration may interact with; you can select the party later in BizTalk Explorer so that you can easily add and remove parties without redeploying your solution.

An example of a role that an orchestration might use would be a shipper. The shipper would have one or two parties associated with it. When the orchestration needs to decide which shipping company to use to ship an item, it can compare the prices of the parties in the shipper role.

Party enlistment is the mechanism that ties a party to a role. In BizTalk Explorer, you can enlist a party in a role, and that will enable the orchestration to interact with the party.

Service link type

A service link type characterizes the relationship between two services or orchestrations by defining the part played by each of the services in the relationship and specifying the port types provided by each role.

Partner self-service

Large enterprises that have large numbers of partners and a complex business process may allow their partners to access, view, and modify the configuration of their role. For example, suppliers should be able to connect to the Trading Partner Management database of the enterprise and modify their own individual configuration (such as URLs and so on.)

Aliases

Aliases refer to the same party by different names. To explain with an analogy, a person named Joe can be known in many ways. Here are a few possibilities:

- His friends may know him by his friendly name, *Joe*.
- His local video rental store may know him by his telephone number, *1-425-555-0123*
- His insurance company may know him by his address, *1234 Main Street*.

All of the earlier examples are aliases for Joe.

Similarly, an organization or organizational unit may be known by its name, by a telephone number, by an e-mail alias, by a signature certificate, or by a DUNS Number (used in some protocols), depending on who is referring to it and what protocol is being used.

BizTalk Server 2006 provides the ability to store multiple aliases for a party. Every party is given a single default alias with a name of **Organization**, with a qualifier as **OrganizationName** and value of the name of the party. This alias is always treated as the default alias. Aliases are provided as triads of name, qualifier, and value. No two parties in the same BizTalk Management database can have the same qualifier value pair. The alias name is used merely for convenience.

Three fields are needed in a BizTalk Server 2006 alias, as opposed to two to support certain BizTalk Server 2000 and BizTalk Server 2002 EDI scenarios.

Default alias name qualifier pairs are provided for standard aliases.

The following list shows the supported alias name qualifier pairs. This list can also be extended.

Alias Name	Alias Qualifier
D-U-N-S (Dun & Bradstreet)	1
Federal Maritime Commission	3

IATA (International Air Transport Association)	4
INSEE (Institut National de la Statistique et des Etudes Economiques)	5
UCC Communications ID (Uniform Code Council Communications)	8
D-U-N-S (Dun & Bradstreet) with 4-digit suffix	9
Department of Defense	10
Drug Enforcement Administration	11
Telephone Number	12
UCS Code	13
EAN (European Article Numbering Association)	14
D&B D-U-N-S Number plus 4-character suffix	16
American Bankers Association	17
AIAG (Automotive Industry Action Group)	18
Health Industry Number	20
Health Care Financing Administration Carrier ID	27
Health Care Financing Administration Fiscal Intermediary	28
Health Care Financing Administration Medicare Provider	29
ISO 6523: Organization Identification	30
DIN (Deutsches Institut fuer Normung)	31
U.S. Federal Employer Identification Number	32
BfA (Bundesversicherungsanstalt fuer Angestellte)	33
National Statistical Agency	34
GEIS (General Electric Information Services)	51

INS (IBM Network Services)	52
Bundesverband der Deutschen Baustoffhaendler	54
Bank Identifier Code	55
KTNet (Korea Trade Network Services)	57
UPU (Universal Postal Union)	58
ODETTE (Organization for Data Exchange through Tele-Transmission in Europe)	59
SCAC (Standard Carrier Alpha Code)	61
ECA (Electronic Commerce Australia)	63
TELEBOX 400 (Deutsche Telekom)	65
NHS (National Health Service)	80
Statens Teleforvaltning	82
Athens Chamber of Commerce	84
Swiss Chamber of Commerce	85
US Council for International Business	86
National Federation of Chambers of Commerce and Industry	87
Association of British Chambers of Commerce	89
SITA (Societe Internationale de Telecommunications)	90
Assigned by seller or seller's agent	91
Assigned by buyer or buyer's agent	92
National Retail Merchants Association	NR
Mutually defined	ZZZ

The example companies, organizations, products, domain names, e-mail addresses, logos, people, places, and events depicted herein are fictitious. No association with any real company, organization, product, domain name, e-mail address, logo, person, places, or events is intended or should be inferred.

Implementing partner management

The steps involved in partner management can be implemented in more than one way in BizTalk Server. The following table lists the linear steps involved in partner management, indicates which of the tools can be used to complete the task, and provides links to more information about completing the task with the chosen tool.

For more information about determining which tool to use to complete a task, see [Determining the Right Tool For Your Task](#).

Task	Orchestration Designer	BizTalk Administration Console	BizTalk Explorer	BizTalk Model Explorer	Object WMI
Create an orchestration in the Orchestration Designer	Creating and Modifying Orchestrations	None	None	None	None
Create role links in the orchestration in Orchestration Designer	Using Role Links	None	None	None	None
Create a receive port	None	None	How to Add a Receive Port Using BizTalk Explorer	Example on ReceivePort Class	MSBTS_ReceivePort Class
Create a receive location	None	None	How to Add a Receive Location Using BizTalk Explorer	Example on ReceiveLocation Class	MSBTS_ReceiveLocation Class
Bind the orchestration	None	None	How to Bind an Orchestration Using BizTalk Explorer	BindOrchestration Sample on Orchestration Binding Using the BizTalk Explorer Object Model BtsOrchestration.Status Property	None

Create a party	None	None	How to Add a Party Using BizTalk Explorer	CreateParty Sample on Partner Management Using the BizTalk Explorer Object Model from Managed Code Party Class	None
Add an alias to the party	None	None	How to Add an Alias to a Party Using BizTalk Explorer	CreateParty Sample on Partner Management Using the BizTalk Explorer Object Model from Managed Code Party.AddNewAlias Method	None
Enlist the party into the role	None	None	Enlisting a Party Using BizTalk Explorer	EnlistParty Sample on Partner Management Using the BizTalk Explorer Object Model from Managed Code Role.AddNewEnlistedParty Method	

Content-Based Routing in BizTalk Explorer

A typical BizTalk Server business process involves receiving, processing, and sending messages. At times, you may receive certain types of messages, such as partner-to-partner correspondence, which do not require intensive processing in an orchestration, and could benefit from a simpler and more efficient solution. Content-based routing is the solution to this scenario.

Content-based routing eliminates the need for message subscription for messages that are deterministically bound to specific ports, and provides additional flexibility for users who want to route messages based on envelope properties or simply based on receive port configuration properties.

Content-based routing provides backward compatibility with the PassThroughScenario that was supported and used extensively in the previous BizTalk Server releases. For BizTalk Server 2006, the dynamic routing of content is implemented by creating filters for specified properties on send ports and send port groups. Note that dynamic routing is based on properties of the document, not necessarily the content of the document. Routing can be performed based on information contained in the envelope of the document or even configuration information from the receive location.

One common scenario involving the need for content-based routing is when you have created a receive port that will only be receiving certain types of documents that should be routed to a particular send port or send port group. In this case, you can create a filter on the send port that checks the receive location specified in the envelope, and if it matches the filter expression, it is routed to the send port without ever having to go through an orchestration.

When a filter is associated with a send port or send port group, all of the conditions in the filter expression must be met for the message to be passed to the port. Each filter expression must contain at least one condition, and each condition is made up of a property from the message context, an operator, such as == or !=, and a value.

It is important to note that filters set for content-based routing operate independently from filters used with orchestrations. Orchestrations still receive messages that do not comply with the filters in the send ports to which they are bound.

If no filter is associated with a send port or a send port group, the port only receives messages from the bound orchestrations.

For more information about the message context properties that you can promote for content-based routing, see **Message Context Properties**.

Implementing content-based routing

The steps involved in content-based routing can be implemented in more than one way in BizTalk Server. The following table lists the linear steps involved in content-based routing, indicates which of the tools can be used to complete the task, and provides links to more information about completing the task with the chosen tool.

For more information about determining which tool to use to complete a task, see [Determining the Right Tool For Your Task](#).

Task	BizTalk Administration Console	BizTalk Explorer	BizTalk Explorer Object Model	WMI
Create a send port	None	How to Add a Static Send Port Using BizTalk Explorer	Example code: SendPort Class	MSBTS_SendPort Class
Add a filter to the send port	None	How to Add a Filter to a Send Port for Content-Based Routing Using BizTalk Explorer	SendPort.Filter Property	MSBTS_SendPort.Filter Property
Optionally, add a map to the send port	None	How to Add Maps to Send Ports for Outbound Normalization Using BizTalk	SendPort.InboundTransforms Property SendPort.OutboundTransforms Property	MSBTS_SendPort.InboundTransforms Property MSBTS_SendPort.OutboundTransforms Property

Document Normalization in BizTalk Explorer

In BizTalk Server 2002, you could associate an XSLT map with a channel in Messaging Manager. In BizTalk Server 2006, maps can be associated with send ports and receive ports. In addition, maps can be executed orchestrations.

Maps associated with one-way send ports are executed before the send pipeline is run. The output of the map is then passed to the send pipeline.

Two-way send ports can have two maps—a map for the outgoing request message and a map for the incoming response message.

Maps associated with one-way receive ports are run after the receive pipeline is run. The output of the map is then passed to the BizTalk MessageBox database.

Two-way receive ports can have two maps—a map for the incoming request messages and a map for the outgoing response messages.

Orchestration Management in BizTalk Explorer

Orchestrations are the entities used by BizTalk Server 2006 to bring business processes to life. These orchestrations may interact with thousands of endpoints or ports, which can be a difficult task to manage. BizTalk Explorer allows you to manage these tasks using a tool in the Visual Studio .NET development environment. Or, for automated management of your orchestrations, you can use the BizTalk Explorer Object Model to write scripts to bind orchestrations to ports, enlist, unenlist, and start and stop orchestrations.

The following is a list of the different types of orchestration ports:

- **Receive ports.** Receives messages into the orchestration.
- **Send ports.** Sends messages from the orchestration to external locations.
- **Request-response receive ports.** Receives messages into the orchestration and returns a response to the caller.
- **Solicit-response send ports.** Sends messages and waits for a response.

Another way to classify orchestration ports is:

- **Direct orchestration ports.** Specifies the location at design time.
- **Dynamic orchestration ports.** Receives the location information at run time. Only send ports can be dynamic, because we cannot receive a receive location address in a message (because we would not know where to receive it).

- **Web service ports.** Some ports can bind to a Web service, while other ports are exposed as Web services.

Direct orchestration ports can be configured with the following binding types:

- **Specify Later.** In Orchestration Designer, the user specifies that the pipeline and endpoint will be made available in BizTalk Explorer through the process of Orchestration binding. These ports show up as candidates for binding during Orchestration binding in Explorer.
- **Specify Now.** The user in Orchestration Designer specifies that the port will receive and send messages based on certain criteria specified in Orchestration Designer. These ports also do not show up for binding in Explorer.

Orchestration States in BizTalk Explorer

Enlisted. Enlisting is the process of associating an orchestration with the physical environment in which it will run including the adapters needed to transport messages to and from the orchestration, the application process in which the orchestration is hosted, and creating the MessageBox subscriptions indicated by the routing. After the orchestration is enlisted, it can be started.

Started. The subscriptions have been activated.

Stopped. The subscriptions have been stopped.

Unenlisted. The orchestration has no subscriptions, send ports, or receive locations associated with it.

Orchestration Binding

The different kinds of Orchestration ports bind to the different kinds of BizTalk Explorer ports during Orchestration binding as per the following rules:

- Orchestration receive ports can bind only to BizTalk Explorer receive ports (one-way or request response).
- Orchestration send ports can bind to BizTalk Explorer send ports or BizTalk Explorer send port groups.
- Orchestration two-way (receive or send) ports can bind only to two-way send ports. Similarly, one-way orchestration ports can only bind to one-way BizTalk Explorer ports. This is to fulfill a run-time requirement, wherein the runtime looked at the BizTalk Explorer port to decide whether the transport needs to behave as a two-way or a one-way transport. Therefore, a send port group can have only one-way send ports in it.
- Orchestration receive ports that have no request-response operations can bind only to one-way BizTalk Explorer receive ports.
- Orchestration receive ports that have one or more request-response operations will show only request-response receive ports to bind to in BizTalk Explorer. For one-way operations on that orchestration port, the send pipeline is ignored.

- Orchestration send ports that have no solicit-response operations can bind only to a one-way send port or send port group (which can contain only one-way send ports). However, the receive pipeline configured for two-way operation on any of these BizTalk Explorer send ports is ignored in this scenario.
- Orchestration send ports that have one or more solicit-response operations will show only solicit-response send ports to bind in BizTalk Explorer. Such ports cannot bind to a send port group.
- Static orchestration send ports will display only static BizTalk Explorer send ports to bind to.
- Dynamic orchestration send ports will display only dynamic BizTalk Explorer send ports to bind to.
- Physically bound orchestration ports create BizTalk Explorer ports after being deployed. These ports can be used in BizTalk Explorer for all bindings just as if they were created in BizTalk Explorer.

Implementing orchestration management

The steps involved in orchestration management can be implemented in more than one way in BizTalk Server. The following table lists the linear steps involved in orchestration management, indicates which of the tools can be used to complete the task, and provides links to more information about completing the task with the chosen tool.

For more information about determining which tool to use to complete a task, see [Determining the Right Tool For Your Task](#).

Task	Tool	BizTalk Administration Console	BizTalk Explorer	BizTalk Explorer Object Model	WMI
Create an orchestration	Orchestration Designer	None	None	None	None
Add logical ports to your orchestration	Orchestration Designer	None	None	None	None
Deploy the Assembly	How to Deploy a BizTalk Assembly from Visual Studio	None	None	None	MSBTS_DeploymentService Method
Create a send port*	None	None	How to Add a Static Send Port Using BizTalk Explorer	Example on SendPort Class	MSBTS_SendPort Class

Create a receive port*	None	None	How to Add a Receive Port Using BizTalk Explorer	Example on ReceivePort Class	MSBTS_ReceivePort Class
Create a receive location*	None	None	How to Add a Receive Location Using BizTalk Explorer	Example on ReceiveLocation Class	MSBTS_ReceiveLocation Class
Bind the receive locations and send ports to the orchestration*	None	None	How to Bind an Orchestration Using BizTalk Explorer	BindOrchestration Sample on Orchestration Binding Using the BizTalk Explorer Object Model BtsOrchestration.Status Property	None
If stopped, start the host service	None	How to Start a Host Instance	None	None	Example on Enumerating Starting All Host Instances WMI MSBTS_Host.Start Method
Enable the receive location	How to Enable a Receive Location	How to Enable a Receive Location	How to Enable and Disable a Receive Location Using BizTalk Explorer	ReceiveLocation.Enable Property	MSBTS_ReceiveLocation.Enable Method
Enlist the orchestration	None	How to Enlist an Orchestration	How to Enlist and Start an Orchestration Using BizTalk Explorer	EnlistOrchestration Sample on Orchestration Binding Using the BizTalk Explorer Object Model BtsOrchestration.Status Property	MSBTS_Orchestration.Enlist Method
Start the orchestration	None	How to Start an Orchestration	How to Enlist and Start an Orchestration Using BizTalk Explorer	StartOrchestration Sample on Orchestration Binding Using the BizTalk Explorer Object Model BtsOrchestration.Status Property	MSBTS_Orchestration.Start Method

*These steps only need to occur if you selected the **Specify Later** option when creating ports in your orchestration. If you created a port using the **Specify Now** option, you do not need to perform these operations as the port was already configured in the orchestration. For more information.

BizTalk Explorer Automation

BizTalk Explorer automation is enabled by the BizTalk Explorer Object Model, a collection of classes and interfaces in the **Microsoft.BizTalk.ExplorerOM** namespace. You can use the BizTalk Explorer Object Model to create tools and scripts to automate the post-deployment tasks that you perform in BizTalk Explorer. You can use the BizTalk Explorer Object Model for such post-deployment tasks as creating ports, binding orchestrations, managing party properties, or any other task where you would use BizTalk Explorer. Automating these tasks can be particularly useful if you are working with a large number of receive locations, send ports, parties, or orchestrations.

Many large and medium scale organizations have business policies that prohibit using development tools, such as Visual Studio in deployment or production environments. Because BizTalk Explorer is a Microsoft Visual Studio .NET tool window, you must have Visual Studio .NET installed to use it. The BizTalk Explorer Object Model can be used in any environment where Microsoft BizTalk Server 2006 is installed, which means that you can use it when you cannot use BizTalk Explorer.

Samples containing the most common tasks for BizTalk Explorer are included in the documentation. For more information, see Automating Business Process Management.

Planning for BizTalk Explorer

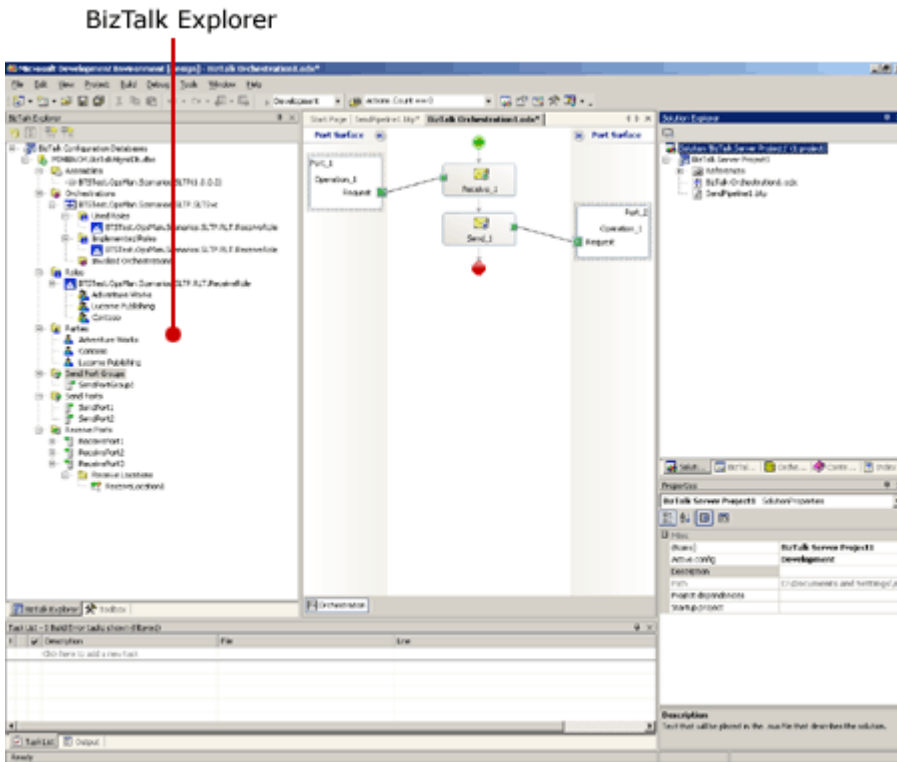
You use the BizTalk® Explorer to configure and manage Microsoft® BizTalk Server artifacts such as orchestrations, send ports, and receive locations.

The following table lists a question that you need to answer in planning for BizTalk Explorer artifacts.

Planning question	Recommendation
Is BizTalk Explorer the correct tool to use to complete your task?	To determine whether the BizTalk Administration console, the BizTalk Explorer Object Model, or Windows Management Instrumentation (WMI) may be better suited to you and your task.

How to Open BizTalk Explorer

Illustrates BizTalk Explorer in the Visual Studio .NET environment.



To open BizTalk Explorer

1. In Visual Studio 2005, in a BizTalk Server project, on the **View** menu, click **BizTalk Explorer**.

For more information about each of the artifacts represented in BizTalk Explorer, see About BizTalk Explorer Artifacts.

Determining the Right Tool For Your Task

Microsoft® BizTalk® Server artifacts can be managed through a user interface or through script. If you prefer to use a user interface, BizTalk Explorer or BizTalk Administration console are your options. These two tools share much of the same functionality but are targeted at different users. BizTalk Explorer is only available through Microsoft® Visual Studio® .NET 2003, which makes it convenient for developers to configure and manage artifacts while working on a project. BizTalk Administration console is a Microsoft Management Console (MMC) snap-in, so it is more convenient for IT Professionals and administrators who are already familiar with MMC and do not have Visual Studio installed on their computer.

If you prefer to manage your BizTalk Server artifacts through code and scripts, you also have two options, the BizTalk Explorer Object Model and Windows Management Instrumentation (WMI). The BizTalk Server Explorer Object Model is a set of classes and interfaces in the **Microsoft.BizTalk.ExplorerOM** namespace. The Help refers to the WMI classes as the core server classes in BizTalk Server. Both the

BizTalk Server Explorer Object Model and the core server classes have API reference pages and samples provided in the documentation and the BizTalk Server SDK.

The following tables list common artifact management tasks and indicate which of the tools can be used to complete the task, as well as links to more information about completing the task with the chosen tool.

For similar tables that outline the tasks involved in managing parties, content-based routing, and managing orchestrations see the following topics: Partner Management in BizTalk Explorer, Content-Based Routing in BizTalk Explorer, and Orchestration Management in BizTalk Explorer.

Assembly tasks

Task	BizTalk Administration Console	BizTalk Explorer	BizTalk Explorer Object Model	WMI
Undeploying an assembly	None	How to Undeploy an Assembly Using BizTalk Explorer	None	MSBTS_DeploymentService.Remove Method

Send port tasks

Task	BizTalk Administration Console	BizTalk Explorer	BizTalk Explorer Object Model	WMI
Add a send port	None	How to Add a Static Send Port Using BizTalk Explorer	Example on SendPort Class	MSBTS_SendPort Class
Start a send port	How to Start a Send Port or Send Port Group	How to Enlist and Start a Send Port Using BizTalk Explorer	Example on SendPort Class SendPort.Status Property	MSBTS_SendPort.Start Method
Edit a send port properties	None	How to Edit Send Port Properties Using BizTalk Explorer	Example on SendPort Class	MSBTS_SendPort Class
Manage send port encryption certificates	None	How to Manage Send Port Encryption Certificates Using BizTalk	Example on SendPort Class SendPort.EncryptionCert Property	MSBTS_SendPort.EncryptionCert Property

		Explorer		
Add a filter to a send port for content-based routing	None	How to Add a Filter to a Send Port for Content-Based Routing Using BizTalk Explorer	CBRSample on Content-Based Routing Using the BizTalk Explorer Object Model Example on SendPort Class SendPort.Filter Property	MSBTS_SendPort.Filter Property
Add a filter to a send port for a messaging only, or pass-through scenario.	None	How to Add a Filter to a Send Port for Passthrough Using BizTalk Explorer	SendPort.Filter Property	MSBTS_SendPort.Filter Property
Add a map to a two-way send port for outbound normalization	None	How to Add Maps to Send Ports for Outbound Normalization Using BizTalk Explorer	CBRSample on Content-Based Routing Using the BizTalk Explorer Object Model Example on SendPort Class SendPort.OutboundTransforms Property	SendPort.OutboundTransforms Property
Add a map to a send port for inbound normalization	None	How to Add Maps to Send Ports for Inbound Normalization Using BizTalk Explorer	SendPort.InboundTransforms Property	SendPort.InboundTransforms Property
Enlist a send port	How to Enlist a Send Port or Send Port Group	How to Enlist and Start a Send Port Using BizTalk Explorer	Example on SendPort Class	MSBTS_SendPort.Enlist Method
Stop a send port	How to Stop a Send Port or Send Port Group	How to Stop and Unenlist a Send Port Using BizTalk Explorer	Example on SendPort Class	MSBTS_SendPort.Stop Method

Unenlist send port	a	How to Unenlist a Send Port or Send Port Group	How to Stop and Unenlist a Send Port Using BizTalk Explorer	Example on SendPort Class	MSBTS_SendPort.UnEnlist Method
Delete send port	a	None	How to Delete a Send Port Using BizTalk Explorer	Example on SendPort Class	MSBTS_SendPort Class

Send port group tasks

Task	BizTalk Administration Console	BizTalk Explorer	BizTalk Explorer Object Model	WMI
Add a send port group	None	How to Add a Send Port Group Using BizTalk Explorer	Example on SendPortGroup Class BtsCatalogExplorer.AddNewSendPortGroup Method	MSBTS_SendPortGroup Class
Add a send port to a send port group	None	How to Add and Delete Send Ports from a Send Port Group Using BizTalk Explorer	Example on SendPortGroup Class BtsCatalogExplorer.AddNewSendPort Method	MSBTS_SendPortGroup2SendProperty
Delete a send port from send port group	None	How to Add and Delete Send Ports from a Send Port Group Using BizTalk	Example on SendPortGroup Class	MSBTS_SendPortGroup2Send

		Explorer		
Add a filter to a send port group	None	How to Add a Filter to a Send Port Group Using BizTalk Explorer	SendPortgroup.Filter Property	MSBTS_SendPortGroup.Filter
Enlist a send port group	How to Enlist a Send Port or Send Port Group	How to Enlist and Start a Send Port Group Using BizTalk Explorer	SendPortGroup.Status Property	MSBTS_SendPortGroup.Enlist
Start a send port group	How to Start a Send Port or Send Port Group	How to Enlist and Start a Send Port Group Using BizTalk Explorer	Example on SendPortGroup Class SendPortGroup.Status Property	MSBTS_SendPortGroup.Start
Stop a send port group	How to Stop a Send Port or Send Port Group	How to Stop and Unenlist a Send Port Group Using BizTalk Explorer	Example on SendPortGroup Class SendPortGroup.Status Property	MSBTS_SendPortGroup.Stop
Unenlist a send port group	How to Unenlist a Send Port or Send Port Group	How to Stop and Unenlist a Send Port Group Using BizTalk	SendPortGroup.Status Property	MSBTS_SendPortGroup.UnEn

		Explorer		
Delete a send port group	None	How to Delete a Send Port Group Using BizTalk Explorer	Example on SendPortGroup Class	MSBTS_SendPortGroup Class

Receive port tasks

Task	BizTalk Administration Console	BizTalk Explorer	BizTalk Explorer Object Model	WMI
Add a receive port	None	How to Add a Receive Port Using BizTalk Explorer	Example on ReceivePort Class	MSBTS_ReceivePort Class
Edit receive port properties	None	How to Edit Receive Port Properties Using BizTalk Explorer	Example on ReceivePort Class ReceivePort Properties	MSBTS_ReceivePort Class
Add a map to a two-way receive port for inbound normalization	None	How to Add Maps to Receive Ports for Inbound Normalization Using BizTalk Explorer	ReceivePort.InboundTransforms Property	MSBTS_ReceivePort.InboundTransforms Property
Add a map to a receive port for outbound normalization	None	How to Add Maps to Receive Ports for Outbound Normalization Using BizTalk Explorer	ReceivePort.OutboundTransforms Property	MSBTS_ReceivePort.OutboundTransforms Property
Delete a receive port	None	How to Delete a Receive Port Using BizTalk Explorer	Example on ReceivePort Class	MSBTS_ReceivePort Class

Receive location tasks

Task	BizTalk Administration Console	BizTalk Explorer	BizTalk Model Explorer	Object	WMI
Add a receive location	None	How to Add a Receive Location Using BizTalk Explorer	Example ReceiveLocation Class	on	MSBTS_ReceiveLocation Class
Edit receive location properties	How to Configure Scheduling for a Receive Location	How to Edit a Receive Location Adapter Properties Using BizTalk Explorer	Example ReceiveLocation Class ReceiveLocation Properties	on	MSBTS_ReceiveLocation Class
Enable a receive location	How to Enable a Receive Location	How to Enable and Disable a Receive Location Using BizTalk Explorer	Example ReceiveLocation Class ReceiveLocation.Enable Property	on	MSBTS_ReceiveLocation.Enable Method
Disable a receive location	How to Disable a Receive Location	How to Enable and Disable a Receive Location Using BizTalk Explorer	ReceiveLocation.Enable Property		MSBTS_ReceiveLocation.Disable Method
Specify a primary receive location	None	How to Specify a Primary Receive Location Using BizTalk	Example ReceiveLocation Class ReceiveLocation.IsPrimary Property	on	MSBTS_ReceiveLocation.IsPrimary Property

		Explorer		
Delete a receive location	None	How to Delete a Receive Location Using BizTalk Explorer	Example on ReceiveLocation Class	MSBTS_ReceiveLocation Class

Orchestration tasks

Task	BizTalk Administration Console	BizTalk Explorer	BizTalk Explorer Object Model	WMI
Browse orchestration artifacts	Using the BizTalk Server Administration Console	Managing Orchestrations Using BizTalk Explorer	EnumerateOrchestrationArtifacts Sample and EnumerateArtifactSample Sample on Browsing BizTalk Server Assembly Artifacts Using the BizTalk Explorer Object Model	MSBTS_Orchestration
Find roles used by an orchestration	None	How to Browse Used Roles of an Orchestration Using BizTalk Explorer	EnumerateOrchestrationArtifacts Sample on Browsing BizTalk Server Assembly Artifacts Using the BizTalk Explorer Object Model BtsOrchestration.UsedRoles Property	None
Find roles implemented by an orchestration	None	How to Browse Implemented Roles of an Orchestration Using BizTalk Explorer	EnumerateOrchestrationArtifacts Sample on Browsing BizTalk Server Assembly Artifacts Using the BizTalk Explorer Object Model BtsOrchestration.ImplementedRoles Property	None
Bind an orchestration	None	How to Bind an Orchestration Using BizTalk Explorer	BindOrchestration Sample on Orchestration Binding Using the BizTalk Explorer Object Model BtsOrchestration.Status Property	MSBTS_DeploymentS Method
Enlist an orchestration	How to Enlist an Orchestration	How to Enlist and Start an Orchestration Using BizTalk Explorer	EnlistOrchestration Sample on Orchestration Binding Using the BizTalk Explorer Object Model BtsOrchestration.Status Property	MSBTS_Orchestration Method
Start an	How to Start an	How to Enlist and Start an	StartOrchestration Sample on Orchestration Binding Using the BizTalk Explorer Object	MSBTS_Orchestration

orchestration	Orchestration	Orchestration Using BizTalk Explorer	Model BtsOrchestration.Status Property	Method
Stop an orchestration	How to Stop an Orchestration	How to Stop and Unenlist an Orchestration Using BizTalk Explorer	StopOrchestration Sample on Orchestration Binding Using the BizTalk Explorer Object Model BtsOrchestration.Status Property	MSBTS_Orchstration.S
Unenlist an orchestration	How to Unenlist an Orchestration	How to Stop and Unenlist an Orchestration Using BizTalk Explorer	UnenlistOrchestration Sample on Orchestration Binding Using the BizTalk Explorer Object Model BtsOrchestration.Status Property	MSBTS_Orchstration.U Method
Unbind an orchestration	None	None	UnbindOrchestration Sample on Orchestration Binding Using the BizTalk Explorer Object Model BtsOrchestration.Status Property	MSBTS_DeploymentS Method
View invoked orchestrations	None	How to Browse Invoked Orchestrations Using BizTalk Explorer	BtsOrchestration.InvokedOrchestrations Property	None

Party tasks

Task	BizTalk Administration Console	BizTalk Explorer	BizTalk Explorer Object Model	WMI
Add a party	None	How to Add a Party Using BizTalk Explorer	CreateParty Sample on Partner Management Using the BizTalk Explorer Object Model from Managed Code Party Class	None
Add a send port to a party	None	How to Manage Party Send Ports Using BizTalk Explorer	CreateParty Sample on Partner Management Using the BizTalk Explorer Object Model from Managed Code Party.SendPorts Property	None

Delete a send port from a party	None	How to Manage Party Send Ports Using BizTalk Explorer	BtsOrchestration.Status Property	None
Add an alias to a party	None	How to Add an Alias to a Party Using BizTalk Explorer	CreateParty Sample on Partner Management Using the BizTalk Explorer Object Model from Managed Code Party.AddNewAlias Method	None
Select a signature certificate for a party	None	How to Manage Party Signature Certificates Using BizTalk Explorer	CreateParty Sample on Partner Management Using the BizTalk Explorer Object Model from Managed Code Party.SignatureCert Property	None
Delete a party	None	How to Delete a Party Using BizTalk Explorer	DeleteParty Sample on Partner Management Using the BizTalk Explorer Object Model from Managed Code Party Class	None
Enlist a party	None	Enlisting a Party Using BizTalk Explorer	EnlistParty Sample on Partner Management Using the BizTalk Explorer Object Model from Managed Code Role.AddNewEnlistedParty Method	None
Unenlist party	None	How to Unenlist Parties Using BizTalk Explorer	UnenlistParty Sample on Partner Management Using the BizTalk Explorer Object Model from Managed Code Role.RemoveEnlistedParty Method	None

Host tasks

Task	BizTalk Administration Console	BizTalk Explorer	BizTalk Explorer Object Model	WMI
Creating a new host	How to Create a New Host	None	None	MSBTS_Host Class
Modifying host properties	How to Modify Host Properties	None	None	MSBTS_Host Class
Delete a host	How to Delete a Host	None	None	MSBTS_Host Class

Starting a host instance	How to Start a Host Instance	None	None	MSBTS_HostInstance.Start Method
Stopping a host instance	How to Stop a Host Instance	None	None	MSBTS_HostInstance.Stop Method
Deleting a host instance	How to Delete a Host Instance	None	None	MSBTS_HostInstance Class
Modifying the properties of a host instance	How to Modify Host Instance Properties	None	None	MSBTS_HostInstance Class

MessageBox database tasks

Task	BizTalk Administration Console	BizTalk Explorer	BizTalk Explorer Object Model	WMI
Adding a new MessageBox database	How to Add a New MessageBox Database	None	None	MSBTS_MsgBoxSetting Class
Configuring properties for a MessageBox database	Managing MessageBox Databases	None	None	MSBTS_MsgBoxSetting Class
Deleting a MessageBox database from a BizTalk group	How to Delete a MessageBox Database	None	None	MSBTS_MsgBoxSetting Class

How to Add or Remove BizTalk Management Databases Using BizTalk Explorer

You must add a BizTalk Management database to BizTalk® Explorer before you can use BizTalk Explorer to work with the projects deployed in the BizTalk Management database. The same BizTalk Management database can be opened from multiple instances of BizTalk Explorer running on the same or different computers. You can also remove a BizTalk Management database from BizTalk Explorer.

To add a BizTalk Management database

1. In Microsoft® Visual Studio® 2005, in a BizTalk Server project, on the **View** menu, click **BizTalk Explorer**.
2. In BizTalk Explorer, right-click the **BizTalk Configuration Databases** node, and then click **Add database**.
3. In the **Connect to BizTalk Configuration Database** dialog box, do the following.

Use this	To do this
SQL Server	Specify the SQL Server that hosts the BizTalk Management database.
Database	Type the name of the BizTalk Management database that you want.

- Click **OK**.

The **BizTalk Configuration database** node under **BizTalk Configuration Databases** with the name in the form of *<ServerName>.<DBName>.<UserName>* appears in BizTalk Explorer.

To remove a BizTalk Management database

- In Visual Studio 2005, in a BizTalk Server project, on the **View** menu, click **BizTalk Explorer**.
- Expand the **BizTalk Configuration Databases** node, right-click the database node that you want to remove, and then click **Remove**.
- In the **BizTalk Server** dialog box, click **Yes**.

The **BizTalk Configuration database** node under **BizTalk Configuration Databases** is removed.

Managing Assemblies Using BizTalk Explorer

You can use BizTalk® Explorer to manage which versions of an assembly are deployed to a BizTalk Management database by deleting assemblies that should no longer be deployed.

In This Section

- Browsing Assemblies Using BizTalk Explorer
- Undeploying an Assembly Using BizTalk Explorer

How to Browse Assemblies Using BizTalk Explorer

Assemblies deployed to any BizTalk BizTalk Management database can be viewed from within BizTalk Explorer.

To browse assemblies

- In Microsoft® Visual Studio® .NET, in a Microsoft® BizTalk Server project, on the **View** menu, click **BizTalk Explorer**.
- In BizTalk Explorer, expand the **Assemblies** collection.

From the **Assemblies** collection, you can undeploy the assembly and review its read-only properties. For information about undeploying the assembly, see Undeploying an Assembly Using BizTalk Explorer.

To review the read-only properties for an assembly

1. In Visual Studio .NET, in a BizTalk Server project, on the **View** menu, click **BizTalk Explorer**.
2. In BizTalk Explorer, expand the **Assemblies** collection, right-click the assembly, and then click **Properties**.

The read-only properties are displayed in the Properties window in Visual Studio .NET.

How to Undeploy an Assembly Using BizTalk Explorer

Each **Assembly** node in BizTalk Explorer represents each version of the assembly in the BizTalk Management database. You can use the **Undeploy** command in BizTalk Explorer to remove an assembly that you do not want deployed. This action removes the selected assembly from the BizTalk Management database and the global assembly cache (GAC).

To undeploy an assembly using BizTalk Explorer

1. In Visual Studio .NET, in a BizTalk Server project, on the **View** menu, click **BizTalk Explorer**.
2. In BizTalk Explorer, expand the **Assemblies** collection, right-click the assembly that you do not want deployed, and then click **Undeploy**.

The assembly is removed from the BizTalk Management database and the GAC.

Managing Send Ports Using BizTalk Explorer

You can use BizTalk® Explorer to do the following:

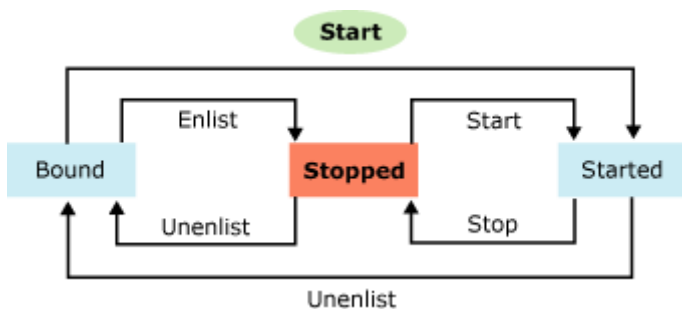
- Add send ports.
- Add one-way static send ports to send port groups.
- Add static one-way or solicit-response send ports.
- Add dynamic one-way or solicit-response send ports.
- Associate encryption certificates with a send port.
- Associate a filter with a send port to create a subscription that routes messages directly to the port without using an orchestration.
- Associate maps with send ports to transform messages without using an orchestration.

The following table lists all of the states and actions available for a send port.

States	Actions
Bound	Enlist
Started	Start
Stopped	Unenlist
Not applicable	Stop

The following illustration shows how you can change the states of the send port by invoking actions.

Illustrates the states and actions of send ports.



In This Section

- Adding a Static Send Port Using BizTalk Explorer
- Adding a Dynamic Send Port Using BizTalk Explorer
- Editing Send Port Properties Using BizTalk Explorer
- Editing Send Port Adapter Properties Using BizTalk Explorer
- Managing Send Port Encryption Certificates Using BizTalk Explorer
- Adding a Filter to a Send Port for Content-Based Routing Using BizTalk Explorer
- Adding a Filter to a Send Port for Passthrough Using BizTalk Explorer
- Adding Maps to Send Ports for Outbound Normalization Using BizTalk Explorer
- Adding Maps to Send Ports for Inbound Normalization Using BizTalk Explorer
- Enlisting and Starting a Send Port Using BizTalk Explorer
- Stopping and Unenlisting a Send Port Using BizTalk Explorer
- Deleting a Send Port Using BizTalk Explorer

How to Add a Static Send Port Using BizTalk Explorer

You can use BizTalk Explorer to add both static and dynamic send ports to a BizTalk Management database. When you add a send port, you are required to select an installed adapter.

You can create static send ports within Orchestration Designer or by using BizTalk Explorer. You specify the adapter type, destination address (URI), and pipeline to use when you create a static send port.

Dynamic send ports do not contain a fixed destination address, only a pipeline. The destination address is determined at run time from a specified property in the message.

The following procedure explains how to add a static send port.

To add a static send port

1. In Microsoft® Visual Studio® 2005, in a Microsoft® BizTalk® Server project, on the **View** menu, click **BizTalk Explorer**.
2. In BizTalk Explorer, right-click the **Send Ports** collection, and then click **Add Send Port**.
3. In the **Create New Send Port** dialog box, select one of the following port types from the drop-down list, and then click **OK**:
 - Static One-Way Port
 - Static Request-Response Port
4. In the **Send Port Properties** dialog box, in the **Name** box, type the name of the send port.
5. For static send ports, do the following.

In the **Send Port Properties** dialog box, expand the **Transport** node, and then select the **Primary** node.

In the right pane, do the following.

Use this	To do this
Transport Type	Select the transport type from the drop-down list. If you select a solicit-response port, only adapters that support solicit-response will be available in the drop-down list. If you create a SOAP adapter, the only available pipeline is PassThruTransmit, to prevent you from modifying the SOAP message.
Address (URI)	Click the ellipsis (...) button to configure the Adapter properties for the send port. For information about configuring the Address property specific to the type of adapter you are creating, see one of the following links:

	<ul style="list-style-type: none"> • How to Configure a Base EDI Send Port • How to Configure an MSMQT Send Port • How to Configure a File Send Port • How to Configure an FTP Send Port • Configuring an HTTP Send Port By Using BizTalk Administrator • How to Configure MQSeries Adapter Receive Locations and Send Ports • How to Configure an MSMQ Send Port • How to Configure an SMTP Send Port • How to Configure a SOAP Send Port • How to Configure a SQL Send Port • How to Configure a Windows SharePoint Services Send Port
Retry Count	<p>Number of times the engine will try to resend if transmission fails.</p> <p>Default Value: 3</p> <p>Allowed range: 0 to 1000 inclusive</p>
Retry Interval	<p>Number of minutes the engine will wait between retries.</p> <p>Default = 5 Minutes</p> <p>Allowed range: 0 to 525600 inclusive (max value is 365 days)</p>
Ordered Delivery	<p>This property is read-only, except for the MSMQT adapter.</p> <p>To guarantee ordered delivery of messages in BizTalk Message Queuing, select True from the drop-down list. This property is False and read-only for all other transports.</p>
Enable Service Window	<p>the The Service window configures the send port to send only at specified times of the day. If disabled, this implies that the send port sends messages all the time.</p> <p>Default Value: False</p>
Start Time	<p>Indicates when the send port should start to send messages. Enabled only if the Service window is enabled, otherwise dimmed.</p>

	Time in HH:MM AM/PM format. Default Value: 12:00 AM (local time)
Stop Time	Indicates when the send port should stop sending messages. Enabled only if the Service window is enabled, otherwise dimmed. Time is in HH:MM AM/PM format. Default Value: 11:59 PM (local time)

If needed, a secondary transport can be configured by selecting **Secondary** in the right pane and configuring the properties as described for the primary transport.

6. In the **Send Port Properties** dialog box, in the left pane, select the **Send** node.
7. In the **Send Port Properties** dialog box, in the right pane, do the following.

Use this	To do this
Tracking Type	<p>To enable message tracking before receiving a message, select the Before Receive check box. This check box is available for one-way and solicit-response send ports.</p> <p>To enable message tracking after receiving a message, select the After Receive check box. This check box is available for one-way and solicit-response send ports.</p> <p>To enable message tracking before sending a message, select the Before Send check box. This check box is available only for solicit-response send ports.</p> <p>To enable message tracking after sending a message, select the After Send check box. This check box is available only for solicit-response send ports.</p>
Priority	<p>Contains the delivery priority of this port. 1 is the highest priority, 10 is the lowest priority.</p> <p>For example, if messages 1, 2, and 3 are published, and 3 goes to the port A with a priority setting of 1, while 1 and 2 go to port B with a priority setting of 10, message 3 will be delivered before messages 1 and 2.</p> <p>To avoid the prioritization of deliveries, set the Priority property to the same value on all ports.</p> <p>Default value: 5</p>
Certificate Name	Select the encryption certificate from the drop-down list, which lists all the public key certificates installed in the Local Computer\Other People store.

Long Name	Contains the long name of the certificate picked for the Certificate name property. Updated automatically. Default Value: Empty
Usage	Contains the usage of the certificate picked for the Certificate name property. Updated automatically. Default Value: Empty
Receive Pipeline	This property is only available when creating a solicit-response send port. You can pick a receive pipeline to use this port as a two-way port. You use the selected pipeline to send a response. Default Value: Empty
Send Pipeline	You can pick a send pipeline to process the messages that you send over this port. Default Value: Empty

8. To apply filters and mapping to the send port, see Adding a Filter to a Send Port for Content-Based Routing Using BizTalk Explorer, Adding a Filter to a Send Port for Passthrough Using BizTalk Explorer, or Adding Maps to Send Ports for Outbound Normalization Using BizTalk Explorer.
9. Click **OK**.

A send port is added to the **Send Ports** collection in the BizTalk Management database. You can edit the send port or add it to a send port group or party.

How to Add a Dynamic Send Port Using BizTalk Explorer

You can use BizTalk Explorer to add both static and dynamic send ports to a BizTalk Management database.

You can create static send ports within Orchestration Designer or by using BizTalk Explorer. You specify the adapter type, destination address (URI), and pipeline to use when you create a static send port.

Dynamic send ports do not contain a fixed destination address, only a pipeline. The destination address is determined at run time from the value of the **OutboundTransportLocation** message context property (in the BTS namespace). If this property is promoted in a published message, that message is eligible to be routed to a dynamic send port. If this property is not promoted, the message cannot be routed to a dynamic send port. This property can be specified within a pipeline by simply promoting a property from the message contents as the **OutboundTransportLocation** using the property promotion features of the schema editor. Alternatively, a custom component could be written to do this programmatically in the case that simple promotion does not suffice. This property can also be specified in the message context explicitly in an Orchestration by setting the **OutboundTransportLocation** property of the message which

the orchestration is about to publish. There is a second related message context property called **OutboundTransportType**. The value of this property will normally be inferred by the messaging engine based on the URL prefix in the value of the **OutboundTransportLocation** property (e.g. **mailto:** will route to the SMTP send handler, **ftp://** will route to the FTP send handler, etc.). However, an application can override this implicit send-handler selection by explicitly identifying a send handler in the value of the **OutboundTransportType** property. This is useful in cases where the protocol prefix might be ambiguous, such as with the **http://** prefix, which could be used with either HTTP or SOAP messages. In such a case, the application would indicate that the SOAP send handler should be used by setting the value of **OutboundTransportType** to SOAP. Be aware that if this property is set by the application, BizTalk Server assumes that the specified transport type is appropriate for the URL provided (i.e. it will not verify, for example, that if SOAP is specified the URL begins with **http://** and not something else, such as **ftp:**, etc.).

This procedure explains how to add a dynamic send port.

To add a dynamic send port using BizTalk Explorer

1. In Visual Studio .NET, in a BizTalk Server project, on the **View** menu, click **BizTalk Explorer**.
2. In BizTalk Explorer, right-click the **Send Ports** collection, and then click **Add Send Port**.
3. In the **Create New Send Port** dialog box, select one of the following port types from the drop-down list, and then click **OK**:
 - Dynamic One-Way Port
 - Dynamic Request-Response Port
4. In the **Send Port Properties** dialog box, in the **Name** box, type the name of the send port.
5. For dynamic send ports, do the following:
 - In the right pane, do the following.

Use this	To do this
Tracking Type	To enable message tracking before receiving a message, select the Before Receive check box. This check box is available only for request-response send ports.
	To enable message tracking after receiving a message, select the After Receive check box. This check box is available only for request-response send ports.
	To enable message tracking before sending a message, select the Before Send check box.
	To enable message tracking after sending a message, select the After Send check box.
Priority	Contains the delivery priority of this port. 1 is the highest priority, 10 is the lowest priority.
	For example, if messages 1, 2, and 3 are published, and 3 goes to the port A with a priority setting of 1, while 1 and 2 go to port B with a priority setting of 10, message 3 will be

	<p>delivered before messages 1 and 2.</p> <p>To avoid the prioritization of deliveries, set the Priority property to the same value on all ports.</p> <p>Default value: 5</p>
Certificate Name	Select the encryption certificate from the drop-down list, which lists all the public key certificates installed in the Local Computer\Other People store.
Long Name	<p>It is updated automatically with the long name of the certificate picked for the Certificate name property.</p> <p>Default Value: Empty</p>
Usage	<p>It is updated automatically with the usage of the certificate picked for the Certificate name property.</p> <p>Default Value: Empty</p>
Receive Pipeline	<p>This property is only available when creating a request-response send port.</p> <p>You can pick a receive pipeline to use this port as a two-way port. You use the selected pipeline to send a response.</p> <p>Default Value: Empty</p>
Send Pipeline	<p>You can pick a send pipeline to process the messages that you send over this send port.</p> <p>Default Value: Empty</p>

6. To apply filters and mapping to the send port, see [Adding a Filter to a Send Port for Content-Based Routing Using BizTalk Explorer](#), [Adding a Filter to a Send Port for Passthrough Using BizTalk Explorer](#), or [Adding Maps to Send Ports for Outbound Normalization Using BizTalk Explorer](#).
7. Click **OK**.

A send port is added to the **Send Ports** collection in the BizTalk Management database. You can edit the send port or add it to a send port group or party.

How to Edit Send Port Properties Using BizTalk Explorer

You can use BizTalk Explorer to edit the properties of a send port; however, you cannot change the port type that you selected when you originally added the send port. If you need a send port with a different port type, you must create a new send port.

To edit the properties of a send port

1. In Visual Studio .NET, in a BizTalk Server project, on the **View** menu, click **BizTalk Explorer**.
2. In BizTalk Explorer, expand the **Send Ports** collection, right-click the send port that contains the properties you want to edit, and then click **Edit**.
3. For static send ports, configure the adapter send port properties. For more information, see Editing Send Port Adapter Properties Using BizTalk Explorer.
4. In the **Send Port Properties** dialog box, expand the **Send** folder in the left pane, and in the **General** section, do the following.

Use this	To do this
Tracking Type	To enable message tracking before receiving a message, select the Before Receive check box. This check box is available only for request-response send ports.
	To enable message tracking after receiving a message, select the After Receive check box. This check box is available only for request-response send ports.
	To enable message tracking before sending a message, select the Before Send check box.
	To enable message tracking after sending a message, select the After Send check box.
Certificate Name	Select the encryption certificate from the drop-down list.
Send Pipeline	It is updated automatically with the long name of the certificate picked for the Certificate name property. Default Value: Empty
Receive Pipeline	It is updated automatically with the usage of the certificate picked for the Certificate name property. Default Value: Empty

5. Click **OK**.

You can use the send port with the new properties.

How to Edit Send Port Adapter Properties Using BizTalk Explorer

You can use BizTalk Explorer to edit the properties for an adapter send port. The adapter send port properties are only available for static send ports.

To edit the properties for an adapter send port

1. In Visual Studio .NET, in a BizTalk Server project, on the **View** menu, click **BizTalk Explorer**.
2. In BizTalk Explorer, expand the **Send Ports** collection, right-click the send port that contains the properties you want to edit, and then click **Edit**.
3. In the **Send Port Properties** dialog box, expand the **Transport** node, and then select the **Primary** node.
4. In the **Send Port Properties** dialog box, in the right pane, do the following.

Use this	To do this
Transport Type	Select the transport type from the drop-down list. If you select a request-response port, only transports that support receipt-response will be available in the drop-down list.
Address (URI)	<p>Click the ellipsis (...) button to configure the Adapter properties for the send port. For information about configuring the adapter-specific properties, see the following topics:</p> <ul style="list-style-type: none"> • How to Configure a Base EDI Send Port • How to Configure an MSMQT Send Port • How to Configure a File Send Port • How to Configure an FTP Send Port • Configuring an HTTP Send Port • How to Configure an SMTP Send Port • Configuring a SOAP Send Port • How to Configure a SQL Send Port
Retry Count	<p>Number of times the engine will try to resend if transmission fails.</p> <p>Default Value: 3</p> <p>Allowed range: 0 to 1000 inclusive</p>
Retry Interval	<p>Number of minutes the engine will wait between retries.</p> <p>Default = 5 Minutes</p> <p>Allowed range: 0 to 525600 inclusive (max value is 365 days)</p>

Ordered Delivery	<p>This property is read-only, except for the MSMQT adapter.</p> <p>To guarantee ordered delivery of messages in BizTalk Message Queuing, select True from the drop-down list. This property is False and read-only for all other transports.</p>
Enable Service Window	<p>The Service window configures the send port to send only at specified times of the day. If disabled, this implies that the send port sends messages all the time.</p> <p>Default Value: False</p>
Start Time	<p>Indicates when the send port should start to send messages. Enabled only if the Service window is enabled, otherwise dimmed.</p> <p>Time in HH:MM AM/PM format.</p> <p>Default Value: 12:00 AM (local time)</p>
Stop Time	<p>Indicates when the send port should stop sending messages. Enabled only if the service window is enabled, otherwise dimmed.</p> <p>Time is in HH:MM AM/PM format.</p> <p>Default Value: 11:59 PM (local time)</p>

5. Click **OK**.

You can use the adapter send port with the new properties.

To configure the rest of the send port, proceed to Editing Send Port Properties Using BizTalk Explorer.

How to Manage Send Port Encryption Certificates Using BizTalk Explorer

You can use BizTalk Explorer to associate a certificate in the Address Book (Local Machine\Other People) store on the local machine with a send port.

To manage send port encryption certificates

1. In Visual Studio .NET, in a BizTalk Server project, on the **View** menu, click **BizTalk Explorer**.
2. In BizTalk Explorer, expand the **Send Ports** collection, right-click the send port that contains the properties you want to edit, then click **Edit**.

3. In the **Send Port Properties** dialog box, expand the **Send** folder in the left pane in the **General** section, click the drop-down button in the value box for the **Certificate Name** property to view the certificates in the Local Machine\Other People store.

How to Add a Filter to a Send Port for Content-Based Routing Using BizTalk Explorer

You can use BizTalk Explorer to create one or more filter expressions using any properties in the message context. The combined expressions filter messages to a send port or a send port group. A filter does not affect the messages that an orchestration sends to the port; it creates an additional source for messages that does not use orchestrations and is particularly helpful in a messaging only or pass-through scenario, when an orchestration is not necessary. In addition, you can define a filter expression that sets conditions for the properties or message fields to determine the messages you want to receive.

For more information about the message context properties that you can promote, see **Message Context Properties**.

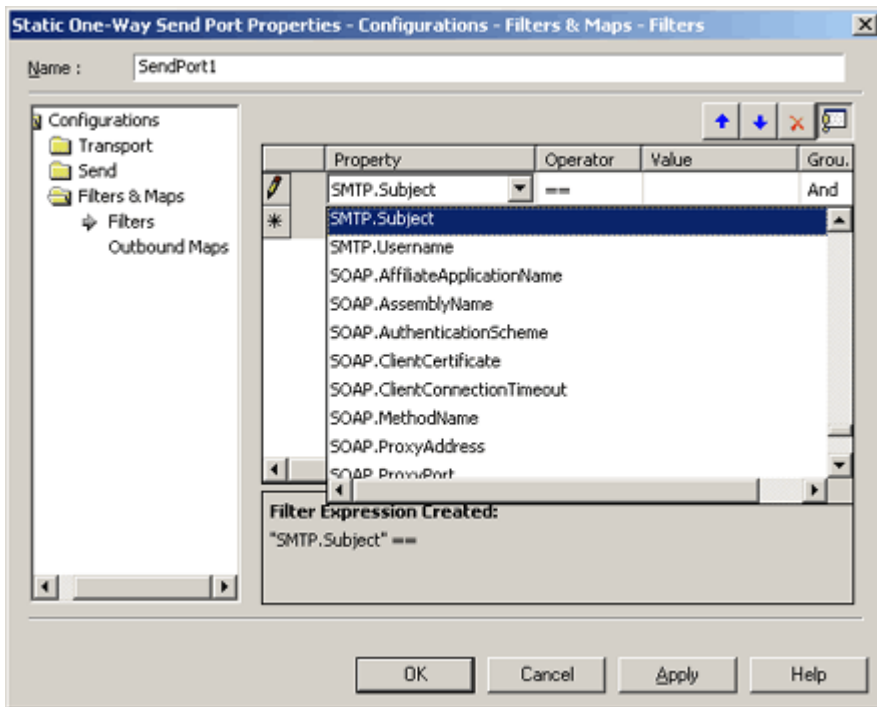
To add a filter to a send port

1. In Visual Studio .NET, in a BizTalk Server project, on the **View** menu, click **BizTalk Explorer**.
2. In BizTalk Explorer, expand the **Send Ports** collection, right-click the send port to which you want to add a filter, and then click **Edit**.
3. In the **Send Port Properties** dialog box, expand the **Filters & Mapping** folder in the left pane, and then select the **Filters** node.

The expression pane displays an empty grid where you can add expressions. In each expression, you compare a message property with a constant. You can change the order in which BizTalk Server evaluates the expressions and how it groups them into the final filter. To view the results of your work, click **Show Filter Expression** on the toolbar.

4. In the expression pane on the right, click **Click here to add a new row** to add a property, operator, value, and group by value to your expression. For example, to subscribe to the messages from a receive port (recPort1), enter the following expression.

5. Illustrates the send port filter.



6. For information about writing expressions and using the Expression Editor, see Filter Expression Dialog Box.
7. After the expression is created, click **OK**.

To edit a filter for a send port

1. In Visual Studio .NET, in a BizTalk Server project, on the **View** menu, click **BizTalk Explorer**.
2. In BizTalk Explorer, expand the **Send Ports** collection, right-click the send port that contains the filter you want to edit, and then click **Edit**.
3. In the **Send Port Properties** dialog box, expand the **Filters & Mapping** folder in the left pane, and then select the **Filters** node.

The expression pane displays a list of the expressions that make up the filter associated with this send port or send port group. You can add or delete expressions, change the order of expressions, or select an expression to edit. To view the results of your changes, click **Show Filter Expression** on the toolbar.

4. In the expression pane on the right, edit the list of filter expressions to build the filter you want, and then click **OK**.

For information about writing expressions and combining them into a filter, see Filter Expression Dialog Box.

How to Add a Filter to a Send Port for Passthrough Using BizTalk Explorer

You can use BizTalk Explorer to create one or more filter expressions using any properties in the message context. The combined expressions filter messages to a send port or a send port group. A filter does not affect the messages that an orchestration sends to the port; it creates an additional source for messages that does not use orchestrations. Also, you can define a filter expression that sets conditions for the properties or message fields to determine the messages you want to receive.

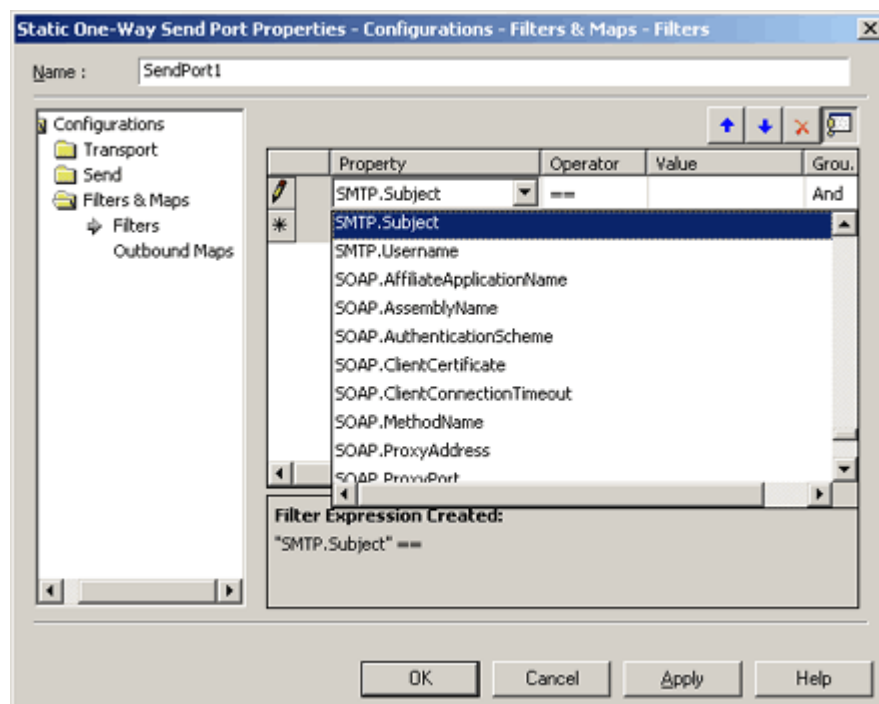
To add a filter to a send port

1. In Visual Studio .NET, in a BizTalk Server project, on the **View** menu, click **BizTalk Explorer**.
2. In BizTalk Explorer, expand the **Send Ports** collection, right-click the send port to which you want to add a filter, and then click **Edit**.
3. In the **Send Port Properties** dialog box, expand the **Filters & Mapping** folder in the left pane, and then select the **Filters** node.

The expression pane displays an empty grid where you can add expressions. In each expression you compare a message property with a constant. You can change the order in which BizTalk Server evaluates the expressions and how it groups them into the final filter. To view the results of your work, click **Show Filter Expression** on the toolbar.

4. In the expression pane on the right, edit the list of filter expressions to build the filter you want, and then click **OK**.

Illustrates the send port filter.



For instance, to subscribe to the messages from a receive port (recPort1) add the following.

```
BTS.ReceivePortName == recPort1
```

For information about writing expressions and using the Expression Editor, see Filter Expression Dialog Box.

To edit a filter for a send port

1. In Visual Studio .NET, in a BizTalk Server project, on the **View** menu, click **BizTalk Explorer**.
2. In BizTalk Explorer, expand the **Send Ports** collection, right-click the send port that contains the filter you want to edit, and then click **Edit**.
3. In the **Send Port Properties** dialog box, expand the **Filters & Mapping** folder in the left pane, and then select the **Filters** node.

The expression pane displays a list of the expressions that make up the filter associated with this send port or send port group. You can add or delete expressions, change the order of expressions, or select an expression to edit. To view the results of your changes, click **Show Filter Expression** on the toolbar.

4. In the expression pane on the right, edit the list of filter expressions to build the filter you want, and then click **OK**.

For information about writing expressions and combining them into a filter, see Filter Expression Dialog Box.

How to Add Maps to Send Ports for Outbound Normalization Using BizTalk Explorer

In BizTalk Server 2006 you can use a map to transform the schema of a message at the send port without processing the message through an orchestration. You can use BizTalk Explorer to associate a map to a send port, and you can add one map for each document schema that you want to use with the send port. You must deploy a map before you can add it to a send port. For more information about using maps, see About Maps.

To add a map to a send port

1. In Visual Studio .NET, in a BizTalk Server project, on the **View** menu, click **BizTalk Explorer**.
2. In BizTalk Explorer, expand the **Send Ports** or **Send Port Groups** collection, right-click the send port to which you want to add a map, and then click **Edit**.
3. In the **Send Port Properties** dialog box, expand the **Filters & Mapping** collection in the left pane, and then select the **Outbound Maps** node.

Only maps deployed in assemblies are shown in the right pane.

4. In the right pane, click the first empty row under **Map** to select a map from the drop-down list, and then click **OK**.
5. The map is added to the send port. The map takes effect almost immediately; caching may slightly delay the application of the map.

How to Add Maps to Send Ports for Inbound Normalization Using BizTalk Explorer

In BizTalk Server 2006, you can use a map to transform the schema of a message at the send port without processing the message through an orchestration. You can use BizTalk Explorer to associate a map to a send port, and you can add one map for each document schema that you want to use with the send port. You must deploy a map before you can add it to a send port. For more information about using maps, see About Maps.

To add a map to a send port

1. In Visual Studio .NET, in a BizTalk Server project, on the **View** menu, click **BizTalk Explorer**.
2. In BizTalk Explorer, expand the **Send Ports** or **Send Port Groups** collection, right-click the send port to which you want to add a map, and then click **Edit**.
3. In the **Send Port Properties** dialog box, expand the **Filters & Mapping** collection in the left pane, and then select the **Inbound Maps** node.

Only maps deployed in assemblies are shown in the right pane.

4. In the right pane, click the first empty row under **Map** to select a map from the drop-down list, and then click **OK**.
5. The map is added to the send port. The map takes effect almost immediately; caching may slightly delay the application of the map.

How to Enlist and Start a Send Port Using BizTalk Explorer

Send ports always have a state. The state of the send port determines whether messages are routed to the port and whether the port processes messages. The state of the send port can be changed by actions made on the send port.

Send ports are always in one of three states: Bound, Started, or Stopped. The actions that you can invoke to change the states are: Enlist, Start, Stop, and Unenlist.

When you create a send port, it is in the Bound state. To move the send port to the Started state, the send port must be started, or enlisted and started. For more information about what functionality is available in each state, see Managing Send Ports Using BizTalk Explorer.

To enlist a send port

1. In Visual Studio .NET, in a BizTalk Server project, on the **View** menu, click **BizTalk Explorer**.
2. In BizTalk Explorer, expand the **Send Ports** collection, right-click the send port to which you want to enlist, and then click **Enlist**.

If the send port was in the Bound state, it is now in the Stopped state.

To start a send port

1. In Visual Studio .NET, in a BizTalk Server project, on the **View** menu, click **BizTalk Explorer**.
2. In BizTalk Explorer, expand the **Send Ports** collection, right-click the send port to which you want to start, and then click **Start**.

If the send port was in the Stopped or Bound state, it is now in the Started state.

How to Stop and Unenlist a Send Port Using BizTalk Explorer

Send ports always have a state. The state of the send port determines whether messages are routed to the port and whether the port processes messages. The state of the send port can be changed by actions made on the send port.

Send ports are always in one of three states: Bound, Started, or Stopped. The actions that can you can invoke to change the states are: Enlist, Start, Stop, and Unenlist.

After a send port is in the Started state, it must be unenlisted to be in the Bound state, or the Stop action must be invoked to move it to the Stopped state. For more information about what functionality is available in each state, see Managing Send Ports Using BizTalk Explorer.

To stop a send port

1. In Visual Studio .NET, in a BizTalk Server project, on the **View** menu, click **BizTalk Explorer**.
2. In BizTalk Explorer, expand the **Send Ports** collection, right-click the send port to which you want to stop, and then click **Stop**. If the send port was in the Started state, it is now in the Stopped state.

To unenlist a send port

1. In Visual Studio .NET, in a BizTalk Server project, on the **View** menu, click **BizTalk Explorer**.
2. In BizTalk Explorer, expand the **Send Ports** collection, right-click the send port to which you want to unenlist, and then click **Unenlist**.

If the send port was in the Started or Stopped state, it is now in the Bound state.

How to Delete a Send Port Using BizTalk Explorer

You can use BizTalk Explorer to delete a static or dynamic send port providing the port meets the following criteria:

- The send port is not bound to an orchestration.
- The send port is not referenced in a party or in a send port group.
- The send port is not in a Started or Enlisted state; it must be in the Bound state.

Attempting to delete a send port that does not meet the previous criteria causes an error.

To delete a send port

1. In Visual Studio .NET, in a BizTalk Server project, on the **View** menu, click **BizTalk Explorer**.
2. In BizTalk Explorer, expand the **Send Ports** collection, right-click the port you want to delete, then click **Delete**.
3. In the **BizTalk Explorer** dialog box, click **Yes** to confirm the deletion.

The send port is deleted from the BizTalk Management database.

Managing Send Port Groups Using BizTalk Explorer

You can use BizTalk® Explorer to do the following:

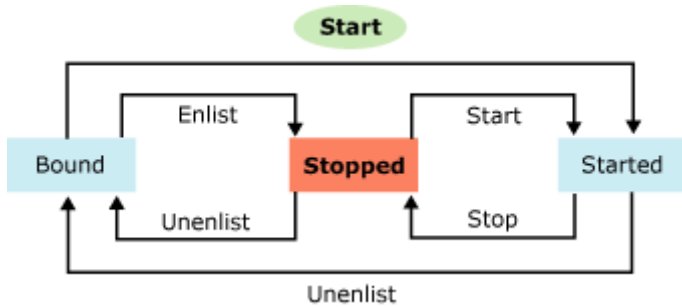
- Add send port groups.
- Add send ports to send port groups.
- Associate a filter with a send port group to create a subscription that routes messages directly to the port without using an orchestration.

The following table lists all of the states and actions available for a send port group.

States	Actions
Bound	Enlist
Started	Start
Stopped	Unenlist
Not applicable	Stop

The following illustration shows how the states and the actions of the send port group can be changed.

Illustrates the states and actions of send port groups.



In This Section

- Adding a Send Port Group Using BizTalk Explorer
- Adding and Deleting Send Ports from a Send Port Group Using BizTalk Explorer
- Adding a Filter to a Send Port Group Using BizTalk Explorer
- Enlisting and Starting a Send Port Group Using BizTalk Explorer
- Stopping and Unenlisting a Send Port Group Using BizTalk Explorer
- Deleting a Send Port Group Using BizTalk Explorer

How to Add a Send Port Group Using BizTalk Explorer

You can use BizTalk Explorer to add a send port group to a BizTalk Management database.

To add a send port group to a BizTalk Managment database

1. In Microsoft® Visual Studio® .NET, in a Microsoft® BizTalk Server project, on the **View** menu, click **BizTalk Explorer**.
2. In BizTalk Explorer, right-click on the **Send Port Groups** collection, and then click **Add Send Port Group**.
3. In the **Send Port Group Properties** dialog box, in the **Name** box, type a name for the send port group.
4. In the **Send Port Group Properties** dialog box, in the right pane, select a send port from the drop-down list.
5. To add a filter to the send port group, see Adding a Filter to a Send Port Group Using BizTalk Explorer.

Click **OK**, and the send port group is added to the BizTalk Management database.

How to Add and Delete Send Ports from a Send Port Group Using BizTalk Explorer

You can use BizTalk Explorer to add and delete a send port to a send port group. You can only add one-way static send ports to a send port group. When you delete a send port from a send port group, you are not deleting the send port from the BizTalk Management database, you are removing the reference to, or dereferencing, the send port. For more information, see [Deleting a Send Port Using BizTalk Explorer](#).

To add a send port to a send port group

1. In Visual Studio .NET, in a BizTalk Server project, on the **View** menu, click **BizTalk Explorer**.
2. In BizTalk Explorer, expand the **Send Port Groups** collection, right-click the send port group to which you want to add a send port, and then click **Edit**.
3. In the **Send Port Group Properties** dialog box, select **Send Ports** in the left pane, click the first blank row in the grid on the right, select a send port from the drop-down list, and then click **OK**.
4. The send port is added to the send port group.

To delete a send port from a send port group

1. In Visual Studio .NET, in a BizTalk Server project, on the **View** menu, click **BizTalk Explorer**.
2. In BizTalk Explorer, expand the **Send Port Groups** collection, right-click the send port group that contains the send port you want to delete, and then click **Edit**.
3. In the **Send Port Group Properties** dialog box, select **Send Ports** in the left pane, in the right pane, click the row selector next to the send port you want to delete, and then click **Delete**.

The send port is deleted from the send port group. You can now add the send port to a different send port group, or delete it from the BizTalk Management database.

How to Add a Filter to a Send Port Group Using BizTalk Explorer

You can use BizTalk Explorer to create one or more filter expressions using any properties in the message context. The combined expressions filter messages to a send port or a send port group. A filter does not affect the messages that an orchestration sends to the send port group; it creates an additional source for messages that does not use orchestrations. Also, you can define a filter expression that sets conditions for the properties or message fields to determine the messages you want to receive.

To add a filter to a send port group

1. In Visual Studio .NET, in a BizTalk Server project, on the **View** menu, click **BizTalk Explorer**.

2. In BizTalk Explorer, expand the **Send Port Groups** collection, right-click the send port group to which you want to add a filter, and then click **Edit**.
3. In the **Send Port Group Properties** dialog box, select the **Filters** node in the left pane.

The expression pane displays an empty grid where you can add expressions. In each expression, you compare a message property with a constant. You can change the order in which BizTalk Server evaluates the expressions and how it groups them into the final filter. To view the results of your work, click **Show Filter Expression** on the toolbar.

4. In the expression pane on the right, click **Click here to add a new row** to add a property, operator, value, and group by value to your expression.

For instance, to subscribe to the messages from a receive port (recPort1), enter the following expression `BTS.ReceivePortName == recPort1`.

For information about writing expressions and using the Expression Editor, see Filter Expression Dialog Box.

After the expression is created, click **OK**. ☐ To edit a filter for a send port group

1. In Visual Studio .NET, in a BizTalk Server project, on the **View** menu, click **BizTalk Explorer**.
2. In BizTalk Explorer, expand the **Send Port Groups** collection, right-click the send port group that contains the filter you want to edit, and then click **Edit**.
3. In the **Send Port Group Properties** dialog box, select the **Filters** node in the left pane.

The expression pane displays a list of the expressions that make up the filter associated with this send port or send port group. You can add or delete expressions, change the order of expressions, or select an expression to edit. To view the results of your changes, click **Show Filter Expression** on the toolbar.

4. In the expression pane on the right, edit the list of filter expressions to build the filter you want, and then click **OK**.

For information about writing expressions and combining them into a filter, see Filter Expression Dialog Box.

How to Enlist and Start a Send Port Group Using BizTalk Explorer

Send port groups always have a state. The state of the send port group determines whether messages are routed to the send port group, and whether the port processes messages. The state of the send port group can be changed by actions made on the send port.

Send port groups are always in one of three states: Bound, Started, and Stopped. The actions that you can invoke to change the states are: Enlist, Start, Stop, and Unenlist.

When send port groups are created, they are in the Bound state. To move the send port group to the Started state, the send port group must be started, or enlisted and started. For more information about what functionality is available in each state and the dependencies between send ports and send port groups, see *Managing Send Ports Using BizTalk Explorer*.

To enlist a send port group

1. In Visual Studio .NET, in a BizTalk Server project, on the **View** menu, click **BizTalk Explorer**.
2. In BizTalk Explorer, expand the **Send Port Groups** collection, right-click the send port group to which you want to enlist, and then click **Enlist**.

If the send port group was in the Bound state, it is now in the Stopped state.

To start a send port group

1. In Visual Studio .NET, in a BizTalk Server project, on the **View** menu, click **BizTalk Explorer**.
2. In BizTalk Explorer, expand the **Send Port Groups** collection, right-click the send port group to which you want to start, and then click **Start**.

If the send port group was in the Stopped or Bound state, it is now in the Started state.

How to Stop and Unenlist a Send Port Group Using BizTalk Explorer

Send port groups always have a state. The state of the send port group determines whether messages are routed to the send port group and whether the port processes messages. The state of the send port group can be changed by actions made on the send port.

Send port groups are always in one of three states: Bound, Started, and Stopped. The actions that you can invoke to change the states are: Enlist, Start, Stop, and Unenlist.

After a send port is in the Started state, it must be unenlisted to be in the Bound state, or the stop action must be invoked to move it to the Stopped state. For more information about what functionality is available in each state and the dependencies between send ports and send port groups, see *Managing Send Ports Using BizTalk Explorer*.

To stop a send port group

1. In Visual Studio .NET, in a BizTalk Server project, on the **View** menu, click **BizTalk Explorer**.
2. In BizTalk Explorer, expand the **Send Port Groups** collection, right-click the send port group to which you want to stop, and then click **Stop**.

If the send port group was in the Started state, it is now in the Stopped state.

To unenlist a send port group

1. In Visual Studio .NET, in a BizTalk Server project, on the **View** menu, click **BizTalk Explorer**.

2. In BizTalk Explorer, expand the **Send Port Groups** collection, right-click the send port group to which you want to unenlist, and then click **Unenlist**.

If the send port group was in the Started or Stopped state, it is now in the Bound state.

How to Delete a Send Port Group Using BizTalk Explorer

You can use BizTalk Explorer to delete a send port group providing the port meets the following criteria:

- The send port group is not bound to an orchestration.
- The send port group is not in the Started or Enlisted states. It must be in the Bound state.

To delete a send port group

1. In Visual Studio .NET, in a BizTalk Server project, on the **View** menu, click **BizTalk Explorer**.
2. In BizTalk Explorer, expand the **Send Port Groups** collection, right-click the send port group that you want to delete, and then click **Delete**.
3. In the **BizTalk Explorer** dialog box, click **Yes** to confirm the deletion.

Managing Receive Ports and Receive Locations Using BizTalk Explorer

You can use BizTalk® Explorer to create and configure receive ports and receive locations. Receive locations are always added to a receive port, so you must create at least one receive port before you can add a receive location.

This section contains:

- Adding a Receive Port Using BizTalk Explorer
- Editing Receive Port Properties Using BizTalk Explorer
- Adding a Receive Location Using BizTalk Explorer
- Editing Receive Location Adapter Properties Using BizTalk Explorer
- Adding Maps to Receive Ports for Inbound Normalization Using BizTalk Explorer
- Adding Maps to Receive Ports for Outbound Normalization Using BizTalk Explorer
- Enabling and Disabling a Receive Location Using BizTalk Explorer
- Specifying a Primary Receive Location Using BizTalk Explorer
- Deleting a Receive Location Using BizTalk Explorer

- Deleting a Receive Port Using BizTalk Explorer

How to Add a Receive Port Using BizTalk Explorer

You can use BizTalk Explorer to add a receive port to a BizTalk Management database. You must add send and receive ports that match the ports on an orchestration before you can bind the orchestration.

To add a receive port

1. In Microsoft® Visual Studio® .NET, in a BizTalk Server project, on the **View** menu, click **BizTalk Explorer**.
2. In BizTalk Explorer, right-click the **Receive Ports** collection, and then click **Add Receive Port**.
3. In the **Create New Receive Port** dialog box, select **One-Way Port** or **Request-Response Port**, and then click **OK**.
4. In the **Receive Port Properties** dialog box, in the **Name** box, type a name for the receive port.
5. In the **Receive Port Properties** dialog box, in the right pane, do the following.

Use this	To do this
Authentication	To disable authentication, select Not Required from the drop-down list.
	To enable authentication and to drop unauthenticated messages, select Required (Drop Messages) from the drop-down list.
	To enable authentication and keep unauthenticated messages, select Required (Keep Messages) from the drop-down list.
Tracking Type	To enable message tracking before receiving a message, select the Before Receive check box. This option is available for one-way and two-way (request-response) ports.
	To enable message tracking after receiving a message, select the After Receive check box. This option is available for one-way and two-way (request-response) ports.
	To enable message tracking before sending a message, select the Before Send check box. This option is available only on two-way (request-response) ports.
	To enable message tracking after sending a message, select the After Send check box. This option is available only on two-way (request-response) ports.
Send Pipeline	Select the send pipeline from the drop-down list.
	This property only applies to request-response receive ports.

- To apply filters and mapping to the send port, see Adding Maps to Receive Ports for Inbound Normalization Using BizTalk Explorer.

You can now add receive locations to the receive port.

How to Edit Receive Port Properties Using BizTalk Explorer

You can use BizTalk Explorer to edit the properties of a receive port.

To edit receive port general properties

- In Visual Studio .NET, in a BizTalk Server project, on the **View** menu, click **BizTalk Explorer**.
- In BizTalk Explorer, expand the **Receive Ports** collection, right-click the receive port you want to edit, and then click **Edit**.
- In the **Receive Port Properties** dialog box, in the right pane, do the following.

Use this	To do this
Authentication	<p>Select the type of party resolution, or authentication, for the receive port.</p> <p>To disable authentication, select Not Required from the drop-down list.</p> <p>To enable authentication and to drop unauthenticated messages, select Required (Drop Messages) from the drop-down list.</p> <p>To enable authentication and keep unauthenticated messages, select Required (Keep Messages) from the drop-down list.</p>
Tracking Type	<p>To enable message tracking before receiving a message, select the Before Receive check box.</p> <p>To enable message tracking after receiving a message, select the After Receive check box.</p> <p>To enable message tracking before sending a message, select the Before Send check box.</p> <p>To enable message tracking after sending a message, select the After Send check box.</p>
Send Pipeline	<p>Select the send pipeline from the drop-down list.</p> <p>This property only applies to request-response receive ports.</p>

- Click **OK**.

5. To apply filters and mapping to the send port, see [Adding Maps to Receive Ports for Inbound Normalization Using BizTalk Explorer](#).

You can now use the receive port with the new properties.

How to Add a Receive Location Using BizTalk Explorer

You can use BizTalk Explorer to add a receive location. Receive locations are created under a receive port.

To add a receive location

1. In Visual Studio .NET, in a BizTalk Server project, on the **View** menu, click **BizTalk Explorer**.
2. In BizTalk Explorer, expand the **Receive Ports** collection, expand the receive port to which you want to add a receive location, right-click the **Receive Locations** collection, and then click **Add Receive Location**.
3. In the **Receive Location Properties** dialog box, in the **Name** box, type a name for the receive location.
4. In the **Receive Location Properties** dialog box, in the left pane, select the **General** node.
5. In the **Receive Location Properties** dialog box, in the right pane, do the following.

Use this	To do this
Transport Type	Select the transport type from the drop-down list. If you change the transport type, you must edit the Address (URI) .
Address (URI)	<p>Click the ellipsis (...) button to configure the Adapter properties for the receive location. For more information about configuring the adapter-specific receive location properties, see the following topics:</p> <p>How to Configure a Base EDI Receive Location How to Configure an MSMQT Receive Location How to Configure a File Receive Location How to Configure an FTP Receive Location How to Configure an HTTP Receive Location How to Configure a SOAP Receive Location How to Configure a SQL Receive Location</p> <p>The URI should be valid for the type of adapter specified. For example, if you have a Transport Type of HTTP, an Address (URI) of someone@example.com is not valid, and the property page will throw an error.</p>
Receive Handler	<p>Select the receive handler from the drop-down list.</p> <p>This is a required field.</p>
Receive	Select the receive pipeline from the drop-down list.

Pipeline	
Start Date	Day to start receiving messages. Select the start date from the pull-down calendar and click on the date. You can change the year by clicking on the year field.
Stop Date	Day to stop receiving messages. Select the stop date from the pull-down calendar and click on the date. You can change the year by clicking on the year field.
Enable Service Window	the The Service window configures the receive location to receive only at specified times of the day. If disabled, this implies that the receive location receives messages round the clock.
Start Time	Enabled only if you enabled the Service window. Otherwise dimmed. Time is in HH:MM AM/PM format. Default Value: 12:00 AM (local time)
Stop Time	Enabled only if you enabled the Service window. Otherwise dimmed. Time is in HH:MM AM/PM format. Default Value: 11:59 PM (local time)

6. Click **OK**.

The receive location is created; you can now enable it.

How to Edit Receive Location Adapter Properties Using BizTalk Explorer

You can use BizTalk Explorer to edit receive location properties.

To edit receive location properties

1. In Visual Studio .NET, in a BizTalk Server project, on the **View** menu, click **BizTalk Explorer**.
2. In BizTalk Explorer, expand the **Receive Ports** collection, expand the receive port and the **Receive Locations** collection, right-click the receive location, and then click **Edit**.
3. In the **Receive Location Properties** dialog box, in the left pane, select the **General** node.

4. In the **Receive Location Properties** dialog box, in the **Name** box, type a name for the receive location.
5. In the **Receive Location Properties** dialog box, in the right pane, do the following.

Use this	To do this
Transport Type	Select the transport type from the drop-down list. If you change the transport type, you must edit the Address (URI) .
Address (URI)	<p>Click the ellipsis (...) button to configure the Adapter properties for the receive location. For more information about configuring the adapter-specific receive location properties, see the following topics:</p> <ul style="list-style-type: none"> • Configuring a Base EDI Receive Location Using BizTalk Explorer • Configuring a BizTalk Message Queuing Receive Location • Configuring a File Receive Location • Configuring an FTP Receive Location • Configuring an HTTP Receive Location • Configuring a SOAP Receive Location • Configuring a SQL Receive Location
Host	Select the host for the receive handler from the drop-down list. This property is only available when you add the first receive location of a transport type to a receive port.
Receive Handler	Select the receive handler from the drop-down list.
Receive Pipeline	Select the receive pipeline from the drop-down list.
Start Date	<p>Day to start receiving messages.</p> <p>Select the start date from the pull-down calendar and click on the date. You can change the year by clicking on the year field.</p>
Stop Date	<p>Day to stop receiving messages.</p> <p>Select the stop date from the pull-down calendar and click on the date. You can change the year by clicking on the year field.</p>
Enable Service	The Service window configures the receive location to receive only at specified times of the day. If disabled, this implies that the receive location receives messages round the

Window	clock.
Start Time	<p>Enabled only if you enabled the Service window. Otherwise dimmed.</p> <p>Time is in HH:MM AM/PM format.</p> <p>Default Value: 12:00 AM (local time)</p>
Stop Time	<p>Enabled only if you enabled the Service window. Otherwise dimmed.</p> <p>Time is in HH:MM AM/PM format.</p> <p>Default Value: 11:59 PM (local time)</p>

- Click **OK**.

Your edits take effect.

How to Add Maps to Receive Ports for Inbound Normalization Using BizTalk Explorer

In BizTalk Server 2006, you can use a map to apply an XSL transform of the message at the receive port without processing the message through an orchestration. You can use BizTalk Explorer to associate a map to a receive port, and you can add one map for each document schema that you want to use with the receive port. You must deploy a map before you can add it to a receive port. For more information about using maps, see About Maps.

To add a map to a receive port

- In Visual Studio .NET, in a BizTalk Server project, on the **View** menu, click **BizTalk Explorer**.
- In BizTalk Explorer, expand the **Receive Ports** collection, right-click the receive port to which you want to add a map, and then click **Edit**.
- In the **Receive Port Properties** dialog box, select **Inbound Maps**.
- In the right pane, click the first empty row under **Map** to select a map from the drop-down list. Only the maps in deployed BizTalk Server assemblies will appear in the drop-down list.
- Click **OK**.

The maps are added to the receive port. The map takes effect almost immediately; caching may slightly delay the application of the map.

How to Add Maps to Receive Ports for Outbound Normalization Using BizTalk Explorer

In BizTalk Server 2006, you can use a map to apply an XSL transform to the response message sent by the receive port without processing the message through an orchestration. You can use BizTalk Explorer to associate a map to a request-response receive port, and you can add one map for each document schema that you want to use with the receive port. You must deploy a map before you can add it to a receive port. For more information about using maps, see About Maps.

To add a map to a receive port

1. In Visual Studio .NET, in a BizTalk Server project, on the **View** menu, click **BizTalk Explorer**.
2. In BizTalk Explorer, expand the **Receive Ports** collection, right-click the request-response receive port to which you want to add a map, and then click **Edit**.
3. In the **Receive Port Properties** dialog box, select **Outbound Maps**.
4. In the right pane, click the first empty row under **Map** to select a map from the drop-down list. Only the maps in deployed BizTalk Server assemblies will appear in the drop-down list.
5. Click **OK**.

The maps are added to the receive port. The map takes effect almost immediately; caching may slightly delay the application of the map.

How to Enable and Disable a Receive Location Using BizTalk Explorer

You can enable and disable a receive location in BizTalk Explorer.

When enabled, messages are received and processed by the receive location. If the receive location is disabled, any messages sent to the receive location will not be processed by BizTalk Server.

To enable a receive location

1. In Visual Studio .NET, in a BizTalk Server project, on the **View** menu, click **BizTalk Explorer**.
2. In BizTalk Explorer, expand the **Receive Ports** collection, expand the **Receive Locations** collection, right-click the receive location to which you want to enable, and then click **Enable**.

The receive location is now enabled.

To disable a receive location

1. In Visual Studio .NET, in a BizTalk Server project, on the **View** menu, click **BizTalk Explorer**.

2. In BizTalk Explorer, expand the **Receive Ports** collection, expand the **Receive Locations** collection, right-click the receive location to which you want to disable, and then click **Disable**.

The receive location is disabled after the cache refresh interval passes. The default value for the cache refresh interval is 60 seconds. For more information about setting the Cache Refresh Interval property, see About the BizTalk Group.

How to Specify a Primary Receive Location Using BizTalk Explorer

You can specify a primary receive location in BizTalk Explorer if you have more than one receive location for a receive port. A primary receive location is the location used to represent a receive port when the port needs to be passed to some other entity. For example, if a receive port needs to be passed to a partner to send me messages, I need to pass a single designated location out of the list of receive locations. This is the receive location that is marked as primary out of the receive locations in a receive port. Users can change the primary nature of a receive location by changing any non-primary receive location to become primary. This automatically sets any other primary receive locations to nonprimary. The simple rule is that one and only one receive location can be primary in a given receive port.

To specify a primary receive location

1. In Visual Studio .NET, in a BizTalk Server project, on the **View** menu, click **BizTalk Explorer**.
2. In BizTalk Explorer, expand the **Receive Ports** collection, expand the **Receive Locations** collection, and select the receive location you want to be the primary receive location.
3. In the Properties window, in the **Value** column for the **Primary** field, select **True** from the drop-down list.

The receive location is now the primary receive location. The receive location that used to be the primary receive location now has a value of **False** for the **Primary** field.

How to Delete a Receive Location Using BizTalk Explorer

You can use BizTalk Explorer to remove a receive location provided the receive port to which the receive location belongs to is not bound to an orchestration.

To delete a receive location

1. In Visual Studio .NET, in a BizTalk Server project, on the **View** menu, click **BizTalk Explorer**.
2. In BizTalk Explorer, expand the **Receive Ports** collection, expand the receive port and the **Receive Locations** collection, right-click the receive location, and then click **Delete**.
3. In the **BizTalk Explorer** dialog box, click **Yes** to confirm the deletion

How to Delete a Receive Port Using BizTalk Explorer

You can use BizTalk Explorer to delete a receive port from a BizTalk Management database provided the port is not bound to an orchestration.

To delete a receive port

1. In Visual Studio .NET, in a BizTalk Server project, on the **View** menu, click **BizTalk Explorer**.
2. In BizTalk Explorer, expand the **Receive Ports** collection, right-click the port you want to delete, then click **Delete**.
3. In the **BizTalk Explorer** dialog box, click **Yes** to confirm the deletion.

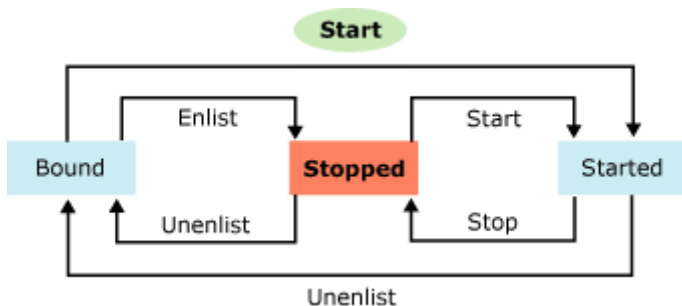
Managing Orchestrations Using BizTalk Explorer

You can use BizTalk® Explorer to manage the port bindings and the state of an orchestration.

The following table lists all of the states and actions available for an orchestration.

States	Actions
Enlisted	Bind
Started	Enlist
Stopped	Start
Unenlisted	Unenlist
Not applicable	Stop

Illustrates the states and actions of send port groups.



In This Section

- Browsing Orchestrations Using BizTalk Explorer
- Browsing Used Roles of an Orchestration Using BizTalk Explorer

- Browsing Implemented Roles of an Orchestration Using BizTalk Explorer
- Browsing Invoked Orchestrations Using BizTalk Explorer
- Binding an Orchestration Using BizTalk Explorer
- Enlisting and Starting an Orchestration Using BizTalk Explorer
- Stopping and Unenlisting an Orchestration Using BizTalk Explorer

How to Browse Orchestrations Using BizTalk Explorer

Orchestrations deployed to any BizTalk Management database can be viewed from within BizTalk Explorer.

To browse orchestrations

1. In Microsoft® Visual Studio® .NET, in a Microsoft® BizTalk Server project, on the **View** menu, click **BizTalk Explorer**.
2. In BizTalk Explorer, expand the **Orchestrations** collection.

From the **Orchestrations** collection, you can bind the orchestration and review its read-only properties.

If different versions of the same assembly are deployed three times, an associated orchestration will be displayed with sequential numbering.

To review the read-only properties for an orchestration

1. In Visual Studio .NET, in a BizTalk Server project, on the **View** menu, click **BizTalk Explorer**.
2. In BizTalk Explorer, expand the **Orchestrations** collection, right-click the orchestration, and then click **Properties**.

The read-only properties are displayed in the Properties window in Visual Studio .NET. The following table contains descriptions for the properties.

Use this	To do this
Name	Contains the name of the node.
Assembly Name	Contains the assembly name.
Status	Specifies the status of the orchestration.

How to Browse Used Roles of an Orchestration Using BizTalk Explorer

All of the Used Roles of the orchestration can be viewed from within BizTalk Explorer.

To browse used roles of an orchestration

1. In Visual Studio .NET, in a BizTalk Server project, on the **View** menu, click **BizTalk Explorer**.
2. In BizTalk Explorer, expand the **Orchestrations** collection, expand the orchestration, and then expand the **Used Roles** collection.

From the **Used Roles** collection, you can review its read-only properties.

To review the read-only properties for used roles

1. In Visual Studio .NET, in a BizTalk Server project, on the **View** menu, click **BizTalk Explorer**.
2. In BizTalk Explorer, expand the **Orchestrations** collection, expand the orchestration, expand the **Used Roles** collection, right-click the role for which you want to view properties, and then click **Properties**.

The read-only properties are displayed in the Properties window in Visual Studio .NET. The following table contains descriptions for the properties.

Use this	To do this
Name	Contains the name of the role node.
Service Link Type	Contains the name of the service link type to which this role belongs.

How to Browse Implemented Roles of an Orchestration Using BizTalk Explorer

All of the implemented roles of an orchestration can be viewed from within BizTalk Explorer.

To browse implemented roles of an orchestration

1. In Visual Studio .NET, in a BizTalk Server project, on the **View** menu, click **BizTalk Explorer**.
2. In BizTalk Explorer, expand the **Orchestrations** collection, expand the orchestration, and then expand the **Implemented Roles** collection.

From the **Implemented Roles** collection, you can review its read-only properties.

To review the read-only properties for implemented roles

1. In Visual Studio .NET, in a BizTalk Server project, on the **View** menu, click **BizTalk Explorer**.
2. In BizTalk Explorer, expand the **Orchestrations** collection, expand the orchestration, expand the **Implemented Roles** collection, right-click the role for which you want to view properties, and then click **Properties**.

The read-only properties are displayed in the **Properties** window in Visual Studio .NET. The following table contains descriptions for the properties.

Use this	To do this
Name	Contains the name of the role node.
Service Link Type	Contains the name of the service link type to which this role belongs.

How to Browse Invoked Orchestrations Using BizTalk Explorer

All the Called or Executed orchestrations of an orchestration can be viewed from within BizTalk Explorer.

To browse invoked orchestrations

1. In Visual Studio .NET, in a BizTalk Server project, on the **View** menu, click **BizTalk Explorer**.
2. In BizTalk Explorer, expand the **Orchestrations** collection, expand the orchestration, and then expand the **Invoked Orchestrations** collection.

From the **Invoked Orchestrations** collection, you can review its read-only properties.

To review the read-only properties for invoked orchestrations

1. In Visual Studio .NET, in a BizTalk Server project, on the **View** menu, click **BizTalk Explorer**.
2. In BizTalk Explorer, expand the **Orchestrations** collection, expand the orchestration, expand the **Invoked Orchestrations** collection, right-click the orchestration for which you want to view properties, and then click **Properties**.

The read-only properties are displayed in the Properties window in Visual Studio .NET. The following table contains descriptions for the properties.

Use this	To do this
Name	Contains the name of the invoked orchestration node.
Assembly	Contains the name of the assembly that contains this orchestration.

Status	Specifies the status of the orchestration.
---------------	--------------------------------------------

How to Bind an Orchestration Using BizTalk Explorer

You can use BizTalk Explorer to bind an orchestration to a BizTalk Server port, such as a receive port or send port, on a per-message per-port level. For example, a one-way inbound port matches a one-way receive port; a request-response inbound port matches a request-response receive port.

An orchestration is fully bound only when all its ports are bound.

To bind an orchestration

1. In Visual Studio .NET, in a BizTalk Server project, on the **View** menu, click **BizTalk Explorer**.
2. In BizTalk Explorer, expand the **Orchestrations** collection, right-click the orchestration you want to bind, and then click **Bind**.
3. If any of the orchestration ports can be bound, in the **Port Binding Properties - <Orchestration> - Configuration - Binding**, in the right pane, select a send port or receive location for each orchestration port. If this dialog box does not appear, the orchestration was bound to its ports in Orchestration Designer.
4. In the **Port Binding Properties - <Orchestration> - Configuration - Host** dialog box, in the right pane, select a host for the orchestration.

The orchestration is bound; you can now enlist it. For instructions on enlisting an orchestration, see [Enlisting and Starting an Orchestration Using BizTalk Explorer](#).

How to Enlist and Start an Orchestration Using BizTalk Explorer

After you bind an orchestration in BizTalk Explorer, you must enlist the orchestration in a host. You must specify the host to enlist the orchestration.

Consider the following items when starting an orchestration:

- An orchestration cannot be started unless all of the invoked orchestrations are started.
- You can start the orchestration only if you enlisted or started the dependent send ports.
- Starting an orchestration fails if any of the receive locations for the orchestration are not completely configured.
- The receive locations can be disabled or enabled while the orchestration is running.
- Enlistment does not work from a remote computer unless you have the assembly in the local GAC. After you have enlisted it from the computer that has the assembly in its GAC, then you will be able to start or stop the orchestration from the remote computer.

To enlist an orchestration

1. In Visual Studio 2005, in a BizTalk Server project, on the **View** menu, click **BizTalk Explorer**.
2. In BizTalk Explorer, expand the **Orchestrations** collection, right-click the orchestration you want to enlist, and then click **Enlist**.

To start an orchestration

1. In Visual Studio 2005, in a BizTalk Server project, on the **View** menu, click **BizTalk Explorer**.
2. In BizTalk Explorer, expand the **Orchestrations** collection, right-click the orchestration you want to start, and then click **Start**.
3. To select any other artifacts that you want to start at the same time, in the **BizTalk Explorer - Express Start** dialog box, do the following.

Use This	To do this
Receive Locations	Start all dependent receive locations.
SendPorts	Start all dependent send ports.
SendPortGroups	Start all dependent send port groups.
BizTalk Host	Start all dependent hosts.
Resume existing Orchestration Instances	Resume any Orchestration instances of this type that were previously suspended (paused) allowing them to complete processing.

4. Click **OK**.

The orchestration, any selected artifacts, and the subscriptions that are associated with the orchestration start.

How to Stop and Unenlist an Orchestration Using BizTalk Explorer

You can stop and unenlist an orchestration by using BizTalk Explorer. Stopping an orchestration deactivates the incoming message subscription, immediately suspending all messages routed to the orchestration. Unenlisting an orchestration removes the orchestration from the host.

To stop an orchestration

1. In Visual Studio .NET, in a BizTalk Server project, on the **View** menu, click **BizTalk Explorer**.
2. In BizTalk Explorer, expand the **Orchestrations** collection, right-click the orchestration you want to stop, and then click **Stop**.

The orchestration stops; you can now unenlist it.

To unenlist an orchestration

1. In Visual Studio .NET, in a BizTalk Server project, on the **View** menu, click **BizTalk Explorer**.
2. In BizTalk Explorer, expand the **Orchestrations** collection, right-click the orchestration you want to unenlist, and then click **Unenlist**.

The orchestration is unenlisted.

Managing Parties Using BizTalk Explorer

You can use BizTalk® Explorer to manage parties and to enlist parties in roles. Roles are part of an orchestration, just like an orchestration port. Parties are created and managed in Explorer, like receive locations and send ports. Enlisting a party under roles is like binding the orchestration port to a receive location or send port.

In This Section

- Adding a Party Using BizTalk Explorer
- Managing Party Send Ports Using BizTalk Explorer
- Adding an Alias to a Party Using BizTalk Explorer
- Managing Party Signature Certificates Using BizTalk Explorer
- Deleting a Party Using BizTalk Explorer

How to Add a Party Using BizTalk Explorer

You can use BizTalk Explorer to add parties to a BizTalk Management database.

To add a party

1. In Microsoft® Visual Studio® .NET, in a Microsoft® BizTalk Server project, on the **View** menu, click **BizTalk Explorer**.
2. In BizTalk Explorer, right-click on the **Parties** collection, and then click **Add Party**.
3. In the **Party Properties** dialog box, in the **Name** box, type a name for the party, and then click **OK**.
4. The party is created; you can now edit the properties of the party and enlist it in a role.

How to Manage Party Send Ports Using BizTalk Explorer

You can use BizTalk Explorer to add send ports to a party, and delete send ports from a party.

To add a send port to a party

1. In Visual Studio .NET, in a BizTalk Server project, on the **View** menu, click **BizTalk Explorer**.
2. In BizTalk Explorer, expand the **Parties** collection, right-click the party to which you want to add a send port, and then click **Edit**.
3. In the **Party Properties** dialog box, in the **Send Ports** section, in the right pane, click the first blank row under **Send Port**.
4. Select a send port from the drop-down list, and click **OK**.

To delete a send port from a party

1. In Visual Studio .NET, in a BizTalk Server project, on the **View** menu, click **BizTalk Explorer**.
2. In BizTalk Explorer, expand the **Parties** collection, right-click on the party from which you want to delete a send port, and then click **Edit**.
3. In the **Party Properties** dialog box, in the left pane, select the **Send Ports** node.
4. In the **Party Properties** dialog box, in the right pane, click the row selector to select the port, and then click **Delete** on the toolbar.

How to Add an Alias to a Party Using BizTalk Explorer

You can use BizTalk Explorer to manage party aliases.

An alias is a unique identifier for a party, and is used by BizTalk Server to retrieve the party at run time. An alias is composed of three elements:

- **Name.** Used to describe the format of the alias. For example telephone, or Microsoft security identifier (SSID).
- **Qualifier.** Used to create a unique key for the alias.
- **Value.** Used to identify the parties to some external organization.

For more information about names, qualifiers, and values, see Partner Management in BizTalk Explorer.

The qualifier-value pair is unique across the organization.

To add an alias to a party

1. In Visual Studio .NET, in a BizTalk Server project, on the **View** menu, click **BizTalk Explorer**.
2. In BizTalk Explorer, expand the **Parties** collection, right-click the party for which you want to create an alias, and then click **Edit**.
3. In the **Party Properties** dialog box, in the **Aliases** section, do the following.

Use this	To do this
Name	Type or select the name from the drop-down list.
Qualifier	Select the qualifier to distinguish between different aliases with the same value. The qualifier-value pair is unique for the entire organization.
Value	Type the name of the alias.

4. Click **OK**.

An alias is added to the party. BizTalk Server can now perform party resolution based on the alias.

You can use BizTalk Explorer to associate public key certificates with a party. BizTalk Server can then use the public key certificate to verify digital signatures on inbound messages and to encrypt outbound messages to the party.

How to Manage Party Signature Certificates Using BizTalk Explorer

You can use BizTalk Explorer to manage party signature certificates.

To select a signature certificate for a party

1. In Visual Studio .NET, in a BizTalk Server project, on the **View** menu, click **BizTalk Explorer**.
2. In BizTalk Explorer, expand the **Parties** collection, right-click on the party to which you want to add a signature certificate, and then click **Edit**.
3. In the **Party Properties** dialog box, in the **Certificate** section, select a signature certificate from the drop-down list, and then click **OK**.

The certificate is associated with the party. BizTalk Server can now resolve the party based on the certificate. For more information about certificates, see **Certificates that BizTalk Server 2006 Uses for Signing Messages**

How to Delete a Party Using BizTalk Explorer

You can use BizTalk Explorer to delete parties that are not enlisted under a role.

To delete a party

1. In Visual Studio .NET, in a BizTalk Server project, on the **View** menu, click **BizTalk Explorer**.
2. In BizTalk Explorer, expand the **Parties** collection, right-click the party that you want to delete, and then click **Delete**.
3. In the **BizTalk Explorer** dialog box, click **Yes** to confirm the deletion.

The party is deleted from the BizTalk Management database.

Enlisting a Party Using BizTalk Explorer

Party enlistment is very similar to orchestration binding because both require that the orchestration role port types are bound to party ports. Thus, all the rules used for orchestration binding relate to party enlistment. However, there are a few points specific to party enlistment that differ from orchestration binding:

- Party enlistment deals only with send ports, because the orchestration role receive ports would bind to locations that are local to the administration group, and are not within the party.
- Party enlistment allows for per-operation level mapping to party ports, as opposed to orchestration port binding where the binding (or mapping) occurs at a port level. However, the rules for which types of party send ports show up apply in the same sense as for orchestration binding as described in the following rules.
- Send port groups cannot be used for party enlistment, because Microsoft® BizTalk® Server 2006 does not support send port groups belonging to a party.
- Each operation for an orchestration role send port type is mapped to a party send port.
- If the operation is one way, both one-way and solicit-response party send ports are shown for that operation (and the receive pipeline in the solicit-response party send port that is picked is ignored).
- If the operation is solicit-response, only solicit-response party send ports are shown for binding during party enlistment.

In This Section

- Browsing Roles Using BizTalk Explorer
- Enlisting Parties Using BizTalk Explorer
- Unenlisting Parties Using BizTalk Explorer

How to Browse Roles Using BizTalk Explorer

Roles used in all orchestrations deployed to any BizTalk Management database can be viewed from within BizTalk Explorer.

To browse roles

1. In Microsoft® Visual Studio® .NET, in a BizTalk Server project, on the **View** menu, click **BizTalk Explorer**.
2. In BizTalk Explorer, expand the **Roles** collection.

From the **Roles** collection, you can enlist the party and review its read-only properties.

To review the read-only properties for a role

1. In Visual Studio .NET, in a BizTalk Server project, on the **View** menu, click **BizTalk Explorer**.
2. In BizTalk Explorer, expand the **Roles** collection, right-click the role, and then click **Properties**.

The read-only properties are displayed in the **Properties** window in Visual Studio .NET. The following table contains descriptions for the properties.

Use this	To do this
Name	Contains the name of the role.
Assembly Name	Contains full name of the assembly that contains the orchestration.
Role Link Type	Contains the name of the link type to which the role belongs

How to Enlist Parties Using BizTalk Explorer

You can use BizTalk Explorer to enlist a party under a role. The enlisted party appears as a node under the role.

To enlist a party

1. In Visual Studio .NET, in a BizTalk Server project, on the **View** menu, click **BizTalk Explorer**.
2. In BizTalk Explorer, expand the **Roles** collection, right-click the role under which you want to enlist a party, and then click **Enlist Party**.
3. In the **Select Party** dialog box, select a party from the drop-down list, and then click **OK**.

4. In the **Enlisting Party Properties - <Party> - Configurations -Binding** dialog box, in the right pane, click the down arrow button to select a send port for each outbound role port, and then click **OK**.

How to Unenlist Parties Using BizTalk Explorer

You can use BizTalk Explorer to unenlist a party from a role. Note, however, that this does not remove the party from the BizTalk Management database.

To remove a party enlistment

1. In Visual Studio .NET, in a BizTalk Server project, on the **View** menu, click **BizTalk Explorer**.
2. In BizTalk Explorer, expand the **Roles** collection and the role, right-click the party for which you want to remove the enlistment, and then click **Remove**.
3. In the **BizTalk Server** dialog box, click **Yes**.

The party is removed from under the role.

How to Refresh BizTalk Explorer

When changes are made to any of the artifacts in the BizTalk Management database outside of the BizTalk® Explorer user interface (UI), the UI must be refreshed for the changes to appear.

To refresh the BizTalk Management database

1. In Microsoft® Visual Studio® .NET, in a Microsoft® BizTalk Server project, on the **View** menu, click **BizTalk Explorer**.
2. In BizTalk Explorer, right-click the BizTalk Management database in which the changes were made, and then click **Refresh**.

The BizTalk Management database artifacts are refreshed in BizTalk Explorer.

Automating Business Process Management

This section describes using the BizTalk® Explorer Object Model from managed code. Developers can use the BizTalk Explorer Object Model to build administrative tools and to dynamically configure BizTalk Explorer objects at run time.

In This Section

- Using the BizTalk Explorer Object Model from Managed Code
- Common Programmatic Tasks Using the BizTalk Explorer Object Model

Using the BizTalk Explorer Object Model from Managed Code

The BizTalk Explorer Object Model is a managed object model that enables programmatic configuration of BizTalk artifacts in the Trading Partner Management database. It enables all the forms of programmatic access that any .NET programming language can provide. This includes programming from any other common language runtime (CLR)-compliant programming language (for example, C#, Visual Basic® .NET, Managed C++, and so on).

The BizTalk Explorer tool available in Microsoft® Visual Studio® .NET makes it easy to manage small numbers of ports, orchestrations, parties, and so on. But when it comes time to add thousands of parties, or delete hundreds of ports, you need a way to write code to perform these tasks for you. This is what the BizTalk Explorer Object Model provides.

All of the APIs in the BizTalk Explorer Object Model are in the **Microsoft.BizTalk.ExplorerOM** namespace, and can be referenced by adding the **Microsoft.BizTalk.ExplorerOM.dll** to your project. The samples in the Common Programmatic Tasks Using the BizTalk Explorer Object Model section provide four complete samples for the most common tasks, and links to the samples that appear in the Explorer Object Model reference pages.

Common Programmatic Tasks Using the BizTalk Explorer Object Model

The topics in this section illustrate common tasks using the BizTalk Explorer Object Model.

In This Section

- Browsing BizTalk Server Assembly Artifacts Using the BizTalk Explorer Object Model
- Orchestration Binding Using the BizTalk Explorer Object Model
- Partner Management Using the BizTalk Explorer Object Model
- Content-Based Routing Using the BizTalk Explorer Object Model
- Managing Artifacts Using the BizTalk Explorer Object Model

Browsing BizTalk Server Assembly Artifacts Using the BizTalk Explorer Object Model

Use the following sample to browse assembly artifacts using the BizTalk Explorer Object Model.

The **EnumerateOrchestrationArtifacts** method enumerates all the orchestrations, and all of the ports, roles, port type operations, used roles, and implemented roles within each orchestration.

The **EnumerateArtifactSample** method connects to the BizTalk Management database, and enumerates all of the deployed assemblies, configured hosts, parties, send port, and aliases.

Orchestration Binding Using the BizTalk Explorer Object Model

The C# code snippets in this topic show how to use the BizTalk Explorer Object Model to do the following:

- Create an instance of the root object of the BizTalk Explorer Object Model. In other words, how to instantiate **BtsCatalogExplorer**.
- Bind orchestration ports to send ports, receive ports, and send port groups.
- Enlist an orchestration.
- Start an orchestration.
- Stop an orchestration.
- Unenlist an orchestration.
- Unbind orchestration ports.
- Commit a set of operations done on the BizTalk Explorer Object Model objects in a transactional manner (that is, either all the changes will be committed or rolled back).

The **BindOrchestration** method shows how to bind an orchestration to a send port or send port group.

The **EnlistOrchestration** method shows how to create the root **BtsCatalogExplorer** object, set the BizTalk Management database connection string, set the orchestration status to enlisted, and commit the changes to the database.

The **StartOrchestration** method shows how to create the root **BtsCatalogExplorer** object, create, set the BizTalk Management database connection string, set the orchestration status to Started, and commit the changes to the database.

The **StopOrchestration** method shows how to create the root **BTSCatalogExplorer** object, set the BizTalk Management database connection string, set the orchestrationstatus to Enlisted, and commit the changes.

The **UnenlistOrchestration** method shows how to create the root **BTSCatalogExplorer** object, set the BizTalk Management database connection string, set the orchestration status to Unenlisted, and commit the changes.

The **UnbindOrchestration** method shows how to create the root **BTSCatalogExplorer** object, set the BizTalk Management database connection string, reset the orchestration bindings, and commit the changes.

Partner Management Using the BizTalk Explorer Object Model

The following C# code snippets illustrate the following concepts of using the BizTalk Explorer Object Model:

- How to create parties.
- How to specify standard aliases and custom aliases for a party.
- How to configure send ports for a party.
- How to browse certificates.
- How to specify a signature certificate for a party.
- How to browse assemblies and roles.
- How to enlist parties under a role.
- How to map party ports to the role port types (operations).
- How to unbind the party ports.
- How to unenlist a party from a role.
- How to delete the party.
- How to commit a set of operations done on the object model objects in a transactional manner (that is, either all the changes will be committed or rolled back).

The **CreateParty** method shows how to create the root **BTSCatalogExplorer** object, set the BizTalk Management database connection string, create a party, create an alias, create send ports, specify a signature certificate, and commit the changes.

The **EnlistParty** method shows how to create the root **BTSCatalogExplorer** object, set the BizTalk Management database connection string, search for a role, enlist a party under the role, and commit the changes.

The **Unenlist** method shows how to create the root **BTSCatalogExplorer** object, set the BizTalk Management database connection string, search for a role, remove an enlisted party, and commit the changes.

The **DeleteParty** method shows how to create the root **BTSCatalogExplorer** object, set the BizTalk Management database connection string, delete a party, and commit the changes.

Content-Based Routing Using the BizTalk Explorer Object Model

The following C# sample shows how to route messages based on their content, using the BizTalk Explorer Object Model. It shows how to do the following:

- Use filters and maps on send ports.
- Perform content based routing.
- Perform document normalization.

For more information about the message context properties that you can promote for content-based routing, see **Message Context Properties**.

The **CBRSample** method shows how to route messages to a US or Canadian (CAN) processing location depending on the CountryCode code indicated in the received PO document.

Managing Artifacts Using the BizTalk Explorer Object Model

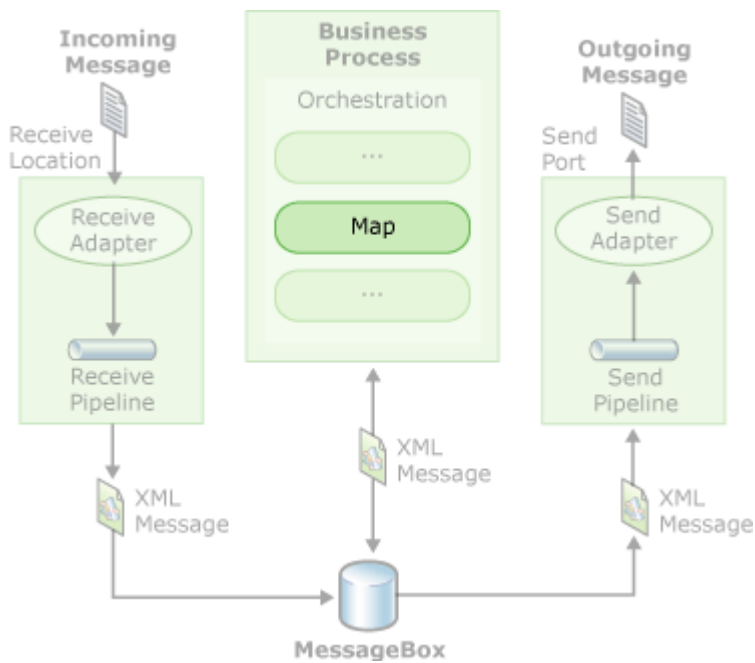
Samples to manage each of these artifacts can be found on the following reference pages for the artifacts:

- **SendPort Class**
- **SendPortGroup Class**
- **ReceivePort Class**
- **ReceiveLocation Class**

Creating Maps Using BizTalk Mapper

BizTalk® Mapper is a tool that runs within the Microsoft® Visual Studio® .NET environment. You use Mapper to create and edit maps to translate or to transform messages. Maps are used in orchestrations, as the following figure suggests, and may also be used in send port message processing.

Maps in Business Processing



Maps enable you to translate and to transform messages. Translation is the process of converting a message from one format to another format, such as converting a flat file into an XML file. Transformation is the process of taking information from one message and inserting it in another message. For example, you might take shipping and billing addresses from a purchase order and insert them in an invoice document.

BizTalk Mapper uses its own graphical system of icons and links to represent the translation and transformation of input instance messages to output instance messages. Mapper uses the same graphical representation of schemas as BizTalk Editor. BizTalk Mapper stores its maps as Extensible Stylesheet Language Transformations (XSLT) stylesheets.

For information about creating schemas, see [Creating Schemas Using BizTalk Editor](#) . For information about using maps within orchestrations, see [Creating Orchestrations Using Orchestration Designer](#) . For instructions about using maps in send port processing, see [Managing Business Processes Using BizTalk Explorer](#)

In This Section

- Planning to Create Maps
- About Maps
- Using BizTalk Mapper
- Creating Maps
- Compiling and Testing Maps

Planning to Create Maps

You use maps to convert input messages conforming to one schema into output messages conforming to a different schema. These conversions may be simple or quite complex, involving calculations and consolidation of information.

The following table lists some of the questions that you need to answer when you plan to create maps.

Planning question	Recommendation
What maps do I need to create?	<p>Make a list of the business documents that you will process with Microsoft® BizTalk® Server 2006. Such a list might include, for example, a purchase order, an invoice, a shipping confirmation, and so on. The list might also include more than one of each such business document, such as when the structure of a purchase order you receive from one trading partner is different than the structure of a purchase order you receive from another trading partner.</p> <p>Go through the list and decide which of the documents need to be in a different format, or which documents are best handled by conversion to a common format.</p>
Where do I need maps?	You can use maps in send ports as well as in orchestrations representing business processes. Consider where you want or need to convert documents.
Do I have old maps to convert?	<p>Many maps from previous versions of BizTalk should migrate without modification. Allocate enough time to test the migrated maps.</p> <p>Maps containing Scripting functoids or custom functoids may require additional work. For more information, see Migrating Functoids.</p>

About Maps

Using BizTalk® Mapper, you define the relationship between an input and an output schema using links and functoids. A link defines a direct data copy of a record or field. Links may directly connect to items in the other schema, or they may form connections to functoids. Functoids perform more complex data manipulations. Examples include:

- Adding the value of two fields in the source schema and copying the result to the destination schema.
- Converting a character to its ASCII format.
- Returning the average of a field in a repeating record and copying the result to a field in the destination schema.

BizTalk® Mapper stores maps in a file with a .btm extension. The file saves design information about the map—the locations of icons representing functoids, the links between schema items and functoids, and other information about the map. When you build or compile the map, BizTalk Mapper converts the information about the map into the corresponding XSLT stylesheet.

The maps that you create can transform or translate data, and they can be specific to a single trading partner or be used with many trading partners. The topics in this section provide an introduction to the concepts involved in mapping schemas.

In This Section

- Overview of Maps
- Links in Maps
- Functoids in Maps
- Map Compilation and Testing

Overview of Maps

Maps consist of several underlying components and depend on schemas you have already created. These schemas in turn rely on XML Schema definition (XSD) language. This section provides an overview of these and other critical topics, such as migrating maps from previous versions of Microsoft® BizTalk® Server 2006.

In This Section

- Map Components
- Transformation vs. Translation
- Source and Destination Schemas
- Types of Mapping Operations
- Specific vs. Generic Maps
- Valid BizTalk Mapper XSLT Encoding Types
- Order of Records and Fields
- XSD Element Groups
- Map Migration from Previous Versions of BizTalk Server
- The Map Grid and Its Pages
- Custom XSLT

Map Components

Files with a .btm extension store most of the components of a map. Items stored in the file include:

- References to source and destination schemas

- Persisted links, including the link properties
- Persisted functoids with their properties such as input parameters
- Other miscellaneous properties such as those associated with the grid and the map itself.

Although BizTalk Mapper compiles the map in the .btm file into an Extensible Stylesheet Transformations (XSLT) file, the XSLT is not part of the file. BizTalk Mapper produces the XSLT for the map only when you compile the project or when you validate the map. BizTalk Mapper packages the XSLT as part of the project assembly.

Transformation vs. Translation

Mapping data relies on two basic operations: data transformation and data translation.

Data transformation is the process of creating a correspondence between records and fields of a source schema to (often different) records and fields in a destination schema. An example of data transformation is to map shipping and billing address information from a purchase order to an invoice. This is the most basic type of mapping. Data transformation can also apply to operations such as:

- Averaging data from a looping record and sending the output to a single field in the destination schema.
- Converting character data to its ASCII format.
- Adding or subtracting data from one or more records and putting the result in a single field in the destination schema.

Data translation is the process of changing the format of an instance message. Data translation is very helpful in solving enterprise application integration problems by translating a given type of message into alternative formats required by existing systems.

Source and Destination Schemas

Each BizTalk map uses two schemas: a source schema and a destination schema. A source schema defines the structure of the instance messages from which you are taking data. The destination schema defines the structure of the instance messages the map produces. For example, if you want to map the shipping and billing information from a purchase order to an invoice, you need a schema to define purchase orders for the source schema and a schema defining invoices for the destination schema.

Schemas used in BizTalk maps must meet the following conditions:

- The source and destination schemas need to be a part of your current BizTalk project, or you must include a reference to the schemas in the assembly so that the schemas can be accessed during run time.
- The schemas used in BizTalk Mapper must be based on the XML Schema definition (XSD) language. BizTalk Editor provides an easy way to create such schemas. For more conceptual, procedural, and

reference information about creating schemas with BizTalk Editor, see [Creating Schemas Using BizTalk Editor and Creating Schemas](#).

In BizTalk Editor, you can create schemas with multiple root nodes. However, if you use a schema with multiple root nodes in a BizTalk map, you must choose which root node (and corresponding substructure) to use in the map. Schemas have a **Root Reference** property identifying which root is primary. If a schema has multiple roots and the **Root Reference** property is set when the schema is first opened as the source or destination schema, BizTalk Mapper uses the specified root. If a schema has multiple roots and the **Root Reference** property is not set, BizTalk Mapper prompts you to choose a root.

If you change the **Root Reference** property of a schema already used in a map, BizTalk Mapper does not notice the change and continues to use the originally specified root.

It is not good practice to have different roots set as the value of the **Root Reference** property in the schema and to have the map use the root. If you want to build different maps using different roots of the same schema, it is best not to set the **Root Reference** property. That way, whenever the schema is used for a new map, you will need to explicitly choose the root.

Editing a schema after including it in a map may result in the loss of links within the map.

Types of Mapping Operations

BizTalk Mapper provides solutions for a variety of mapping scenarios ranging from simple parent-child tree-type operations to detailed, complex operations involving looping records and hierarchies. The complexity of a mapping scenario largely depends on your preferences and business needs—XML Schema definition (XSD) language gives you considerable flexibility in defining structured formats. Almost all mapping scenarios fall into one of two categories: basic mapping and complex mapping.

Basic Mapping

Basic mapping is the most common type of mapping you can create. In a basic map, input and output items have a one to one relationship. An input item maps to one and only one output item. Although many types of transformations and translations are possible with basic mapping, such as using multiple functoids and cascading functoids to manipulate the value being copied, the underlying scenario remains relatively simple. Basic mapping operations also include mapping fields from two different parent records (occurring only once) to fields under a single parent record in the destination schema.

Complex Mapping

Complex mapping involves records or fields that occur multiple times for a single instance of the **Record** or **Field Element** node in the schema tree. Such nodes have their **Max Occurs** property set to a value greater than one (1), indicating there may be more than one corresponding element in an instance message. When a BizTalk map uses this type of variable count mapping (also known as looping), the Extensible Stylesheet Language (XSL) style sheet compiler must be able to determine the proper loop path over which to iterate to produce the required output.

In general, you can link a field in a looping record in the source schema to a field in a looping record in the destination schema. The number of corresponding elements in an input instance message drives the number of elements created in the output instance message. For example, consider the following XSD fragments from hypothetical source and destination schemas.

Source schema fragment

Destination schema fragment

In these fragments:

- **SrcLoopingRecord**, a **Record** node in input instance messages, can occur from 1 to 5 times. It also contains child **Field Element** nodes **Field1** (a string) and **Field2** (an integer) that occur once for each instance of their parent.
- **DstLoopingRecord**, a **Record** node in output instance messages, can occur 0 or more times, unbounded. It also contains child **Field Element** nodes **FieldA** (a string) and **FieldB** (an integer) that occur once for each instance of their parent.

Assuming that Field1 is mapped to FieldA and Field2 is mapped to FieldB, and that the following fragment from an input instance message has processed those mappings, the following fragment from an output instance message would be produced.

Input instance message fragment

Output instance message fragment

The number of occurrences of the **SrcLoopingRecord** element in the input instance message (3) drives the number of occurrences of the **DstLoopingRecord** element in the output instance message.

A type of mapping not supported by BizTalk Mapper is having multiple loop paths. This type of mapping involves fields from two or more looping records in the source schema being mapped to fields within a single looping record in the destination schema. This presents a problem—there is no effective way to determine the number of elements to produce in the output instance message. Multiple loop paths result in a map compilation warning indicating that the destination node has multiple source loop paths. However, this is only a warning, and the number of iterations in the first source loop path are used to determine the number of elements produced in the output instance message. You can take explicit control of looping behavior by using the **Looping** functoid.

Microsoft® BizTalk® Server 2006 introduces a new kind of looping called table-driven looping. Table-driven looping is useful when your output instance message needs to be based on data from the input instance message combined with one or more constants, links from the source schema, or functoids. In such cases, the output instance message can have multiple records based on data from a single record in the input instance message that is combined with different constants, or based on data coming from multiple records in the input instance message. For more information about table-driven looping using the **Table Looping** and **Table Extractor** functoids, see Table Looping and Table Extractor Functoids.

Specific vs. Generic Maps

When you create a map that transforms or translates data, there are two approaches that you can take: specific or generic. You use a specific map to meet the needs of a particular trading partner. Use a specific map when you have very specific business needs or business agreements with your trading partner. The advantage of a specific map is that it is customized to meet the needs of the business functions you have with specific trading partners. The map itself is easier to maintain because you need only define the map to consider the requirements for one trading partner. The disadvantage of a specific map is that it usually cannot be used with more than one trading partner. Thus, if you have multiple trading partners, you will

need more maps. If you multiply the number of trading partners by the number of different message types exchanged with those trading partners, you will need to allocate enough time and resources to manage all of the associated maps.

Generic maps, on the other hand, are designed to be used with multiple trading partners. Therefore, instead of developing and maintaining multiple schemas for a particular business document, you create one schema for each type of business document, and then use them with all of your trading partners. While using one map with multiple trading partners saves time and resources, these maps are not customized and may not meet the needs of all cases.

The maps that you create will likely be a mix of specific and generic maps depending on the nature of your business.

Valid BizTalk Mapper XSLT Encoding Types

BizTalk Mapper supports different types of XSLT encoding. You use the **XSLT Encoding** grid property to set the XSLT encoding type that you prefer. The following list shows the encoding formats that are available in the drop-down list associated with the **XSLT Encoding** grid property:

- None
- UTF-8
- Arabic (1256)
- Baltic (1257)
- Central-European (1250)
- Chinese (GB18030)
- Cyrillic (1251)
- Greek (1253)
- Hebrew (1255)
- Japanese-Shift-JIS (932)
- Korean (949)
- Little-Endian-UTF16 (1200)
- Simplified-Chinese-GBK (936)
- Thai (874)

- Traditional-Chinese-Big5 (950)
- Turkish (1254)
- Vietnamese (1258)
- Western-European (1252)

If you do not find the encoding you want to use in this list, you can enter a different encoding value. Make sure the value you provide is valid because BizTalk Mapper does not test the value you enter.

Order of Records and Fields

Extensible Stylesheet Language Transformations (XSLT), as used by BizTalk Mapper, does not guarantee the output order of output elements and attributes. This is because BizTalk Mapper generates XSLT by examining the destination schema structure, and then propagating back through the grid pages to extract values from the source schema structure. For example, if you want to create an output file that has the BillTo Address record listed before the ShipTo Address record, you must ensure that the BillTo Address precedes the ShipTo Address record in the destination schema.

XSD Element Groups

The use of certain structures in a schema may create variations in the Extensible Stylesheet Language Transformations (XSLT) that BizTalk Mapper generates. One case is including a schema in your map that defines sequence, choice, or all element groups. For example, if you use a schema that includes a **Choice Group** node, it is possible for you to create a map that requires two or more of the children of the **Choice Group** node to appear in an output instance message. In this case, BizTalk Mapper displays a warning when you compile the map. The warning tells you that only one of the required fields you have mapped may be populated in the same iteration of the parent loop at run time. BizTalk Mapper will not give you an error that your mapping logic is incorrect.

Another situation in which you might generate variations in the XSLT is when the following conditions are met:

- **Record A** has a child **Field Element B**.
- **Record A** and child **Field Element B** occur once.
- **Record A** is part of a **Choice Group** that repeats.

In this situation, BizTalk Mapper generates XSLT that contains iteration logic to handle the possibility of the many variations of the source records.

Map Migration from Previous Versions of BizTalk Server

You can open maps created in previous versions of BizTalk Server in the version of BizTalk Mapper included in Microsoft BizTalk Server 2006. Before opening such a map, you must change its file extension to .btm and add it to a BizTalk project.

Because of the differences between Microsoft BizTalk Server 2006 and previous versions of BizTalk Server, you must take the following manual steps:

- Maps created in previous versions of BizTalk Server did not have a paged grid—all links and functoids are placed in a single grid page. If you want to organize these links and functoids onto different grid pages, you will need to create the grid pages and move the links and functoids manually. For more information about creating grid pages, and moving functoids and links from one grid page to another, see *Working with Grid Pages*.
- Source and destination schemas were embedded in the maps themselves in previous versions of BizTalk maps. Now, the schemas are separate files referenced by the maps. When you open a map created in a previous version of BizTalk Server, BizTalk Mapper extracts the source and destination schemas from the map and processes these schemas with an XDR-to-XSD transformation. This creates separate schema files in the file system. You must add the schema files manually to your BizTalk project. The schemas can be added to the same BizTalk project as the map, or they can be added to a different BizTalk project. After the files are added to the appropriate BizTalk project, the source and destination schemas in the migrated map are replaced with references and the map source is updated to represent the new structure.
- For more information about adding schemas to a BizTalk project, see *Adding Existing Schemas*.
- BizTalk Server 2006 does not support the migration of maps containing custom Extensible Stylesheet Language Transformations (XSLT) code. For more information about the custom XSLT features of BizTalk Server 2006, see *Custom XSLT*.

The Map Grid and Its Pages

The links that connect nodes in the source and destination schemas, representing the data transformations that will be performed at run time, are displayed graphically in the Grid view. In previous versions of BizTalk Mapper, this view was called the mapping grid, and all links and functoids appeared on that single, two-dimensional surface. In Microsoft® BizTalk® Server 2006, the Grid view consists of multiple pages, called grid pages, or sometimes layers. Multiple pages enable you to organize your links so that any one page does not become too cluttered and confusing.

New maps begin with a single grid page, but you can easily add new grid pages as necessary to keep your map organized and easy to maintain. You can name grid pages to describe their purpose. You can reorder them so that related grid pages are near each other in the tab order. And, most importantly, you can move links and functoids from one grid page to another grid page, enabling you to reorganize your map over time as new requirements emerge.

Custom XSLT

If you are familiar with Extensible Stylesheet Language Transformations (XSLT) code, you can use it to customize, extend, or replace BizTalk maps. The simplest way to use XSLT is with the **Scripting** functoid. The **Scripting** functoid accepts scripts in many .NET languages, including XSLT. For more information about using XSLT with the **Scripting** functoid, see *Scripting Using Inline XSLT and XSLT Call Templates*.

If you have XSLT code you have been using to convert one instance message into another, you can use that code directly in place of a BizTalk map. For information about replacing BizTalk maps with your XSLT code, see *Mapping without Maps*.

Links in Maps

Links specify the basic function of copying data from an element or attribute in an input instance message to an element or attribute in an output instance. You create links between records and fields in the source and destination schemas at design time. This drives the creation, at run time, of an output instance message conforming to the destination schema from an input instance message conforming to the source schema.

BizTalk® Mapper supports one-to-one links and one-to-many links. For example, a link can connect a single record or field from the source schema to a single record or field in the destination schema. A link can also connect a single record or field from the source schema to multiple records or fields in the destination schema.

Links can also connect multiple records or fields from the source schema to a functoid, which then connects to a single record or field in the destination schema. In general, direct links from multiple source records or fields to a single destination record or field are not valid and produce a warning. One exception to this is the **Looping** functoid. For more information about the **Looping** functoid, see [Looping Functoid](#).

The topics in this section describe concepts related to creating and working with links in BizTalk Mapper.

In This Section

- Types of Links
- Link Creation
- Link Configuration
- Record-to-Record Linking
- Links To and From the Any Element and anyAttribute Nodes
- Compiler Directives and Links

Types of Links

The following list summarizes the various types of links available in BizTalk Mapper:

- **Elastic links.** The term elastic applies to a link as it is being created, before both ends of the link are connected. For example, if you are dragging a link from a field in the source schema, but you have not yet connected it to its corresponding field in the destination schema, the link is elastic. As you complete the connection, the elastic becomes a fixed link.
- **Fixed Links.** The term fixed applies to a link that you have explicitly constructed to represent the movement of a value from a source instance message to a destination instance message, or at least a part of that movement. Fixed links that directly connect a **Record** or **Field** node in the source schema to a **Record** or **Field** node in the destination schema represent a straight copying of data at run time. Fixed links connecting to a functoid at one end or the other represent part of the movement of data from an input instance message to an output instance message at run time. Several of these

together, completing the link between the source and destination schemas, represent the entire movement of an item of data.

- **Partial links.** The term partial applies to links for which you cannot currently see the exact **Record** or **Field** node to which they are connected in the source or destination schemas. This occurs when the **Record** or **Field** node to which they are attached is not shown because one of its ancestor nodes is collapsed in the tree representation of the schema.
- **Compiler links.** The term compiler applies to links that BizTalk Mapper creates automatically when you build a BizTalk project. For example, if you configure table-driven looping, the compiler links show the relationship and the links between the records and fields in the source schema and the records and fields in the destination schema. This type of link can be generated automatically by compiler directives, or it can be user-directed.

BizTalk Mapper, by default, displays these various types of links using different colored lines to help you distinguish the detail of your maps. You can change the colors used for these different kinds of links with the **Options** command on the **Tools** menu. For more information about how changing the colors used to render different categories of links, see Customizing Colors and Font in BizTalk Mapper.

Link Creation

BizTalk Mapper helps you automate some elements involved in link creation. Simple link creation is similar to simple data types. There are more sophisticated forms of link creation that are more like structure assignment in a programming language. An example is a single link creation that specifies how multiple items of data are to be moved from input instance messages to corresponding output instance messages.

You create links using the following methods:

- **Simple link creation.** In simple link creation, you produce a link by dragging. Dragging a field in the source schema to a field in the destination schema causes the creation of an element or attribute in an output instance message and inserts the value of the element or attribute in the message. Such links can be made directly between **Record** and **Field** nodes in the source and destination schema, or they can include one or more functoids in a link path between **Record** and **Field** nodes in the source and destination schemas.
- **Structure links.** In creating structure links, you produce multiple simple links at the same time between **Record** and **Field** nodes in the source and destination schemas that have the same relative structure. To use structure linking, the structure of the relevant parts of the two schemas must be the same. For more information about configuring structure links, see Linking Records Automatically.
- **Name-matching links.** When you use this method, you create multiple simple links at the same time between **Record** and **Field** nodes in the source and destination schemas based on the names of the **Records** and **Field** nodes. To use name-matching linking, the structure of the source and destination schemas must be very similar, but not exactly the same. For more information about configuring name-matching links, see Linking Records Automatically.

Link Configuration

Links have properties that are displayed in the Microsoft® Visual Studio® .NET Properties window when a link is selected in the displayed grid page. For reference information about the properties associated with links, see **Link Properties**.

The following properties configure links in a map:

- **Source Links.** You use the **Source Links** property to configure the nature of the data from an input instance message that will be used to construct the output instance message. The default, and most common choice, is to use the element or attribute value from the input instance message. Other choices are possible, including the use of the name of the element or attribute from the input instance message. For more information about this property and its available choices, see **Link Properties**.
- **Target Links.** You use the **Target Links** property to configure how BizTalk Mapper will match node-hierarchy levels. By default, source schema hierarchies are flattened as the output instance message is created. You can also choose to have hierarchies preserved, matching links from either the top down or from the bottom up. For more information about this property and its available choices, see **Link Properties**.
- **Label.** You use the **Label** property to create a more readable name for the link than the XPath value that is used by default. Configuring this property is especially helpful when the link is used as an input for table-driven looping. For more information about this property and its available choices, and about table-driven looping, see **Link Properties** and **Table Looping and Table Extractor Functoids**, respectively.

Record-to-Record Linking

In Microsoft® BizTalk® Server 2006, you can use BizTalk Mapper to create multiple links between similar portions of the source and destination schemas at the same time. In previous versions of BizTalk Server, you had to create such links individually, one at a time. There are two distinct types of record-to-record linking, each appropriate to different scenarios based on the degree of similarity of the structures of the source and destination schema records being linked, as follows:

- **Structure linking.** Use structure linking when the structure of the records being linked in your source and destination schemas are the same or very similar.
- **Name-matching linking.** Use name-matching linking when the structure of the records being linked in your source and destination schemas are still similar, and with matching record and field names, but with more structural exceptions than are workable with structure linking.

Record-to-Record Linking: Structure Links

Use structure linking when the structure of the records that you want to link in your source and destination schemas is the same or almost exactly the same.

If the structure of your schemas is exactly the same, you need only add one link at the root node of each schema.

If the structure of particular records in your schemas is exactly the same, you need only add one link between those records in each schema.

To create a structure link between portions of your source and destination schemas that has matching structures, including the entire schemas where applicable, press and hold the SHIFT key while linking the relevant nodes. After you link the nodes, links are automatically created for all of the subordinate records and fields in the source and destination schemas.

For information about how to configure record-to-record linking as using either structure linking or name-matching linking, see [Linking Records Automatically](#).

Record-to-Record Linking: Name-Matching Links

Use name-matching links when the structure of the records that you want to link in your source and destination schemas is very similar, but not exactly the same. For example, the source or destination schema might have more subordinate records or fields within the nodes to be linked than in the other schema.

To create a name-matching link between portions of your source and destination schemas that have nearly matching structures, including the entire schemas where applicable, press and hold the SHIFT key while linking the relevant nodes. After you link the nodes, links are automatically created for all of the subordinate records and fields in the source and destination schemas that are named the same and have the same substructure.

For information about how to configure record-to-record linking as using either structure linking or name-matching linking, see [Linking Records Automatically](#).

Links To and From the Any Element and anyAttribute Nodes

You must use the **Mass Copy** functoid to map elements containing **any** or **anyAttribute** nodes. BizTalk mapper does not support direct linking between **any** or **anyAttribute** nodes. The **any** and **anyAttribute** nodes, in combination with the **Mass Copy** functoid, are most commonly used to map sections of HTML documents.

Compiler Directives and Links

You can configure the following two compiler directives on a link-by-link basis:

- What data from the input instance message is retrieved and copied as the relevant element or attribute value in the output instance message.
- How node-matching is performed between the source and destination schemas.

This section provides a detailed explanation of these options available for these directives.

In This Section

- Data Transformation Configuration

- Node-Hierarchy Level Matching

Data Transformation Configuration

When mapping from an element, a typical Extensible Stylesheet Language Transformations (XSLT) looks as follows.

If the element **BCT01** contains mixed content, the use of `text()` makes it possible to access the first text only up to the point of the first subelement, if any. If `text()` were not used in this XSLT statement, the result would be that all text content, plus any text content of subelements, would be mapped as one string of text. Configuring the **Source Links** property for a link allows you to control the source of the data that is copied into the structure defined by the destination schema.

When you select a link in the displayed grid page, one of the properties displayed in the Visual Studio .NET Properties window is the **Source Links** property. You can choose between the following possible values for each link in your map:

- **Copy Text Value.** Use this value, which is the default, to copy the value of the element or attribute in the input instance message. For example, if the relevant element is **BoldExample**, as follows:
- The value copied into the relevant element or attribute in the output instance message is "This is a ". For mixed content elements like this, this result may not be what is desired. But because mixed content elements are relatively rare, the **Copy Text Value** setting for the **Source Links** property is probably appropriate in most cases.
- **Copy Name.** Use this value to copy the name of the node in the input instance message. For the example in the **Copy Text Value** description, the result is "BoldExample", which is the actual name of the element.
- **Copy Text and Subcontent Value.** Use this value to concatenate the values of the node and all values of its child nodes in the input instance message. For the example in the **Copy Text Value** description, the result is "This is a Bold Text example.", which might very well be the appropriate result for elements defined to contain mixed content.

Node-Hierarchy Level Matching

BizTalk Mapper enables you to configure a link property to control how the compiler matches node hierarchies between the source and destination schemas. When you create a link from a field in the source schema to a field in the destination schema, BizTalk Mapper automatically adds compiler links. These compiler links depend on the matching you select.

When you select a link in the displayed grid page, one of the properties displayed in the Visual Studio .NET Properties window is the **Target Links** property. You can choose among the following possible values for each link in your map:

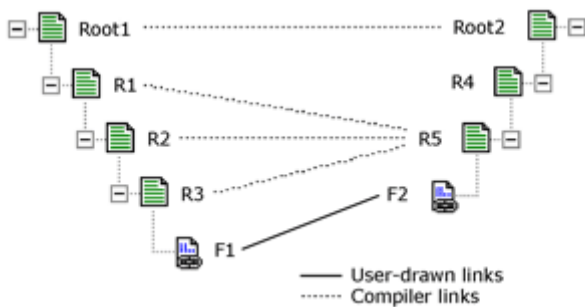
- **Flatten Links.** Use this value to flatten all source hierarchies to the parent record in the destination schema node.

- **Match Links Top-Down.** Use this value to match node levels from the top of the schemas to the bottom of the schemas.
- **Match Links Bottom-Up.** Use this value to match node levels from the bottom of the schemas to the top of the schemas.

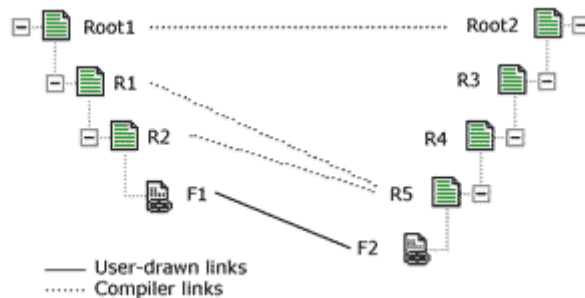
Flatten Links

In this mode, all the source hierarchies are flattened to the parent record of the destination node. In the first case, the source schema is more complex than the destination schema. In the second case, the destination schema is more complex.

Flatten Links



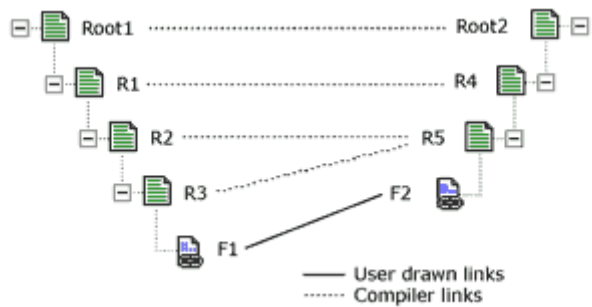
Flatten Links, Second Case



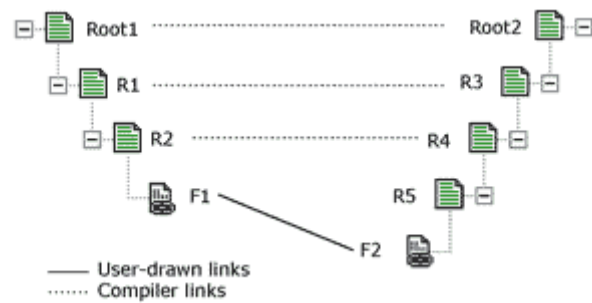
Match Links Top-Down

This mode matches level to level from the top down. In the first case, the source schema is more complex than the destination schema. In the second case, the destination schema is more complex.

Top-Down Matching



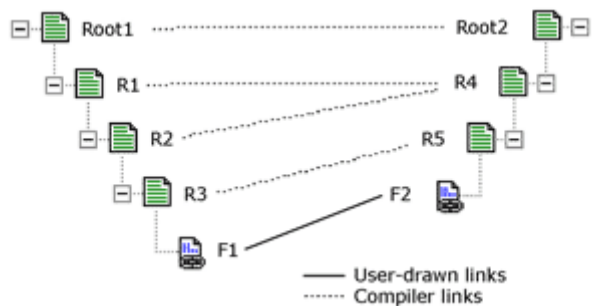
Top-Down Matching, Second Case



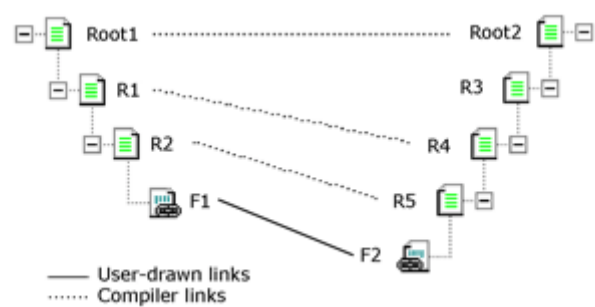
Match Links Bottom-Up

This mode matches level to level from the bottom up. In the first case, the source schema is more complex than the destination schema. In the second case, the destination schema is more complex.

Bottom-Up Matching



Bottom-Up Matching, Second Case



How BizTalk Mapper Processes Link Types

Because you can set the **Target Links** property to different values for different links, BizTalk Mapper needs a way to resolve the different settings when they may conflict.

For example, if you use a flatten compiler directive, a top-down compiler directive, and a bottom-up compiler directive for links from **Field** nodes to **Field** nodes in the destination schema, and these nodes share the same parent **Record** node, BizTalk Mapper ignores the conflicting top-down and bottom-up compiler directives and treats all the links as if they were set to the flatten compiler directive.

The following table shows how BizTalk Mapper treats links to **Field** nodes in the same **Record** node in the destination schema, based on the settings for the **Target Links** property for the links within the same **Record** node.

Flatten	Top-down	Bottom-up	Result
0 or more	1 or more	1 or more	BizTalk Mapper treats all the links as if they were set to the flatten compiler directive.
1 or more	1 or more	0	BizTalk Mapper treats all the links as if they were set to the top-down compiler directive.
1 or more	0	1 or more	BizTalk Mapper treats all the links as if they were set to the bottom-up compiler directive.

The top-down and bottom-up compiler directives take precedence over the flatten compiler directive, but cancel each other out when both are present.

Functoids in Maps

BizTalk® Mapper supports complex structural transformations from records and fields in the source schema to records and fields in the destination schema. Functoids perform calculations by using predefined formulas and specific values, called arguments. These calculations are executed based on the designated order of the records and fields.

By selecting a functoid from the Toolbox, dragging it to the grid page, and linking it to nodes in the source schema and destination schema, data can be added together, date or time information can be modified, data can be concatenated, or other operations can be performed. For example, the **Addition** functoid adds values and the **Record Count** functoid returns the total count of a looping record in an instance message. Categories of functoids that you can find in the Toolbox include: Advanced, Conversion, Cumulative, Database, Date and Time, Logical, Mathematical, Scientific, and String.

The topics in this section describe how to migrate functoids from previous versions of BizTalk Server, what functoids are, what they do, and how to use them.

In This Section

- Functoid Categories
- Functoid Input Parameters
- Basic Functoids
- Advanced Functoids
- Cascading Functoids
- Functoid Properties
- Migrating Functoids

Functoid Categories

BizTalk functoids are divided into categories according to their intended use. For example, database functoids are designed for extracting data from a database at run time, mathematical functoids are used to perform mathematical operations, and so on. In BizTalk Mapper, functoids appear by category in the Visual Studio .NET Toolbox. The following table shows the functoid categories, briefly describes the category, and shows the list of functoids in each category, including links to their corresponding reference pages.

Functoid description	category	Category	Functoids in category
Advanced (Advanced Reference) Primarily used with looping records. Also used for running arbitrary script or compiled code.	(Advanced)	Functoids	Index, Iteration, Looping, Mass Copy, Record Count, Scripting, Table Extractor, Table Looping, Value Mapping, Value Mapping (Flattening)
Conversion (Conversion Reference) Used to convert to and from ASCII,	(Conversion)	Functoids	ASCII to Character, Character to ASCII, Hexadecimal, Octal

and between numeric bases.	
Cumulative (Cumulative Functoids Reference) Used to perform mathematical operations in looping records, such as averages and concatenation.	Cumulative Average, Cumulative Concatenate, Cumulative Maximum, Cumulative Minimum, Cumulative Sum
Database (Database Functoids Reference) Used to extract data from a database and use it in destination instance messages.	Database Lookup, Error Return, Format Message, Get Application ID, Get Application Value, Get Common ID, Get Common Value, Set Common ID, Value Extractor
Date and Time (Date and Time Functoids Reference) Used to retrieve the current date and time, and to calculate delta times.	Add Days, Date, Date and Time, Time
Logical (Logical Functoids Reference) Used to perform a variety of logical operations, such as greater than and logical existence.	Equal, Greater Than, Greater Than or Equal To, Less Than, Less Than or Equal To, Logical AND, Logical Date, Logical Existence, Logical Numeric, Logical OR, Logical String, Not Equal
Mathematical (Mathematical Functoids Reference) Used to perform a variety of mathematical operations, such as addition and multiplication.	Absolute Value, Addition, Division, Integer, Maximum Value, Minimum Value, Modulo, Multiplication, Round, Square Root, Subtraction
Scientific (Scientific Functoids Reference) Used to perform a variety of scientific operations, such as logarithms and trigonometry.	10ⁿ, Arc Tangent, Base-Specified Logarithm, Common Logarithm, Cosine, Natural Exponential Function, Natural Logarithm, Sine, Tangent, X^Y
String (String Functoids Reference) Used to perform a variety of string functions, such as trimming and	Lowercase, Size, String Concatenate, String Extract, String Find, String Left, String Left Trim, String Right, String Right Trim, Uppercase

concatenation.

Functoid Input Parameters

A critical aspect of using functoids in your maps is properly configuring the input parameters to the functoid. While the order of input parameters is not critical for all functoids (such as the **Addition** functoid, which displays the same associate properties that one expects from addition), many functoids must have their input parameters specified in the correct order.

There are two ways that input parameters to a functoid can be created:

- By creating links into the left side of a functoid in the displayed grid page. The order in which you create the links is the order in which the associated input parameters are passed to the functoid.
- By opening the **Configure Functoid Inputs** dialog box and adding new input parameters. This dialog box is also important because it allows you to reorder the input parameters to a functoid so that they are in the correct order. For example, if you create links to the functoid in the incorrect order, you can go to this dialog box and make the necessary corrections. For step-by-step instructions on configuring the input parameters for a functoid, see [Configuring Functoid Input Parameters](#).

You must use the **Configure Functoid Inputs** dialog box to create input parameters that are constants.

When you provide a label for a link or functoid using the **Label** property in the Properties window when the link or functoid is selected, that label will be shown in the **Configure Functoid Inputs** dialog box rather than the corresponding XPath of a schema node, or the name of a functoid being used as an input parameter. With labels, it will be much easier to tell which parameter is which, and to get them into the correct order.

For definitive information about the correct order of functoid input parameters, see the individual functoid topics in the **Functoid Reference**.

Basic Functoids

Any functoid not in the **Advanced** category is considered a basic functoid. Basic functoids are generally easy to use, having few and, often, straightforward input parameters to be configured. This section provides conceptual information about the basic functoids, organized by category.

For step-by-step instructions about how to add a basic functoid to a BizTalk map, see [How to Add Basic Functoids to a Map](#).

For reference information about all functoids, including basic functoids, see **Functoid Reference**.

In This Section

- [Conversion Functoids](#)
- [Cumulative Functoids](#)

- **Database Functoids7**
- Date and Time Functoids
- Logical Functoids
- Mathematical Functoids
- Scientific Functoids
- String Functoids

Conversion Functoids

Conversion functoids are used to convert between numbers and their ASCII equivalents, and to convert base 10 numbers to their base 8 and base 16 equivalents.

All of the conversion functoids take a single input parameter.

The **Conversion** functoids are: **ASCII to Character**, **Character to ASCII**, **Hexadecimal**, and **Octal**.

Cumulative Functoids

Cumulative functoids reduce a series of values to a single value such as a sum, a concatenated string, or an average.

All **Cumulative** functoids accept two input parameters:

1. The value to accumulate. This value is numeric for all **Cumulative** functoids except the **Cumulative Concatenate** functoid which expects a string value. The value is a link, often from a **Field Attribute**, **Field Element**, or **Record** node (with its **Mixed** property set to **True**).
2. The scope within which the value is accumulated. This argument is optional. The argument indicates how closely related the specified values must be to be accumulated.

The following table shows the values for the scoping parameter and its effect:

Scoping Parameter Value	Effect
0 (zero)	Accumulate the value over the entire instance message. The default.
1 (one)	Accumulate the value of element or attribute values with the same parent element.
2	Accumulate the value of element or attribute values with the same grandparent element.
3 or greater	Accumulate the value of element or attribute values of progressively wider scope

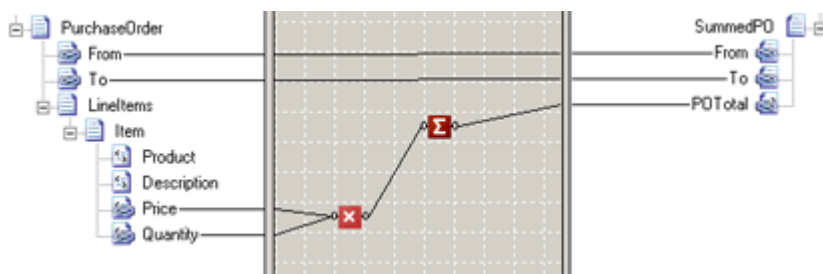
following the preceding pattern (great-grandparent, great-great-grandparent, etc.).

An example of using a **Cumulative** functoid might be summing costs across a purchase order. The following code is an example of a purchase order.

The **Max Occurs** property for the **Item** record would, of course, be **unbounded**. This indicates that the item record loops, and BizTalk Mapper compiles this record as a loop.

The following figure shows a map using a **Multiplication** functoid and a **Cumulative Sum** functoid to aggregate item records from an incoming purchase order and output the results in the **POTotal** field:

Cumulative Functoid Map



The map produces the following output with the preceding data and with the default scoping parameter value, 0 (zero):

In this example, all the **Item** records under the **LineItems** record participate in the accumulation—the default for the scoping parameter indicates accumulating the values across the entire message. The **Price** and **Quantity** fields are sent to a **Multiplication** functoid. The output of the **Multiplication** functoid becomes the input to the **Cumulative Sum** functoid. The output of the **Cumulative Sum** functoid is the accumulated value as the **Item** records are traversed in the input purchase order.

Changing the scoping parameter to 1 (one) and modifying the output schema slightly, yields output like the following:

Setting the scoping parameter to 1 (one) indicates accumulating values for elements or attributes with the same parent. Here the **Price** and **Quantity** fields have **Item** as a parent so that the functoid sums values for each individual **Item**.

If the scoping parameter is 2, the functoid accumulates values for elements or attributes with the same grandparent. The grandparent of the **Price** and **Quantity** fields is the **LineItems** record. Because there is only one **LineItems** record in the instance message, the result is the same as using the default value:

The **Cumulative Average**, **Cumulative Minimum**, and **Cumulative Maximum** functoids behave similarly to the **Cumulative Sum** functoid. The **Cumulative String** concatenates strings rather than aggregating numeric values.

The **Cumulative** functoids are: **Cumulative Average**, **Cumulative Concatenate**, **Cumulative Maximum**, **Cumulative Minimum**, and **Cumulative Sum**.

Database Functoids

Database functoids extract data from a database for use in an output instance message. The following is a list of the **Database** functoids and how you can use them:

- **Database Lookup.** Use the **Database Lookup** functoid to extract information from a database and store it as a Microsoft® ActiveX® Data Objects .NET (ADO.NET) recordset. This functoid requires four input parameters in the following order:
 - A lookup value
 - A database connection string
 - A table name
 - A column name for the lookup value.
- **Error Return.** Use the **Error Return** functoid to capture error information, such as database connection failures, that occur during run time. This functoid requires one input parameter: a link from the **Database Lookup** functoid.
- **Format Message.** Returns a formatted and localized string using argument substitution and, potentially, ID and value cross-referencing.
- **Get Application ID.** Retrieves an identifier for an application object.
- **Get Application Value.** Retrieves an application value.
- **Get Common ID.** Retrieves an identifier for a common object.
- **Get Common Value.** Retrieves a common value.
- **Remove Application Value.** Removes an application value.
- **Set Common ID.** Sets and returns an identifier for a common object.
- **Value Extractor.** Use the **Value Extractor** functoid to extract data from the specified column in a recordset returned by the **Database Lookup** functoid. This functoid requires two input parameters: a link to the **Database Lookup** functoid and a column name.

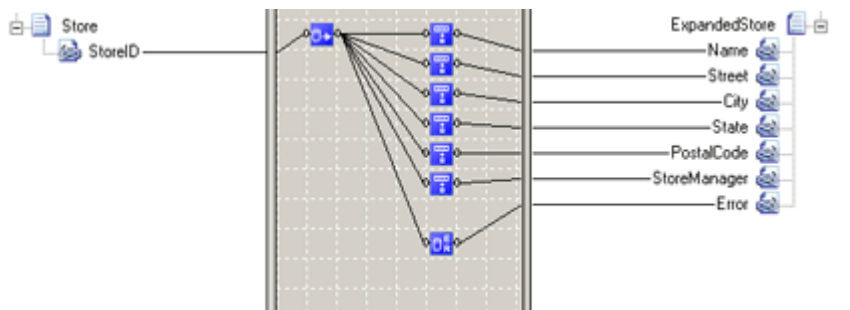
Six of the **Database** functoids—**Get Application ID**, **Get Application Value**, **Get Common ID**, **Get Common Value**, **Set Common ID**, and **Format Message**—are **CrossReferencing** functoids. These functoids translate IDs and values from an input message into the IDs and values needed in the output message. For more information, see **Database Functoids Reference**.

The following example uses some of the **Database** functoids. Consider a large retail manufacturer with stores spread over a large geographical area. To keep track of the stores, headquarters assigns each store a unique code called a **StoreID**. Additionally, headquarters associates the following information with each **StoreID**:

- StoreName
- StoreAddress
- City
- PostalCode
- StorePhoneNumber
- StoreManager

This information is stored in a database and is distributed to trading partners on a regular basis. For the manufacturer, all purchasing is done by headquarters, not the stores. When headquarters sends a purchase order to the trading partners, it is common for multiple stores to receive merchandise ordered through the single purchase order. Instead of sending name and address information for each store that is to receive merchandise, headquarters simply sends the **StoreID**. To insert the name and address information into the advanced ship notice, the trading partner uses the **Database** functoids to automatically insert this information into the output instance message. The following figure shows how a trading partner can implement the replacement of the StoreID in a map.

Database Functoid Map



In the figure, the source schema represents an incoming purchase order; the destination schema represents an advanced ship notice. The **Database Lookup** functoid finds the appropriate record from the appropriate database table. The **Value Extractor** functoids extract the appropriate column from the lookup record. The **Error Return** functoid outputs a string containing error information if there are errors (such as connection failures) at run time.

In the previous example, the first input parameter is taken from the **StoreID** field of the incoming purchase order, and the remaining three input parameters are constants configured in the **Configure Functoid Inputs** dialog box for the **Database Lookup** functoid. It is possible to create links from the source schema to supply values for all four input parameters.

Date and Time Functoids

Date and Time functoids can be divided into two categories. The first category contains a single functoid, **Add Days**, which is used to add a specified number of days to a specified date and time value. This can be useful when a field in the output instance message is supposed to include a date and time estimate in

the future. For example, the **Add Days** functoid can be used to generate an estimated shipping date based a fixed delta from the date that an order was received.

The second category includes all of the remaining functoids in the **Date and Time** category. They are used to provide a timestamp at run time so that the date and time at which message transformation is being performed can be included in the output instance message.

The **Date and Time** functoids are: **Add Days**, **Date**, **Date and Time**, and **Time**.

Logical Functoids

Logical functoids are used to perform the following types of operations:

- Perform specific logical tests at run time. The **Logical OR** and the **Logical AND** functoids can be used to determine whether a record is created in a destination instance message, such as the following:

If ShipTo **OR** OrderedBy are present, create BillTo address record.

You can also use these functoids in conjunction with the **Looping** functoid to configure how many times a record loops.

- Control whether a specific record is created in a destination instance message at run time. Functoids such as **Logical Numeric**, **Less Than**, and **Greater Than** can be used to control whether a record is created.

If the result of one of these logical functoids is **True**, the corresponding record in the destination instance message is generated. If the result is **False**, the corresponding record in the destination instance message is not generated.

The output of a **Logical** functoid can also be accepted as input to other functoids in a map. If both a **Logical** functoid and a looping functoid are linked together, and then linked to a record in the destination schema, the looping functoid is used only when the **Logical** functoid output is **True**.

You can also use **Logical** functoids with the **Value Mapping** or **Value Mapping (Flattening)** functoids to control whether a record in the destination instance message is created.

The **Logical** functoids are: **Equal**, **Greater Than**, **Greater Than or Equal To**, **Less Than**, **Less Than or Equal To**, **Logical AND**, **Logical Date**, **Logical Existence**, **Logical Numeric**, **Logical OR**, **Logical String**, and **Not Equal**.

Mathematical Functoids

Mathematical functoids are used to perform a variety of basic mathematical operations.

The number of input parameters to the **Mathematical** functoids varies from functoid to functoid, and where reasonable, these functoids accept large numbers of inputs. For example, the **Division** functoid requires exactly two input parameters (the dividend and the divisor), while the **Addition** functoid accepts between 2 and 100 input parameters, resulting in their sum.

The **Mathematical** functoids are: **Absolute Value**, **Addition**, **Division**, **Integer**, **Maximum Value**, **Minimum Value**, **Modulo**, **Multiplication**, **Round**, **Square Root**, and **Subtraction**.

Scientific Functoids

Scientific functoids are used to perform a variety of standard trigonometric, logarithmic, and exponential calculations.

With the exception of the **Base-Specified Logarithm** and **X^Y** functoids, which each take two input parameters, the **Scientific** functoids all take a single parameter.

The four trigonometric **Scientific** functoids (**Arc Tangent**, **Cosine**, **Sine**, and **Tangent**) all use radians rather than degrees as the units for their relevant input or output parameters. A radian is a unit of measure of angles, such that there are 2π radians in a circle. It follows that:

- 2π radians equals 360 degrees
- 1 radian = $180/\pi$ degrees
- 1 degree = $\pi/180$ radians

If your input or output instance messages use degrees as their unit of measure for angles, you will need to use a **Mathematical** functoid in conjunction with a trigonometric **Scientific** functoid to achieve the correct result.

The **Scientific** functoids are: **10ⁿ**, **Arc Tangent**, **Base-Specified Logarithm**, **Common Logarithm**, **Cosine**, **Natural Exponential Function**, **Natural Logarithm**, **Sine**, **Tangent**, and **X^Y**.

String Functoids

String functoids are used to manipulate strings in standard ways such as conversions to all uppercase or all lowercase, string concatenation, determination of string length, white space trimming, and so on.

Two **String** functoids refer to the numeric position of a character in a string: **String Extract** and **String Find**. These functoids begin counting character positions at 1, not 0.

The two string trimming functoids, **String Left Trim** and **String Right Trim**, remove all white space characters (spaces, tabs, and so on) from the specified string.

The **String** functoids are: **Lowercase**, **Size**, **String Concatenate**, **String Extract**, **String Find**, **String Left**, **String Left Trim**, **String Right**, **String Right Trim**, and **Uppercase**.

Advanced Functoids

Advanced functoids fall into four groups, according to their use:

- **Managing looping records.** The **Index**, **Iteration**, **Looping**, **Record Count**, **Table Extractor**, and **Table Looping** functoids are used in various combinations when the input instance message

contains sections with an unpredictable number of repeating elements, as represented by looping records in the source schema.

- **Conditional mapping.** The **Value Mapping** and **Value Mapping (Flattening)** functoids are used to provide conditional mapping from an input instance message to an output instance message. When their first input parameter is true, the second input parameter is put into the specified element or attribute in the output instance message; otherwise, that element or attribute is not created in the output instance message.
- **Arbitrary scripting.** The **Scripting** functoid is used to run arbitrary script or compiled code when an input instance message is being mapped to an output instance message. Such script or compiled code can be created so that it accepts input parameters from the source instance message, from configured constant values, from the output of another functoid, or some combination thereof.
- **Simple mapping.** The **Mass Copy** functoid can be used to copy an entire element, including its subelements to an arbitrary depth, from an input instance message to an output instance message.

This section provides detailed descriptions and examples of how the **Advanced** functoids can be used in BizTalk maps.

The **Advanced** functoids are: **Index Functoid**, **Iteration Functoid**, **Looping Functoid**, **Mass Copy Functoid**, **Record Count Functoid**, **Scripting Functoid**, **Table Looping and Table Extractor Functoids**, **Value Mapping Functoid**, and **Value Mapping (Flattening) Functoid**.

In This Section

- **Index Functoid**
- Iteration Functoid
- Looping Functoid
- Mass Copy Functoid
- Record Count Functoid
- Table Looping and Table Extractor Functoids
- Value Mapping Functoid
- Value Mapping (Flattening) Functoid
- Scripting Functoid

Index Functoid

The **Index** functoid enables you to select information from a specific record in a series of records. Each **Index** functoid collects information from a single field.

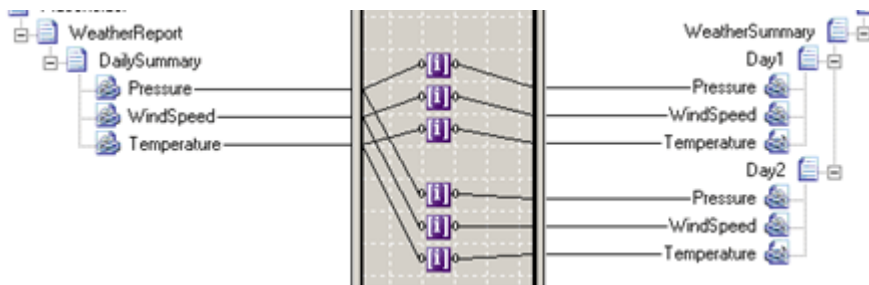
Certain records typically occur many times in an input file. For example, in a weather report, the **DailySummary** element might occur many times. The **DailySummary** element might include attributes for the temperature, the barometric pressure, and the wind speed. The following code is an example of a weather report.

In the underlying schema, the **Max Occurs** property for the **DailySummary** record would be set to unbounded to indicate a recurring or looping record. BizTalk Mapper compiles this record as a loop.

Suppose you want to collect weather information for the first two **DailySummary** records of the weather report. In BizTalk Mapper, each attribute from the **DailySummary** record of the incoming source schema can be connected to an **Index** functoid. In turn, each **Index** functoid can specify the **DailySummary** record from which to draw the information: the first or second. The **Index** functoids can then be connected to the appropriate fields of the destination schema.

The following figure shows **Index** functoids used in this way.

Index Functoid Example



To get the daily summary information for the first day, the upper set of three **Index** functoids have their index values set to 1. To get the daily summary information for the second day, the lower set of three **Index** functoids have their index values set to 2.

Index functoids use the **Configure Functoid Inputs** dialog box to set their input parameters. The first input parameter identifies a field within a looping record in the source schema. The second and succeeding input parameters specify the particular record. You can specify multiple index values to select a record within nested repeating structures. The index value for the innermost structure is the second parameter. The index value for the next outermost structure would be the third parameter, and so on. For example, suppose that the preceding **DailySummary** records were inside **WeeklyData** records. To retrieve the **Pressure** from the first **DailySummary** in the second **WeeklyData**, the second parameter would be 1 and the third parameter would be 2.

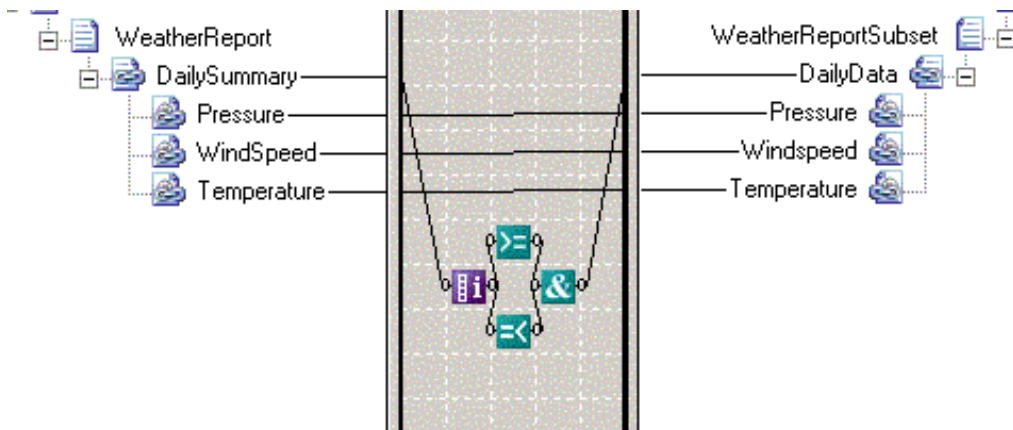
Notice that this example assumes the **Pressure** field does not repeat. If the field did repeat, the indices would be off—the count would begin with the **Pressure** field, rather than the **Daily Summary**.

Iteration Functoid

The **Iteration** functoid outputs the index of the current record in a looping structure, beginning at 1 for the first record, 2 for the second record, and so on.

The following figure shows an **Iteration** functoid combined with a **Greater Than or Equal To** functoid, a **Less Than or Equal To** functoid, and an **And** functoid to create the equivalent of a **For** loop.

Iteration Functoid Map



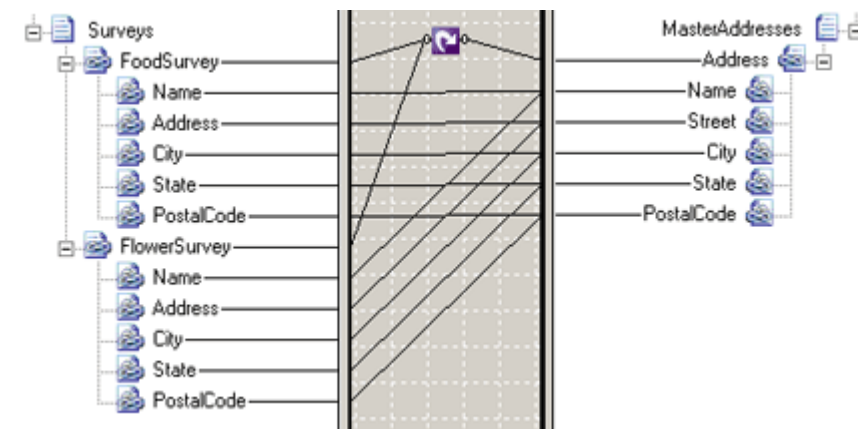
Assume that the **Greater Than or Equal To** functoid tests for values greater than or equal to 2, and the **Less Than or Equal To** functoid tests for values less than or equal to 4. In that case, the **And** functoid will return **true** only for records 2, 3, and 4. Thus, the output instance will contain records two, three, and four of the input instance message.

Looping Functoid

The **Looping** functoid combines multiple records or fields in the source schema into a single record in the destination schema.

The following figure shows a **Looping** functoid used in a map to combine addresses collected from two different surveys into a single master address list.

Looping Functoid Map



The **FoodSurvey** and **FlowerSurvey** looping records of the source schema are mapped to the looping **Address** record of the destination schema. If an input instance message has three **FoodSurvey** records and two **FlowerSurvey** records, the **Looping** functoid combines these to create five **Address** records in the output instance message.

The following code is a sample input instance message.

This input instance message produces the following output instance message when processed by the map in the preceding figure.

The **FoodSurvey** and **FlowerSurvey** message addresses have been combined. The combined message does not indicate the source of each address. If you want to track the source, add a **Source** attribute to the **Address** record of the **MasterAddress** schema and map a constant value. To set this value, connect the **FoodSurvey** field to the new **Source** field. On the connector line, modify the **Link Properties | Compiler | Source Links** property to "Copy name". Repeat this process for the **FlowerSurvey** field. Reprocessing the input message from above yields the following output:

Relationships among nodes affect the behavior of the **Looping** functoid. For example, linking both a child node and its parent in the source schema to the **Looping** functoid prevents the destination node from being created.

Functoids are also affected by the relationships among source nodes. Connecting a functoid to non-sibling child fields of source nodes of the **Looping** functoid may produce unexpected results. For example, using the **String Concatenate** functoid to combine the **FoodSurvey** Name field and **FlowerSurvey** Address field into the Address Name field in **MasterAddress** would produce the following output instance message:

Notice how the **Name** field is missing for **FoodSurvey** source messages but is present for **FlowerSurvey** source messages.

The **Looping** functoid is a powerful construct that you can use to create conditional loops and to map schemas to catalogs. There are also some effects of overlapping **Looping** functoid paths you need to take into account.

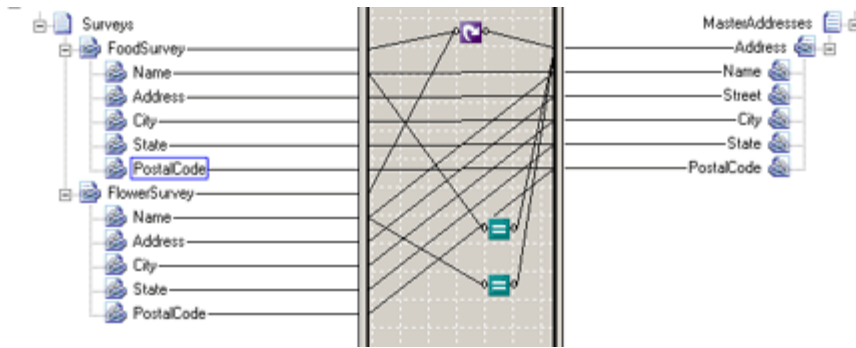
In This Section

- Conditional Looping
- Flat Schema to Catalog
- Loop Paths

Conditional Looping

You can add conditions to a **Looping** functoid by linking the output of a **Looping** functoid and a **Logical** functoid to the same destination record. The destination records are created only when the logical condition is met.

Conditional Looping Map



In the preceding figure, the first **Equal** functoid compares the **Name** field under **FoodSurvey** to "Wendy Wheeler". The second **Equal** functoid compares the **Name** field under **FlowerSurvey** to "Kelly Focht". Thus, the map creates the destination **Address** records only for the two names. Using the sample data from the **Looping** functoid example, the output instance message would appear as follows.

Flat Schema to Catalog

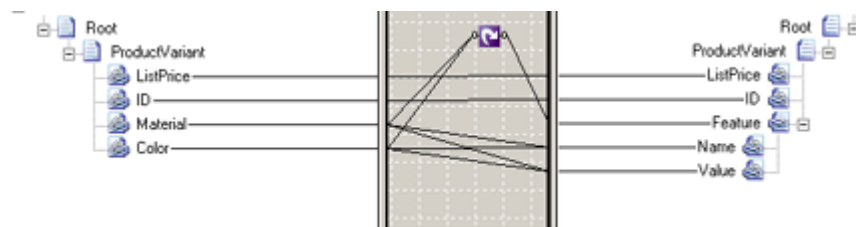
You can use the **Looping** functoid to convert a flat schema to an hierarchical schema by mapping a single record to multiple records. This is a common operation in converting flat schemas to Microsoft Commerce Server catalogs.

The following code shows a portion of a catalog listing product variants with each variant as its own record.

Expanding this portion of the catalog would convert some or all of the **ProductVariant** attributes into records.

The following figure shows a map that performs this conversion.

Looping Functoid, Flat Schema Map



For this type of map to work correctly, you must do the following:

- For each link connecting to the **Name** field in the destination schema, set the source-schema link properties to copy the name. For more information, see [Link Configuration](#) and **Link Properties**
- For each link connecting to the **Value** field in the destination schema, set the source-schema link properties to copy the value (the default).

- For the link connecting the **Looping** functoid to the record named **Feature** in the destination schema, set the destination-schema link properties to match links top-down.

For the inverse of this mapping, converting a catalog schema to a flat schema, see Value Mapping (Flattening) Functoid.

Loop Paths

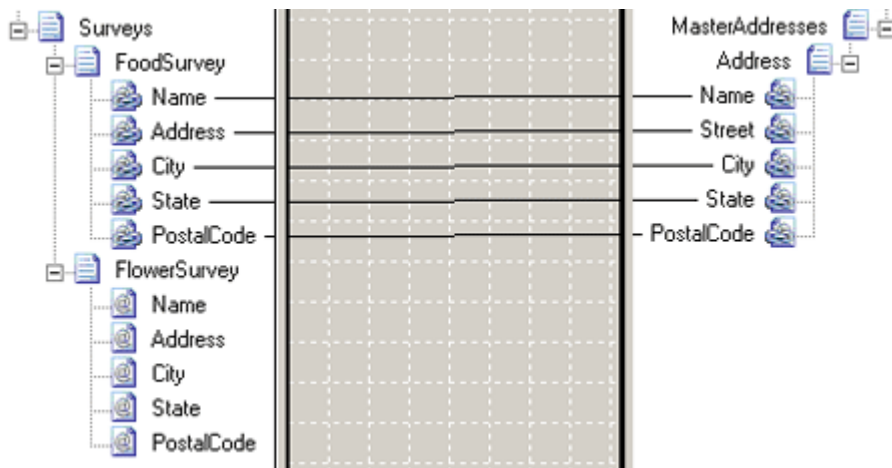
An element in a schema is looping if its Max Occurs property is greater than 1. A loop path occurs when you draw a link between a looping element in the source schema and a looping element in the destination schema.

Configuring a Loop Path

BizTalk mapper automatically handles the looping records when you create a loop path.

You can configure a loop path in a map by linking a field in a looping record in the source schema to a field that is in a looping record in the destination schema. The figure below shows a map that copies only food survey records into a master address list.

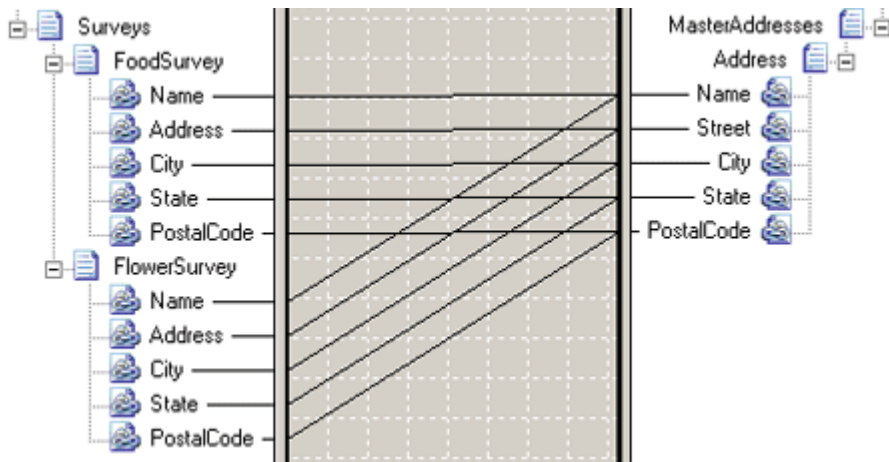
Loop Path Map



Multiple Loop Paths

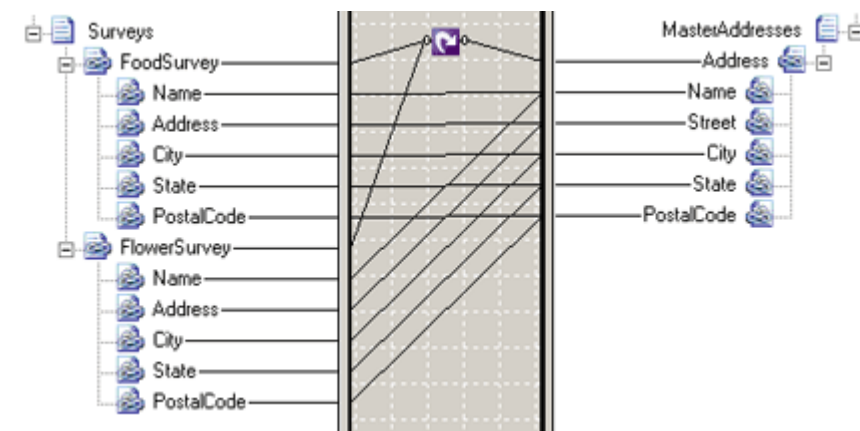
A multiple loop path occurs in a map when you link fields contained by two or more looping records to fields contained in a single looping record. The following figure shows an attempt to combine addresses collected from two different surveys into a single master address list.

Map With Multiple Loop Paths (Incorrect)



This map will not produce the expected results. When the Mapper encounters multiple loop paths during compilation, it produces a warning and selects the first loop path by default. In order to combine the two different addresses into a single master address list, use a **Looping** functoid as shown in the map below.

Looping Functoid Map (Correct)



The **Looping** functoid should be used instead of multiple loop paths in the following scenarios:

1. When the Mapper does not produce the desired output in a multiple loop paths scenario.
2. To combine multiple repeating structures in an input instance message into a single repeating structure in the output instance message.
3. To convert a flat schema to a hierarchical schema by mapping a single record to multiple records. This is a common operation in converting flat schemas to Microsoft Commerce Server catalogs.

Record Count Functoid

The **Record Count** functoid counts records in the input instance message.

The **Record Count** functoid has one input and one output. The input is a link from a looping record in the source schema. The output of the **Record Count** functoid is the count of the looping record in an actual input instance message.

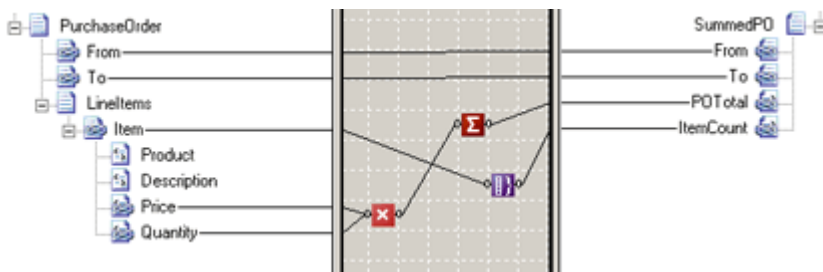
Looping records correspond to elements that repeat an unpredictable number of times in an input instance message. For example, in a purchase order, the **Item** element might occur many times. And, the **Item** element might include products, descriptions, prices, and quantities. The following code is a simplified example of such a purchase order.

The **Max Occurs** property for the **Item** record is set as unbounded. This indicates that the **Item** record loops, and BizTalk Mapper compiles this record as a loop.

Suppose you want to find the total number of **Item** elements in the purchase order input instance message and place the result in a field in the output instance message.

The following figure shows a **Record Count** functoid that counts the number of items in an incoming purchase order and puts that value in the **ItemCount** field in the **SummedPO** output instance message.

Record Count Functoid Map



Notice that the **Max Occurs** property for the **Item** record would be **unbounded**. This indicates that the **Item** record loops, and BizTalk Mapper compiles this record as a loop.

For the preceding sample purchase order instance message, which contained two **Item** elements, the value of the **ItemCount** field will be set to 2.

Table Looping and Table Extractor Functoids

In a map, you commonly have one structure in the output instance message for each structure in the input instance message. However, there are cases when you need an input instance structure to produce multiple output instance structures. Table-driven looping enables you to create maps generating such multiple structures.

Table-driven looping uses the **Table Looping** functoid and the **Table Extractor** functoid. The **Table Looping** functoid has an internal table you configure. For each input record or field, the **Table Looping** functoid outputs the rows of the table, one at a time. For example, if there are ten records in the input instance message and two rows in the internal table of the **Table Looping**, the functoid outputs a total of twenty rows, two for each of the ten records. The **Table Extractor** functoid extracts the desired item from a row and passes it on to the output instance message.

For reference information about these functoids, see the **Table Looping Functoid** and **Table Extractor Functoid** reference pages.

In This Section

- Table Looping Functoid
- Table Extractor Functoid
- Table-Driven Looping Configuration
- Table-Driven Looping Example

Table Looping Functoid

The **Table Looping** functoid enables you to create a table of output values to use in creating the output instance message. The data in the table can consist of links and constants. The first input parameter to the **Table Looping** functoid configures the scope, or how many times the records will loop. The second input parameter for the **Table Looping** functoid determines how many columns are in the table, and the remaining inputs define possible cell values for the table, in no particular order. For more information about working with these properties, see Table-Driven Looping Configuration and Table-Driven Looping Example.

Table Extractor Functoid

The **Table Extractor** functoid determines which data to extract from each row of the looping grid as the **Table Looping** functoid presents the rows. You need one **Table Extractor** functoid for each output field. For example, suppose your input instance message had an address in a single format, but your output instance message needed the address in two formats with each format tagged. Then you would need a **Table Extractor** functoid for the tag and for each element of the address. For more information about configuring the **Table Extractor** functoid, see Table-Driven Looping Configuration and Table-Driven Looping Example.

Table-Driven Looping Configuration

Follow these steps to configure table-driven looping in your map:

- **Add a Table Looping functoid to the map.** You need only one **Table Looping** functoid per table-driven looping instance. For example, if you are using table-driven looping to derive BillTo and ShipTo information, you need only one **Table Looping** functoid in your map. However, if you are using table-driven looping to derive BillTo and ShipTo information and StoreName and StoreAddress information, you might need two **Table Looping** functoids in your map.
- **Add one more Table Extractor functoid to the displayed grid page.** Add as many **Table Extractor** functoids as you need for each **Table Looping** functoid. The number of **Table Extractor** functoids depends on the number of fields in the destination schema. For example, if you only have an **AddressCode** in your source schema and CompanyName, Address, City, State, PostalCode, and AttentionName in your destination schema, you need to add six **Table Extractor** functoids to the displayed grid page.

- **Configure the Table Looping functoid with the appropriate inputs.** First, link the **Table Looping** functoid to the input instance record or element. Also link it to the structure in the output instance message. Next, configure the inputs by using the **Configure Functoid Inputs** dialog box. For more information about how to configure this property, see *Editing Functoid Properties and Input Parameters*. The list of inputs you enter must be comprehensive and complete because this is the data that you will use to configure the **Table Looping Grid** property. The inputs must be defined as follows:
 - **First input.** The first input parameter is the link to the input instance message record or field. The **Table Looping** functoid loops once for each instance of the record or field.
 - **Second input.** The second input parameter defines the number of columns in the looping grid. The grid may contain up to 228 columns.
 - **Remaining inputs.** The remaining inputs for the **Table Looping** functoid consist of a list of all of the possible values that may appear in the **Table Looping Grid**.
 - For step-by-step instructions about how to configure inputs for the **Table Looping** functoid, see *Adding Table Looping and Table Extractor Functoids to a Map*, and in particular, steps 3 through 8.
- **Configure the Table Looping Grid property of the Table Looping functoid.** Use the **Table Looping Grid** property to open the **Configure Table Looping** dialog box in which you configure the cells in the looping grid.

For step-by-step instructions about how to configure the looping grid, see *Adding Table Looping and Table Extractor Functoids to a Map*, and in particular, steps 9 and 10.

- **Configure the Table Extractor functoids.** Use the **Input Parameters** property to configure the **Table Extractor** functoid inputs as follows:
 - **First input.** The first input parameter to a **Table Extractor** functoid is the **Table Looping** functoid.
 - **Second input.** The second input parameter specifies the column in the row from which to extract data.

For step-by-step instructions about how to configure the **Table Extractor** functoids associated with a **Table Looping** functoid, see *Adding Table Looping and Table Extractor Functoids to a Map*, and in particular, steps 11 through 16.

Table-Driven Looping Example

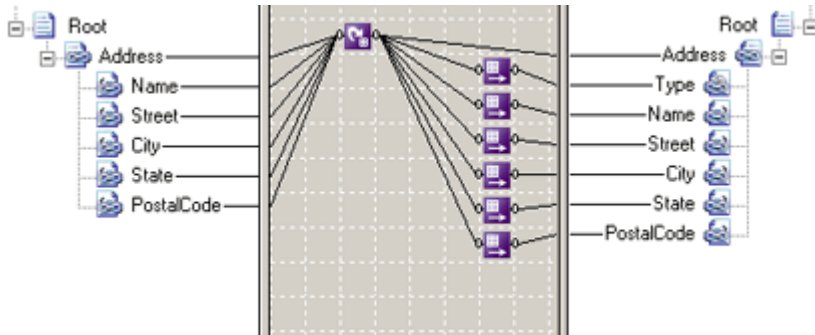
This section briefly describes a map using the **Table Looping** and **Table Extractor** functoids. For detailed information about selecting, placing, linking, and configuring the functoids, see *Adding Table Looping and Table Extractor Functoids to a Map*.

Suppose you have a list of addresses that you need to use in a document requiring separate ship-to and bill-to addresses. The addresses might appear like the following code.

One form the output could take would be the following code, duplicating the addresses but marking them with attributes.

The following figure shows a map using the **Table Looping** functoid and **Table Extractor** functoids to generate the desired output instance message.

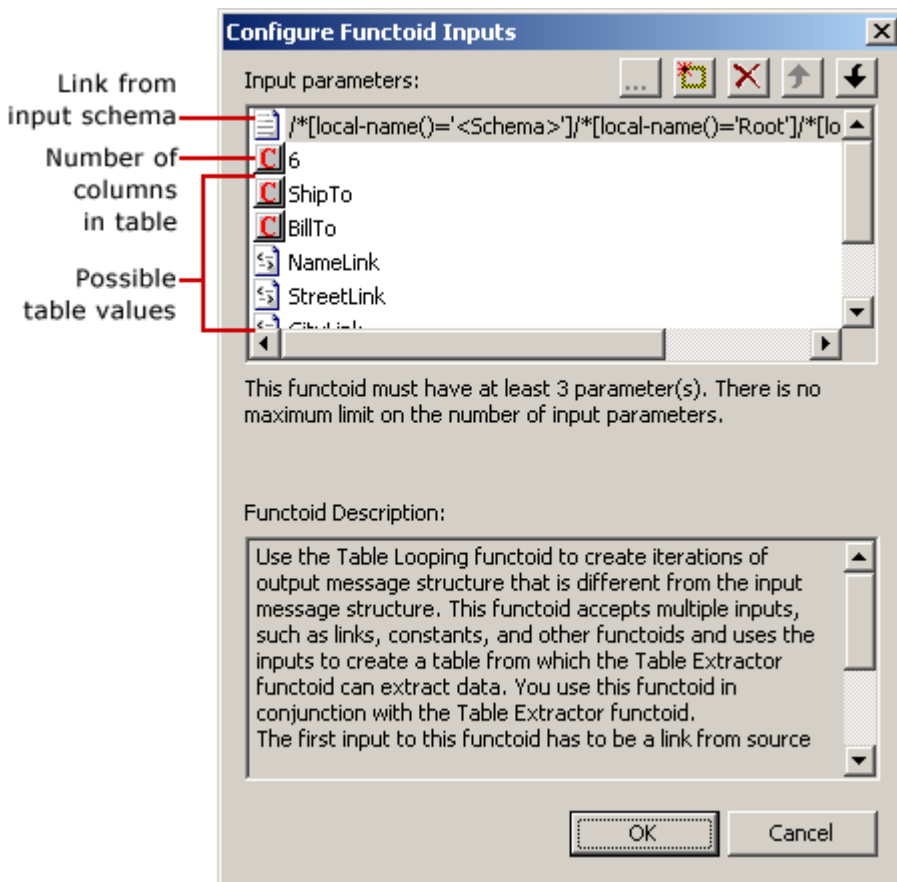
TableLooping and Extractor Functoids



Notice that the **Table Looping** functoid links to the record-level element in both the input and output schemas. The link ensures the creation of the enclosing structure and, thus, the creation of the elements within the record. Also notice that there is one **Table Extractor** functoid for each field in the output schema.

The link to the record in the input schema is the first parameter in the **Configure Functoid Inputs** dialog box, as the following figure shows.

Configure Functoid Inputs Dialog Box

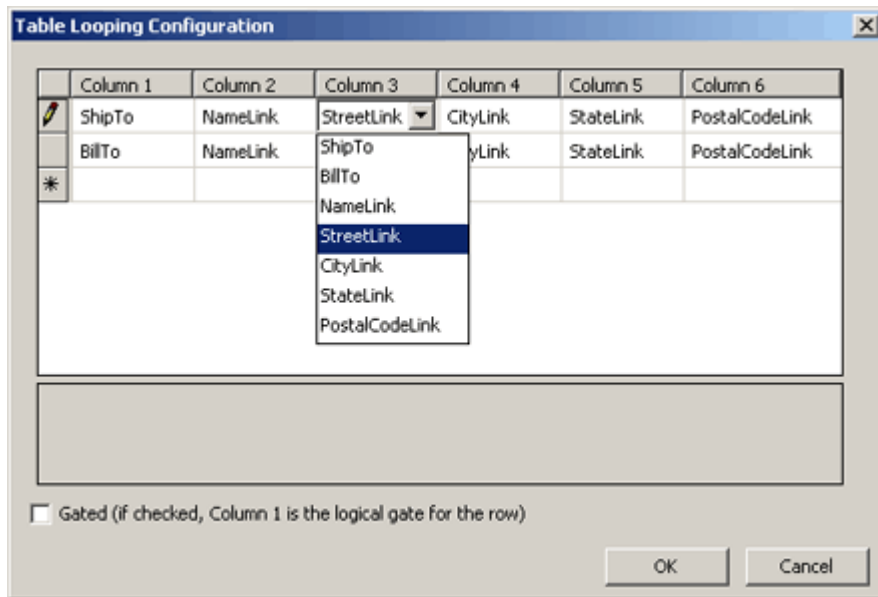


The second parameter, an integer constant of 6, is the number of columns in the grid table of the functoid: one column each for the address type, name, street, city, state, and postal code. Following the second parameter is a list of all of the values that may appear in the grid table. These include string constants for the address type ("ShipTo", "BillTo"), as well as links to the fields of the address. Notice that the links to the address fields have names. Naming the links in the map simplifies constructing the table. Otherwise, full paths appear in the **Table Looping Configuration** dialog box.

After you have configured the **Table Looping** functoid, you can construct the table using the **Table Looping Configuration** dialog box. The dialog appears when you click on the ellipsis () button associated with the **Table Looping Grid** property in the **Properties** window.

The following figure shows the **Table Looping Configuration** dialog box for the **Table Looping** functoid.

Table Looping Grid Configuration Dialog



Notice that there are six columns as specified in the **Configure Functoid Inputs** dialog: one column for each field in the output schema. The dropdown shows the possible values for a field, also as specified by the third and following parameters in the **Configure Functoid Inputs** dialog. The table has two rows, one for each type of record in the output schema. Because there are two rows, this map produces two records for every input record. If there were four rows, there would be four output records for each input record.

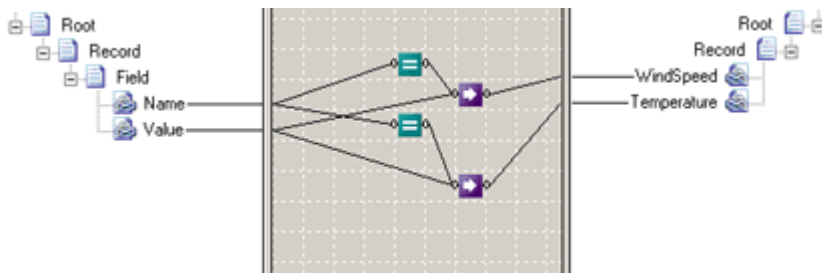
As the **Table Looping** functoid takes each record, it fills in the table with the values from the record, and then sends one row at a time to the **Table Extractor** functoids. The **Table Extractor** functoids each extract one value from the table row and pass it on to the linked field in the output instance message.

Value Mapping Functoid

The **Value Mapping** functoid returns the value of its second parameter if its first parameter is true. A common use of the functoid is to change the attributes of a field into the attributes of a record. Use the Value Mapping (Flattening) Functoid to flatten a portion of an input message by converting multiple records into a single record.

The following figure shows a map with the **Value Mapping** functoid used to change the attributes of a field into the attributes of a record.

Value Mapping Functoid Map



The following code shows an input instance message in which pairs of names and values are assigned to **Name** and **Value** attributes.

The preceding map can convert this message into one in which the values are assigned to attributes with the corresponding names in separate records.

The **Equal** functoids test the values of the **Name** attribute. The first **Equal** functoid tests for the value of **Name** being "WindSpeed." When the **Name** is "WindSpeed," the first **Equal** functoid returns **True**. This, in turn, allows the **Value Mapping** functoid to set the value of the **WindSpeed** attribute in the output instance message.

Suppressing the Creation of Empty Tags

To suppress empty tags, use the Value Mapping functoid to control if a tag gets created or not. If the value is evaluated to true, the destination field will be created; otherwise the destination field will not be created. In a looping scenario, use a logical functoid and connect it to the destination record or field. If the condition is evaluated to false, the tag will not be created. See Conditional Looping for an example.

Forcing the Creation of Empty Tags

To force empty tags to be created, you can add a value in the Value property of the destination field or link a **Concatenate** functoid to the destination field. In BizTalk Server 2006, it is possible to force the generation of empty tags by selecting the "<empty>" value in the Value property of the destination field. In this case the field will be created with the empty value.

Value Mapping (Flattening) Functoid

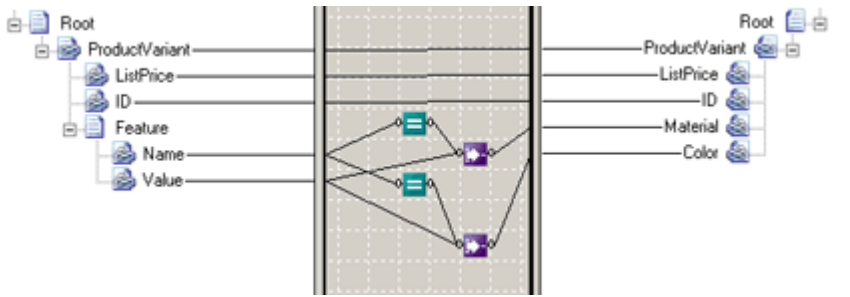
The **Value Mapping (Flattening)** functoid enables you to flatten a portion of an input instance message by converting multiple records into a single record. This is a common operation in converting Microsoft Commerce Server catalogs.

The following code shows a portion of a catalog listing product variants with each feature of the variant in a separate record.

Flattening this portion of the catalog would convert the **Feature** records into attributes of the **ProductVariant** record.

The following figure shows a map that performs this conversion.

Value Mapping (Flattening) Functoid Map



The **Value Mapping (Flattening)** functoid returns the value of its second parameter if its first parameter is true. In this map, the first **Equal** functoid tests to see if the **Name** attribute is equal to "Material". If the attribute is equal to "Material", the **Equal** functoid returns **True**. In turn, this causes the **Value Mapping (Flattening)** functoid to assign the value of the **Value** attribute to the field in the output message.

Scripting Functoid

The **Scripting** functoid enables you to use custom script or code at run time to perform functions otherwise not available. For example, you can call a .NET assembly at run time by using the **Scripting** functoid and writing your own custom functions.

The **Scripting** functoid in Microsoft® BizTalk® Server 2006 extends the number of supported languages from previous versions of BizTalk. Microsoft® BizTalk® Server 2006 supports the following languages for the **Scripting** functoid:

- C# .NET
- JScript .NET
- Visual Basic .NET
- Extensible Stylesheet Language Transformations (XSLT)
- XSLT Call Templates

Another significant difference between the current **Scripting** functoid and earlier versions is that the script need no longer be created and stored in the functoid itself. Instead, you can create the script in a separate .NET assembly and reference the assembly through the **Script** property. Having the script in a separate assembly enables you to use the same script in more than one map. Additionally, you may be able to purchase **Scripting** functoid assemblies from third-party vendors.

You can use **Scripting** functoids created in previous versions of BizTalk Mapper with the current version of BizTalk Mapper. However, you must migrate the functoids first. For more information about how to migrate **Scripting** functoids, see *Migrating Functoids*.

When you add a **Scripting** functoid to a map, you need to configure the script the functoid uses. If you select a **Scripting** functoid, the **Script** property is enabled in the **Properties** window. If you click the ellipsis (...) button for this property, the **Configure Functoid Script** dialog box opens.

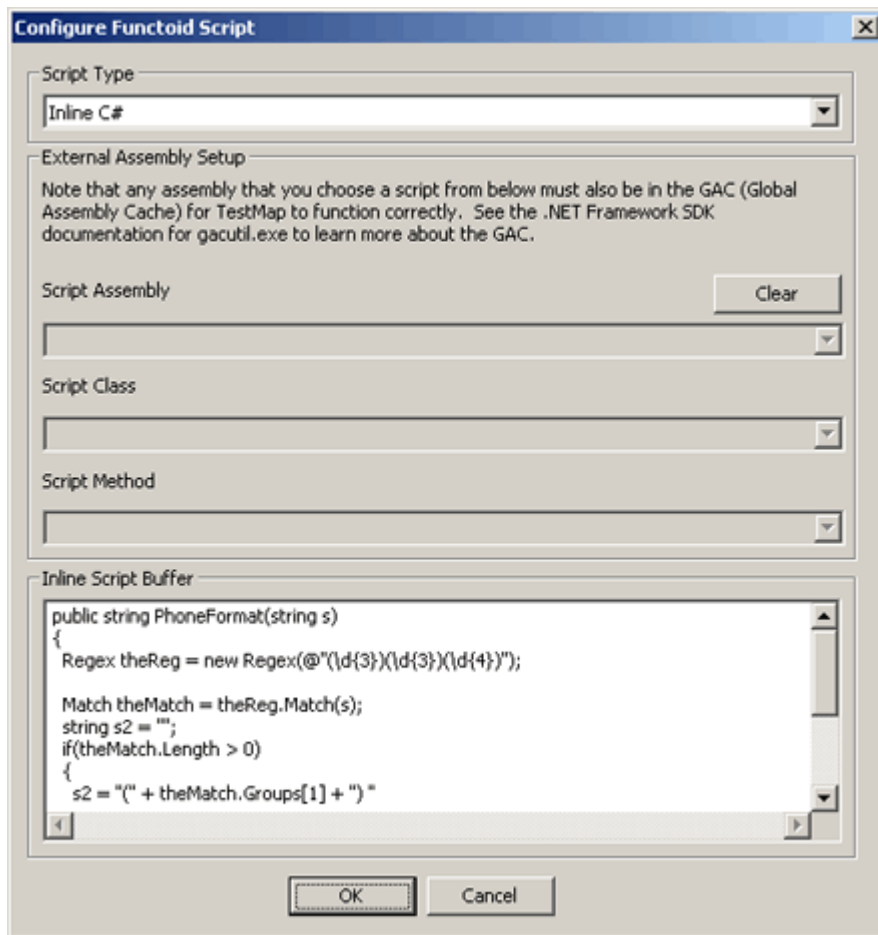
The following table shows the fields of this dialog box.

Configure Functoid Script dialog box field	Description
Script Type	<p>Use this field to select the type of script you want to use in this Scripting functoid.</p> <p>Values:</p> <ul style="list-style-type: none"> • External Assembly. Use this value if you want to associate this Scripting functoid with an assembly in the global assembly cache (GAC). • Inline C#. Use this value if you want to associate this Scripting functoid with C# code in the Inline Script Buffer property. • Inline JScript .NET. Use this value if you want to associate this Scripting functoid with JScript .NET script in the Inline Script Buffer property. • Inline Visual Basic .NET. Use this value if you want to associate this Scripting functoid with Visual Basic .NET code in the Inline Script Buffer property. • Inline XSLT. Use this value if you want to associate this Scripting functoid with XSLT in the Inline Script Buffer property. • Inline XSLT Call Template. Use this value if you want to associate this Scripting functoid with XSLT call templates in the Inline Script Buffer property.
Script Assembly	<p>Select the assembly to associate with this Scripting functoid. Only assemblies referenced in the Project window appear in this list. Note also that you must register assemblies in the GAC.</p> <p>This field is only available when Script Type is set to External Assembly.</p>
Script Class	<p>Select the class within the chosen assembly that you want this Scripting functoid to use.</p> <p>This field is only available when Script Type is set to External Assembly.</p>
Script Method	<p>Select the method within the chosen class that you want this Scripting functoid to use.</p>

<p>Inline Script Buffer</p>	<p>Write or copy the inline script to be used into this text box. Valid languages and scripts include: C#, JScript .NET, Visual Basic .NET, XSLT, and XSLT call templates.</p> <p>This field is only available when Script Type is set to one of the Inline settings.</p>
------------------------------------	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

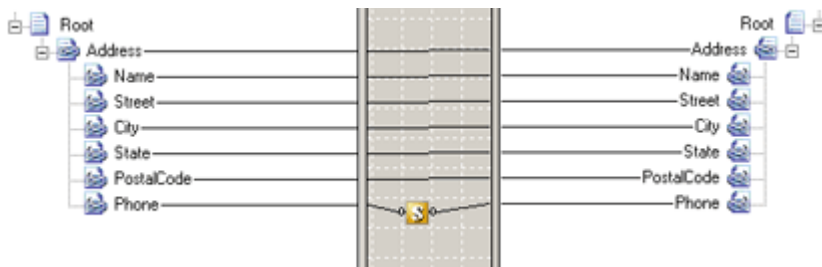
The following figure shows the dialog box with Inline C# selected and code entered in the buffer.

Configure Functoid Script Dialog



The following figure shows how the **Scripting** functoid appears in a map using the C# .Net script to reformat a telephone number.

Scripting Functoid Map



In This Section

- Scripting Using External Assemblies
- Scripting Using Inline C#, JScript .NET, and Visual Basic .NET
- Scripting Using Inline XSLT and XSLT Call Templates

Scripting Using External Assemblies

Scripting with external assemblies is the preferred way to use scripting in Microsoft® BizTalk® Server 2006. External assemblies provide the several advantages:

- Easy code sharing
- Simpler maintenance
- Easier debugging

Re-using the script only requires setting the **Script** property of the **Scripting** functoid. Because the script is stored outside of the map, you can modify the script without changing the map. And you can use the full array of Visual Studio 2005 debugging tools to ensure your script runs correctly.

For a sample function housed in an external assembly, see XML Tools (BizTalk Server Samples Folder).

Notice in the sample that functions used in the **Scripting** functoid may not be **static** (**shared**).

Scripting Using Inline C#, JScript .NET, and Visual Basic .NET

Inline scripts are convenient for custom code that you are unlikely to use elsewhere in your application.

BizTalk saves inline scripts in the Extensible Stylesheet Language Transformations (XSLT) stylesheet defining the map. Because of this, inline scripts may use the same namespaces as any other XSLT stylesheet script. The following table shows the available namespaces.

Namespace	Description
System	The System class.
System.Collection	The collection classes.
System.Text	The text classes.
System.Text.RegularExpressions	The regular expression classes.
System.Xml	The core XML classes.
System.Xml.Xsl	The XSLT classes.
System.Xml.XPath	The XPath classes.
Microsoft.VisualBasic	The Visual Basic script classes.

For more information about namespaces and data types, search on "XSLT Stylesheet Scripting using <msxsl:script>" and on "System.Xml.Xsl.XslTransform" in the .NET Framework collection.

Caution Avoid using the same method signature more than once. When several Scripting functoids have the same method signature, BizTalk selects the first implementation and disregards the others.

In addition to being convenient for one-time scripts, inline scripts are also useful for declaring global variables for use among a number of scripts. For example, in a C# inline script, you could place the following line of code outside of any class.

This creates an **ArrayList**, *statusList*, available to all inline scripts in the map.

For a sample inline script, see XML Tools (BizTalk Server Samples Folder).

Scripting Using Inline XSLT and XSLT Call Templates

You can directly write Extensible Stylesheet Language Transformations (XSLT) stylesheets for use in the **Scripting** functoid. This enables you to perform transformations, that links and built-in functoids may not be able to represent. There are two kinds of XSLT scripts: inline XSLT and XSLT call templates. When you select either in the **Script type** dropdown in the **Configure Functoid Script** dialog box, sample code appears that you may use.

Inline XSLT scripts and inline XSLT call templates may call functions in external assemblies. Making such calls requires setting the **Custom Extension XML** property of the grid. For more information, see **Custom Extension XML (Grid Property)**.

Inline XSLT

An inline XSLT script may only produce output. The **Scripting** functoid may not have any input links. The functoid must also directly link to a record or field in the destination schema.

In addition, the script is responsible for creating the target node and any structures underneath it.

The following input instance message contains two elements representing contact information.

The following inline XSLT script, entered in the script buffer, converts the **Contact** and **ContactType** fields to attributes.

The script produces the following output, assuming an appropriate output schema, when run against the preceding input instance message.

Notice that the absence of links to the **Scripting** functoid does not prevent the XSLT script from getting data from the input instance message. The script specifies paths to the input instance values.

For another example of an inline XSLT script, see XML Tools (BizTalk Server Samples Folder).

Inline XSLT Call Templates

Like an inline XSLT script, an inline XSLT call template must connect directly to a destination node. However, an inline XSLT call template may use links from the source schema and from other functoids.

The call template is responsible for creating the destination node and any of its substructures.

A sample XSLT call template that concatenates two elements appears in the **Input Script Buffer** when you select **Inline XSLT Call Template** in the **Script type** dropdown.

For another example of an inline XSLT call template, see XML Tools (BizTalk Server Samples Folder).

Cascading Functoids

Functoids are said to be cascaded when one functoid is linked to another functoid before it is linked to a record or field in the destination schema. For example, you can create cascading functoids in which two concatenated strings produce a third string fed into a field in the destination schema.

When you cascade functoids, you can create multiple, consecutive transformations in the grid page. While there is no limit to the number of functoids that you can cascade in a page, use cascading wisely. Complex cascading can increase the time to transform an input instance message into an output instance message, significantly reducing run time performance.

Functoid Properties

Functoids have properties that are available for examination and modification in the Visual Studio .NET Properties window. Several functoid properties are only available for particular functoids and are disabled for other functoids. One functoid property, Input Parameters, is especially important to almost all

functoids, and will probably be the most frequently used. For more information about the Input Parameters property, see Functoid Input Parameters.

All functoid properties are categorized as General properties. For a complete listing of functoid properties, see Functoid Properties.

Migrating Functoids

When you migrate a map from previous versions of BizTalk Server to BizTalk Server 2006, any functoids included in the map are also migrated. If the functoids you migrate do not include **Scripting** functoids, no additional migration tasks are required. However if your map includes **Scripting** functoids or custom functoids, you may have additional steps to perform.

In previous versions of BizTalk Server, all custom script included with a **Scripting** functoid was written inline. That is, when you created the functoid, all the script the functoid called during run time was stored with the functoid. If you wanted to use the same script with a different functoid, you either copied and pasted it from one **Scripting** functoid to another, or you rewrote the script from scratch.

BizTalk Server 2006 copies existing inline scripts with the functoids when you migrate a map. However, not all of the scripts may function correctly. BizTalk Server 2006 uses Visual Basic .NET and JScript .NET rather than the VBScript and JScript used in previous versions. The .NET versions of the languages include some changes in syntax.

You will need to rewrite custom functoids. BizTalk Server 2006 expects custom functoids to use the .NET framework. It cannot use the older, COM-based custom functoids. Custom functoids can be rewritten to use the .NET framework. For sample code of a custom functoid, see Custom Functoid (BizTalk Server Sample).

An alternative is to wrap the functionality of the custom functoid in an external assembly and call this assembly through a **Scripting** functoid. The following section describes this process.

To migrate your custom functoids

1. Re-create the functionality of the functoid in a .NET language, such as Microsoft® Visual Basic® .NET, JScript® .NET, or Microsoft® Visual C#® .NET.
2. Create an assembly to contain the new functionality.
3. Register the assembly in the global assembly cache (GAC).
4. Create a reference between the map that contains the **Scripting** functoid and the assembly that contains the rewritten functionality.
5. Configure the **Script** property for the **Scripting** functoid. This property determines what script the **Scripting** functoid calls during run time. You must match the value of this property to the language into which you converted your custom script. For more information about how to configure the Script property, see Editing Functoid Properties and Input Parameters and Scripting Functoid.

6. Build the BizTalk project that contains the map with the **Scripting** functoid.
7. Validate and test the map.

Map Compilation and Testing

BizTalk® Mapper enables you to validate the map to uncover error and warning conditions. You can also test the map against input instance data.

Settings of map and schema properties have a large effect on validation and testing. Topics in this section cover these properties where appropriate.

In This Section

- Map Compilation
- Map Testing

Map Compilation

When you validate maps, the BizTalk Mapper compiler component generates an Extensible Stylesheet Language Transformations (XSLT) style sheet. This creates a compiled map that transforms an instance message defined by the source schema to an instance message defined by the destination schema. Compiling a map enforces the structural rules and transformations specified in the grid pages.

Transformations, such as links, are processed in the same order that records and fields appear in the destination schema. For example, when BizTalk Mapper reaches a destination **Record** or **Field** node with a link, BizTalk Mapper compiles the properties of the link. The action might be a simple copy value from a record or field in the source schema, or the action might involve multiple calculations using functoids and multiple records and fields.

BizTalk Mapper generates warnings in the **Output** window and **Task List** window when the compiler encounters a situation that might yield incorrect output. For example, if a functoid requires one input parameter and has no input parameters, BizTalk Mapper generates a warning in the **Output** window. In general, you should not use a map in a production environment if it is generating warnings.

A link to the generated XSLT style sheet also appears in the **Output** window when the map compiles correctly.

BizTalk Server uses the compiled map to perform the translation of an input instance message to an output instance message.

Map Testing

After your map has compiled without warnings, it is time to test it. This section describes various aspects of testing maps.

In This Section

- Using Map File Properties for Testing
- Validating Instance Data
- Empty Node Values in Source Instance Messages
- Optional Nodes

Using Map File Properties for Testing

The map **Property Pages**, available by right-clicking on the map in the Solution Explorer, control many important behaviors during testing:

- The validation of input and output instances against source and destination schemas
- The choice of whether to create an input instance or use an existing one
- The format of input and output test instances

For information about the map properties, see **Map Property Reference**.

Validating Instance Data

During the process of testing a map, you may want to validate instance data against the source and destination schemas to verify that the instance data adheres to the schema structure. To validate an instance message against the source or destination schema, use the **Validate Instance** command available in BizTalk Editor.

For more information about how to generate and validate instance data by using BizTalk Editor, see Instance Message Generation and Validation and Instance Validation. The **Value** property of the node of a schema also affects how BizTalk generates instance data. For more information see **Schema Node Properties**.

Empty Node Values in Source Instance Messages

There may be times when you do not want content in all of the schema nodes when you test a map.

To create empty node values

1. Generate instance data using BizTalk Editor. For more information about generating instance data, see Generating Instance Messages.
2. Open the input instance message in a text editor, delete the data from elements and attributes that you want to be empty, and then save the modified instance file.

3. Configure BizTalk Mapper to use the file you just modified when the schema is validated and tested. You set the file in the **Property Pages** dialog box for the schema. For more information about properties you can set in the **Property Pages** dialog box, see **Map File Properties**.

Optional Nodes

Using optional nodes will produce a warning when you test your map. There are two conditions in which a source node may be considered optional:

- The actual record or field is optional.
- The source record or field is required, but a parent, grandparent, and farther up the hierarchy is optional.

You may have situations when you have records and fields in a source schema that are optional, but the destination schema requires the corresponding records or fields.

In both of the possible conditions, testing your map produces a warning in the **Output** window. In addition, the output data will not include the target node.

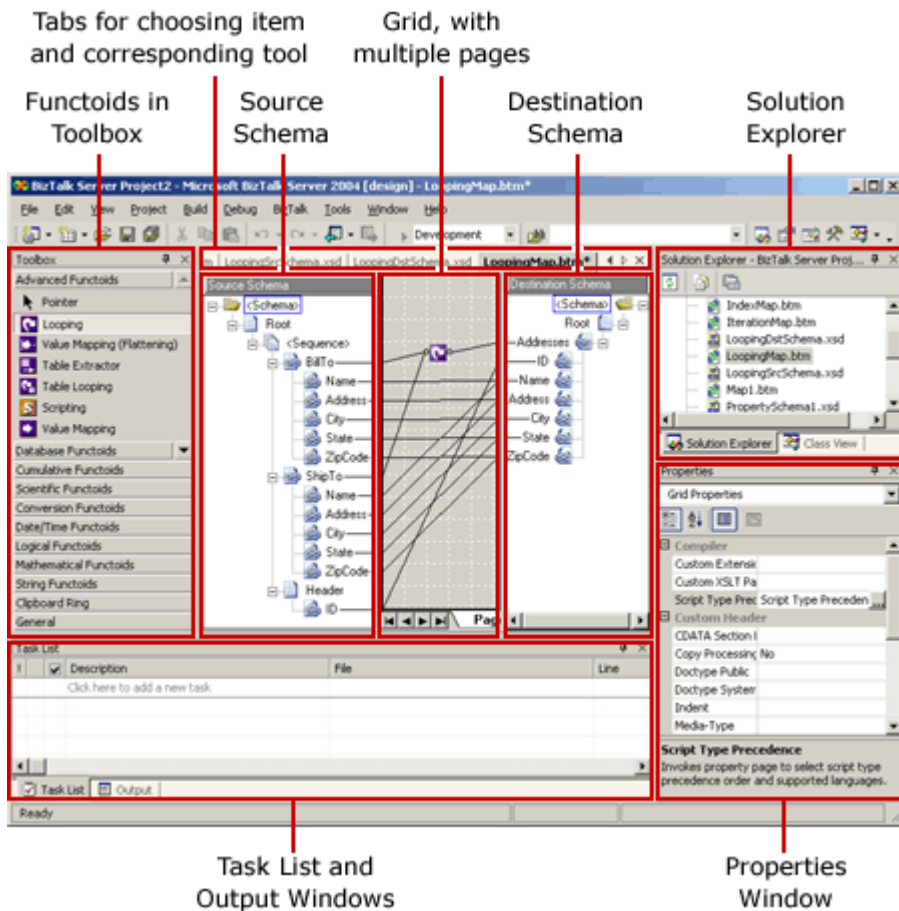
Using BizTalk Mapper

BizTalk® Mapper resides in the Microsoft® Visual Studio® 2005 shell. Some of the functionality in BizTalk Mapper relies on the user interface elements of the Visual Studio 2005 shell. For example, you use the **File**, **Edit**, and **View** menus just as you would for other development in Visual Studio 2005. Information about this common functionality is available from the **Help** menu.

BizTalk Mapper becomes active when you add a new map to a BizTalk project, when you open an existing map (a .btm file), or when you reactivate a map by clicking its tab in the main Visual Studio 2005 editing window.

The following figure shows the views BizTalk® Mapper uses within Microsoft® Visual Studio® 2005.

BizTalk Mapper Views



The views have the following functions:

- **Source schema tree view.** This view shares the main Visual Studio .NET editing window with the destination Schema Tree view and the Grid view, and is located to their left.

As its name suggests, this view displays the schema that describes the instance messages that are the source of the mapping defined by the map. The links that define the mapping lead from the source Schema Tree view to the Grid view, and, ultimately, to the destination Schema Tree view.

Other than serving as the source of links, and a minor exception related to the testing of maps, this view is read-only.

For more information about how BizTalk schemas are represented in a Schema tree view, see BizTalk Representation of Schemas .

- **Destination schema tree view.** This view shares the main Visual Studio .NET editing window with the source Schema Tree view and the Grid view, and is located to their right.

As its name suggests, this view displays the schema that describes the instance messages that are the destination of the mapping defined by the map. The links that define the mapping lead into the destination Schema Tree view from the Grid view, and ultimately from the source schema tree view.

Other than serving as the destination of links and a minor exception related to the testing of maps, this view is read-only.

For more information about how BizTalk schemas are represented in a Schema Tree view, see [BizTalk Representation of Schemas](#).

- **Grid view.** This view shares the main Visual Studio .NET editing window with the source Schema Tree view and the destination schema tree view, and is located between them, with the source Schema Tree view to the left and the destination Schema Tree view to the right.

As its name suggests, this view plays a critical role in the definition of maps, containing the links and functoids that control how data in a source instance message is transformed into an instance message that conforms to the destination schema.

The Grid view can have multiple layers, called grid pages, allowing you to organize complex maps into logical subdivisions of mappings. Grid pages generally use more space than can be displayed at one time, and there are several effective ways to scroll within a grid page.

You actively work in this view to construct your map.

- **Visual Studio .NET Toolbox window.** You use this view to display the functoids available for use in BizTalk maps, and as the source of the drag-and-drop operations to place functoids in a grid page.

The functoids shown in the Toolbox are organized according to their categories. For more information about the available functoids, see [Functoids in Maps1](#) and [Functoid Reference](#).

- **Visual Studio .NET Properties window.** You use this view, and its associated dialog boxes, to examine and set the properties of the links and functoids that you create to define your map.

When you select a link or functoid in a grid page in the Grid view, select a schema node in the source or destination schema tree views, or select a map in the **Solution Explorer** window, the corresponding properties of that link, functoid, schema node, or map appear in the **Properties** window using the standard Visual Studio conventions. For example, the properties are grouped into categories, and can be displayed according to these categories or alphabetically.

For detailed information about the different sets of properties that are available for links, functoids, schema nodes, or the map itself, see [Map Property Reference](#) and [Schema Property Reference](#).

- **Visual Studio .NET Task List and Output windows.** You use these views to examine the results of validating, compiling, and testing your BizTalk maps in much the same way that these views are used when compiling source code and building other types of projects.

In addition to these views, you can interact with several dialog boxes. You usually open these dialog boxes when you are editing a complex property such as the input parameters to a functoid.

You often use the Solution Explorer window in conjunction with BizTalk Mapper. For example, to create a new map, right-click the BizTalk project in the **Solution Explorer** window, click **Add**, click **Add New Item**, and then use the **Add New Item** dialog box to name and create a new map.

In This Section

- Using BizTalk Mapper Commands
- Working with Grid Pages
- How to Change Views Sizes
- How to Customize Colors and Font in BizTalk Mapper
- How to Expand and Collapse the Schema Trees

Using BizTalk Mapper Commands

When BizTalk® Mapper becomes active, it adds a menu called **BizTalk** to the Microsoft® Visual Studio® .NET shell. This menu provides access to the BizTalk Mapper commands and their functionality. When BizTalk Mapper is active, the **BizTalk** menu provides the commands that are specific to editing BizTalk maps.

Where applicable, BizTalk Mapper uses existing Visual Studio .NET menu items for commands that have obvious parallels to standard application functionality. For example, when BizTalk Mapper is active, the **Save** command on the **File** menu saves changes to the map that is currently being edited.

The following table describes the commands within BizTalk Mapper that can be used in the process of developing maps.

Command locations, if any) (menu	Description
Open Map (File Open File...)	Opens a BizTalk map for editing in BizTalk Mapper. Also available on the shortcut menu in Solution Explorer when a map is selected, and by double-clicking a map in Solution Explorer.
Close Map (File Close)	Closes BizTalk Mapper for the current map, prompting to save any unsaved changes. Also available by using the standard Close([X]) button in the upper-right corner of the main editing window in Visual Studio .NET.
New Map (File Add New Item... Map) (Project Add New Item... Map)	Creates a new BizTalk map for editing and opens BizTalk Mapper. Select the map template in the Add New Item dialog box. Also available on the shortcut menu for the BizTalk project in Solution Explorer (Add Add New Item).
Open Source Schema	Allows selection of the source schema in new maps using the BizTalk Type

<i>(in source Schema Tree view of new map only)</i>	Picker dialog box.
Open Destination Schema <i>(in destination Schema Tree view of new map only)</i>	Allows selection of the destination schema in new maps using the BizTalk Type Picker dialog box.
Save Map (File Save map.btm) (File Save map.btm As...) (File Save All)	Saves the BizTalk map currently being edited under its own name, under a new name, or as part of saving all unsaved changes, respectively.
Create Link <i>(drag-and-drop only)</i>	Creates a link between schema nodes in the source and destination schema tree views or between functoids in a grid page. You can create a link by: <ul style="list-style-type: none"> • Dragging a schema node to another schema node in the other schema tree or to a functoid. • Dragging a functoid to a schema node or to another functoid. You can create links by dragging in either direction; in other words, you can start at the source (left end) or the destination (right end) of the link.
Add Functoid <i>(drag-and-drop only)</i>	Adds a functoid to the displayed grid page. You can add a functoid to a grid page by dragging the functoid from the tab in the Visual Studio .NET Toolbox that corresponds to the category of the functoid.
Delete Link and Delete Functoid (Edit Delete) (BizTalk Delete)	Deletes the currently selected links or functoids in the displayed grid page, with confirmation prompting. When applicable, this command is also available on the shortcut menu for the selected links or functoids.
Find Schema Node (Edit Find and Replace	Provides search functionality for node names in the Schema Tree views, which can be useful with large schemas.

Find)	
Properties (View Properties Window) (BizTalk Properties)	<p>Provides access to the properties of links, functoids, source and destination Schema Tree view nodes, and to higher-level objects like the maps themselves.</p> <p>Also available on the shortcut menu for the selected link or functoid, but not for nodes in the schema trees. Pressing F4 also displays the Properties window.</p>
Add Grid Page (BizTalk Add Page)	<p>Adds a new grid page, also known as a layer, to the Grid view.</p> <p>Also available on the shortcut menu for the grid page tabs.</p>
Delete Grid Page (BizTalk Delete Page)	<p>Deletes the displayed grid page, also known as a layer, with confirmation prompting.</p> <p>Also available on the shortcut menu for the grid page tabs.</p>
Reorder Grid Pages (BizTalk Reorder Pages)	<p>Allows reordering of the grid pages, also known as layers, using the Reorder Pages dialog box.</p> <p>Also available on the shortcut menu for the grid page tabs.</p>
Rename Grid Page (BizTalk Rename Page)	<p>Allows renaming of the displayed grid page, also known as a layer.</p> <p>Also available on the shortcut menu for the grid page tabs.</p>
Expand Tree Node (BizTalk Expand Tree Node)	<p>Completely expands the currently selected (and at least partially collapsed) Schema, Group, or Record node in the source or destination Schema Tree view.</p> <p>When applicable, this command is also available on the shortcut menu for the selected node.</p>
Collapse Tree Node (BizTalk Collapse Tree Node)	<p>Collapses the currently selected (and at least partially expanded) Schema, Group, or Record node in the source or destination Schema Tree view.</p> <p>When applicable, this command is also available on the shortcut menu for the selected node.</p>
Grid Preview (BizTalk Grid Preview)	<p>Allows rapid and accurate scrolling within the displayed grid page using the Grid Preview dialog box.</p> <p>This command is also available on the shortcut menu for grid pages.</p>
Replace Schema	Allows replacement of the source or destination schema using the BizTalk

(BizTalk Schema)	Replace	Type Picker dialog box. Also available on the shortcut menu for the source and destination Schema Tree views.
BizTalk Mapper Options (Tools Options BizTalk Mapper)		Allows configuration of various colors in the Grid view, including selection, link, grid, and background. The font used in the schema node display can also be changed using this command.

The following table describes additional map commands that are available on the shortcut menu for the selected map in Solution Explorer.

Schema command	Description
Open	Opens the selected map in BizTalk Mapper. Double-clicking a map also opens it in BizTalk Mapper.
Open With	Allows the selected map to be opened in a variety of XML editors, including BizTalk Mapper.
Test Map	Tests the selected map.
Validate Map	Validates the map.
Exclude From Project	Removes the currently selected map from the current BizTalk project. Use the Add Existing Item command to re-add a map that was previously excluded from the current BizTalk project.
Cut, Copy, Paste	Use these commands to perform the standard Visual Studio .NET behavior of cutting, copying, and pasting entire maps within a BizTalk project.
Delete	Permanently deletes the currently selected map, with a confirmation prompting.
Rename	Allows the currently selected map to be renamed, in place.
Properties	Opens the Property Pages dialog box for the currently selected map, in which some of the properties of the map can be examined and set.

Working with Grid Pages

Creating maps for many business documents can involve numerous links and functoids, often enough to make the grid complex and difficult to understand. The solution to this potential complexity is the introduction of pages to the grid. You can isolate different aspects of your mapping into different pages, and then view and work with those pages separately.

Pages are represented as tabs in the grid. You can add new pages, rename pages, delete pages, move between and within them, and so on. This section provides step-by-step instructions for these various page operations.

In This Section

- How to Manage Grid Pages
- How to Move Between and Within Grid Pages
- How to Move Functoids and Links Between Grid Pages

How to Change Views Sizes

In the process of developing BizTalk maps, you may find that you want to change the sizes of the source or destination schema tree views, or the size of the Grid view. This topic provides step-by-step instructions for these various tasks.

To make the schema tree views or the Grid view taller or shorter

1. Move the mouse pointer to the bottom edge of the Visual Studio .NET main editing window, which displays the schema tree views on either side of the Grid view, until the cursor changes to the standard window vertical resizing icon.
2. Click and hold the left mouse button and drag the window edge either up or down.

You have vertically resized the schema tree views and the Grid view by vertically resizing the entire main editing window.

To make the Schema Tree views or the Grid view wider or more narrow

1. Move the mouse pointer to one of the two pane dividers in the Visual Studio .NET main editing window, which divides the schema tree views from the Grid view between them, until the cursor changes to the standard window horizontal resizing icon.
2. Click and hold the left mouse button and drag the pane edge either left (narrower) or right (wider).

You have horizontally resized one of the schema tree views and the Grid view by changing the amount of the main editing window dedicated to each of these views.

You can also make the schema tree views and the Grid view wider or narrower by horizontally resizing the entire main editing window.

How to Customize Colors and Font in BizTalk Mapper

You can change the colors associated with various types of links and the color of selected objects in the Grid view. You can also change the font used to display the schema tree nodes. This topic provides step-by-step instructions for making such changes.

To change the colors and font used in BizTalk Mapper

1. In Visual Studio .NET, on the **Tools** menu, click **Options**.
2. In the **Options** dialog box, click the **BizTalk Mapper** folder, and, if necessary, expand the **Color** or **Font** sections by clicking their plus (+) icons.
3. Change the colors for the various types of links, compiler warnings, grid foreground, and grid background by using the drop-down color picker associated with each color property.

The following color choices are available:

- **Selected Objects.** Controls the color of the eight squares around a selected functoid and the color of selected links.
 - **Partial Links.** Controls the color of partial links, which are links that exist for a field or record whose parent record is collapsed.
 - **Fixed Links.** Controls the color of fixed links, which are simple value-copy links in a grid page.
 - **Compiler Links.** Controls the color of compiler links, which are the compiler directive links. They are links that are automatically created when a link is set from a field in the source schema tree to a field in the destination schema tree.
 - **Elastic Links.** Controls the color of elastic links, which are simple value-copy links that are dragged from the source schema tree to the destination schema tree. After the link is made, the color of the link changes to the color for fixed links.
 - **Compiler Warnings.** Controls the color of compiler warnings.
 - **Grid Foreground.** Controls the color of the dashed lines in grid pages.
 - **Grid Background.** Controls the color of the background in grid pages.
4. Change the font used in the views in the main editing window by using the **Font** dialog box associated with the property in the Font section.

Access the **Font** dialog box by using the ellipsis () button located at the right end of the **Schema Tree Font** property value box.

How to Expand and Collapse the Schema Trees

When developing BizTalk maps, you are likely to need to expand and collapse the source and destination schema trees to expose or to hide the various schema nodes. This topic provides step-by-step instructions for expanding and collapsing the schema tree.

To completely expand all or part of a schema tree

1. Select the node under which you want to completely expand the schema tree.

The selected node must be a node with a plus (+) or minus (-) icon next to it.

2. On the **BizTalk** menu, or on the shortcut menu for that node, click **Expand Schema Node**.

All nodes below the selected node are completely expanded.

If the schema tree below the selected node is already completely expanded, the **Expand Schema Node** menu item is dimmed.

To completely collapse all or part of a schema tree

1. Select the node under which you want to completely collapse the schema tree.

The selected node must be a node with a plus (+) or minus (-) icon next to it.

2. On the **BizTalk** menu, or on the shortcut menu for that node, click **Collapse Schema Node**.

All nodes below the selected node are completely collapsed.

If the schema tree below the selected node is already collapsed, the **Collapse Schema Node** menu item is dimmed.

This operation will not remember the individual expand and collapse settings of the nodes below the selected node. When you re-expand the collapsed node, previous settings are lost, and you must expand the schema tree below that point node-by-node, or expand it entirely. ☐ To expand a node of a schema tree

1. Click the plus (+) icon next to the node you want to expand.

The selected node is expanded. Whether or not its descendent nodes are expanded or collapsed depends on how the selected node was collapsed and their previous expand and collapse settings.

To collapse a node of a schema tree

1. Click the minus (-) icon next to the node you want to collapse.

The selected node is collapsed.

This operation will remember the individual expand and collapse settings of the nodes below the selected node. When you re-expand the collapsed node by using the plus (+) icon, the previous individual settings are not lost, and the schema tree below that point returns to its previous state.

Creating Maps

The primary user interface for BizTalk® Mapper is displayed on a tab within the Microsoft® Visual Studio® .NET editing window. This display is divided into three panes. The left pane displays the source schema as a tree on the left side. The right pane displays the destination schema as a tree. The middle pane displays the grid as multiple pages. To indicate how you want to map data from the source schema to the destination schema, you draw lines between the records and fields you want to map. These lines are called "links", and they are the most basic way to specify the mapping of data. For more information about linking records and fields, see [Links in Maps](#) .

If you want to implement more advanced mapping methods, you can use functoids, tools available in BizTalk Mapper tabs within the Visual Studio .NET Toolbox. Functoids enable you to create more complex maps. Examples of more complex map elements include:

- Adding the values in two fields in a source schema and putting the result in a field in the destination schema.
- Calculating the average value of a field in a looping record and putting the result in a field in the destination schema.
- Writing a custom script to manipulate instance data as appropriate for your business needs.

For more information about functoids, see [Functoids in Maps](#) .

BizTalk Mapper can support many different mapping scenarios from simple parent-child relationships to detailed, complex looping of records and hierarchies. All data processed by Microsoft® BizTalk® Server 2006 at run time must be in XML. All non-XML data must be translated to an equivalent XML format before mapping. Similarly, when the mapping process is complete, BizTalk Server 2006 uses the output of a mapping operation to create a file format that is recognized by the trading partner or application to which the data is sent.

BizTalk Mapper includes a compiler. This tool-level component generates the Extensible Stylesheet Language Transformations (XSLT) needed to transform or translate input instance messages to output instance messages.

This section provides task-specific information about using BizTalk Mapper to create the mapping between two schemas. It assumes that you already have BizTalk Mapper open, and have chosen your source and destination schemas.

In This Section

- [Managing Maps Within Projects](#)
- [Using Links to Specify Record and Field Mappings](#)
- [Using Functoids to Create More Complex Mappings](#)

- How to Create a Map without Maps

Managing Maps Within Projects

This section provides step-by-step instructions for working with entire maps—for example, the steps involved in creating a map, specifying the schemas in the map, and saving maps. For information about building the mapping structure using links and functoids, see [Using Functoids to Create More Complex Mappings](#).

In This Section

- How to Create New Maps
- How to Add Existing Maps
- How to Open, Save, Close, and Rename Maps
- How to Replace Schemas

How to Create New Maps

To build a new BizTalk map, there are three high-level steps to perform:

1. Create the new map within a BizTalk project.
2. Associate source and destination schemas with the map.
3. Build the set of links and, perhaps, intermediate functoids specifying how the source schema maps to the destination schema.

In the current context, the first two of these three steps is considered "creating" the map. The third step is considered "building" the map. For step-by-step instructions on many of the tasks related to the process of building the map, see [Using Functoids to Create More Complex Mappings](#).

To create a new map within a BizTalk project

1. Right-click a BizTalk project in Solution Explorer, click **Add**, and then click **Add New Item**.
2. In the **Add New Item** dialog box, in the **Templates** area, click **Map**.
3. Select the text in the **Name** box, type a name for the map, and then click **Open**.

BizTalk Mapper opens in the Microsoft® Visual Studio® .NET editing window, showing three distinct panes, side-by-side. From left to right, these panes show the source schema, the grid (which may have multiple pages), and the destination schema.

4. In BizTalk Mapper, in the left pane, click **Open Source Schema**.

5. In the **BizTalk Type Picker** dialog box, expand the **Schema** node in the tree, if necessary, select the appropriate source schema, and then click **OK**.

If only a single root exists in the source schema, or a root node has been established for the schema using the **Root Reference** property of the **Schema** node, the source schema opens in the left pane, and you can proceed to step 7.

6. If the source schema has multiple root nodes, and no root node has been established for the source schema using the **Schema** node's **Root Reference** property, in the **Root Node for Source Schema** dialog box, select the appropriate root node, and click **OK**.

7. In BizTalk Mapper, in the right pane, click **Open Destination Schema**.

8. In the **BizTalk Type Picker** dialog box, expand the **Schema** node in the tree, if necessary, select the appropriate destination schema, and then click **OK**.

If only a single root exists in the destination schema, or a root node has been established for the destination schema using the **Root Reference** property of the **Schema** node, the destination schema opens in the right pane, and you will not need to perform step 9.

9. If the destination schema has multiple root nodes, and no root node has been established for the destination schema using the **Root Reference** property of the **Schema** node, in the **Root Node for Target Schema** dialog box, select the appropriate root node, and click **OK**.

The destination schema opens in the right pane.

How to Add Existing Maps

There may be times when you want to add an existing map to a BizTalk project. Before doing so, you must ensure that the source and destination schemas of the map are included in the BizTalk project to which you are adding the map; or, referenced by the corresponding .NET assembly.

To add an existing map to a BizTalk project

1. Right-click a BizTalk project in Solution Explorer, point to **Add**, and then click **Add Existing Item**.
2. In the **Add Existing Item** dialog box, browse to the folder containing the map to be added, select it, and then click **Open**.

The map opens in BizTalk Mapper. The newly added map also appears as a child of the current BizTalk project in Solution Explorer.

How to Open, Save, Close, and Rename Maps

In Microsoft® BizTalk® Server 2006, maps exist as files in the file system with .btm extensions. Nevertheless, it is much more common to work with maps as items in a BizTalk project, from which you perform operations such as opening, saving, and closing maps.

To open a map

1. In Solution Explorer, select the map you want to open.
2. On the **View** menu, click **Open**.

The map opens in BizTalk Mapper.

To save a map

1. In Solution Explorer, select the map you want to save.
2. On the **File** menu, click **Save <Name of Map>**.

To save a map to a new location

1. In Solution Explorer, select the map you want to save to a new location.
2. On the **File** menu, click **Save <Name of Map> As**.
3. In the **Save File As** dialog box, browse to the folder location where you want to save the map.
4. In the **File name** box, type a name for the file, and then click **Save**.

To rename a map

1. In Solution Explorer, right-click the map that you want to rename, and then click **Rename**.
2. In the editing box that appears in the location of the map in Solution Explorer, type the new name for the map, or alter its existing name, and then press ENTER.

To close a map

1. In Solution Explorer, select the map you want to close.
2. On the **File** menu, click **Close**.

How to Replace Schemas

There may be times when you want to replace either the source or destination schema in an existing map, such as when you receive an updated schema from a trading partner.

To replace a source or destination schema

1. Right-click in either the source or destination schema tree view, and then click **Replace Schema**.
2. In the **BizTalk Type Picker** dialog box, expand the **Schema** node in the tree, if necessary, select the appropriate schema, and then click **OK**.

If only a single root exists in the replacement schema, or a root node has been established for the replacement schema using the **Root Reference** property of the **Schema** node, the replacement schema opens in the relevant pane, and you will not need to perform step 3.

3. If multiple root nodes exist in the destination schema, and no root node has been established for the destination schema using the **Root Reference** property of the **Schema** node, in the **Root Node for <Source/Target> Schema** dialog box, select the appropriate root node, and then click **OK**.

The replacement schema opens in the relevant pane.

Using Links to Specify Record and Field Mappings

In BizTalk® Mapper, a link is the way you associate a data item in the source schema with a data item in the destination schema. Typically, in a completed map there are many links between the source schema and the destination schema. All together the links specify how the data in the source instance messages will be transformed into a semantically equivalent, but syntactically distinct, destination instance messages.

This section provides task-specific information about creating new links, working with existing links, creating links automatically, and other linking operations.

In This Section

- How to Create Links
- How to Edit Link Properties
- How to Link Records Automatically
- How to Manage Existing Links
- How to Select Multiple Links
- How to Configure Node Hierarchy Matching
- How to Set the Source Links Compiler Value
- How to Show and Hide Compiler Links

How to Create Links

Creating a link from a **Record** or **Field** node in a source schema to a **Record** or **Field** node in a destination schema is the most basic activity in creating maps. This topic provides step-by-step instructions for several variations of this activity, including creating links to and from functoids. For additional information about working with functoids, see *Using Functoids to Create More Complex Mappings*.

The instructions in this topic assume you already have a BizTalk map open, and that you have chosen source and destination schemas for the map. For more information about opening maps and choosing schemas for the map, *Managing Maps Within Projects*.

To create links between Field and Record nodes

1. In BizTalk Mapper, drag a **Field** or **Record** node from the source schema tree to a **Field** or **Record** node in the destination schema tree.

- Or -

2. In BizTalk Mapper, drag a **Field** or **Record** node from the destination schema tree to a **Field** or **Record** node in the source schema tree.

There are several things to consider when creating links:

- The data type of a **Field** or **Record** node in the source schema tree should match the data type of a **Field** or **Record** node to which it is linked in the destination schema tree.
- If a **Field** or **Record** node in the source schema is optional, and a particular source instance message does not contain the corresponding element or attribute, BizTalk Mapper will not create a corresponding element or attribute in the destination instance message, even if the **Field** or **Record** nodes have a direct link between them in the map.
- You cannot link to a **Field** or **Record** node in the destination schema that has a constant value associated with it. On the other hand, you can link to a required **Field** or **Record** node in the destination schema that has a default value associated with it. Note, however, that when you test the map, the default value will be used.
- You cannot create a link to or from the **Any Element**, **Any Attribute**, **Sequence Group**, or **Choice Group** nodes. For more information about these types of nodes, see Any Element Nodes, Sequence Group Nodes, and Choice Group Nodes, respectively.
- You may need to expand the schema trees to view the fields that you want to map. For more information, see Expanding and Collapsing the Schema Trees.

To create links between Record or Field nodes and functoids

1. In BizTalk Mapper, drag a **Record** or **Field** node from the source or destination schema to a functoid in a grid page.

- Or -

2. Drag the functoid from a grid page to a **Record** or **Field** node in the source or destination schema.

When you create a link between a **Record** or **Field** node in the source schema and a functoid, you are creating an input to that functoid. When you create a link between a **Record** or **Field** node in the destination schema and a functoid, you are creating an output from that functoid.

To create links between functoids

1. In BizTalk Mapper, drag one functoid to another functoid in a grid page.

How to Edit Link Properties

Links have several properties that appear in the Visual Studio .NET Properties window and that you can set: **Label**, **Source Links**, and **Target Links**. Setting the **Label** property provides a short description of the link and is especially helpful in table-driven looping. The **Source Links** property determines whether the link represents the text value of an element or the name of the element. The **Target Links** property enables you to control the order in which links are processed in the map.

For information about the **Label** property, see Managing Existing Links. For information about setting and using the **Source Links** and **Target Links** properties, see Configuring Node Hierarchy Matching and Setting the Source Links Compiler Value.

How to Link Records Automatically

There are two ways to create record-to-record links automatically:

- **By node name.** Using this technique, BizTalk Mapper attempts to match the **Record** and **Field** nodes within the **Record** nodes being linked according to the names of the corresponding nodes, regardless of their structure, within the **Record** nodes being linked.
- **By structure.** Using this technique, BizTalk Mapper attempts to match the **Record** and **Field** nodes within the **Record** nodes being linked according to the structures of those **Record** nodes, regardless of names of the corresponding nodes within those structures.

This topic provides step-by-step instructions for switching between these two techniques.

To choose the type of automatic record-to-record linking done by BizTalk Mapper and generate links

1. With the relevant map open in BizTalk Mapper, click in the Grid view to display the grid properties in the Visual Studio .NET Properties window.
2. Set the **Autolink By** property in the **General** category to either **Node Name** or **Structure** using the drop-down list.
3. Hold down the SHIFT key and draw a link from a source schema element to a destination schema element.

BizTalk Mapper creates links between elements under the newly elements according to the Autolink By property.

How to Manage Existing Links

Sometimes you may need to change the source or destination of a link, name or rename a link, or delete a link. This topic provides step-by-step instructions for performing these types of link operations.

To change the source or destination of a link

1. In BizTalk Mapper, in a grid page, click a link to select it.

The endpoints of a selected link in the grid page are highlighted with small blue boxes.

2. Drag either endpoint of the link to the new schema node or functoid to which you want it to connect it.

As you drag an endpoint, the cursor becomes a crosshair. If you point to a schema node or functoid (or other object) which cannot accept the link, the cursor becomes a circle with a line through it.

To name or rename a link

1. In BizTalk Mapper, in a grid page, click a link to select it.

The endpoints of a selected link in the grid page are highlighted with small blue boxes.

2. In the Visual Studio .NET Properties window, provide a (new) name for the link using the **Label** property.

To delete a link

1. In BizTalk Mapper, in a grid page, click a link to select it.

The endpoints of a selected link in the grid page are highlighted with small blue boxes.

2. On the **View** menu, click **Delete**.

You can also press the **DELETE** key or right-click the link and click **Delete** on the shortcut menu.

3. In the confirmation dialog box, click **Yes**.

How to Select Multiple Links

In some circumstances, you may want to perform operations on more than one link at a time. In such cases, you can select all of the links of interest and then perform the operation. This is useful, for example, when you want to move a set of functoids and their connecting links from one grid page to another.

To select multiple links

1. In BizTalk Mapper, in a grid page, click the first link to select it.

The endpoints of the first selected link in the grid page are highlighted with small blue boxes.

2. In BizTalk Mapper, in the same grid page, hold down the **CTRL** or the **SHIFT** key, and then click another link to select it as well.

The endpoints of this selected link in the grid page are also highlighted with small blue boxes.

3. Repeat step 2 as required to select all of the links of interest.

How to Configure Node Hierarchy Matching

When you create a link in a map, BizTalk automatically creates compiler links to implement the link you've drawn. The **Target Links** property of a link controls how BizTalk draws the compiler links. For more information about the **Target Links** property, see Node-Hierarchy Level Matching.

To set the Target Links link property

1. In BizTalk Mapper, in a grid page, click a link to select it.

The endpoints of a selected link in the grid page are highlighted with small blue boxes.

2. In the Visual Studio .NET Properties window, set the **Target Links** property to one of the following choices:
 - **Flatten Links.** The hierarchy in the source record node is flattened to the linked-to record node in the destination schema.
 - **Match Links Top Down.** Node matching is performed level-to-level from the top down.
 - **Match Links Bottom Up.** Node matching is performed level-to-level from the bottom up.

How to Set the Source Links Compiler Value

You can use the **Source Links** property of a link to specify how a value is retrieved from the source node and applied to the destination node. This topic explains the available choices and how to choose among them.

To set the Source Links link property

1. In BizTalk Mapper, in a grid page, click a link to select it.

The endpoints of a selected link in the grid page are highlighted with small blue boxes.

2. In the Visual Studio .NET Properties window, set the **Source Links** property to one of the following choices:
 - **Copy Name.** The name of the node in the source schema is used as the value of the linked node in the destination schema.
 - **Copy Text Value.** The value corresponding to the node in the source schema (element data or an attribute value) is used as the value of the linked node in the destination schema. This choice is the default. For example, `<Node>Hello<Name>Chris</Name>Cannon</Node>` would result in "Hello".

- **Copy Text and Subcontent Value.** The value corresponding to the record node, and the value of all its child nodes, and their child nodes, in the source schema (element data and attribute values) are combined as the value of the linked node in the destination schema. For example, `<Node>Hello<Name>Chris</Name>Cannon</Node>` would result in "Hello Chris Cannon".

How to Show and Hide Compiler Links

When you compile a map, BizTalk Mapper creates additional links, known as compiler links to account for all linking needed in the map. Some of these links are only implied by the links you created. When you compile, or test, a map, the final line in the Visual Studio .NET Output window allows you to show or hide these additional compiler links in the main window. By default, the compiler links appear as red dashed lines.

To show or hide compiler links

1. In Solution Explorer, right-click the map whose compiler links you want to view, and then click **Test Map**.
2. In the Visual Studio .NET Tasks List window, scroll to the end and double-click the line that says **Double-click here to show/Hide compiler links**.

To change the display state of compiler links from shown to hidden, or from hidden to shown, just double-click that line again.

To test a map, you must configure the properties for the input and output instances. For more information about how to configure these properties, see [Configuring Map Validation and Test Parameters](#).

Using Functoids to Create More Complex Mappings

Functoids play a crucial role in many mapping scenarios. Without functoids, you can copy element and attribute data, but you cannot, to any significant extent, manipulate the values themselves. Using functoids, almost any transformation is possible. For example, with a functoid you can take two values from entirely different locations, add them together, and place the sum in the destination schema.

Functoids appear in the Visual Studio 2005 Toolbox, one toolbox tab per category, when you are editing a BizTalk map. After you open the Toolbox and choose a category of functoids by clicking on the corresponding tab, you drag the functoid onto a grid page. Then you create input and output links between the functoid and either schema nodes or another functoid. Input links correspond to input parameters and lead to a functoid from the left; an output link corresponds to the output parameter and leaves a functoid to the right.

Like other map elements, functoids have properties. One of the most important properties of a functoid is its set of input parameters. For more information, see [How to Add Basic Functoids to a Map](#).

This section provides step-by-step instructions for working with functoids within BizTalk maps. For reference information about functoids, organized by category, see **Functoid Reference**.

This section contains:

- How to Open the Functoid Toolbox
- How to Add Basic Functoids to a Map
- Adding Advanced Functoids to a Map
- Editing Functoid Properties and Input Parameters
- How to Select Multiple Functoids
- How to Replace Functoids
- How to Delete Functoids

How to Open the Functoid Toolbox

After you create a map and select the source and destination schemas, you can place functoids on the grid. The functoids appear in the Visual Studio .NET Toolbox, one toolbox tab per functoid category.

To open the functoid toolbox

1. In Visual Studio .NET, with a BizTalk map open and active in the Editor window, on the **View** menu, click **Toolbox**.
2. **Important** If the functoid tabs are not visible in the Toolbox, ensure that you have BizTalk Mapper open by having selected a BizTalk map for editing in the main editing window.

How to Add Basic Functoids to a Map

Many functoids are very simple to use. These are referred to here as basic functoids to distinguish them from the functoids in the **Advanced** category. Basic functoids comprise the remaining categories of functoids such as Conversion, Cumulative, Database, Date and Time, Logical, Mathematical, Scientific, and String.

Basic functoids, in general, have simple types, such as numbers and strings, as their input and output links.

Using a basic functoid involves adding it to a grid page, creating input links to the functoid, coming from the left, and creating output link from the functoid, leaving to the right. This topic provides step-by-step instructions for these operations.

To add a basic functoid to a map

1. With the Visual Studio .NET Toolbox active, click the appropriate tab to select the category of the functoid you want to use.

The list of available functoids in the chosen category appears.

2. Drag the functoid you want to use from the Toolbox to the appropriate location on a grid page.

To create input links to a basic functoid

1. Drag a record or field node from the source schema to the basic functoid in the displayed grid page.

- Or -

In the displayed grid page, drag another functoid, which is located farther to the left, to the basic functoid to which you want to create an input link.

- Or -

Drag the basic functoid in the displayed grid page to a record or field node in the source schema.

- Or -

In the displayed grid page, drag the basic functoid to which you want to create an input link to another functoid that is located farther to the left.

2. Repeat step 1 as necessary to establish the complete set of input links (though perhaps not the entire set of input parameters) to the basic functoid.

To create the output link from a basic functoid

1. Drag a record or field node from the destination schema to the basic functoid in the displayed grid page.

- Or -

2. In the displayed grid page, drag another functoid, which is located farther to the right, to the basic functoid to which you want to create the output link.

- Or -

3. Drag the basic functoid in the displayed grid page to a record or field node in the destination schema.

- Or -

4. In the displayed grid page, drag the basic functoid to which you want to create the output link to another functoid that is located farther to the right.

Adding Advanced Functoids to a Map

Functoids in the **Advanced** category are more complex to understand and use than the functoids in the other categories, together classified as basic functoids. You use advanced functoids when you need more complex maps. These maps might need to count records, loop through records with a variable number of subrecords, or run an arbitrarily complex script.

This section provides step-by-step instructions for adding functoids in the **Advanced** category to your maps, including correctly setting their input and output parameters.

In This Section

- How to Add Index Functoids to a Map
- How to Add Iteration Functoids to a Map
- How to Add Looping Functoids to a Map
- How to Add Mass Copy Functoids to a Map
- How to Add Record Count Functoids to a Map
- How to Add Scripting Functoids to a Map
- How to Add Table Looping and Table Extractor Functoids to a Map
- How to Add Value Mapping Functoids to a Map
- How to Add Value Mapping (Flattening) Functoids to a Map

How to Add Index Functoids to a Map


The **Index** functoid enables you to select information from a specific record in a series of looping records. Each **Index** functoid selects information from a single field.

For conceptual information about the **Index** functoid, see Index Functoid.


To add the Index functoid to a map and configure it

1. With the Visual Studio .NET Toolbox active, click the **Advanced Functoids** tab to select that category of functoids.

The list of advanced functoids in the chosen category appears.

2. Drag the **Index** functoid  from the Toolbox to the appropriate location on a grid page.
3. To establish the field input parameter for the **Index** functoid, create an input link by dragging a field within a looping record from the source schema to the **Index** functoid, or dragging the **Index** functoid to a field within the looping record in the source schema.
4. To establish at least one index input parameter for the **Index** functoid, perform the following steps:
 - a. With the **Index** functoid selected, click the ellipsis (...) button associated with the **Input Parameters** property in the **Properties** window.

The **Configure Functoid Inputs** dialog box appears.

- b. In the **Configure Functoid Inputs** dialog box, click the **Insert New Parameter** button  above the **Input parameters** list box.
 - c. In the edit box that appears within the **Input parameters** list box, type the index input parameter as a numeric value. Repeat as necessary if additional index values are required, and then click **OK**.
5. To use the output parameter from the **Index** functoid, create an output link by dragging the **Index** functoid to a field in the destination schema, or by dragging a field in the destination schema to the **Index** functoid.

How to Add Iteration Functoids to a Map

The **Iteration** functoid outputs the index of the current record in a looping structure, beginning at 1 for the first record, 2 for the second record, and so on.

For conceptual information about the **Iteration** functoid, see [Iteration Functoid](#).

To add the Index functoid to a map and configure it

1. With the Visual Studio .NET Toolbox active, click the **Advanced Functoids** tab to select that category of functoids.

The list of advanced functoids in the chosen category appears.

2. Drag the **Index** functoid

Iteration functoid

from the Toolbox to the appropriate location on a grid page.

3. To establish the looping record input parameter for the **Iteration** functoid, create an input link by dragging a looping record from the source schema to the **Iteration** functoid, or by dragging the **Iteration** functoid to a looping record in the source schema.
4. To use the output parameter from the **Iteration** functoid, create an output link by dragging the **Iteration** functoid to a field in the destination schema, or by dragging a field in the destination schema to the **Iteration** functoid.

How to Add Looping Functoids to a Map

The **Looping** functoid combines multiple records or fields in the source schema into a single record in the destination schema.

For conceptual information about the **Looping** functoid, see [Looping Functoid](#).

Under certain conditions, some functoids might not behave as expected when they are used in a map with a **Looping** functoid. If such a functoid meets the following conditions, it does not produce the expected results:

- The functoid has more than one input link.
- Two or more of the functoid input links are linked to child fields of the input records to the **Looping** functoid, where the child fields are not siblings.
- The functoid has an output link that is linked to a child field of the output record of the **Looping** functoid.

To add the **Looping** functoid to a map and configure it

1. With the Visual Studio .NET Toolbox active, click the **Advanced Functoids** tab to select that category of functoids.

The list of advanced functoids in the chosen category appears.

2. Drag the **Looping** functoid

Looping functoid

from the Toolbox to the appropriate location on a grid page.

3. To establish the input parameters for the **Looping** functoid, create an input link by dragging a record or field from the source schema to the **Looping** functoid, or dragging the **Looping** functoid to a record or field in the source schema. Repeat as required to include all of the relevant input records or fields to the **Looping** functoid.
4. To use the output parameter from the **Looping** functoid, create an output link by dragging the **Looping** functoid to a record or field in the destination schema, or by dragging a record or field in the destination schema to the **Looping** functoid.

How to Add Mass Copy Functoids to a Map

The **Mass Copy** functoid enables your maps to use schemas that include **any** and **anyAttribute** elements. These elements are, in essence, wildcards provided in the XML Schema definition language to match unknown structures or sets of attributes.

For conceptual information about the **Mass Copy** functoid, see [Mass Copy Functoid](#).

To add the **Mass Copy** functoid to a map and configure it

1. With the Visual Studio .NET Toolbox active, click the **Advanced Functoids** tab to select that category of functoids.

The list of advanced functoids in the chosen category appears.

2. Drag the **Mass Copy** functoid

Mass Copy functoid

from the Toolbox to the appropriate location on a grid page.

3. To establish the input parameter for the **Mass Copy** functoid, create an input link by dragging a record from the source schema to the **Mass Copy** functoid, or dragging the **Mass Copy** functoid to a record in the source schema.
4. To use the output parameter from the **Mass Copy** functoid, create an output link by dragging the **Mass Copy** functoid to a record in the destination schema, or by dragging a record in the destination schema to the **Mass Copy** functoid.

How to Add Record Count Functoids to a Map

The **Record Count** functoid enables you to generate a count of the number of times a record occurs in an instance message.

For conceptual information about the **Record Count** functoid, see Record Count Functoid.

To add the Record Count functoid to a map and configure it

1. With the Visual Studio .NET Toolbox active, click the **Advanced Functoids** tab to select that category of functoids.

The list of advanced functoids in the chosen category appears.

2. Drag the **Record Count** functoid

Record Count functoid

from the Toolbox to the appropriate location on a grid page.

3. To establish the input parameter for the **Record Count** functoid, create an input link by dragging a looping record from the source schema to the **Record Count** functoid, or dragging the **Record Count** functoid to a looping record in the source schema.
4. To use the output parameter from the **Record Count** functoid, create an output link by dragging the **Record Count** functoid to a field in the destination schema, or by dragging a field in the destination schema to the **Record Count** functoid.

How to Add Scripting Functoids to a Map

The **Scripting** functoid enables you to use custom script or code at run time to perform functions otherwise not available. For example, you can call a COM object at run time by using the **Scripting** functoid and writing your own custom script.

For conceptual information about the **Scripting** functoid, see Scripting Functoid.

To add the Scripting functoid to a map and configure it

1. With the Visual Studio .NET Toolbox active, click the **Advanced Functoids** tab to select that category of functoids.

The list of advanced functoids in the chosen category appears.

2. Drag the **Scripting** functoid

Scripting functoid



from the Toolbox to the appropriate location on a grid page.

3. Select the **Scripting** functoid that you just added to the displayed grid page.
4. In the Visual Studio .NET Properties window, click the ellipsis (...) button associated with the **Script** property.

The **Configure Functoid Script** dialog box appears.

5. In the **Configure Functoid Script** dialog box, in the **Script Type** drop-down list, select the type of your script.
6. If you selected **External Assembly** as the script type, use the **Script Assembly**, **Script Class**, and **Script Method** drop-down lists, in that order, to select the assembly, class, and method, respectively, to associate with this **Scripting** functoid.
8. If you selected something other than **External Assembly** as the script type (one of the inline choices), use the **Inline Script Buffer** text box to enter your script in the language you selected.
10. Click **OK**.
11. If your script or the associated method in an external assembly requires input parameters, create the appropriate number and type of input links as you would for a basic functoid.
12. In most circumstances, your **Scripting** functoid will produce an output value used to populate a field in the destination schema, or as input to another functoid, in much the same way that basic functoids do.

How to Add Table Looping and Table Extractor Functoids to a Map

The **Table Looping** and **Table Extractor** functoids are used together. The **Table Looping** functoid has an internal table you configure. For each input record or field, the **Table Looping** functoid outputs the rows of the table, one at a time. The **Table Extractor** functoid extracts the desired item from a row and passes it on to the output instance message.

For conceptual information about the **Table Looping** and **Table Extractor** functoids, see Table Looping and Table Extractor Functoids.

To add the Table Looping and Table Extractor functoids to a map and configure it

1. With the Visual Studio .NET Toolbox active, click the **Advanced Functoids** tab to select that category of functoids.

The list of advanced functoids in the chosen category appears.

2. Drag the **Table Looping** functoid

Table Looping functoid



from the Toolbox to the appropriate location on a grid page.

3. Drag a record or field from the source schema to the newly added **Table Looping** functoid. As the first input parameter to the **Table Looping** functoid, the number of occurrences of this record or field in an instance message will control the number of times this functoid produces output. For example, if a looping record is dragged to the functoid, and an instance message that has 10 occurrences of this record is processed, and the table grid has been configured with one row of sources of column data, the **Table Looping** functoid will iterate 10 times, producing 10 output rows for extraction by a **Table Extractor** functoid, and allowing 10 destination records to be easily constructed.
4. Drag a record or field from the destination schema to the **Table Looping** functoid. This link ensures the creation of the node in the destination schema.
5. Select the newly added **Table Looping** functoid, and in the **Properties** window, click the ellipsis (...) button associated with its **Input Parameters** property.

The **Configure Functoid Inputs** dialog box appears.

6. In the **Configure Functoid Inputs** dialog box, click the **Insert New Parameter** button

Insert New Parameter button



to create the second input parameter and type a number that represents the number of columns that will be available in the table you are creating for this **Table Looping** functoid.

7. In the **Configure Functoid Inputs** dialog box, continue clicking the **Insert New Parameter** button

Insert New Parameter button



and entering any constant values that appears in your configured table grid. The order in which you create these constants is not important in this dialog box as long as the first and second parameter values, the number of rows and columns, respectively, retain their positions at the beginning of the input parameter list. When complete, click **OK**.

The **Configure Functoid Inputs** dialog box closes.

8. Drag zero or more record or field nodes from the source schema to the **Table Looping** functoid that you recently added. Each of these record and field nodes is added to the end of the input parameter list, and therefore will be available when the table grid is configured in a latter step. Like the table data constants added earlier (not the row and column count constants), the order in which these record and field nodes are added is not ultimately relevant.
9. Because labeled links are displayed in the **Configure Functoid Inputs** and **Table Looping Configuration** dialog boxes using their label rather than their XPath, it is extremely helpful to provide such labels for any links that connect as input to the **Table Looping** functoid, such as those created in step 7. (Labeling the scoping link created in step 3 is not especially important.) To label a link, follow these steps:
 - Select a link in the displayed grid page.
 - In the Visual Studio .NET Properties window, provide a descriptive name for the **Label** property. For example, you might give a name like "link2ndAuthro" to a link coming from a field called "Second Author".
10. Select the newly added **Table Looping** functoid, and in the **Properties** window, click the ellipsis (...) button associated with the **Table Looping Grid** property associated with that functoid.

The **Table Looping Configuration** dialog box appears.

11. In the **Table Looping Configuration** dialog box, use the drop-down lists associated with each table cell to configure at least one, and possibly multiple, rows in the grid. The choices available in the drop-down lists are the constants and links that you have configured in steps 6-8 as input parameters 3 and up to the **Table Looping** functoid. (Input parameters 1 and 2 do not appear in these drop-down lists.) When complete, click **OK**.

The **Table Looping Configuration** dialog box closes.

12. Drag as many **Table Extractor** functoids

Table Extractor functoid



from the Toolbox to the displayed grid page as needed.

13. To create the first input parameter for one of the **Table Extractor** functoids added in step 9, drag it to the relevant **Table Looping** functoid to its left.

14. To create the second input parameter for the same **Table Extractor** functoid, select the functoid, and in the **Properties** window, click the ellipsis (...) button associated with its **Input parameters** property.

The **Configure Functoid Inputs** dialog box appears.

15. In the **Configure Functoid Inputs** dialog box, click the **Insert New Parameter** button

Insert New Parameter button

to create the second input parameter and type the number of the column in the table grid of the corresponding **Table Looping** functoid from which you want to extract data. Click **OK**.

The **Configure Functoid Inputs** dialog box closes.

16. To use the output of the **Table Extractor** functoid, drag the **Table Extractor** functoid to a record or field node in the destination schema, or drag a record or field node in the destination schema to the **Table Extractor** functoid. The element or attribute value in a destination instance message corresponding to this record or field node in the destination schema will be populated with the value from (in the case of constants), or the value indicated by (in the case of links), the specified cell in the table grid.
17. Repeat steps 12, 13, 14, and 15 for each of the **Table Extractor** functoids added in step 11.

How to Add Value Mapping Functoids to a Map

The **Value Mapping** functoid returns the value of its second parameter if its first parameter is true. A common use of the functoid is to change the attributes of a field into the attributes of a record.

For conceptual information about the **Value Mapping** functoid, see Value Mapping Functoid.

To add the Value Mapping functoid to a map and configure it

1. With the Visual Studio .NET Toolbox active, click the **Advanced Functoids** tab to select that category of functoids.

The list of advanced functoids in the chosen category appears.

2. Drag the **Value Mapping** functoid

Value Mapping functoid



from the Toolbox to the appropriate location on a grid page.

3. To establish the first input parameter for the **Value Mapping** functoid, create an input link by dragging the **Value Mapping** functoid to a record or field node in the source schema, or to another

functoid to its left, that has a Boolean data type and that will produce an input value of "true" or "false".

4. To establish the second input parameter for the **Value Mapping** functoid, create an input link by dragging the **Value Mapping** functoid to a record or field node in the source schema, or by dragging a record or field node in the source schema to the **Value Mapping** functoid.
5. To use the output parameter from the **Value Mapping** functoid, create an output link by dragging the **Value Mapping** functoid to a record or field in the destination schema, or by dragging a record field in the destination schema to the **Value Mapping** functoid.

How to Add Value Mapping (Flattening) Functoids to a Map

The **Value Mapping (Flattening)** functoid enables you to flatten a portion of an input instance message by converting multiple records into a single record. This is a common operation in converting Microsoft Commerce Server catalogs.

For conceptual information about the **Value Mapping (Flattening)** functoid, see [Value Mapping \(Flattening\) Functoid](#).


To add the Value Mapping (Flattening) functoid to a map and configure it

1. With the Visual Studio .NET Toolbox active, click the **Advanced Functoids** tab to select that category of functoids.

The list of advanced functoids in the chosen category appears.

2. Drag the **Value Mapping (Flattening)** functoid

from the Toolbox to the appropriate location on a grid page.

3. To establish the first input parameter for the **Value Mapping (Flattening)** functoid, create an input link by dragging the **Value Mapping (Flattening)** functoid to a record or field node in the source schema, or to another functoid to its left, that has a Boolean data type and that will produce an input value of "true" or "false."

4. To establish the second input parameter for the **Value Mapping (Flattening)** functoid, create an input link by dragging the **Value Mapping (Flattening)** functoid to a record or field node in the source schema, or by dragging a record or field node in the source schema to the **Value Mapping (Flattening)** functoid, or insert a constant.
5. To use the output parameter from the **Value Mapping (Flattening)** functoid, create an output link by dragging the **Value Mapping (Flattening)** functoid to a record or field in the destination schema, or by dragging a record field in the destination schema to the **Value Mapping (Flattening)** functoid.

Editing Functoid Properties and Input Parameters

Functoid properties can be categorized as follows:

- **Label and Input parameters.** These two properties are read/write and are available for all functoids. The **Label** property provides a mechanism for providing a descriptive name for a particular instance of a functoid, which may help in maintaining maps. The **Input parameters** property provides access to the **Configure Functoid Inputs** dialog box, which plays a central role in functoid configuration.
- **Script and Table Looping Grid.** These two properties provide access to dialog boxes that are only applicable to the **Scripting** and **Table Looping** functoids, respectively. These dialog boxes are the **Configure Functoid Script** dialog box and the **Table Looping Configuration** dialog box.
- **Name, Help, Maximum Input Parameters, and Minimum Input Parameters.** These four are informational and always read-only.

For conceptual information about these functoid properties, see [Functoid Properties](#) .

This section provides step-by-step instructions for working with, and specifically modifying, the properties of functoids, including general instructions for configuring input parameters for functoids, configuring script for the **Scripting** functoid, and configuring the table grid for the **Table Looping** functoid.

In This Section

- How to Label a Functoid
- How to Configure Functoid Input Parameters
- How to Configure the Scripting Functoid
- How to Configure the Table Looping and Table Extractor Functoids

How to Label a Functoid

You can use the **Label** property to provide a name for a particular instance of a functoid. This can be helpful during the long-term maintenance of a BizTalk map by documenting the purpose of the functoid.

To label a functoid

1. Select the functoid you want to label in the displayed grid page.
2. In the Visual Studio .NET Properties window, provide a new or replacement name for the selected functoid in the **Label** property.

How to Configure Functoid Input Parameters

Properly configuring the input parameters to the functoids in your map is one of the most important, and potentially error-prone, aspects of using functoids. There are two main ways in which functoid input parameters are configured, as follows:

By dragging schema nodes and functoids to each other to create the visible input links that connect to the left side of a functoid. There is a one-to-one correspondence between input links created in this fashion and an input parameter to the relevant functoid.

By directly editing the list of input parameters using the **Configure Functoid Inputs** dialog box. This dialog box is accessed through the **Input parameters** property of the relevant functoid, available in the Visual Studio .NET Properties window when the functoid is selected in the displayed grid page.

Although the dragging method of establishing functoid input parameters provides a convenient way to specify input parameters that involve XPath specifications into the source schema, the **Configure Functoid Inputs** dialog box is the definitive mechanism for viewing all of the input parameters to a functoid, for creating and modifying any constant parameters, and for re-arranging the order of the input parameters when necessary.

Many functoids require particular parameters in a particular order, and most functoids have limits on the number of parameters they accept. Information about how many parameters a particular functoid expects can be found in several places, as follows:

A summary is provided the **Input parameters** list and the **Functoid Description** in the **Configure Functoid Inputs** dialog box.

The **Maximum Input Parameters** and the **Minimum Input Parameters** properties of the functoid, available in the Visual Studio .NET Properties window when the relevant functoid is selected in the displayed grid page, provide a numeric indication of the number of input parameters expected by that functoid.

The most complete description of each functoid, in page reference format, appears in the **Functoid Reference**, provided in this documentation.

This topic provides step-by-step instructions for configuring the input parameters for a functoid using both of these methods.

Creating schema node and functoid input parameters

The generic step-by-step instructions for creating schema node and functoid input parameters by dragging is provided in the procedure called "To create input links to a basic functoid" in Adding Basic Functoids to a Map.

To delete an input parameter by deleting the input link

1. In the relevant grid page, select the input link corresponding to the input parameter to be deleted by clicking anywhere along the link.
2. On the **Edit** menu, click **Delete**.
3. Click **Yes** in the **confirmation** dialog box.

To open the Configure Functoid Inputs dialog box

1. In the relevant grid page, click on the relevant functoid to select it.
2. In the Visual Studio .NET Properties window, click the ellipsis (...) button associated with the **Input parameters** property.

The **Configure Functoid Inputs** dialog box opens.

To insert constant input parameters

1. In the **Configure Inputs** dialog box, click the **Insert New Parameter** button



, type the value for the new parameter in the edit box within the **Input parameters** list, and then press **ENTER**.

To edit existing constant input parameters

1. In the **Configure Functoid Inputs** dialog box, in the **Input parameters** list, click the existing constant input parameter that you want to edit.

The existing constant input parameter is selected.

2. Click the **Edit Selected Parameter** button

The **Functoid Constant Value** dialog box opens, displaying the current value of the constant input parameter.

3. In the **Functoid Constant Value** dialog box, in the edit box, make the appropriate changes to the constant value, and then click **OK**.

The **Functoid Constant Value** dialog box closes, and the new value is now displayed in the relevant location in the **Configure Functoid Inputs** dialog box.

To delete existing input parameters within the **Configure Functoid Inputs** dialog box

1. In the **Configure Functoid Inputs** dialog box, in the **Input parameters** list, click the existing input parameter that you want to delete.

The existing input parameter is selected.

2. Click the **Delete Selected Parameter** button The selected existing input parameter is deleted from the **Input parameters** list, and an adjacent input parameter is selected (normally the following input parameter, unless the last input parameter in the list is deleted).

To change the order of existing input parameters

1. In the **Configure Functoid Inputs** dialog box, in the **Input parameters** list, click the existing input parameter that you want to move to a different position in the ordered list of input parameters.


The existing input parameter is selected.

2. Click the **Move Up Selected Parameter** button



to move the parameter up in the **Input parameters** list. Repeat as necessary until the selected input parameter is in the proper position in the **Input parameters** list.

- Or -

Click the **Move Down Selected Parameter** button  to move the parameter down in the **Input parameters** list. Repeat as necessary until the selected input parameter is in the proper position in the **Input parameters** list.

How to Configure the Scripting Functoid

Step-by-step instructions for configuring the **Script** property of a **Scripting** functoid are provided as steps 3 through 6 of Adding Scripting Functoids to a Map. Steps 7 and 8 are also relevant with respect to configuring the input parameters, if any, and the output parameter of a **Scripting** functoid.

How to Configure the Table Looping and Table Extractor Functoids

Step-by-step instructions for configuring the **Table Looping Grid** property of a **Table Looping** functoid are provided as steps 9 and 10 of Adding Table Looping and Table Extractor Functoids to a Map. Steps 3 through 8 are also relevant with respect to configuring the table grid because they must be performed first to populate the drop-down lists available in each cell of the table grid.

How to Select Multiple Functoids

When dealing with large maps, you may want to perform operations on multiple functoids at the same time. In such cases, you can select all of the functoids of interest and then perform the operation. This is useful, for example, when you want to move a set of connected functoids, and their connecting links, from one grid page to another.

To select multiple functoids at the same time

- In BizTalk Mapper, in a grid page, click the first functoid to select it.

The first selected functoid in the grid page is highlighted with eight small blue boxes around it.

- In BizTalk Mapper, in the same grid page, hold down the **CTRL** or the **SHIFT** key, and then click another functoid to select it as well.

This selected functoid in the grid page is also highlighted with eight small blue boxes around it.

- Repeat step 2 as required to select all of the functoids of interest.

How to Replace Functoids

On rare occasions, you may want to replace an existing functoid with another functoid. This will be rare because different functoids typically take distinct sets of input parameters. Therefore, replacing a functoid will often not require less effort than deleting the existing functoid and adding the new functoid.

To replace an existing functoid with a different functoid

1. Display the grid page that contains the functoid you want to replace. For more information about working with grid pages, including switching between grid pages, see [Working with Grid Pages](#).
2. With the Visual Studio .NET Toolbox active, click the appropriate tab to select the category of the replacement functoid.

The list of available functoids in the chosen category appears.

3. Drag the replacement functoid from the Toolbox and place it on an existing functoid in the relevant grid page.

How to Delete Functoids

When developing maps, you will sometimes need to delete an existing functoid. Deleting a functoid also deletes the associated input and output links, as well as any additional input parameters, script specification, and table grid configuration.

To delete an existing functoid

1. Display the grid page that contains the functoid you want to delete. For more information about working with grid pages, including switching between grid pages, see [Working with Grid Pages](#).
2. Select the functoid you want to delete by clicking on it.
3. On the **Edit** menu, click **Delete**.
4. Click **Yes** in the confirmation dialog box.

How to Create a Map without Maps

If you have XSLT code you have been using to convert instance messages, you can use that code directly instead of creating a map.

Using your XSLT code involves creating an empty map and setting its **Custom XSLT Path** grid property. If your XSLT code uses external .NET assemblies, you will also need to create a custom extension XML file. For information about the structure of a custom extension XML file, see **Custom Extension XML (Grid Property)**.

To use custom XSLT code in place of a map

- Create an empty BizTalk map in your project and set the source and destination schemas.

For information about creating the map and setting the schemas, see [Creating Maps](#).

- In the Grid view, click on the mapper grid.

The Properties window shows the **Grid** properties.

- In the Properties window, select **Custom XSLT Path** and click on the ellipsis () button.

The **Select Custom XSLT File** dialog box opens.

- Navigate to the file and click the **Open** button.

The dialog box closes and the **Custom XSLT Path** property is set.

Compiling and Testing Maps

After you have developed your map, the next step is to validate and test it. This section provides step-by-step instructions for validating and testing maps.

In This Section

- How to Resolve Map Warnings and Errors
- How to Add Constant Values
- How to Configure Map Validation and Test Parameters
- How to Validate Maps
- How to Test Maps
- How to Create Multi-Part Mappings
- How to Specify XSLT Output Settings

How to Resolve Map Warnings and Errors

When you compile a map, you may find that warnings and errors result from the compilation process.

To resolve warnings and errors found when building a map

- For link and functoid errors, in the **Task List** window, double-click any description.

The link or functoid associated with the error is highlighted. Resolve the issue.

- Review the **Description** area in the **Task List** window to determine additional errors.
- Click the **Output** window tab to view additional warning and error message information.
- After you have resolved all issues, right-click the map file name in Solution Explorer, and then click **Test Map**.

How to Add Constant Values

When testing maps, you will sometimes want to set constant values for use during the test.

To set constant values for nodes in the source or destination schema trees

1. Select a record or field in the source or destination schema trees.
2. In the Properties window, in the **General** category, use the **Value** property to enter a value for the selected record or field.