

ChainBuilder ESB

Visual Enterprise Integration™

Version 1.0

Message Format Editor Guide



©Copyright 2007
Bostech Corporation
2800 Corporate Exchange Drive
Suite 260
Columbus, OH 43231

Acknowledgements

This document contains proprietary information that is the property of Bostech Corporation. Any reproduction, disclosure, or transfer of this document or the information contained herein without the express written consent of Bostech Corporation is strictly prohibited.

The use of the information contained in this document and the implementation of any of its techniques are the sole responsibility of the client and depend on the client's ability to evaluate the information and implement it into the client's operational environment.

Except for any express written warranties made by it, Bostech Corporation makes no warranties or representations with respect to any information contained herein, whether express, implied, statutory, or otherwise, in fact or in law, including without limitation, any implied warranties of merchantability or fitness for a particular purpose; and in no event shall Bostech Corporation be liable for any special, consequential, indirect, punitive, or exemplary damages in connection with the use of the information contained herein. The information contained in this document is subject to change at any time without notice.

Trademarks

The following trademarks and acknowledgments apply to the information presented in this manual:

- ChainBuilder is a registered trademark of Bostech Corporation.
- Adobe and Acrobat Reader are registered trademarks of Adobe, Inc.
- Java is a registered trademark of Sun Microsystems, Inc.
- Windows (NT, 2000, XP, and Server 2003), .NET Framework, Internet Information Services (IIS) are registered trademarks of Microsoft Corporation.

Credits

The following third-party products are used within the ChainBuilder product, and acknowledgments apply to the information presented in this manual:

- Acrobat Reader is created and licensed by Adobe, Inc.
- This product includes software developed by the Apache Software Foundation (<http://www.apache.org/>)
- This product includes software developed by Eclipse (<http://www.eclipse.org/>)

Table of Contents

| | |
|---|----|
| 1. Introduction..... | 1 |
| 1.1. Overview | 1 |
| 1.2. Introduction to Message Formats | 1 |
| 1.2.1. Fixed Format..... | 1 |
| 1.2.2. Variable Format - Non-Tagged (CSV) | 1 |
| 1.2.3. Variable Format - Tagged | 2 |
| 1.3. Introduction to the Message Format Editor..... | 2 |
| 1.3.1. Accessing the Message Format Editor..... | 2 |
| 1.3.2. Root Node..... | 7 |
| 1.3.3. Message Node..... | 8 |
| 1.3.4. Element Node..... | 11 |
| 1.3.5. Group Node..... | 12 |
| 2. Creating Fixed Format Messages..... | 13 |
| 2.1. Creating a new Message | 13 |
| 2.2. Adding Child Elements | 13 |
| 3. Creating Non-Tagged Variable Format Messages | 16 |
| 3.1. Creating a new Message | 16 |
| 3.2. Adding Child Elements | 17 |
| 4. Creating Tagged Variable Format Messages..... | 18 |
| 4.1. Creating a new Message | 19 |
| 4.2. Adding Child Elements | 20 |
| 4.3. Adding Groups..... | 20 |
| 5. Advanced Features | 22 |
| 5.1. Include and Import..... | 22 |
| 5.1.1. Using Include | 22 |
| 5.1.2. Using Import..... | 22 |
| 5.2. Global Element Definitions and References | 22 |
| 5.2.1. Defining a Global Element..... | 23 |
| 5.2.2. Referencing a Global Element | 23 |
| 6. Format Tester..... | 24 |
| 7. ChainBuilder ESB Community..... | 27 |

1. Introduction

1.1. Overview

ChainBuilder ESB is built upon the JBI standard and because of this XML messages can be easily configured, mapped, and processed. However, there are vast number of legacy systems that still exist and have their own unique messaging requirements. In order to meet these requirements to handle non-XML formatted messages, the Message Format Editor was created. This document will cover how to create MDL (Message Definition Language) messages within the ChainBuilder ESB Message Format Editor.

1.2. Introduction to Message Formats

The ChainBuilder ESB Message Format Editor can be configured to handle three different types of message formats: a fixed message format, a variable message format, and a tagged variable message format.

1.2.1. Fixed Format

The fixed format message is a message structure where the length of the message and its elements is of a known value. For example, a name message may be of a fixed length, left justified, and space filled, lets say that the message has the following definition:

| <i>Name</i> | <i>Length</i> | <i>Example</i> |
|-----------------------|---------------|----------------|
| Message Type | 4 | NAME |
| First | 25 | John |
| Middle Initial | 1 | Q |
| Last | 25 | Public |

Then the example data, within braces, would look like:

```
[NAME]John          QPublic          ]
```

1.2.2. Variable Format - Non-Tagged (CSV)

The variable format message is a message structure where the length of the message is unknown and each element within the message is separated by a message delimiter. Often the delimiter is a comma and the messages are referred to a CSV (Comma Separated Values) file. However, within ChainBuilder ESB this delimiter is configurable. Again, for example, let us use the name message type as it is configured below:

| <i>Name</i> | <i>Delimiter</i> | <i>Example</i> |
|-----------------------|------------------|----------------|
| Message Type | , | NAME |
| First | , | John |
| Middle Initial | , | Q |

| | |
|-------------|--------|
| Last | Public |
|-------------|--------|

Then the example data, within braces, would look like:
[NAME,John,Q,Public]

1.2.3. Variable Format - Tagged

The tagged message format deals with messages that contain tags and delimiters to denote different fields within a message. Again, we will use the name message to show an example of a tagged message format:

| <i>Name</i> | <i>TAG</i> | <i>Delimiter</i> | <i>Example</i> |
|-----------------------|------------|------------------|----------------|
| Message Type | 1: | , | 1:NAME |
| First | 2: | , | 2:John |
| Middle Initial | 3: | , | 3:Q |
| Last | 4: | , | 4:Public |

Then the example data, within braces, would look like this:
[1:NAME,2:John,3:Q,4:Public]

1.3. Introduction to the Message Format Editor

1.3.1. Accessing the Message Format Editor

The Message Format Editor can be accessed from the within the Package Explorer within the ChainBuilder ESB application. The Message Format Editor opens and edits Message Definition Language files. These files are located within the src/formats directory within a ChainBuilder ESB Project or a Service Assembly Project, see Figure 1.

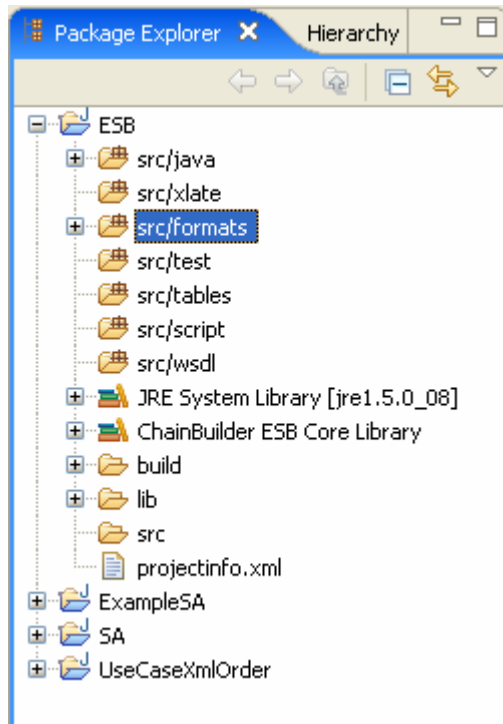


Figure 1

A Message Definition Language, herein referred to as MDL, can be created here by right-clicking on the formats file directory, selecting new, Message Format File, Figure 2.

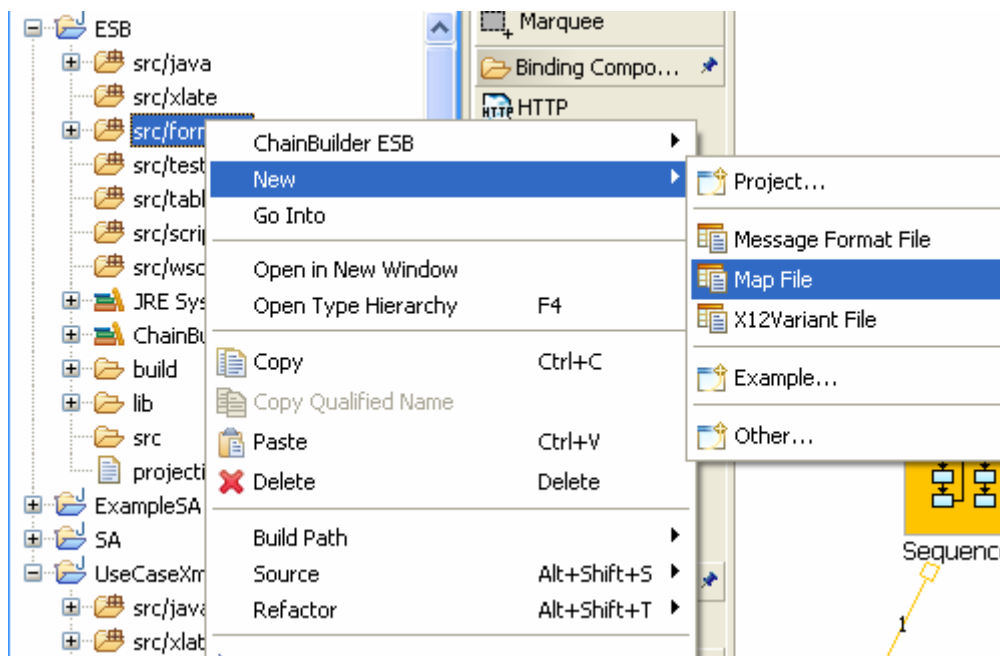


Figure 2

Upon selecting to create a new Message Format File, a window will appear asking you to name the file, Figure 3.

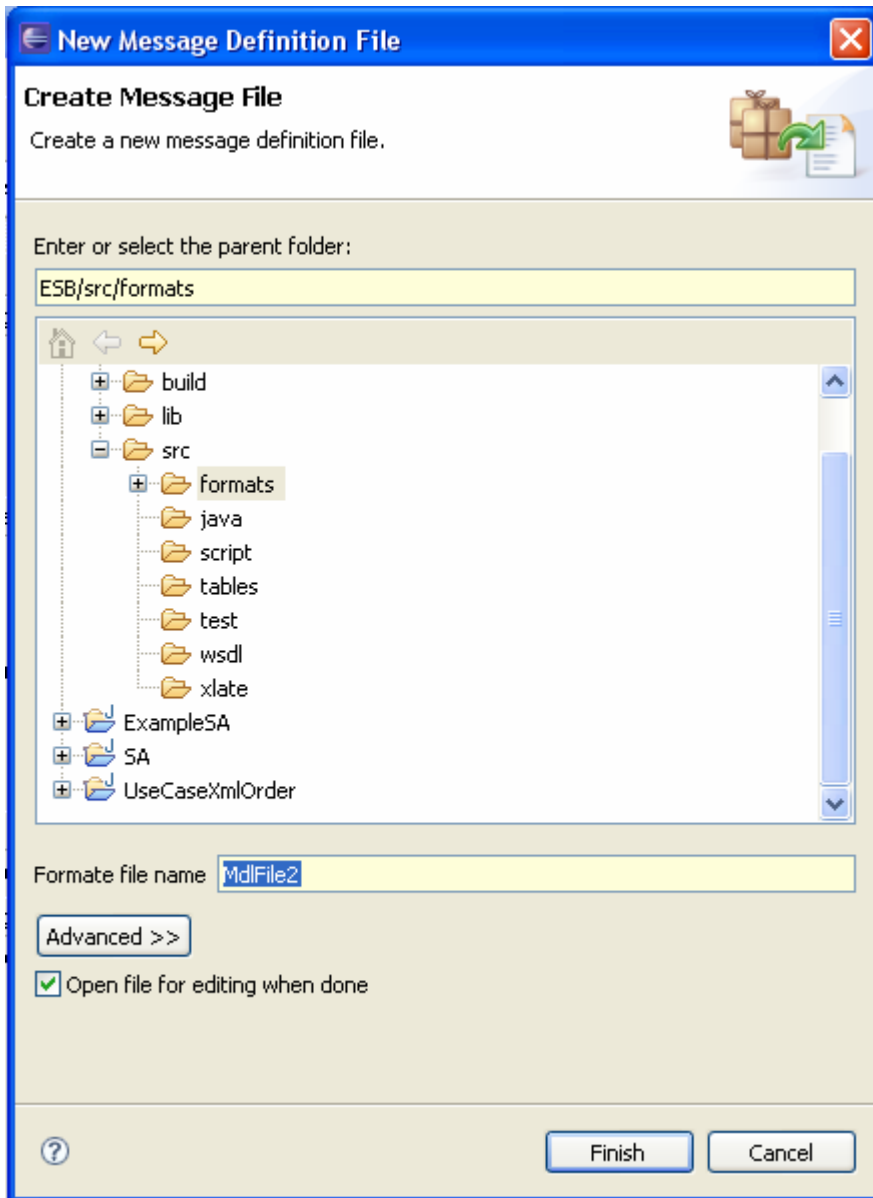


Figure 3

Name the MDL file, which will contain the message definitions that will be created. Often the name of this file will reflect the name of the system that ChainBuilder ESB is connecting to i.e., lab.mdl, oracle.mdl, terminal.mdl, etc. Keep in mind that the .mdl extension will be appended into the MDL file name when the file is created.

Once the new MDL file has been created, the ChainBuilder ESB IDE is changed to the Message Format perspective. There are three areas: the Menu Bar, the Definitions View, and the Properties View.

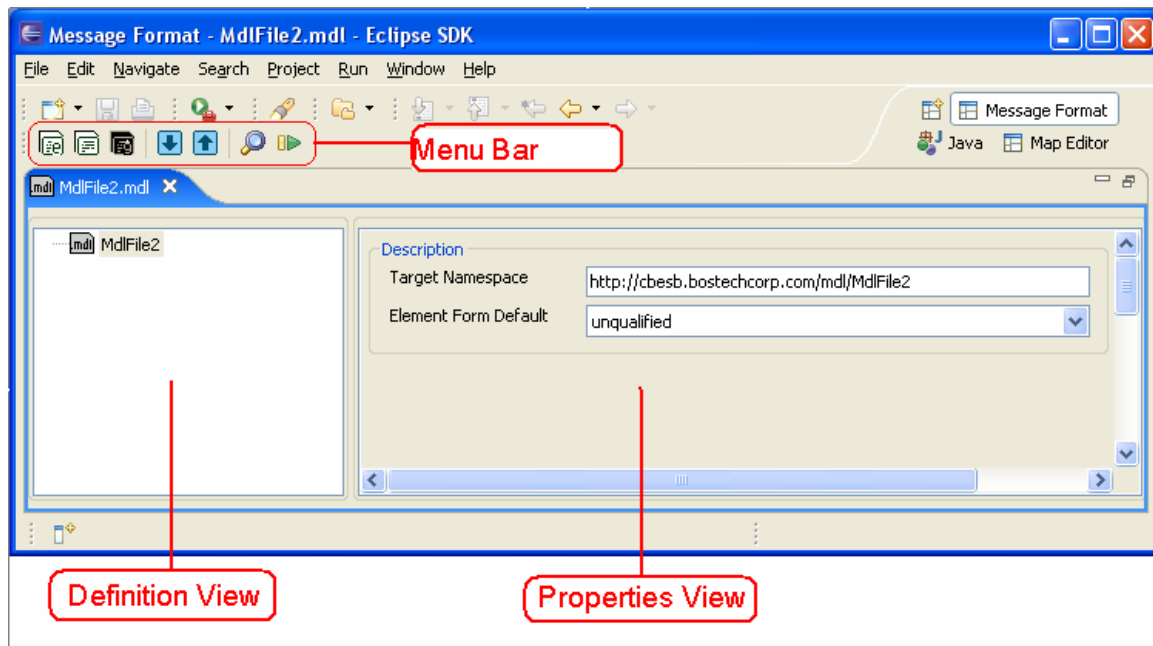


Figure 4

Definitions View

The definitions view is a tree structure that is the main presentation for message and element definitions. It gives the user the ability to generate messages and elements using a tree structure. Each node within the tree can be Message or an Element, with the exception of the Root Node. There is only one Root Node within the MDL file tree. Figure 5 gives an outline of the nodes within the tree.

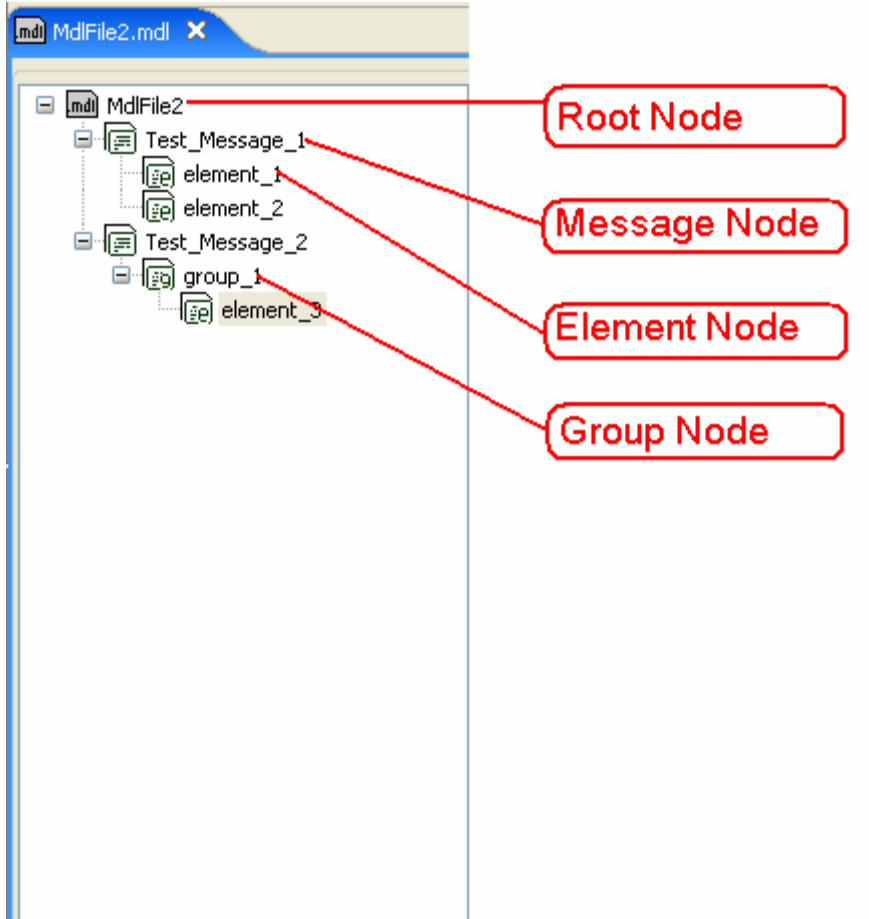



Figure 5


Properties View


The properties view displays the detailed setting for each node within the definition tree. The contents of the properties view will change according to the node type that is selected and the properties that have been set for a nodes parent.

Menu Bar

The Menu Bar gives you quick access to actions that are of use when creating and editing MDL files. The actions that are available are as follows:

 Insert Element – Used to insert an element at the selected node within Definition View section of the screen.

 Insert Message – Used to insert a message at the selected node within the Definition View section of the screen.

 Insert Group – Used to insert a group at the selected node within the Definition View section of the screen.

The Insert Group, Message, and Element buttons will only be enabled within nodes in the message definition tree where they are applicable, otherwise these buttons will be disabled.



Search – Can search for Message, Element, and Group names within an MDL file.





Move Node – Can be used to move a node within the Definition View tree. The up and down arrows can move a node up and down the tree structure.

1.3.2. Root Node

The Root Node represents the entire MDL document and there is only one root node available for a document.

Root Node Actions

There are two possible actions that can be applied to a Root Node, New Message and New Element, these can be created by using the toolbar Insert Message icon, , or by using the Insert Element icon, . The actions can also be applied by using the context-sensitive pull down menu that appears when you right click on the root node, see Figure 6.

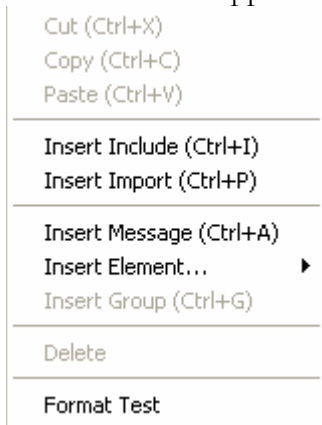


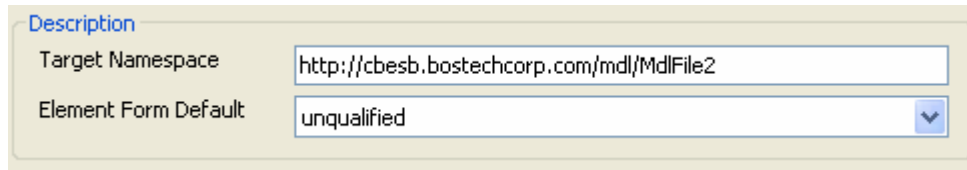
Figure 6

- Insert Message: Will add a new Message Node to the tree. This represents a message tag in the MDL document. There may be many Messages within an MDL document.
- Insert Element: On the root level, creating an Element actually creates a Global Element Definition, which is described in a later section. In a nutshell the Global Element Definition allows a user to create an element that can be used in many different messages. This saves the user the hassle of having to create a unique element for each message that may contain basically the same element structure.

Root Node Properties

The root node has two properties that can be set, the Target Namespace and the Element Form Default. See Figure 6.

- Target Namespace, optional: The namespace is used when a parsed message is represented as XML. This is equivalent to the target namespace of an XML Schema definition.
- Element Form Default, required if Target Namespace is set: The element defines how the Target Namespace is applied to the MDL file. There are two values that can be applied here, *qualified* and *unqualified*. If the Element Form default has been set to *qualified* then the MDL is processed like an XML message and is equivalent to the XML Schema elementFormDefault attribute. It determines if non-global elements in the message are namespace qualified or not.



The screenshot shows a 'Description' tab in a software interface. It contains two fields: 'Target Namespace' with a text input field containing the URL 'http://cbesb.bostechcorp.com/mdl/MdlFile2', and 'Element Form Default' with a dropdown menu currently showing 'unqualified'.

Figure 7

1.3.3. Message Node

The message node does what its name implies, it represents a message tag within the MDL definition. Message nodes will only appear as a direct child of the root node.

Message Node Actions

There are a number of actions that can be applied to a message:

- Delete: Removes a message from the MDL document.
- Copy: Makes a copy of the message node. This can be helpful when creating many messages with few differences between them.
- Insert Element: Allows the user to add an element to a message. The following Element Insert actions can be performed, see Figure 8:
 - o As Child: Creates a new element within the message tag.
 - o As Reference: Adds a reference to a global element definition. The Global Element definition is described in section 1,3.2
 - o As Sibling: Inserting an element as a sibling from a Message Node will create a new element.

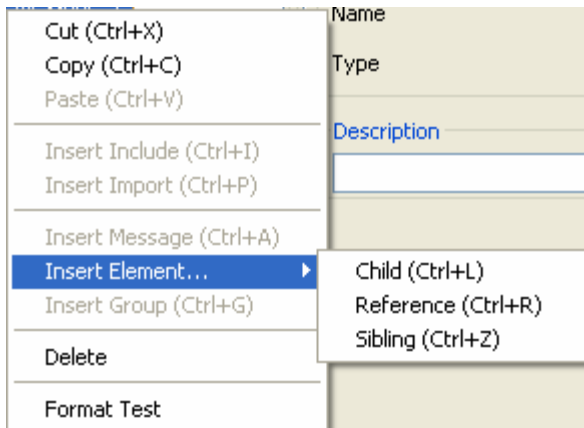



Figure 8

In addition a Message Node may be moved up or down the same level of the MDL tree by using the up or down arrows within the tool bar, .

Message Node Properties

The default properties for a Message Node, Figure 9, are described as follows:

- Name: The name of the message within the MDL document. All message, element and group names must be valid XML element names.
- Type: The type of message format that the current message definition represents. There are three types of formats; fixed, variable, or leaf. Also, depending on the value of Type, additional properties may be displayed. The default type for a message is set to fixed.
- Description: Allows the user to add a brief sentence or two to describe the purpose of the message.

Figure 9

For the Variable type additional properties can be set for a message. These properties are defined as follows:

- Delimiter: Defines the delimiter that will be used within the message to separate elements within the message.
- Escape Char: Defines the escape character within a message if a delimiter value is encountered within a message that is actual data for the message.
- Quote Char: Defines the quote character. The Quote Char is used as a pair in the data to escape a sequence of characters. For example, if the Quote Char is " and the

delimiter is ~, and the data was abc"~~~~", then the multiple ~ chars would be escaped.

- ID Method: There are two possible ID Methods that can be set for a variable message, position and tag.
 - o position: When position ID Method is selected, the child elements in the message are identified by the order they occur in the message. If the delimiter property has been set to a semi-colon then every character up to that semi-colon is considered part of that message. The semi-colon is not used.
 - Repeat Delimiter: Unique to the position ID Method, this setting defines a repeating element within the message.

The screenshot shows the 'Properties' dialog for the 'Position' ID Method. The 'ID Method' dropdown is set to 'position'. The 'Delimiter' field contains a comma (,). The 'Quote Char' field is empty. The 'Escape Char' field is empty. The 'Repeat Delimiter' field is empty.

Figure 10 Position ID Method Properties

- o tag: The tag method is used to define a message which may contain tags. When the tag ID Method is selected, the child elements in the message are identified by the value of a tag at the beginning of the data in the element. The tag can be defined as fixed or variable length.
 - Tag Length: Sets the tag length when the ID Method is set to tag and the tag is of a fixed length.

The screenshot shows the 'Properties' dialog for the 'Tag' ID Method. The 'ID Method' dropdown is set to 'tag'. The 'Fixed Length' radio button is selected. The 'Delimiter' field contains a comma (,). The 'Quote Char' field is empty. The 'Tag Length' field contains the value '2'.

Figure 11 Tag ID Method set to Fixed Length Properties

- Tag Delimiter: Sets the tag delimiter when the ID Method is set to tag and the tag is of a variable length. For example, in the case of X12 standard, segment names such as "ST" is of variable length.

The screenshot shows the 'Properties' dialog for the 'Tag' ID Method. The 'ID Method' dropdown is set to 'tag'. The 'Variable Length' radio button is selected. The 'Delimiter' field contains a comma (,). The 'Quote Char' field is empty. The 'Tag Delimiter' field contains a vertical bar (|).

Figure 12 Tag ID Method set to Variable Length Properties

1.3.4. Element Node

There are two different types of element nodes, a Global Element node or as a Local Element that is singularly defined for a specific message or element. The Global Element node can be referenced from multiple points within a message or an element. A Local Element node is defined as a child within a message node or within another element node.

Element Node Actions

The following is a list of actions that can be performed on an element node:

- Delete: Removes the element definition and all its sub-elements.
- Insert Element
 - o As Child: Inserts a sub-element within the currently selected element.
 - o As Reference: Allows the user to set the current element to a Global Element.
 - o As Sibling: Inserts an element on the same node as the current element.
- Insert Group: Inserts a group of sub-element to this element. Only available if the current element is of type variable and has an ID Method of tag.

Element Node Properties

An element has three basic properties: Name, Type, and Description. All other properties that an element may have are derived from the elements parent node. There are different properties when the parent node's type is set to Fixed, Variable, or Leaf.

- Default Properties:
 - o Name: The unique name for the element.
 - o Type: The type of element that is being created; Fixed, Variable, or Leaf.
 - o Leaf: The end node of a tree.
- Element Properties when the parent node is of type Fixed:
 - o Element Properties when the element is of type Fixed:
 - Length: The length of the element.
 - Fill Char: The fill characters that will be used if data does not occupy the entire length of the element.
 - Justification: Left or right justified.
 - o Element Properties when the element is of type Variable:
 - Contains the Length, Fill Char, and Justification properties.
 - Also contains the ID Method attributes that are the same as a message.
 - o Element Properties when the element is of type Leaf:
 - Contains the Length, Fill Char, and Justification properties.
 - Data Type: Defines the type of data that this element will contain. Currently only the string data type is supported.

1.3.5. Group Node

A group node may only appear as a child of a message or element that whose type is variable and whose ID Method is set to tag. What a group node allows the user to do is to define a number of repeatable elements within a tagged formatted message. For example, let's say that the current message that you are defining has a requirement for 1 to 10 family member names. The first, middle, and last names can be defined as sub-elements within a group of names.

Group Node Actions

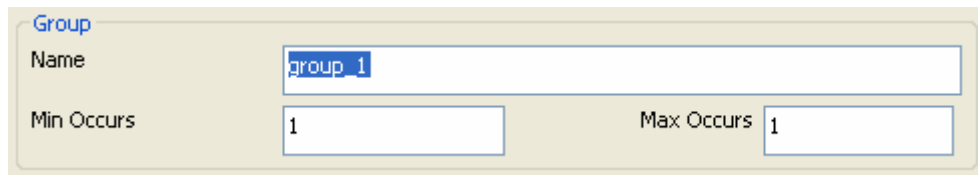
The following is a list of actions that can be defined within a group:

- Delete: Used to remove a group and all its sub-elements.
- Insert Element:
 - o As Child: Inserts a sub-element within the current group.
 - o As Reference: Inserts a sub-element reference to a Global Element.
 - o As Sibling: Inserts an element on the same level as the current group element.
- Insert Group: Inserts a group sub-element.

Group Node Properties

A group node only has three properties:

- Name: The name of the group node
- Min Occurs: The minimum number of times a group may occur.
- Max Occurs: The maximum number of times a group may occur.



| | | |
|------------|---------|--------------|
| Group | | |
| Name | group_1 | |
| Min Occurs | 1 | Max Occurs 1 |

Figure 12 Group Properties

2. Creating Fixed Format Messages


The goal of this section is to give the user the basic tools and understanding on how to create a fixed format message within the ChainBuilder ESB Message Format Editor.

For example, let's say that we have a fixed formatted message for a persons address that contains the following definition:

| <i>Name</i> | <i>Length</i> | <i>Example</i> |
|-----------------------|---------------|--------------------|
| Message Type | 3 | ADD |
| First Name | 25 | John |
| Middle Initial | 1 | Q |
| Last Name | 25 | Public |
| Street | 50 | 123 Any Street Dr. |
| City | 25 | Cleveland |
| State | 2 | OH |
| Zip Code | 5 | 43000 |

2.1. Creating a new Message

If you do not have a MDL file created already, follow the steps outlined in section 1.3.1 to create a new MDL file. For our purposes here we will create an MDL file with the name "SystemERP".

Once you have a root node available there are two ways to create a new message, right click on the root node then select Insert Message, or you may use the Insert Message icon, , to create a new message.

For our example here the name will be "FixedMessage" and the Type will be set to fixed. The Description attribute is an optional value.

2.2. Adding Child Elements

When looking at our example message structure we can clearly see that there are 8 fields that can be created. However, to show the flexibility of the Message Format Editor we will only have three sub-elements directly beneath the message element.


- 1) Right click on the message, select Insert Element → As Child.
- 2) Set the name for the element to "Message Type"
- 3) Set the Type of the element to leaf.
- 4) Set the length to 3

At this point you have defined a message within one element, Message Type. One element down, seven more to go.

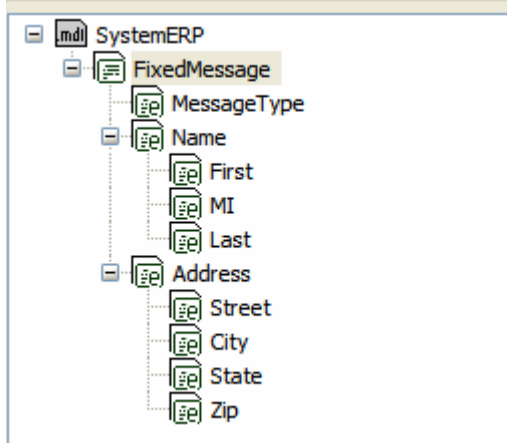
What we would like to show now is how to create an element with sub-elements. If you notice there are really two distinct parts to the sample message; one is for a name and the other is for an address. So let's go ahead and create two elements as siblings to the "Message Type" element, one called "Name" and the other called "Address"

- 1) Right click on the element "Message Type" and try and insert an element, you will notice that this cannot be done. A leaf type element cannot have elements inserted. Instead right click on the message "FixedMessage", select Insert Element → Child.
- 2) Set the name for the element to be "Address"
- 3) Set the type to be fixed.
- 4) Set the length to be the total length of the elements that will make up the address, in this case 82.
- 5) Repeat above steps for "Name." The name field has a length of 51.


Now we have a message with three sub elements defined. Let's continue now and create sub-elements for both the Address and Name elements. Basically, for each sub-element do the following steps:

- 1) Right click on the selected element, select Insert Element → Child or you can select the Insert Element icon  and it will insert a child element for you.
- 2) Set the name of the element to be what has been defined within the specification, First Name, City, State, etc.
- 3) Set the Type to be leaf.
- 4) Set the length of the field. 25 for First Name, 2 for State, etc..
- 5) Repeat above steps for the remaining fields.

When you look at your message format it should look something like this:



You will notice that the Name element is below the Address element and our definition states that the name values will come before the address values. An element can be moved within a message and to other messages simply by using the arrow keys within the tool bar. Try moving the elements around a bit to see where an element can and cannot go. Once the

message has been completed you can save the file by using normal windows protocol, Alt-F-S, File → Save, or by using the  icon.

3. Creating Non-Tagged Variable Format Messages

The goal of this section is to give the user the basic tools and understanding on how to create a variable format message within the ChainBuilder ESB Message Format Editor.


For example, let's say that we have a variable formatted message for a person's address that contains the following definition:

| <i>Name</i> | <i>Sub-Group Name</i> | <i>Delimiter</i> | <i>Example</i> |
|---------------------|-----------------------|------------------|--------------------|
| Message Type | | , | ADD |
| Name | | , | |
| | First | ~ | John |
| | Middle | ~ | Q |
| | Last | ~ | Public |
| Address | | , | |
| | Street | ~ | 123 Any Street Dr. |
| | City | ~ | Cleveland |
| | State | ~ | OH |
| | Zip | ~ | 43000 |

In addition the following are attributes of the address message: The escape character is a \.

3.1. Creating a new Message

If you do not have a MDL file created already, follow the steps outlined in section 1.3.1 to create a new MDL file. For our purposes here we will create an MDL file with the name "SystemCRM".

Once you have a root node available there are two ways to create a new message, right click on the root node then select Insert Message, or you may use the Insert Message icon, , to create a new message.

For our example here the name of the message will be "VariableMessage" and the Type will be set to variable. The Description attribute is an optional value.

You will notice that when the type of message has been set to variable that additional attributes for the variable message have appeared within the properties view. Being that our message is a variable message within delimiters and no tags are present the value of the attributes under properties should be the following:

- ID Method: Set to position, there are no tags in the message and the message is defined by delimiters.
- Delimiter: The default value for a variable message is a comma, which is the delimiter required by elements within the address message.

- Escape Character: The escape character for the variable message is a \ and it should be entered here.
- Quote Char: If quotes are allowed within the message we are receiving then the value of the quote would go here. Being that one was not defined within the message specifications we will assume one is not needed, leave blank.
- Repeat Delimiter: If a section within the message is repeatable and it is defined by a delimiter that value would go here.

The screenshot shows the 'Message Definition' section with the following fields:

- Name:** VariableMessage
- Type:** variable

The 'Properties' section contains:

- ID Method:** position
- Delimiter:** ,
- Escape Char:** \
- Quote Char:** (empty)
- Repeat Delimiter:** (empty)

The 'Description' section contains:

- A Variable Message Example

Figure 13 Variable Message Properties

3.2. Adding Child Elements

Our message structure for the Address message is unique. The message contains two elements, Name and Address. In addition the elements Name and Address have their own elements, or sub-elements that need to be defined within the Message Format Editor.

The first thing we need to do is create an element for the message called “Message Type”, follow these steps:

- 1) Right click on the “VariableMessage” message element, select Insert Element → As Child.
- 2) Set the name for the element to “MessageType”
- 3) Set the Type of the element to leaf.
- 4) Set the length to 3.

For our purposes here, we will go ahead walk through the example of creating the Name element and its attributes which will give you all the information needed in creating the Address element on your own.

- 1) Right click on the “Variable Message” element, select Insert Element → As Child.
- 2) Set the name for the element to “Name.”

- 3) Set the Type to variable. This will allow for child nodes to be placed within the element “Name.”
- 4) The delimiter for the element Name is different than that from the Message node, it is a ~. Because of this the Delimiter value needs to be set as a ~.
- 5) Select Apply to set the new element.
- 6) Right click on the “Name” element, select Insert Element → As Child.
- 7) Set the Name of the element to be First and set the type to leaf, select Apply.
- 8) Repeat steps 6 and 7 for MI and Last names.
- 9) Once the name element has been completed you can go ahead and use the same steps to produce the Address element as well.

Once the variable message has been completed it should look something like figure 14.

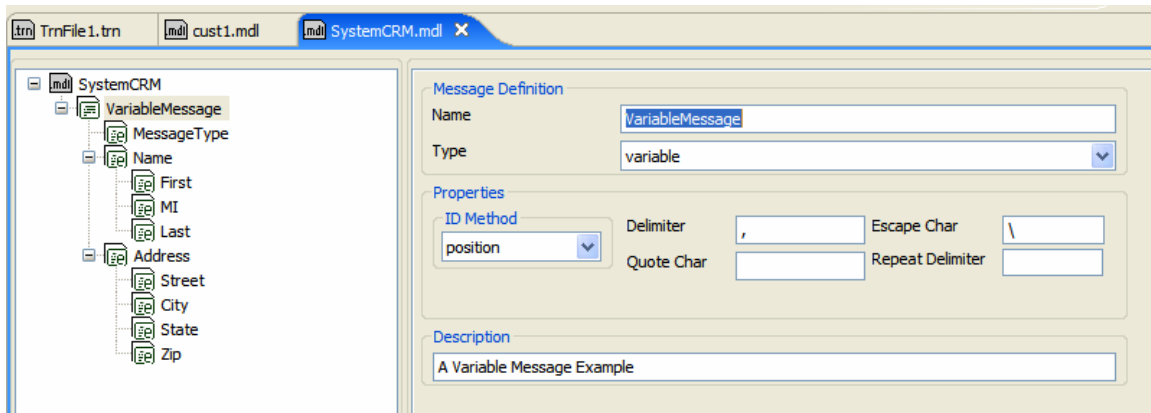


Figure 14

4. Creating Tagged Variable Format Messages


The goal of this section is to give the user the basic tools and understanding on how to create a tagged variable format message within the ChainBuilder ESB Message Format Editor.

For example, let’s say that we have a tagged variable formatted message for a person’s address that contains the following definition:

| <i>Name</i> | <i>Sub-Group Name</i> | <i>Tag</i> | <i>Example</i> |
|----------------------|-----------------------|------------|---------------------|
| Message Type | | M1 | ADD |
| Name Group | Name | N1 | 1 to 10 Name values |
| | First | | John |
| | Middle | | Q |
| | Last | | Public |
| Address Group | Address | A1 | |
| | Street | | 123 Any Street Dr. |
| | City | | Cleveland |
| | State | | OH |
| | Zip | | 43000 |

4.1. Creating a new Message

If you do not have a MDL file created already, follow the steps outlined in section 1.3.1 to create a new MDL file. For our purposes here we will create an MDL file with the name “SystemHR”.

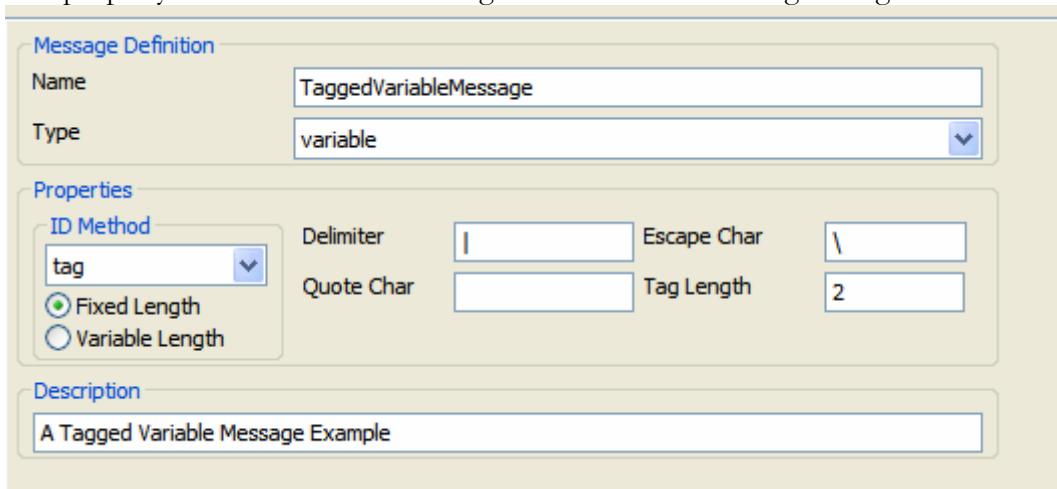
Once you have a root node available there are two ways to create a new message, right click on the root node then select Insert Message, or you may use the Insert Message icon, , to create a new message.

For our example here the name of the message will be “TaggedMessage” and the Type will be set to variable. The Description attribute is an optional value.

You will notice that when the type of message has been set to variable that additional attributes for the variable message have appeared within the properties view. Being that our message is a tagged variable message the value of the attributes under properties should be the following:

- ID Method: Set to tag and leave as a Fixed Length. The reason for leaving as a fixed length is that the tag value of Message Type, Name, and Address are all of a fixed length of 2.
- Delimiter: Not needed.
- Escape Character: If an escape character is needed it would be placed here.
- Quote Char: If quotes are allowed within the message we are receiving then the value of the quote would go here. Being that one was not defined within the message specifications we will assume one is not needed, leave blank..
- Tag Length: Set to 2, the length of the Tags being used.

The property screen of the new message should look something like figure 15.



The screenshot shows the 'Message Definition' dialog box. It has three main sections:

- Message Definition:**
 - Name: TaggedVariableMessage
 - Type: variable (dropdown menu)
- Properties:**
 - ID Method: tag (dropdown menu)
 - Fixed Length: (selected)
 - Variable Length:
 - Delimiter: |
 - Escape Char: \
 - Quote Char: (empty)
 - Tag Length: 2
- Description:**
 - A Tagged Variable Message Example

Figure 15

4.2. Adding Child Elements

Our message structure for the Address message is unique. The message contains many elements and one group. We will go ahead and define how to add the Message Type element to the Tagged Message.


The first thing we need to do is create an element for the message called “MessageType”, follow these steps:

- 1) Right click on the “TaggedVariableMessage” message element, select Insert Element → As Child.
- 2) Set the name for the element to “MessageType”
- 3) Set the Type of the element to leaf.

These steps can be repeated for adding the Street, City, State, and Zip elements.

4.3. Adding Groups

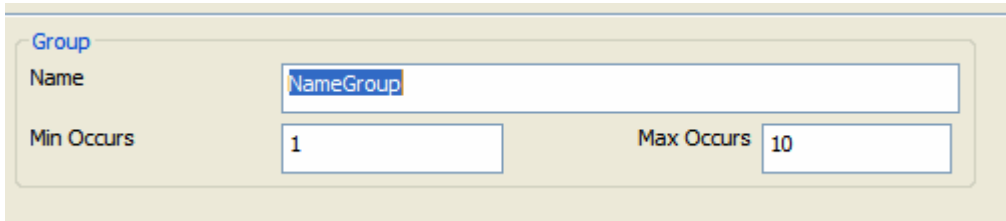
With tagged messages comes the added convenience of being able to add groups. Groups are what they sound like, a grouping of repeatable elements. For instance if you had more than three addresses within a message, it would be time consuming and not make much sense to create three different Street, City, State, and Zip elements (a total of 12 elements) when one group can be defined with Street, City, State, and Zip elements that is repeatable 3 times.

A group can be added by right clicking the message with a type set to variable and an ID Method set to tag, and selecting Insert Group, or selecting the Insert Group icon  will insert a group as well.

For our example specification we have a group called NameGroup that is repeatable up to 10 times. The properties for the group being created should be set as follows:

- Name: NameGroup
- Min Occurs: 1
- Max Occurs: 10

An example of this is given in Figure 16.



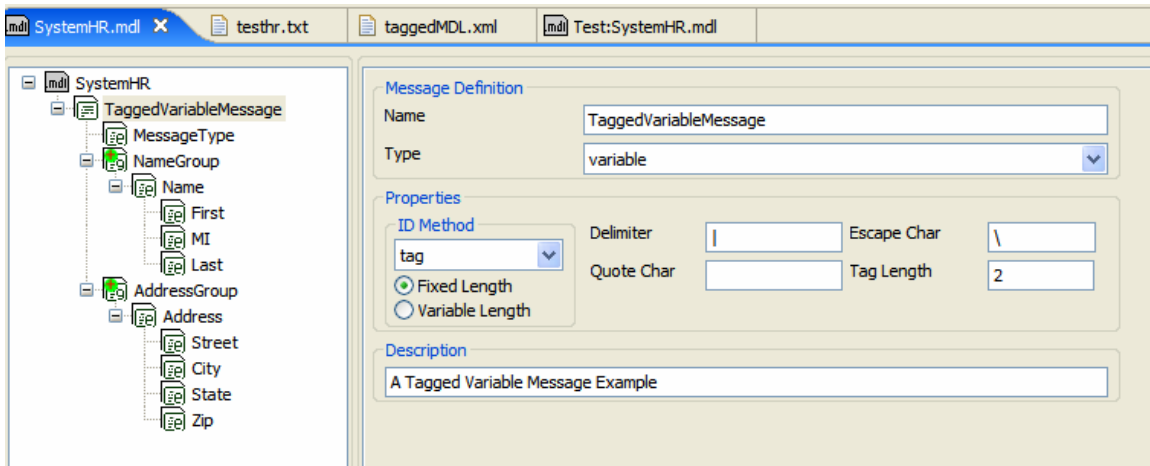
Group

Name

Min Occurs Max Occurs

Figure 16

The resulting MDL definition for the example is shown in Figure 17.



SystemHR.mdl x testhr.txt taggedMDL.xml Test:SystemHR.mdl

SystemHR

- TaggedVariableMessage
 - MessageType
 - NameGroup
 - Name
 - First
 - MI
 - Last
 - AddressGroup
 - Address
 - Street
 - City
 - State
 - Zip

Message Definition

Name

Type

Properties

ID Method

Delimiter Escape Char

Quote Char Tag Length

Fixed Length Variable Length

Description

Figure 17

5. Advanced Features

5.1. *Include and Import*

There are instances when creating a MDL that a lot of the elements and or messages may be reused in certain circumstances. If a system that ChainBuilder ESB is connecting to has a set of messages with a similar structure, each file would require all components that make up the message. This most likely would force you, the user, to re-define the same components for each individual message. ChainBuilder ESB provides the include and import functions in Message Format Editor to alleviate user's burden.

The ChainBuilder ESB Format Editor provides the ability to define multiple messages in a single file. It also support namespaces and allows include and import tags, so definitions can be broken down into multiple files for better re-use.

An MDL file can contain multiple includes and imports.

5.1.1. Using Include

The include function allows the user to include all the elements and messages within a previously defined MDL file and include them in the current MDL file.

To include an already produced MDL file do the following:

- 1) Right click on the root node and select Insert Include from the drop down menu.
- 2) Select the MDL file you would like to include in the current MDL definition.

5.1.2. Using Import

An import of a file is similar to the include of a file in that it allows for the inclusion of an MDL file within the new MDL file. However, it differs in that the imported file can have a target namespace defined for it. The namespace is used when a parsed message is represented as XML. This is equivalent to the target namespace of an XML Schema definition.

To import an MDL file and set its target namespace do the following:

- 1) Right click on the root node and select Insert Import from the drop down menu.
- 2) Select the MDL file you would like to import.
- 3) Set the target namespace of the MDL file.

5.2. *Global Element Definitions and References*

There are a great many systems that contain elements in them that can be reused. As an example lets take a Name definition that contains a first name, MI, last name. This name could be present within many messages like a address message, insurance provider message, financial provider, etc. It would become redundant to create a new Name element for use

within each message structure. Because of this ChainBuilder ESB has the capability to create a Global Element that can be referenced by many messages. For our example, we would create a global element called Name that could be referenced by many messages. This means that the Name element would only be defined once, instead of many times.

5.2.1. Defining a Global Element

A global element is an element that resides under the root node of the MDL tree and does not reside within a message. To create a global element within an MDL file do the following:

- 1) Right click on the Root node, Select Insert Element → As child or Right click on a message, Select Insert Element → As Sibling
- 2) Define the element as any element within a message would be defined.

At this point the Global Element is ready to be referenced.

5.2.2. Referencing a Global Element

Referencing a Global Element is an easy process, the only requirement is that a Global Element has already been created and is ready to be used. To use a Global Element within a Message do the following:

- 1) Right click the Message where you want the Global Element to reside.
- 2) Select Insert Element → As Reference.
- 3) There is Reference property with a pull down list of available Global Elements.
- 4) Select the one you would like to use.
- 5) Set the required attributes for the message.

6. Format Tester

After you define the message, you can run a test on the message definition by choosing a test data file. The testing feature in the Format Editor will display the parsed result based on the message definition. You can then visually examine if the result is what you expect to ensure that the message definition is correct before it is used in the Map Editor or Parser component.

You can test the format definition by clicking the green “Test” button on the toolbar or click “Format Test” in the context menu. This will open the Test Editor and bring up a new dialog window for you to choose test data as shown in Figure 18.

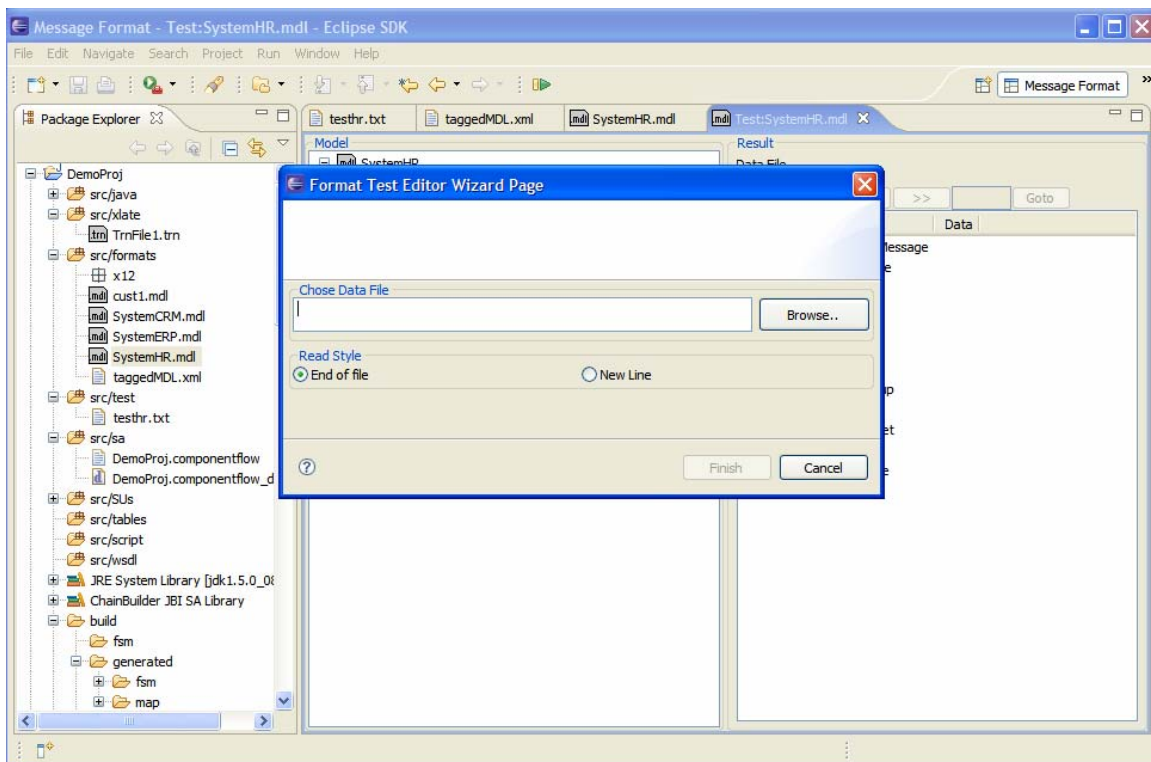


Figure 18

Click the “Cancel” button. You will close the dialog and the Format Tester.

Click the “Browse...” button. You will then be prompted to select a file that contains test data. as shown in Figure 19:

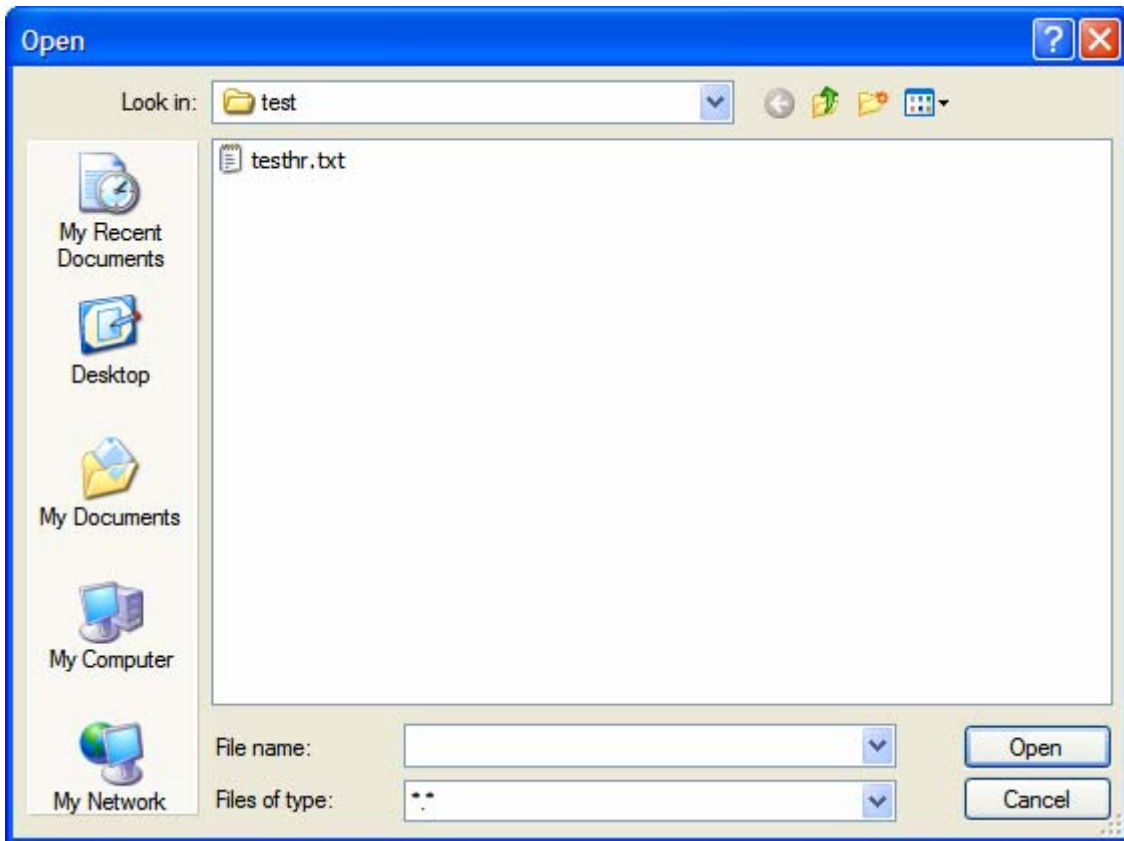


Figure 19

Select a file that contains a single sample record and click the “Open” button. The test data in the file will then be parsed using the message definition and the results will be displayed on the right side of the window as shown in Figure 20 for the example defined in the section 4.

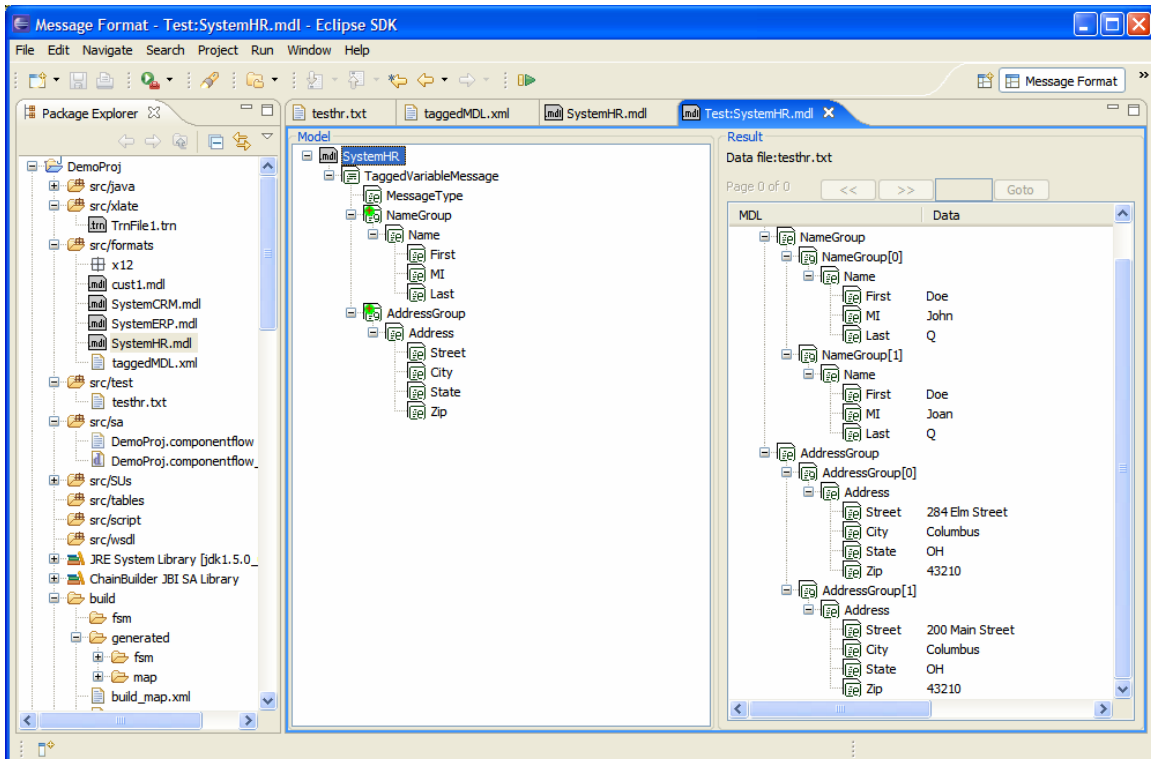


Figure 20

When you finish reviewing the test output, just close the window.

7. ChainBuilder ESB Community

ChainForge.net is the internet's premier destination to share ChainBuilder and JBI knowledge with your peers.

Join the ChainBuilder ESB Community:

<http://www.chainforge.net/community>

As a member you can view content and contribute to a Forum:

<http://www.chainforge.net/community/forums.html>

Read ChainBuilder ESB related Blogs:

<http://www.chainforge.net/blogs>