Fixed Length Message Responder

This topic introduces the Fixed Length Message Responder and explains configuration options that are unique to this type of Message Responder.

Sections include:

About the Fixed Length Message Responder

Fixed Length Message Responder is a Message Responder that is designed to simplify the use of fixed length by allowing you to utilize XML. You can model your fixed length response payload as an XML document; the responder then automatically converts the XML to fixed length before sending the message. If the responder receives a fixed length message, the responder can convert the message to XML so that you can define message correlations using XPaths or attach tools. Message Responders are protocol agnostic. The transport protocol or API to access a responder is defined in the deployment configuration of the PVA.

About Fixed Length Format

Fixed length is a data format for which there is no standard definition. In general, fixed length data formats consist of a number of records. Each record has one or more fields, each of which is of a fixed length (hence the name "fixed length format"). These fields typically do not have separators between them.

Example

For a simple example, consider a message where each record is on a different line and simply contains two fields:

- last name
- first name

Each field is 10 characters long, is right-aligned, and is padded by spaces.

```
Smith         John
Doe           Jane
```

A record in a fixed length format may have nested sections that also have records with fixed length formats. These nested sections could have a variable number of records. Consider a second fixed length format that is based on the first example. Assume that for each person, we want to specify a variable number of pets owned by that person. We will start with the data format we used above, but for each record we will add the following:

- A number field that specifies the number of pets; this is 2 characters long, padded with 0.
- A variable number of sections, where each section contains the type of pet and the name of the pet. The number of sections is equivalent to the number in the field that specifies the number of pets. The field for type of pet is 5 characters long, right-aligned. The field for the name of pet is 10 characters long, right-aligned.

Here is an example of such data.

```
Smith       John00
Doe         Jane02  Cat       Fluffy  Dog       Spot
```

The first person record has 0 pets, and the second person record has 2 pets. The pets sub-section has pet records that can appear a variable number of times.

Configuring Your Fixed Length Data Format: Overview

Since many different fixed length formats exist (and there is no single standard), Parasoft Virtualize is easily configured to recognize any fixed length message format that the organization uses. In order to configure and validate fixed length messages in Virtualize, you need to perform the following one-time configuration:

1. Define a data model that describes your fixed length data format.
2. Register the data model that so Virtualize can use it in the Fixed Length Responder and XML Converter.

Defining a Data Model that Describes your Fixed Length Data Format

There are two main steps required to define a data model that describes your fixed length data format:

Creating a New Data Model Definition File

The first step in working with fixed length messages in Virtualize is to define a data model that describes your fixed length data format.

To create a new data model:

1. Choose **File> New> Data Model Definition File**.
2. Specify the name of the file into which the data model definition will be stored and a location that is relative to your workspace. (If you want to save the file somewhere outside of the workspace, first save it in the workspace and then move it later.) We recommend saving the data model file in the workspace so that you can check it into source control along with your tests. This way it is easy to get the latest updated data model files for use in your test projects.
3. Click **Finish**.

The wizard will create a file with the extension .datamodel in the workspace location that was selected. The file does not appear within the Virtual Asset Explorer. You can find it by switching to Navigator view and looking within the project where it was saved.

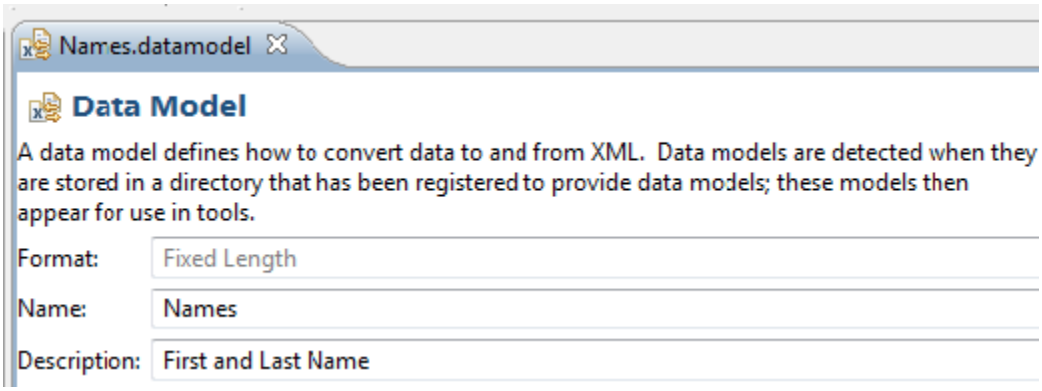
Editing the Data Model Definition

The next step is to use the data model editor to tell Virtualize how your fixed length messages are structured. Virtualize will use this information to convert these messages to and from XML.

The data model editor opens after a new data model definition is created. You can also access it by double-clicking the Navigator node that represents the data definition file.

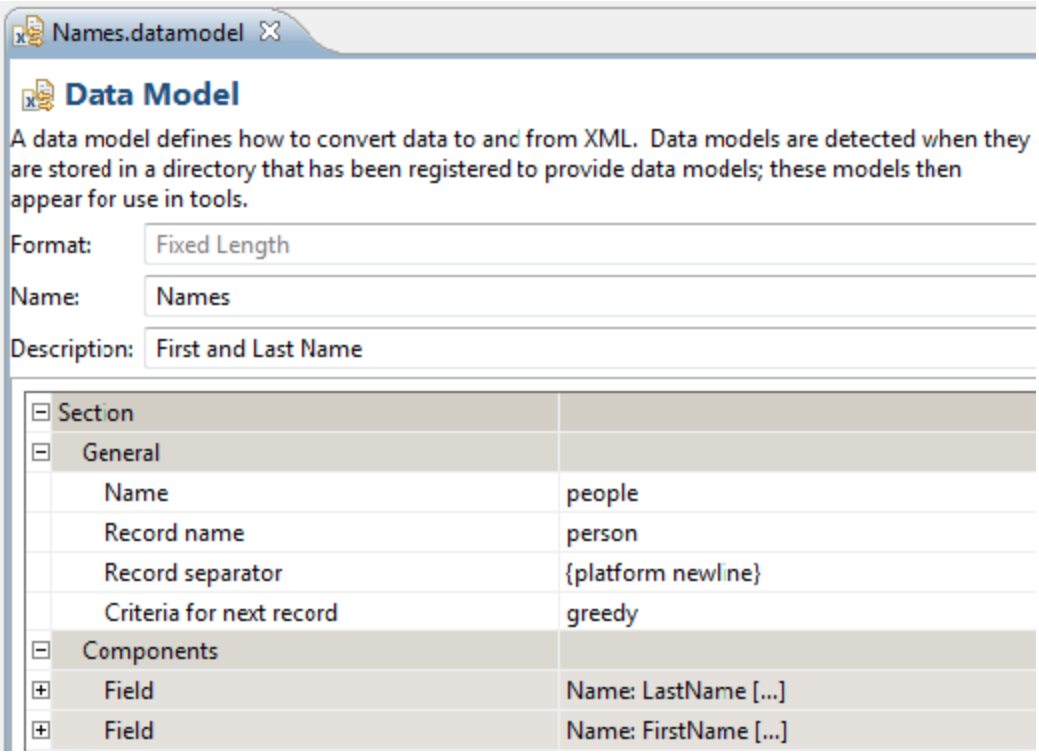
1. Enter a name and description for the data format

This name will be used to identify your message format in applicable tools.



2. Specify general information about the section

Next, enter the general details about the section as a whole: its name, the name of its records, how these records are separated, and the criteria for next record.



Name

Specify the name of the section. This name appears in message configuration interfaces and is used as the XML tag name in the converted XML (illegal XML characters get stripped out in the XML). It can be set to anything.

Record Name

Specify the name of each record within the section. This name appears in message configuration interfaces and is used as the XML tag name in the converted XML (illegal XML characters get stripped out in the XML). It can be set to anything.

Record Separator

Indicate what separator should be used between different records within the section. You can either select one of the available options (double-click the field to open a drop-down), or enter a custom separator.

Available options are:

- {platform newline} - The platform newline for the operating system ("\r\n" on Windows, "\n" on Linux)
- {if} - "\n"
- {crlf} - "\r\n"
- {cr} - "\r"
- {space}
- {tab}

Criteria for Next Record

Specify criteria that will enable the parser to know that it is done reading this section when the parser is converting from fixed length to XML. This is important when there is additional content that appears after this section that should not be confused as being part of this section. The parser needs to know how many records appear within this section before it moves on from this section.

Valid values are:

- **fixedCount** - Specifies that there are a fixed number of records within this section. The parser moves on from this section once it has seen this number of records. When this option gets chosen, a new descriptor named Record Count appears; here, you can specify the number of records that appear in this section.
- **greedy** - Specifies that all remaining content in the file should be considered as part of this section. This essentially means there are no other fields or sections that appears after this section. In this case, the parser "greedily" eats up all remaining content and applies it to this section.
- **xpath** - This option is used when the number of records in this section is dynamically determined by other content in the document. One example is when anther field in the document specifies the number of records that are in this section. Another example is when each record in the section starts with a delimiter. See below for more details on this option.

The xpath option specifies the condition under which the parser continues applying content to this section. The parser considers that it is still in this section as long as this condition is true; once the condition becomes false, the parser moves on from this section. The condition gets applied at the beginning of reading a record in order to determine whether or not the following content is a new record in this section.

When this option is chosen, a new descriptor named XPath appears. You can enter an xpath expression that describes the condition under which the content should still be applied to this document.

When the condition is evaluated, the XPath gets applied to the XML document that is being created from the fixed length document.

The default XPath is "count($section/*) < 1". Let's examine what this means. $section is a special XPath variable that applies only in this context. It refers to the XML element that represents this section. "$section/*" gets the collection of all XML elements that are children of the section element. Basically, it gets all the XML elements that represent the records in this section. "count()" is a normal XPath function that counts the number of nodes in the nodeset that is passed to it. So "count($section/*)" ends up returning the number of records that have been added to the XML document so far. The full expression then essentially means "keep adding records to this section as long as the number of records that have already been added to this document is less than 1" - meaning by default to only add one record.

However, in many cases, the number of records to read is specified in some other field in the document. In this case, you would use an XPath that references the field within the document that specifies the number of records. Remember that the XPath gets evaluated against the XML document that is being created, so it references an XML element within the XML document. If a document has a field that has been named "NumberOfPets", then a corresponding XPath to locate that element might be "$section/../NumberOfPets". This means "start at the XML element representing this section, go up one element to the parent, and then get the NumberOfPets element". This would find a "NumberOfPets" element that is a sibling of the current section. In this case, the full XPath expression might be
"count($section/*) < $section/../NumberOfPets".

There are two available XPath functions that are unique to this context: fixedlen:peek() and fixedlen:peek-starts-with()

fixedlen:peek()

This function takes a single integer as an argument. It peeks at the upcoming content within the fixed length document that has not yet been read and converted to XML. This is used in cases where the determination of whether there are more records to read for this section is based on some content that appears later in the file. For example, consider a case where the record starts with a "<" character:

```
Doe        Jane<  Cat        Fluffy><  Dog        Spot>
```

In this case, your XPath expression would be "fixedlen:peek(1) = '<'".

fixedlen:peek-starts-with()

This function takes a single string as an argument. It peeks at the upcoming content within the fixed length document that has not yet been read and converted to XML. This is used in cases where the determination of whether there are more records to read for this section is based on some kind of delimiter that signals the start of the record. For example, consider a case where the record starts with a "<" character:

```
Doe        Jane<  Cat        Fluffy><  Dog        Spot>
```

In this case your XPath expression would be "fixedlen:peek-starts-with('<')".

The fixedlen:peek-starts-with() function is the equivalent of the XPath expression "starts-with(fixedlen:peek(1), '<')".

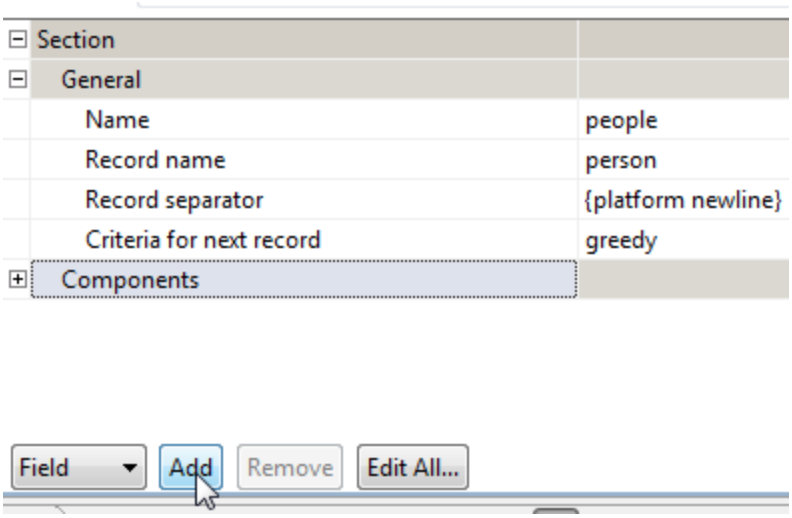
The default option is greedy. This is the option that you will typically use—unless you are working within a nested sub-section and it is not the last content in the document.

3. Specify component fields and subsections

Use the Components section to provide details about what fields and/or sub-sections are contained by the section. Multiple fields and/or sub-sections can be added here.

To add a field:

1. Select the **Components** node.
2. Set the box in the lower left of the panel to **Field**.
3. Click **Add**.

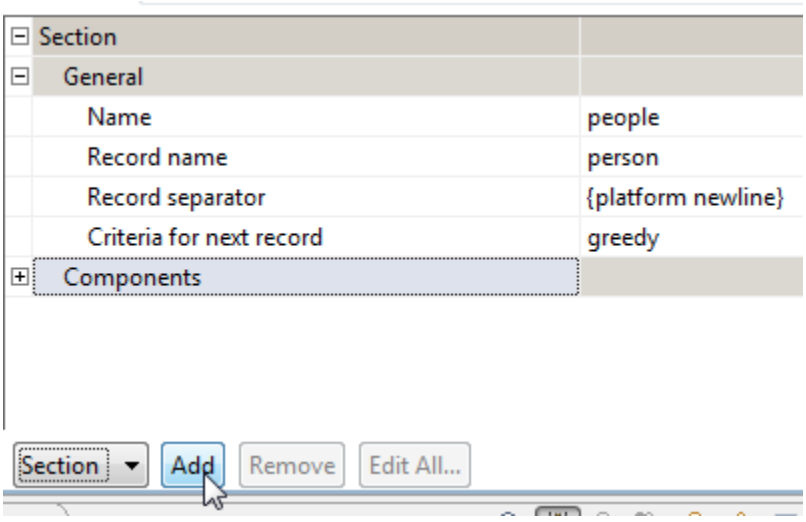


For each field, specify the following settings:

- **Name** - Enter the name of the field. This name is used in appropriate tool controls and as the XML tag name in the converted XML (illegal XML characters get stripped out in the XML). It can be set to anything.
- **Type** - Select one of the following values:
  - **decimal** - Indicates that this field contains a decimal value.
  - **integer** - Indicates that this field contains an integer value.
  - **string** - Indicates that this field contains a string value.
- **Length** - Specify the length of the field.
- **Alignment** - Specify whether the value in the field (when shorter than the length of the field) should appear on the left or the right.
- **Padding** - Specify what character to use when the value in the field is shorter than the length of the field. Generally you use a space for string fields and a space or '0' for number fields. To use a space specify {space}. Otherwise, any single alphanumeric character is valid here.

To add a subsection:

1. Select the **Components** node.
2. Set the box in the lower left of the panel to **Section**.
3. Click **Add**.



For details on completing general section details, see 2. Specify general information about the section.

Editing in a Tree

The initial view for the data model editor shows the data model in a tree-based table. The tree format allows you to easily see the hierarchy and structure of the data model. It is in a table format so that you can edit values within the context of the data model structure.

In this tree-table view, the fields that can be edited fall into the following categories:

- **Free-form text** - Simply go to the field and type whatever text you want to enter.

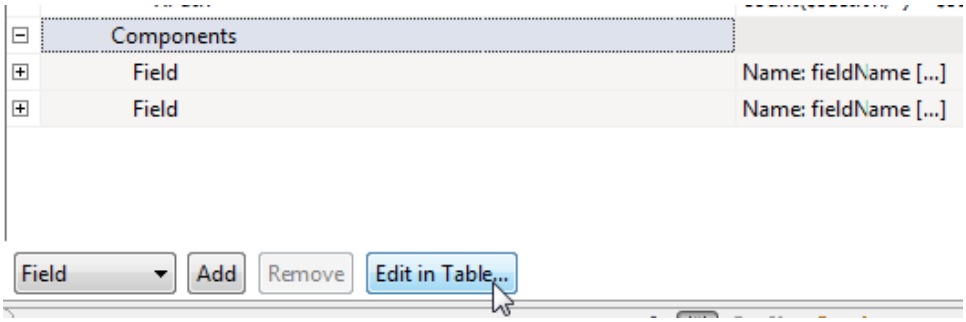
- **Combo box selections** - These fields only allow you to select predefined values for the field. You can double-click in the field to bring up a combo box that allows you to select your choice. Or you can tab into the field, press the first letter of one of the options in the combo (which will make that option appear), and then press enter to move to the next field.
- **Editable combo box selections** - You can either double-click in the field to bring up a combo box that allows you to select your choice. Or you can type whatever value into the field that you would like to enter.

Editing in a Table

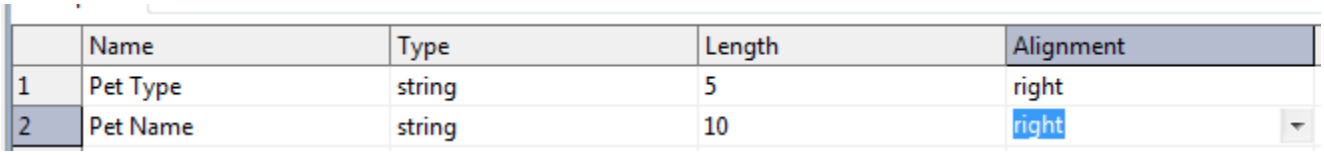
When you're entering data for a large number of fields, you might want to switch from the tree format to the table format, which is designed to provide a fast and easy way to enter field descriptors for a single section.

To switch to table view:

1. Select the **Components** node for the section you want to edit.
2. Click the **Edit in Table** button.



The table view resembles a spreadsheet. Each field in the section is a single row in the table, and each column in the table corresponds to one of the field descriptors. You can add more fields in this view by simply modifying the last row in the table and pressing an arrow key or the Enter key.



	Name	Type	Length	Alignment
1	Pet Type	string	5	right
2	Pet Name	string	10	right

You can also use the following buttons:

- Move Rows Up - Highlighting one or more rows and clicking this button will move the rows up within the table.
- Move Rows Down - Highlighting one or more rows and clicking this button will move the rows down within the table.
- Delete Rows - Highlighting one or more rows and clicking this button will delete the selected rows.
- Template for new row - This specifies what type each new field should have by default when a new row is added to the table.

To return to the main view for the data model, click the **Return to Full Data Model** button.

Switching Between Views

There may be cases where you have a number of fields and sections defined in your data model, and you realize that you need to add a new field in the middle of those you already have defined.

To achieve this:

1. Add the new field in the full data model view or in the section table view.
2. Switch to the table section view (click **Edit in Table**).
3. In that view, move rows up and down. This will correspond to moving your fields and sections within your data model.
4. Switch back to the full data model view (click **Return to Full Data Model**) to see the change reflected there.

Testing the Data Model

As the data model is being defined, it you probably will want to test it to ensure that it is accurately reflecting your fixed length data format.

To do this:

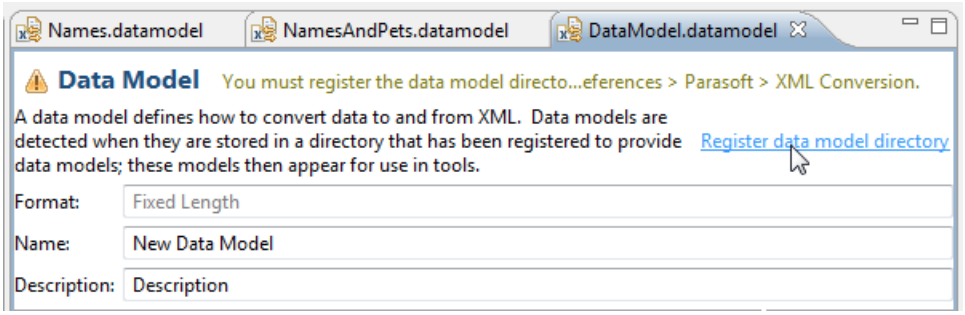
1. Ensure that the data model is registered (as described in Registering the Data Model)
2. Add it to a Fixed Length Responder (as described in Using the Data Model in Virtualize Tools).
3. Configure the responder with the desired message.

Registering the Data Model

Each data model that you have defined needs to be registered in Virtualize before the associated formats can be used. This registration has to be performed on every Virtualize installation that you want to access your fixed length format(s). If you have both Virtualize and SOAtest installed, you can register the model once, then use it for both products.

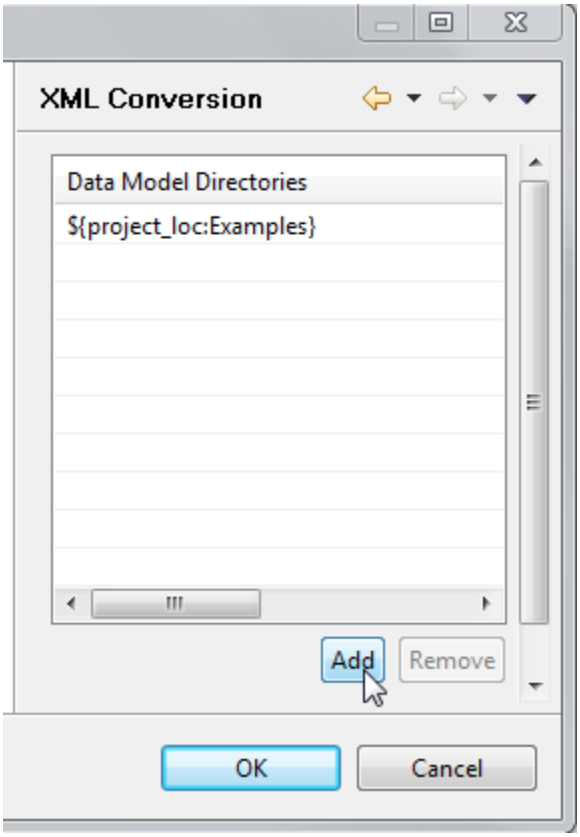
There are two ways to register a data model:

- If you're in the data model editor, click the **Register data model directory** link. This will register the entire directory that contains the data model being edited.



- Otherwise:

  a. Choose **Parasoft> Preferences**.
  b. Select **Parasoft > XML Conversion**.
  c. Click **Add**.



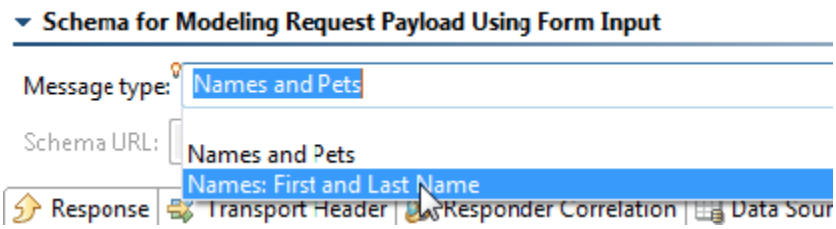
  d. Specify the data model directory that contains your data models.

Using the Data Model in Virtualize Tools

Once registered, the data model can be used in the Fixed Length Message Responder as well as in the XML Converter. You can create a Fixed Length Message Responder tool directly from the Add Responder wizards.

In order to perform fixed length message configuration or validation, you must select your data model in the **Message type** combo box in each of those tools.

You can now configure the Fixed Length Message Responder as you could configure any other Message Responder. For details on configuring standard Message Responder behavior (e.g., correlations, performance profiles, etc.), see Message Responder Overview.



Sample Generation

When you select a message type as shown above, Virtualize fills the Form Input view with sample XML that can be converted to fixed length data without requiring modification.

The data model definition for the fixed length message type determines the content of the generated XML. The XML contains the sections and fields specified in the definition.

An XML element is added for each section. Each section element contains elements for records. The section's **Criteria for next record** setting determines how many records are added to the section:

Criteria	Records Added
fixedCount	The "Record count" option determines the number of records.
greedy	Exactly one record is added.
xpath	The XPath is evaluated when generating the XML. Another record is added if the XPath evaluates to true. Virtualize always adds at least one record, and—in the unlikely event that the XPath evaluates to true many times—never adds more than 1000 records. (Because of these restrictions on the number of records added, there is the small chance that using XPath criteria will result in sample XML that will not convert to fixed length data without modification.)

Fields are filled with default values. Integer and decimal fields are set to the number 1. String fields are set to the value a. Virtualize uses this arbitrary value rather than no value (the empty string) to clarify where each field begins and ends if you convert the sample XML to fixed length format.