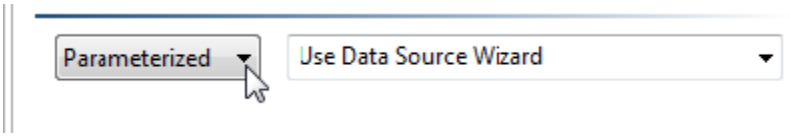
_Parameterizing Tests with Values Extracted from Another Test

You can parameterize tests (for data-driven testing) by extracting values from one test and then using them available in another test. This is accomplished with tools such as:

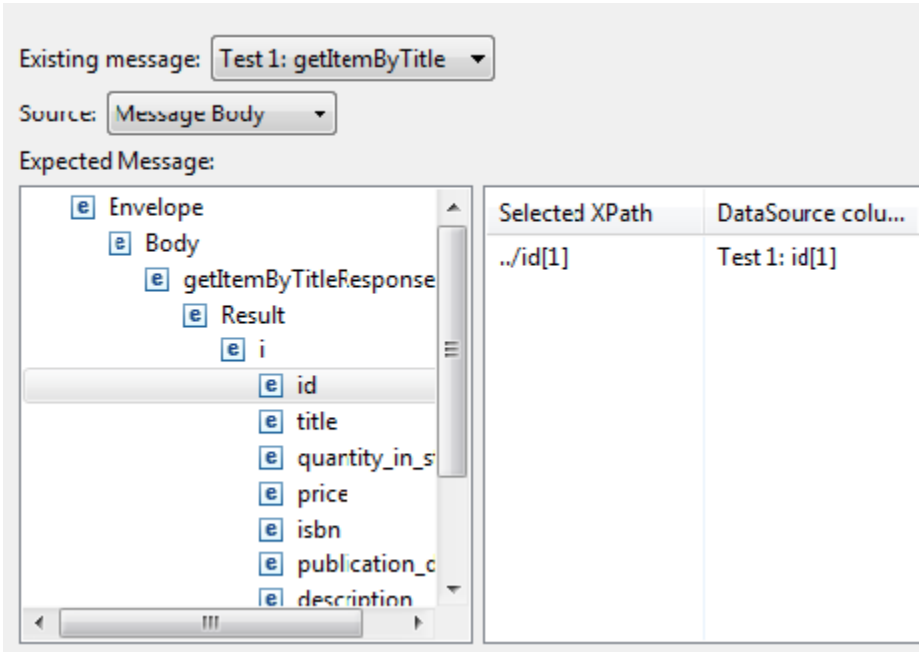
- XML Data Bank
- Header Data Bank
- JSON Data Bank
- Text Data Bank
- Writable data sources (described in Configuring a Writable Data Source)

- Object Data Bank
- Browser Data Bank

For example, if you want to parameterize one of test 3's request value with the id value from test 1's response, you can would set that field to Parameterized....



then use the data source wizard to indicate that you want to use the value of the id element from test 1 (which will be stored in an XML data bank tool that is automatically added to the test suite).



The following section explains two different strategies for passing values from test to test. In addition to this test-to-test value passing, values can also be passed in the following ways:

- Through variables: Variables can used to parameterize all or parts of text field values in tool editors. For example, a text field could reference environment variables, a data source column, a data bank column, or a variable defined for the current suite. For details, see Parameterizing Tools with Variables.
- Through scripting: This is primarily used when you need to pass an object of a type other than a String. You can only access these values through scripting. This is done using com.parasoft.api.Context get(String) and put(String, Object). For details, see Extensibility and Scripting Basics.

Passing Values from One Test to Another

Parasoft SOAtest supports two primary ways to achieve test chaining, (i.e., extract values from one test to use in a subsequent test):

- Using a Data Bank
- Using a Data Bank with a Writable Data Source

Using a Data Bank – Capture values for single run

This approach is used in simple cases when values from a single test execution need to be captured for subsequent reuse.

For example, you can configure a test suite that tests a bank's Web service transactions. Test 1 of that test suite can log on to the service using a User ID, then the SOAP response would return a session ID back to Test 1. Test 2 of that test suite can be configured to use the session ID from Test 1 to perform transactions. You can configure any of the tests in a test suite to use SOAP response parameters as SOAP request parameters.

A test scenario for this case might look like this:

- Test 1: Log on and Retrieve ID
    - XML Data Bank: Extract user ID
- Test 2: Use ID to invoke service 1
- Test 3: Use ID to invoke service 2
- ...

In this case, we are assuming that every time the test suite executes, we use a different ID throughout the scenario, but it is a single ID for each scenario run.

It is important to note that the same principle applies to all SOAtest data bank tools (XML, Browser, Header, JSON, Object, Text). You can extract a single value into any of these data bank tools, then use it in subsequent tests (as a parameterized value, as well as through scripting with com.para-soft.api. ScriptingContext.getValue(String, String). Extracted values are stored as strings.

This functionality is designed to suit a wide variety of customized needs; the example given above is just one sample application.

> **Tutorial**
>
> For a step-by-step example of how this functionality works, see Storing Results to Be Used in Subsequent Tests.

Using a Data Bank with a Writable Data Source – Capture values to iterate on multiple runs

This case is useful when you need to extract a list of values of an unknown count, then have a test or test suite iterate over every single value that has been extracted. Assume the following test scenario:

- Data Sources:
    - Writable: Captured User IDs
- Set-Up Test: Get User IDs
    - XML Data Bank: Extract User IDs and write them into a writable data source
- Test 1: Use captured IDs to invoke service 1 (parameterized with "Captured User IDs")
- Test 2: Use captured IDs to invoke service 2 (parameterized with "Captured User IDs")
- …

In that case, the "Get User Ids" test assumes a service that returns a list (a sequence) of User Ids, and each User ID is extracted and written to the desired writable data source column. The exact number of user IDs can be unknown or variable. After the "Get User IDs" test executes and the Writable Data Source is populated with the values, the scenario proceeds to the remainder of the test suite and the subsequent tests (Test 1 and Test 2) can be parameterized with that writable data source (e.g., by selecting the appropriate data source then parameterizing fields vs. the appropriate columns in that data source). This means that the tests can iterate over it and use up all the IDs that have been written into it by "Get User IDs". In other words, they will execute for every ID that is retrieved.

Another case where writable data sources are useful if the test that is generating the values retrieves the values one at a time, but iterates over a data source and stores the extracted values into a Writable Data Source. In this case, a sample test scenario might be:

- Data Sources:
    - Table: Usernames
    - Writable: Captured User IDs
- Set-Up Test: Get and Store User ID (parameterized with Usernames)
    - XML Data Bank: Extract a User ID and write it into a writable data source
- Test 1: Use captured IDs to invoke service 1 (parameterized with "Captured User IDs")
- Test 2: Use captured IDs to invoke service 2 (parameterized with "Captured User IDs")
- …

In this example, the "Get and Store User ID" test step will run multiple times: once per each username provided by the "Usernames" data source. For each run, it will write a User ID to the "Captured User IDs" data source. Now that the "Captured User IDs" writable data source got populated (with user IDs), the remainder of the test suite will use these IDs to execute Test 1 and Test 2 with them.

For details on writable data sources, see Configuring a Writable Data Source.

then use the data source wizard to indicate which element's value you want to use. Or, you could manually configure values to be stored to a data bank or writable data source, then parameterize the response vs. those saved responses.

In addition to this tool-to-tool value passing, values can also be passed in the following ways:

- Through variables: Variables can used to parameterize all or parts of text field values in tool editors. For example, a text field could reference environment variables, a data source column, a data bank column, or a variable defined for the current suite. For details, see Parameterizing Tools with Variables.
- Through scripting: This is primarily used when you need to pass an object of a type other than a String. You can only access these values through scripting. This is done using com.parasoft.api.Context get(String) and put(String, Object). For details, see Extensibility and Scripting Basics.

General Rule of Thumb

A general rule of thumb is:

- To extract content from a message only once as part of a scenario, then use a Data Bank by itself.
- To extract a list of values then have another test iterate over them and use them one by one, use an XML Data Bank with a writable data source.