Consuming the Data in Tools

Repository data—like data from Excel, CSV, and other data sources—is consumed by Parasoft messaging tools via a data source. You define a data source that specifies where to access the appropriate data and (optionally) what subset of the available data you want to use. Then, you populate tools by parameterizing values against this data source.

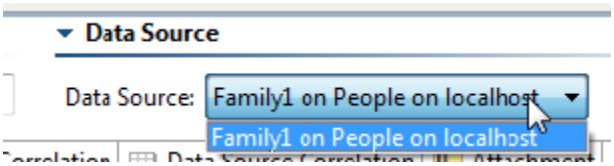
In SOAtest, repository data can be consumed in messaging client tools, such as the SOAP Client, REST client, Messaging Client, and other client-oriented tools.

In Virtualize, repository data can be consumed in Message Responders, SQL Responders, and client-oriented validation tools. For details on how repository data is used in SQL Responders, see SQL Responder. All other cases are discussed below.

Hierarchical Parameterization with Form Views

To use form views (e.g., Form Input or Form JSON) to parameterize tool values with data from a repository:

1. Ensure that you have a data source that connects to the desired repository, and that this data source is available to the suite including the tools you want to parameterize.
2. In the tool that you want to parameterize, do the following:
    a. Ensure that the repository data source is selected in the Data Source area (in the top right of the editor).



    b. Open the desired form view.
    c. To populate all items with the matching Data Repository columns, right-click the tree and choose **Populate**. (Note that the attribute exclusion option lets you control whether optional attributes are automatically added by the populate process). Or, for each item you want to populate with a repository value, select **Parameterized**, then select the name of the column that contains the values you want to use.

Notes:

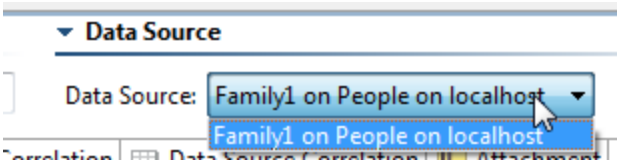
> **Tip: Accessing Data Values from Custom Tools**
>
> When working with custom tools (i.e., tools added via the Extension Framework), you access values from a data repository column by manally entering the column name in the applicable tool text fields. For example, if you want to access the values in a key column called "loanAmount" in a Data Repository, you would enter `${loanAmount (key)}` in the appropriate tool field.

- A complex element should be parameterized with a record list column
- A simple element should be parameterized with a primitive or primitive list column

Hierarchical Parameterization with Literal View

To use Literal view to parameterize tool values with data from a repository:

1. Ensure that you have a data source that connects to the desired repository, and that this data source is available to the suite including the tools you want to parameterize.
2. In the tool that you want to parameterize, do the following:
    a. Ensure that the repository data source is selected in the Data Source area (in the top right of the editor).



    b. Open the Literal view.
    c. Edit the message, using "ParasoftColumn" to specify which Data Repository column should be used to parameterize that level in the hierarchy of the message. For details, see the guidelines below in JSON Guidelines and XML Guidelines.

Note that "ParasoftColumn" is applicable only for complex elements; simple elements are parameterized by accessing data source values using the {$} syntax.

For example, assume you have the following data repository data:

```
body:
        info: "some info about stuff"
        excludedSimple: "[parasoft_exclude]"
        optionalSimple: "[parasoft_null]"
        excluded: []
           optional: null
            items: [
                {
                        item: [
                                { name: "bill", last: "mclaren", id: "12345"}
                                { name: "steve", last: "smith", id: "45678" }
                        ]
                }
        ]
        codes: [
                {
                        code: [ "abc", "efg", "hij" ]
                }
        ]
```

Here is a JSON message that is parameterized to use that data:

```
{
	"body" : {
		"ParasoftColumn" : "body",
		"info" : "${info}",
		"excludedSimple" : "${excludedSimple}",
		"optionalSimple" : "${optionalSimple}",
		"excluded" : {
			"ParasoftColumn" : "excluded",
			"excludedChild" : "${excludedChild}"
		},
		"optional" : {
			"ParasoftColumn" : "optional",
			"optionalChild" : "${optionalChild}"
		},
		"items" : [
			"ParasoftColumn: items",
			{
				"ParasoftColumn" : "item",
				"name" : "${name}",
				"last" : "${last}",
				"id" : ${number:id}
			}
		],
		"codes" : [
			"ParasoftColumn: codes",
			"${code}"
		]
	}
}
```

Once the parameterization is completed, the expected JSON message would look like this:

```
{
	"body" : {
		"info" : "some info about stuff",
		"optionalSimple" : null,
		"optional" : null,
		"items" : [
			{
				"name" : "bill",
				"last" : "mclaren",
				"id" : 12345
			},
			{
				"name" : "steve",
				"last" : "smith",
				"id" : 45678
			}
		],
		"codes" : [
			"abc",
			"efg",
			"hij"
		]
	}
}
```

Here is an XML message that is parameterized to use that data:

```
<root>
	<body ParasoftColumn="body">
		<info>${info}</info>
		<excludedSimple>${excludedSimple}</excludedSimple>
		<optionalSimple>${optionalSimple}</optionalSimple>
		<excluded ParasoftColumn="excluded">
			<excludedChild>${excludedChild}</excludedChild>
		</excluded>
		<optional ParasoftColumn="optional">
			<optionalChild>${optionalChild}</optionalChild>
		</optional>
		<optional>${optional}</optional>
		<items customerAttr="val" ParasoftColumn="items">
			<item ParasoftColumn="item">
				<name>${name}</name>
				<last>${last}</last>
				<id>${id}</id>
			</item>
		</items>
		<codes ParasoftColumn="codes">
			<code>${code}</code>
		</codes>
	</body>
</root>
```

Once the parameterization is completed, the expected XML message would look like this:

```
<root>
	<body>
		<info>some info about stuff</info>
		<optionalSimple xsi:nil="true"/>
		<optional xsi:nil="true"/>
		<items customerAttr="val">
			<item>
				<name>bill</name>
				<last>mclaren</last>
				<id>12345</id>
			</item>
			<item>
				<name>steve</name>
				<last>smith</last>
				<id>45678</id>
			</item>
		</items>
		<codes>
			<code>abc</code>
			<code>efg</code>
			<code>hij</code>
		</codes>
	</body>
</root>
```

JSON Guidelines

- An object or array should be parameterized with a record list column.
- A simple object field should have its content value parameterized with a primitive column. For number or a boolean value, the content value should be ${number:<value>} or ${boolean:<value>}.
- A simple array item should have its content value parameterized with a primitive list column. For number or a boolean value, the content value should be ${number:<value>} or ${boolean:<value>}.
- You can parameterize with variables as well as with primitive or primitive list columns from repository data sources.
- If an object or array is parameterized with a data repository column, there shouldn't be any other elements at the same level with that same name.
- The root object or array is not parameterized.

XML Guidelines

- A complex element should be parameterized with a record list column.

- A simple element should have its content value parameterized with a primitive or primitive list column.
- You can parameterize with variables as well as with primitive or primitive list columns from repository data sources.
- If an element is parameterized with a data repository column, there shouldn't be any other elements at the same level with that same name.
- The root element is not parameterized.