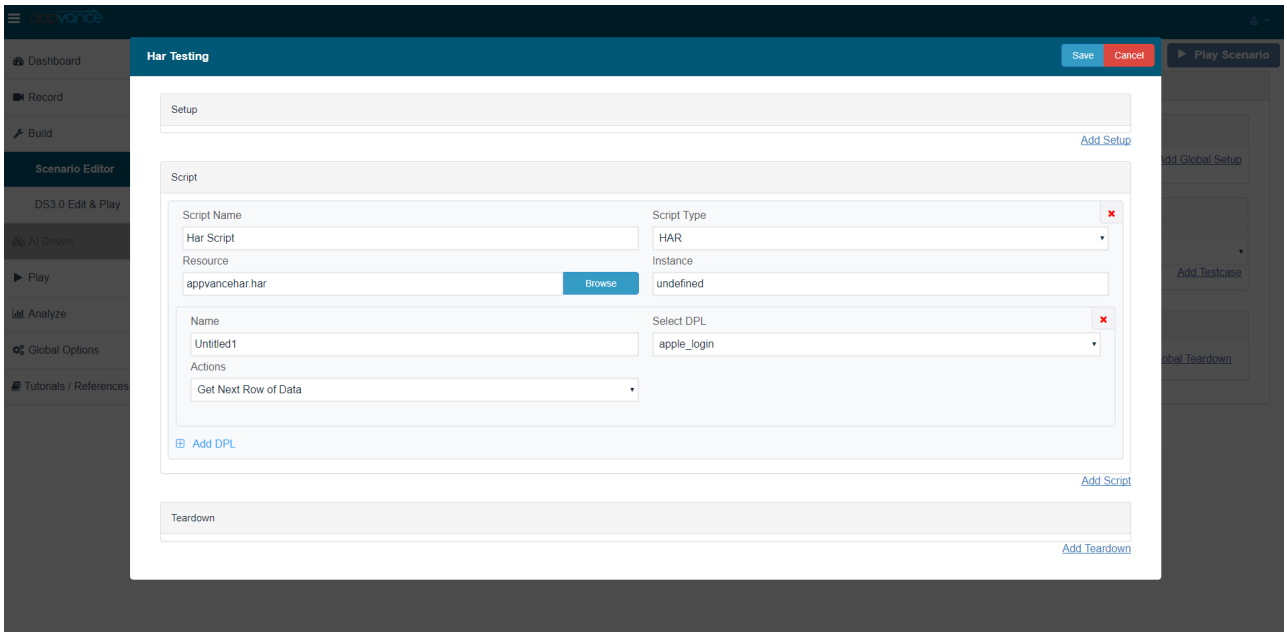
Configuring Appvance DPLs

AIQ implements a Data Production Library (DPL) as an extensible framework to provide operational test data to test script runners at test runtime. The DPL framework is general in the sense that it can get data from a variety of sources, including a relational database, comma-separated-value file, and data generating objects. A DPL also provides a safe way to store data between test steps. AIQ comes with pre-built DPLs and makes it easy for software developers to add their own custom DPLs.

DPLs work in conjunction with script runners at test runtime. Test scenarios define the DPL and script runner interaction. The AIQ Tutorials give instructions on using the pre-built DPLs.

AIQ users configure Test Use Cases in the Appvance Scenario Builder to use operational test data from a DPL. To see this in action follow these steps:

1. Start AIQ
2. Click Build from the Start Screen. The Scenario Builder opens with a blank page.
3. Click File, then New to create a new empty scenario. AIQ sets the default name of the new scenario to New-Scenario.
4. In the Info tab, which is opened by default, select Functional Test for this example.
5. Click the Resources tab.
6. Click Add Data Source.
7. Choose "HashDPL - Reads CSV" in the DPL Type drop-down menu.
8. Click Browse to select a CSV data file.
9. Provide a name for the DPL in the Name box.
10. Click Test Cases on the left-hand side.
11. Under Iteration Name: Iteration 1, click Add a Test Case Execution
12. Provide a Test Case Name at the top of the pop-up window.
13. Select Type Selenium, provide a Script Name and then click Browse to find your appropriate Selenese file.
14. Click Add DPL and under Select DPL choose the DPL you created earlier.
15. Choose Get Next Row of Data from the Actions menu.



16. Provide a name for the DPL action and then click Save at the top.
17. Click File at the top and then Save and choose a location to save the Test Scenario
    This configuration tells the AIQ Selenium script runner to look into the Selenium test and replaces values in the test with values from the data.csv file.
    The DPL provides data from the data.csv file as A IQ runs the Selenium script, at test run time.
    The DPL uses the first row in the CSV file to identify field mapping names. Insert these mapping names into the Selenium file where you want AIQ to inject the test data from the CSV file.
    After the mapping name row comes the rows of the data file containing the actual data.
    The DPL Action determines the row of data AIQ provides to the test.
    In a load test, setting the DPL Action to Get Next Row of Data tells AIQ to advance to the next row of data.
    When the DPL reaches the bottom of the table, the DPL starts at the top.

Sheets | Charts | Smar

	A	B	C	D	E
1	name	password	product	validate	
2	bert	biz	1000	1000	
3	bert	biz	1001	1001	
4	bert	biz	1002	1002	
5	bert	biz	1003	1003	
6	bert	biz	1004	1004	
7	bert	biz	1005	1005	
8	bert	biz	1006	1006	
9	bert	biz	1007	1007	
10					
11					

The Selenium, SoapUI, and Command-Line script runners provide support for data mapping from DPLs.

Script runners for Appvance Designer Script, Java, Jython, JRuby, VB, .NET, Groovy, and the other support scripting languages receive DPL data in a dictionary structure (map).

For TestNG engine to work with DPLs, it is necessary to need to define the @DataProvider annotation inside the TestNG, an AIQ will override that method providing the DPL values as arguments for the @Test method:

```
public class TestNG {
@DataProvider(name = "DPLDataProvider")
public Object[][] ValidDataProvider3() throws Exception {
return new Object[][] {
{'C', 65}, {'c', 97},
{'D', 66}, {'d', 98}
};
}
@Test(dataProvider = "DPLDataProvider")
public void basicTest(final char character, final int ascii) throws Exception {
System.out.println("character " + character + " ascii " + ascii);
}
}
```

In this case, you need to use CSV DPL in your Scenario. AIQ will do the casting of the arguments to the ones expected in the @Test method.

If you want to use HashMap instead (sending arguments as map dictionaries) then you need to add an AIQ provided annotation @TestNGParams (tools.jar):

```
public class TestNG {
@DataProvider(name = "DPLDataProvider")
public Object[][] ValidDataProvider3() throws Exception {
return new Object[][] {
{'C', 65}, {'c', 97},
{'D', 66}, {'d', 98}
};
}
@Test(dataProvider = "DPLDataProvider")
@TestNGParams(params = {"character","ascii"})
public void basicTest(final char character, final int ascii) throws Exception {
System.out.println("character " + character + " ascii " + ascii);
}
}
```

DPLs have the ability to share data from one test to the next. For example, TheLocker DPL takes the results from a test step authored in SoapUI as input to a second test step authored in Appvance Designer Script.

SoapUI support was removed in release 5.2.0 of AIQ.

The above example shows how to use data from a CSV file. DPLs work in the same way with data from a Relational Database (RDBMS DPL, RDBMS Map DPL) and other data-generating objects.

Last updated: July 25, 2024

If you have any feedback on this topic, please contact us at documentation@appvance.ai

---

Revision #5
Created 2026-07-31 15:09:32 UTC by Admin
Updated 2026-10-14 13:51:27 UTC by AI_Mananger