MICRO FOCUS

Silk Test 17.0

Keyword-Driven Testing

Micro Focus
The Lawn
22-30 Old Bath Road
Newbury, Berkshire RG14 1QN
UK
http://www.microfocus.com

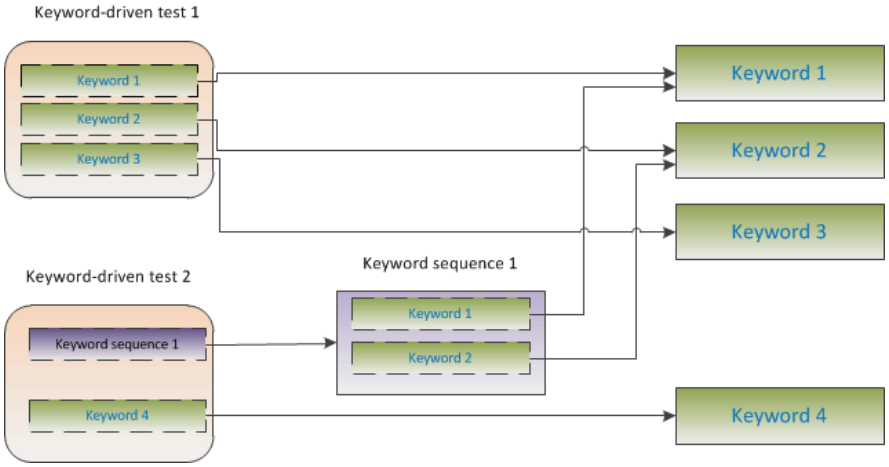
2016-05-04

Contents

Keyword-Driven Tests

Keyword-driven testing is a software testing methodology that separates test design from test development and therefore allows the involvement of additional professional groups, for example business analysts, in the test automation process. Silk Central and Silk Test support the keyword-driven testing methodology and allow a very close collaboration between automation engineers and business analysts by having automation engineers develop a maintainable automation framework consisting of shared assets in the form of keywords in Silk Test. These keywords can then be used by business analysts either in Silk Test to create new keyword-driven tests or in Silk Central to convert their existing manual test assets to automated tests or to create new keyword-driven tests.

- A *keyword-driven test* is an executable collection of keywords. A keyword-driven test can be played back just like any other test.
- A *keyword sequence* is a keyword that is a combination of other keywords. Keyword sequences bundle often encountered combinations of keywords into a single keyword, enabling you to reduce maintenance effort and to keep your tests well-arranged.
- A *keyword* is a defined combination of one or more actions on a test object. The implementation of a keyword can be done with various tools and programming languages, for example Java or .NET.



There are two phases required to create keyword-driven tests:

1. Designing the test.
2. Implementing the keywords.

Note: The following topics describe how you can automate manual tests with Silk Central, and how you can perform keyword-driven testing with Silk Central and Silk Test. Any tasks that are performed with Silk Test are described based on Silk4J, the Silk Test plug-in for Eclipse. The steps for Silk4NET and Silk Test Workbench are similar. For additional information on performing keyword-driven testing with a specific Silk Test client, refer to documentation of the Silk Test client.

Test Automation Overview

Test automation is the process of using software to control the execution of tests by using verifications to compare the *actual* outcomes of the execution with the *expected* outcomes. In the Silk Test context, test automation includes automated functional and regression tests. Automating your functional and regression tests enables you to save money, time, and resources.

A *test automation framework* is an integrated system that sets the rules of automation of a specific product. This system integrates the tests, the test data sources, the object details, and various reusable modules.

These components need to be assembled to represent a business process. The framework provides the basis of test automation and simplifies the automation effort.

Advantages of Keyword-Driven Testing

The advantages of using the keyword-driven testing methodology are the following:

- Keyword-driven testing separates test automation from test case design, which allows for better division of labor and collaboration between test engineers implementing keywords and subject matter experts designing test cases.
- Tests can be developed early, without requiring access to the application under test, and the keywords can be implemented later.
- Tests can be developed without programming knowledge.
- Keyword-driven tests require less maintenance in the long run. You need to maintain the keywords, and all keyword-driven tests using these keywords are automatically updated.
- Test cases are concise.
- Test cases are easier to read and to understand for a non-technical audience.
- Test cases are easy to modify.
- New test cases can reuse existing keywords, which amongst else makes it easier to achieve a greater test coverage.
- The internal complexity of the keyword implementation is not visible to a user that needs to create or execute a keyword-driven test.

Keywords

A *keyword* is a defined combination of one or more actions on a test object. The implementation of a keyword can be done with various tools and programming languages, for example Java or .NET. In Silk Test, a keyword is an annotated test method (`@Keyword`). Keywords are saved as keyword assets.

You can define keywords and keyword sequences during the creation of a keyword-driven test and you can then implement them as test methods. You can also mark existing test methods as keywords with the `@Keyword` annotation. In Java, keywords are defined with the following annotation:

```
@Keyword("keyword_name")
```

A *keyword sequence* is a keyword that is a combination of other keywords. Keyword sequences bundle often encountered combinations of keywords into a single keyword, enabling you to reduce maintenance effort and to keep your tests well-arranged.

A keyword can have input and output parameters. Any parameter of the test method that implements the keyword is a parameter of the keyword. To specify a different name for a parameter of a keyword, you can use the following:

```
// Java code
@Argument("parameter_name")
```

By default a parameter is an input parameter in Silk4J. To define an output parameter, use the class `OutParameter`.

Note: To specify an output parameter for a keyword in the **Keyword-Driven Test Editor**, use the following annotation:

```
${parameter_name}
```

In the **Keyword-Driven Test Editor**, you can use the same annotation to use an output parameter of a keyword as an input parameter for other keywords.

Example

A test method that is marked as a keyword can look like the following:

```
// Java code
@Keyword("Login")
public void login(){
    ... // method implementation
}
```

or

```
// Java code
@Keyword(value="Login", description="Logs in with the given
name and password.")
public void login(@Argument("UserName") String userName,
  @Argument("Password") String password,
  @Argument("Success") OutParameter success) {
    ... // method implementation
}
```

where the keyword logs into the application under test with a given user name and password and returns whether the login was successful. To use the output parameter as an input parameter for other keywords, set the value for the output parameter inside the keyword.

Note: If you are viewing this help topic in PDF format, this code sample might include line-breaks which are not allowed in scripts. To use this code sample in a script, remove these line-breaks.

- The keyword name parameter of the `Keyword` annotation is optional. You can use the keyword name parameter to specify a different name than the method name. If the parameter is not specified, the name of the method is used as the keyword name.
- The `Argument` annotation is also optional. If a method is marked as a keyword, then all arguments are automatically used as keyword arguments. You can use the `Argument` annotation to specify a different name for the keyword argument, for example *UserName* instead of *userName*.

Integrating Silk Test with Silk Central

Integrate Silk Test and Silk Central to enable collaboration between technical and less-technical users.

When Silk Test and Silk Central are integrated and a library with the same name as the active Silk Test project exists in Silk Central, the **Keywords** view under **Silk4J** > **Show Keywords View** displays all keywords from the Silk Central library in addition to any keywords defined in the active Silk Test project.

Note: The Silk Central connection information is separately stored for every Silk Test user, which means every Silk Test user that wants to work with keywords and keyword sequences from Silk Central must integrate Silk Test with Silk Central.

Integrating Silk Test with Silk Central provides you with the following advantages:

- Test management and execution is handled by Silk Central.
- Keywords are stored in the Silk Central database (upload library) and are available to all projects in Silk Central.
- Manual tests can be directly automated in Silk Central and the created keyword-driven tests can be executed in Silk Test from Silk Central.

Note: In Silk Test, you can edit and execute keyword-driven tests that are located in Silk Test, and you can execute keyword-driven tests that are stored in Silk Central. To edit a keyword-driven test, which is stored in Silk Central, open the keyword-driven test in the **Keyword-Driven Test Editor** and click **Edit**.

The following steps show how you can integrate Silk4J with Silk Central. The steps for Silk4NET and Silk Test Workbench are similar. For additional information on performing keyword-driven testing with a specific Silk Test client, refer to the documentation of the Silk Test client.

1. From the menu, select **Silk4J** > **Silk Central Configuration**. The **Preferences** dialog box opens.
2. Type the URL of your Silk Central server into the **URL** field.

   For example, if the Silk Central server name is *sctm-server*, and the port for Silk Central is 13450, type `http://sctm-server:13450`.
3. Type a valid user name and password into the corresponding fields.
4. Click **Verify** to verify if Silk Test can access the Silk Central server with the specified user.
5. Click **OK**.

Uploading a Keyword Library to Silk Central

To work with Silk Central, ensure that you have configured a valid Silk Central location. For additional information, see *Integrating Silk Test with Silk Central*.

To automate manual tests in Silk Central, upload keywords that you have implemented in a Silk Test project as a keyword library to Silk Central, where you can then use the keywords to automate manual tests.

1. In Silk Test, select the project in which the keyword-driven tests reside.
2. Ensure that a library with the same name exists in Silk Central (**Tests** > **Libraries**).
3. In the toolbar, click **Upload Keyword Library**.
4. *Optional:* Provide a description of the changes to the keyword library.
5. *Optional:* Click **Configure** to configure the connection to Silk Central.
6. *Optional:* To see which libraries are available in the connected Silk Central instance, click on the link.
7. Click **Upload**.

   Caution: If the keyword library in Silk Central is already assigned to a different automation tool or another Silk Test client, you are asked if you really want to change the type of the keyword library. Upload the library only if you are sure that you want to change the type.

Silk Test creates a keyword library out of all the keywords that are implemented in the project. Then Silk Test saves the keyword library with the name `library.zip` into the output folder of the project. The library is validated for consistency, and any changes which might break existing tests in Silk Central are listed in the **Upload Keyword Library to Silk Central** dialog box. Finally, Silk Test uploads the library to Silk Central. You can now use the keywords in Silk Central. Any keyword-driven tests in Silk Central, which use the keywords that are included in the keyword library, automatically use the current implementation of the keywords.

Uploading a keyword library from a project that was created in Silk Test 15.5

To upload keyword libraries from Silk Test projects that were created with Silk Test 15.5, you need to edit the `build.xml` file of the project.

1. In the **Package Explorer**, expand the folder of the project from which you want to upload the keyword library.
2. Open the `build.xml` file.

3. Add the keyword assets directory of the project to the JAR build step of the *compile* target:

```
<fileset dir="Keyword Assets" includes="**/*.kwd"
erroronmissingdir="false" />
```

4. Add the following target for the keyword library:

```
<target name="build.keyword.library" depends="compile">
  <java
classname="com.borland.silk.kwd.library.docbuilder.DocBuilder"
 fork="true">
    <classpath refid="project.classpath" />

    <arg value="AutoQuote Silk4J Library" />
    <arg value="${output}" />
    <arg value="${output}/library.zip" />
  </java>
</target>
```

The new `build.xml` file should look like the following:

```
<?xml version="1.0" encoding="UTF-8"?>
<project name="AutoQuote" default="compile">

  <property name="src" value="src" />
  <property name="bin" value="build" />
  <property name="output" value="output" />
  <property name="lib" value="lib" />
  <property name="buildlib" value="buildlib" />

  <path id="project.classpath">
    <fileset dir="${lib}" includes="*.jar"
excludes="*source*" />
    <fileset dir="${buildlib}" includes="*.jar"
excludes="*source*" />
  </path>

  <target name="clean">
    <delete dir="${output}" />
  </target>

  <target name="compile" depends="clean">
    <mkdir dir="${output}" />

    <delete dir="${bin}" />
    <mkdir dir="${bin}" />

    <componentdef name="ecj"
classname="org.eclipse.jdt.core.JDTCompilerAdapter"
classpathref="project.classpath" />
    <javac srcdir="${src}" destdir="${bin}" debug="true"
source="1.7" target="1.7" encoding="utf-8"
includeantruntime="false">
      <classpath refid="project.classpath" />
      <ecj />
    </javac>

    <jar destfile="${output}/tests.jar" >
      <fileset dir="${bin}" includes="**/*.class" />
      <fileset dir="${src}" includes="**/*" excludes="**/
*.java" />
      <fileset dir="Object Maps" includes="**/*.objectmap"
erroronmissingdir="false" />
      <fileset dir="Image Assets" includes="**/*.imageasset"
erroronmissingdir="false" />
      <fileset dir="Verifications" includes="**/*.verification"
```

```
erroronmissingdir="false" />
      <fileset dir="Keyword Assets" includes="**/*.kwd"
erroronmissingdir="false" />
    </jar>

    <copy todir="${output}" overwrite="true">
      <fileset dir="${lib}" includes="*.jar"
excludes="*source*" />
    </copy>
    <delete dir="${bin}" />
  </target>

  <target name="build.keyword.library" depends="compile">
    <java
classname="com.borland.silk.kwd.library.docbuilder.DocBuilder"
fork="true">
      <classpath refid="project.classpath" />

      <arg value="AutoQuote Silk4J Library" />
      <arg value="${output}" />
      <arg value="${output}/library.zip" />
     </java>
  </target>
</project>
```

Uploading a Keyword Library to Silk Central from the Command Line

Upload an external keyword library to Silk Central from a Java-based command line to integrate Silk Central and your keyword-driven tests into your continuous integration build system, for example Jenkins.

To upload your keyword library to Silk Central from a Java-based command line:

1. Select **Help** > **Tools** in Silk Central and download the **Java Keyword Library Tool**.
2. Call the command line tool that is contained in the downloaded `jar` file with the following arguments:
   - `java`
   - `-jar com.borland.silk.keyworddriven.jar`
   - `-upload`
   - `Library` name of the library in Silk Central to be updated, or created if it does not yet exist.
   - `Package` name of the library package (`zip` archive) to be uploaded.
   - `Hostname:port` of the Silk Central front-end server.
   - `Username` of the Silk Central user.
   - `Password` of the Silk Central user.
   - `Update information`, describing the changes that were applied to the library, in quotes.
   - `[-allowUsedKeywordDeletion]`, an optional flag to allow the deletion of keywords that are used in a test or keyword sequence. By default, an error is raised if used keywords are attempted to be deleted.

   The following example outlines the command line to upload a library to Silk Central:

```
java -jar com.borland.silk.keyworddriven.jar -upload
"My library" "./output/library.zip" silkcentral:19120 scLogin
scPassword "Build xy: Implemented missing keywords"
```

Creating a Keyword-Driven Test by Automating a Manual Test

To convert a manual test to a keyword-driven test in Silk Central:

1. In the menu, click **Tests** > **Details View** .
2. Right-click the manual test in the **Tests** tree and select **Automate with...** > **Keyword-Driven Test** from the context menu. The **Keyword-Driven Test Properties** dialog box appears.
3. Select the **Library** which contains the keywords that you want to use.

   The library is required to store the set of keywords that the keyword-driven test uses.
4. Click **Finish**.

Automating a manual test as keyword-driven test generates the following:

- The manual test is converted to an automated keyword-driven test, containing a draft keyword for each test step. These keywords are added to the library that has been selected for this test.
- If the test contains calls to shared steps that already exist as keyword sequences, these keyword sequences are referenced.
- If the test contains calls to shared steps that don't exist as keyword sequences, these shared steps objects are referenced and enhanced to also be keyword sequences, containing a draft keyword for each test step. This also applies to nested shared steps (shared steps that reference other shared steps). If the hierarchy of nested shared steps is more than 30 levels deep, draft keywords for calls to shared steps below that level are created.
- If the test contains calls to shared steps that are in a different library, draft keywords for these steps are created and referenced.

If your test steps contain parameters that you want to access in the generated keywords, the **Action Description** of a test step must use the correct syntax. Parameters use the syntax `${<name>}`, for example `${username}`. When converting a manual test to a keyword-driven test, these parameters are automatically added to the generated keyword.

Creating a Keyword-Driven Test in Silk Central

1. In the menu, click **Tests** > **Details View** .
2. Select a container or folder node in the **Tests** tree where you want to insert a new test, or select the test that you want to edit.
3. Click  (**New Child Test**) on the toolbar, or click  (**Edit**).

   The **New Test**/**Edit Test** dialog box appears.
4. Type a name and description for the test.

    **Note:** Silk Central supports HTML formatting and cutting and pasting of HTML content for **Description** fields.
5. Select the test type **Keyword-Driven Test** from the **Type** list.
6. Select the **Library** which contains the keywords that you want to use.

   The library is required to store the set of keywords that the keyword-driven test uses.
7. Click **Finish**.

Managing Keywords in a Test in Silk Central

Tests > **Details View** > **<Test>** > **Keywords**

The **Keywords** page enables you to manage the keywords of the selected keyword-driven test. The following actions are possible:

Task	Steps
Opening a test or keyword sequence in Silk Test	Click **Open with Silk Test** to open the selected test or keyword sequence in Silk Test.
Adding a keyword	1. Click **New Keyword** at the bottom of the keywords list, or right-click a keyword and select **Insert Keyword Above** from the context menu.  **Note:** You can let Silk Test recommend keywords based on their usage. Toggle the recommendations on or off with **Enable Recommendations** or **Disable Recommendations** in the context menu. For additional information, see *Which Keywords Does Silk Test Recommend?*. 2. Select a keyword from the list of available keywords or type a new name to create a new keyword. 3. Click **Save**. Alternatively, double click an existing keyword in the **All Keywords** pane on the right or drag and drop it.  **Tip:** You can select multiple keywords with **Ctrl+Click**. When dropping them, they will be sorted in the order that you selected them in.
Deleting a keyword	Click  in the **Actions** column of the keyword that you want to delete. Click **Save**.
Changing the order of keywords	Drag and drop a keyword to the desired position. Click **Save**.
Creating a keyword sequence (a keyword consisting of other keywords)	1. Select the keywords that you want to combine in the keywords list. Use **Ctrl+Click** or **Shift+Click** to select multiple keywords. 2. Right-click your selection and click **Combine**. 3. Specify a **Name** and **Description** for the new keyword sequence.
Extracting keywords from a keyword sequence	Right-click a keyword sequence and click **Extract keywords**. The original keyword sequence is then replaced by the keywords that it contained, but it is not removed from the library. Click **Save**.
Defining parameters for a keyword sequence	1. Click **Parameters** above the keywords list. The **Parameters** dialog box appears. 2. Click **Add Parameter**. 3. Specify a **Name** for the new parameter. If the parameter is an outgoing parameter (delivers a value, instead of requiring an input value), check the **Output** checkbox. 4. Click **OK**. 5. Click **Save**.
Editing a draft keyword	1. Click  in the **Actions** column of the draft keyword that you want to edit. 2. Select a **Group** or specify a new group for the keyword.

Task	Steps
	3. Type a **Description** for the keyword. This information is valuable for the engineer who will implement the keyword. 4. Click **OK**. 5. *Optional:* Click into a parameter field to add parameters for the keyword. If the keyword is implemented with Silk Test, these parameters will appear in the generated code stub. 6. Click **Save**.
Searching for a keyword	Use the search field in the **All Keywords** pane on the right to find a specific keyword. When you enter alphanumeric characters, the list is dynamically updated with all existing matches. Tips for searching: • The search is case-insensitive: `doAction` will find `doaction` and `DOAction`. • Enter only capital letters to perform a so-called *CamelCase* search: `ECD` will find `Enter Car Details`, `Enter Contact Details` and `EnterContactDetails`. • Keyword and group names are considered: `test` will find all keywords that contain `test` and all keywords in groups where the group name contains `test`. • ? replaces 0-1 characters: `user?test` will find `userTest` and `usersTest`. • * replaces 0-n characters: `my*keyword` will find `myKeyword`, `myNewKeyword` and `my_other_keyword`. • `<string>.` only searches in group names: `group.` will find all keywords in groups where the group name contains `group`. • `.<string>` only searches in keyword names: `.keyword` will find all keywords that contain `keyword`. • `<string>.<string>` searches for a keyword in a specific group: `group.word` will find `myKeyword` in the group `myGroup`. • Use quotes to search for an exact match: `'Keyword'` will find `Keyword` and `MyKeyword`, but not `keyword`.

Which Keywords Does Silk Test Recommend?

When you add keywords to a keyword-driven test or a keyword sequence in the **Keyword-Driven Test Editor**, Silk Test recommends existing keywords which you might want to use as the next keyword in your test. The recommended keywords are listed on top of the keywords list, and are indicated by a bar graph, with the filled-out portion of the graph corresponding to how much Silk Test recommends the keyword.

Silk Test recommends the keywords based on the following:

- When you add the first keyword to a keyword-driven test or a keyword sequence, Silk Test searches for similar keywords that are used as the first keyword in other keyword-driven tests or keyword sequences. The keywords that are used most frequently are recommended higher.
- When you add additional keywords to a keyword-driven test or a keyword sequence, which already includes other keywords, Silk Test recommends keywords as follows:
    - If there are keywords before the position in the keyword-driven test or the keyword sequence, to which you add a new keyword, Silk Test compares the preceding keywords with keyword

combinations in all other keyword-driven tests and keyword sequences and recommends the keywords that most frequently follow the preceding combination of keywords.
    - If there are no keywords before the position in the keyword-driven test or the keyword sequence, but there are keywords after the current position, then Silk Test compares the succeeding keywords with keyword combinations in all other keyword-driven tests and keyword sequences and recommends the keywords that most frequently precede the succeeding combination of keywords.
- Additionally, Silk Test takes into account how similar the found keywords are. For example, if both the name and group of two keywords match, then Silk Test recommends these keywords higher in comparison to two keywords for which only the name matches.
- If you have established a connection with Silk Central, any keywords included in keyword-driven tests, which belong to the keyword library that corresponds to the current project, are also considered.

Using Parameters with Keywords

A keyword or a keyword sequence can have input and output parameters. This topic describes how you can handle these parameters with Silk Test.

In the **Keyword-Driven Test Editor**, you can view any defined parameters for a keyword or a keyword sequence and you can edit the parameter values.

In the **Keywords** window, you can see which parameters are assigned to a keyword or a keyword sequence when you hover the mouse cursor over the keyword or keyword sequence.

Input parameters for simple keywords

You can define and use input parameters for keywords in the same way as for any other test method.

The following code sample shows how you can define the keyword `setUserDetails` with the two input parameters `userName` and `password`:

```
@Keyword
public void setUserDetails(String userName, String password) {
  ...
}
```

Output parameters for simple keywords

You can define a return value or one or more output parameters for a keyword. You can also use a combination of a return value and one ore more output parameters.

The following code sample shows how you can define the keyword `getText` that returns a string:

```
@Keyword
public String getText() {
  return "text";
}
```

The following code sample shows how you can define the keyword `getUserDetails` with the two output parameters `userName` and `password`:

```
@Keyword
public void getUserDetails(OutParameter userName, OutParameter password) {
  userName.setValue("name");
  password.setValue("password");
}
```

Parameters for keyword sequences

You can define or edit the parameters for a keyword sequence in the **Parameters** dialog box, which you can open if you click **Parameters** in the **Keyword Sequence Editor**.

Example: Keywords with Parameters

This topic provides an example of how you can use keywords with parameters.

As a first step, create a keyword-driven test which contains the keywords that you want to use. You can do this by recording an entire keyword-driven test, or by creating a new keyword-driven test and by adding the keywords in the keyword-driven test editor.

In this example, the keyword-driven test includes the following keywords:

Start application	This is the standard keyword that starts the AUT and sets the base state.
Login	This keyword logs into the AUT with a specific user, identified by a user name and a password.
GetCurrentUser	This keyword returns the name of the user that is currently logged in to the AUT.
AssertEquals	This keyword compares two values.
Logout	This keyword logs the user out from the AUT.

The next step is to add the parameters to the keywords. To do this, open the test scripts of the keywords and add the parameters to the methods.

To add the input parameters `UserName` and `Password` to the keyword `Login`, change

```
@Keyword("Login")
public void login() {
  ...
}
```

to

```
@Keyword("Login")
public void login(String UserName, String Password) {
  ...
}
```

To add the output parameter `UserName` to the keyword `GetCurrentUser`, change

```
@Keyword("GetCurrentUser")
public void getCurrentUser() {
  ...
}
```

to

```
@Keyword("GetCurrentUser")
public void getCurrentUser(OutParameter CurrentUser) {
  ...
}
```

The keyword-driven test in the **Keyword-Driven Test Editor** should look similar to the following:

Now you can specify actual values for the input parameters in the **Keyword-Driven Test Editor**. To retrieve the value of the output parameter `UserName` of the keyword `GetCurrentUser`, provide a variable, for example *${current user}*. You can then pass the value that is stored in the variable to subsequent keywords.

		Keyword	Parameters	
1		Start application		
2		Login	*UserName*	*Password*
3		GetCurrentUser	${current user}	
4		AssertEquals	John Smith	${current user}
5		Logout		

Creating a Keyword-Driven Test in Silk Test

Before you can create a keyword-driven test in Silk Test, you have to select a project.

Use the **Keyword-Driven Test Editor** to combine new keywords and existing keywords into new keyword-driven tests. New keywords need to be implemented as automated test methods in a later step.

This topic shows you how to create a keyword-driven test in Silk4J. The steps for Silk4NET and Silk Test Workbench are similar. For additional information on performing keyword-driven testing with a specific Silk Test client, refer to the documentation of the Silk Test client.

1. Click **Silk4J** > **New Keyword-Driven Test**. The **New Keyword-Driven Test** dialog box opens.
2. Type a name for the new test into the **Name** field.
3. Select the project in which the new test should be included.

   By default, if a project is active, the new test is created in the active project.

   Note: To optimally use the functionality that Silk Test provides, create an individual project for each application that you want to test, except when testing multiple applications in the same test.
4. Click **Finish**.
5. Click **No** to create an empty keyword-driven test. The **Keyword-Driven Test Editor** opens.
6. Perform one of the following actions:
   - To add a new keyword, type a name for the keyword into the **New Keyword** field.
   - To add an existing keyword, expand the list and select the keyword that you want to add.
7. Press `Enter`.
8. Repeat the previous two steps until the test includes all the keywords that you want to execute.
9. Click **Save**.

Continue with implementing the keywords or with executing the test, if all keywords are already implemented.

Recording a Keyword-Driven Test in Silk Test

Before you can create a keyword-driven test in Silk Test, you have to select a project.

Note: You cannot record keyword-driven tests on Mozilla Firefox versions prior to version 41.

To record a single keyword, see *Recording a Keyword*.

To record a new keyword-driven test:

1. Click **Silk4J** > **New Keyword-Driven Test**. The **New Keyword-Driven Test** dialog box opens.
2. Type a name for the new test into the **Name** field.
3. Select the project in which the new test should be included.

   By default, if a project is active, the new test is created in the active project.

   Note: To optimally use the functionality that Silk Test provides, create an individual project for each application that you want to test, except when testing multiple applications in the same test.
4. Click **Finish**.
5. Click **Yes** to start recording the keyword-driven test.
6. If you have set an application configuration for the current project and you are testing a Web application, the **Select Browser** dialog box opens. Select the browser on which you want to record the keyword.
7. Depending on the dialog that is open, perform one of the following:
   - In the **Select Application** dialog box, click **OK**.
   - In the **Select Browser** dialog box, click **Record**.
8. In the application under test, perform the actions that you want to include in the first keyword.

   For information about the actions available during recording, refer to the documentation of the Silk Test client .
9. To specify a name for the keyword, hover the mouse cursor over the keyword name in the **Recording** window and click **Edit**.

   Note: Silk Test automatically adds the keyword *Start application* to the start of the keyword-driven test. In this keyword, the applications base state is executed to enable the test to replay correctly. For additional information on the base state, refer to the documentation of the Silk Test client.
10. Type a name for the keyword into the **Keyword name** field.
11. Click **OK**.
12. To record the actions for the next keyword, type a name for the new keyword into the **New keyword name** field and click **Add**. Silk Test records any new actions into the new keyword.
13. Create new keywords and record the actions for the keywords until you have recorded the entire keyword-driven test.
14. Click **Stop**. The **Record Complete** dialog box opens.

Silk Test creates the new keyword-driven test with all recorded keywords.

Setting the Base State for a Keyword-Driven Test in Silk Test

When you execute a keyword-driven test with Silk Test and the keyword-driven test calls a base state keyword, Silk Test starts your AUT from the base state.

During the recording of a keyword-driven test, Silk Test searches in the current project for a base state keyword, which is a keyword for which the `isBaseState` property is set to *true*.

- If a base state keyword exists in the current project, Silk Test inserts this keyword as the first keyword of the keyword-driven test.
- If there is no base state keyword in the project, Silk Test creates a new base state keyword with the name *Start application* and inserts it as the first keyword of the keyword-driven test.

To manually mark a keyword as a base state keyword, add the `isBaseState` property to the `Keyword` annotation, and set the value of the property to *true*:

```
@Keyword(value = "Start application", isBaseState = true)
public void start_application() {
  // Base state implementation
}
```

Implementing a Keyword in Silk Test

Before implementing a keyword, define the keyword as part of a keyword-driven test.

The following steps show how you can implement a keyword in Silk4J. The steps for Silk4NET and Silk Test Workbench are similar. For additional information on performing keyword-driven testing with a specific Silk Test client, refer to the documentation of the Silk Test client.

To implement a keyword for reuse in keyword-driven tests:

1. Open a keyword-driven test that includes the keyword that you want to implement.
2. In the **Keyword-Driven Test Editor**, click **Implement Keyword** to the left of the keyword that you want to implement. The **Select Keyword Location** dialog box opens.
3. Click **Select** to select the package and class to which you want to add the keyword implementation.
4. *Optional:* Define the package name for the new keyword implementation in the **Package** field.
5. Define the class name for the new keyword implementation in the **Class** field.
6. Click **OK**.
7. Perform one of the following actions:
   - To record the keyword, click **Yes**.
   - To create an empty keyword method, click **No**.
8. If you have set an application configuration for the current project and you are testing a Web application, the **Select Browser** dialog box opens. Select the browser on which you want to record the keyword.
9. Click **Record**.

   For additional information on recording, refer to the documentation of the Silk Test client.

If an implemented keyword is displayed as not implemented in the **Keywords** window, check **Project** > **Build Automatically** in the Eclipse menu.

Implementing Silk Central Keywords in Silk Test

Before implementing Silk Central keywords, define the keywords as part of a keyword-driven test in Silk Central.

The following steps show how you can implement Silk Central keywords in Silk4J. The steps for Silk4NET and Silk Test Workbench are similar. For additional information on performing keyword-driven testing with a specific Silk Test client, refer to the documentation of the Silk Test client.

To implement a Silk Central keyword in Silk Test:

1. Create a project in Silk Test with the same name as the keyword library in Silk Central, which includes the keyword-driven test.
2. If the keyword library in Silk Central has no type assigned, click **Silk4J** > **Upload Keyword Library** to set the library type.
3. *Optional:* To implement a specific keyword in Silk Test from Silk Central, open the **Keywords** tab of the library in Silk Central and click **Implement with Silk Test** in the **Actions** column of the keyword.

4. In the Silk Test menu, click **Silk4J** > **Show Keywords View**.
5. In the **Keywords** view, double-click the keyword-driven test.

   To update the **Keywords** view with any changes from Silk Central, click **Refresh**.
6. In the toolbar, click **Record Actions**.
7. If you have set an application configuration for the current project and you are testing a Web application, the **Select Browser** dialog box opens. Select the browser on which you want to record the keyword.
8. Click **Record**.

   For additional information on recording, refer to the documentation of the Silk Test client.
9. Record the actions for the first unimplemented keyword.
10. When you have recorded all the actions for the current keyword, click **Next Keyword**.
11. To switch between keywords in the **Recording** window, click **Previous Keyword** and **Next Keyword**.
12. Click **Stop**. The **Record Complete** dialog box opens.

Note: You cannot delete keywords or change the sequence of the keywords in a keyword-driven test from Silk Central, as these tests are read only in Silk Test.

If an implemented keyword is displayed as not implemented in the **Keywords** window, check **Project** > **Build Automatically** in the Eclipse menu.

Recording a Keyword in Silk Test

You can only record actions for a keyword that already exists in a keyword-driven test, not for a keyword that is completely new. To record a new keyword-driven test, see *Recording a Keyword-Driven Test*.

To record the actions for a new keyword:

1. Open a keyword-driven test that includes the keyword that you want to record.
2. In the **Keyword-Driven Test Editor**, click **Implement Keyword** to the left of the keyword that you want to implement. The **Select Keyword Location** dialog box opens.
3. Click **Select** to select the package and class to which you want to add the keyword implementation.
4. *Optional:* Define the package name for the new keyword implementation in the **Package** field.
5. Define the class name for the new keyword implementation in the **Class** field.
6. Click **OK**.
7. If you have set an application configuration for the current project and you are testing a Web application, the **Select Browser** dialog box opens. Select the browser on which you want to record the keyword.
8. Click **Record**. The **Recording** window opens and Silk Test starts recording the actions for the keyword.
9. In the application under test, perform the actions that you want to test.

   For information about the actions available during recording, refer to the documentation of the Silk Test client .
10. Click **Stop**. The **Record Complete** dialog box opens.

The recorded actions are displayed in the context of the defined class.

Editing a Keyword-Driven Test

Note: In Silk Test, you can edit and execute keyword-driven tests that are located in Silk Test, and you can execute keyword-driven tests that are stored in Silk Central. To edit a keyword-driven test, which is stored in Silk Central, open the keyword-driven test in the **Keyword-Driven Test Editor** and click **Edit**.

To edit a keyword-driven test:

1. Open the keyword-driven test in the **Keyword-Driven Test Editor**.
   a) Under **Tests** > **Details View** in the Silk Central menu, expand the project in which the keyword-driven test resides.
   b) Select the keyword-driven test in the **Tests** tree.
   c) Select the **Keywords** tab.
2. To add a new keyword to the keyword-driven test:
   a) Click into the **New Keyword** field.
   b) Type a name for the new keyword.
   c) Press `Enter`.
3. To edit an existing keyword, click **Open Keyword** to the left of the keyword.

   Note: Silk Central has the ownership of any keyword that has been created in Silk Central, which means any changes that you make to such keywords are saved in Silk Central, not in Silk Test.
4. To remove the keyword from the keyword-driven test, click **Delete Keyword** to the left of the keyword.

   The keyword is still available in the **Keywords** window and you can re-add it to the keyword-driven test at any time.
5. To save your changes, click **Save**.

Combining Keywords into Keyword Sequences

Use the **Keyword-Driven Test Editor** to combine keywords, which you want to execute sequentially in multiple keyword-driven tests, into a keyword sequence.

1. Open the keyword-driven test that includes the keywords that you want to combine.
2. In the **Keyword-Driven Test Editor**, press and hold down the `Ctrl` key and then click the keywords that you want to combine.
3. Right-click on the selection and click **Combine**. The **Combine Keywords** dialog box opens.
4. Type a name for the new keyword sequence into the **Name** field.
5. *Optional:* Type a description for the new keyword sequence into the **Description** field.
6. Click **Combine**.

The new keyword sequence opens and is also displayed in the **Keywords** window. You can use the keyword sequence in keyword-driven tests.

Note: Like any other keyword, you cannot execute a keyword sequence on its own, but only as part of a keyword-driven test.

Replaying Keyword-Driven Tests from Eclipse

The following steps show how you can replay a keyword-driven test in Silk4J. The steps for Silk4NET and Silk Test Workbench are similar. For additional information on performing keyword-driven testing with a specific Silk Test client, refer to the documentation of the Silk Test client.

1. In the **Project Explorer**, navigate to the keyword-driven test asset that you want to replay.
2. Richt-click the asset name.
3. Choose **Run As** > **Keyword-Driven Test**.
4. *Optional:* To open the **Run Configurations** dialog box, choose **Run As** > **Run Configurations**.
5. *Optional:* In the **Run Configurations** dialog box, you can select a different test or project.
6. *Optional:* In the **Global variables** grid of the **Run Configurations** dialog box, you can set the values of any variables that are used for the execution of the keyword-driven test. These values are used whenever you execute the keyword-driven test asset.

a) Type a **Variable Name** and a **Value** for the variable into the corresponding fields.
b) Type **Enter** to add a new line to the grid.
c) Repeat the previous two steps until you have set the values of all the global variables that you want to use.

When executing keyword-driven tests that are part of an automation framework and that are managed in a test management tool, for example Silk Central, you can also add a new *.properties* file to a project to set the values of global variables for the entire project. For additional information, see *Replaying a Keyword-Driven Test with Specific Variables*.

7. *Optional:* To close the **Run Configurations** dialog box and to start the execution of the keyword-driven test asset, click **Run**.
8. If you are testing a Web application, the **Select Browser** dialog box opens. Select the browser and click **Run**.

   Note: If multiple applications are configured for the current , the **Select Browser** dialog box is not displayed.
9. *Optional:* If necessary, you can press both **Shift** keys at the same time to stop the execution of the test.
10. When the test execution is complete, the **Playback Complete** dialog box opens. Click **Explore Results** to review the TrueLog for the completed test.

Replaying Silk Test Tests from Silk Central

To access Silk Test tests from Silk Central, you need to store the Silk Test tests in a JAR file in a repository that Silk Central can access through a source control profile.

To replay functional tests in Silk Test from Silk Central, for example keyword-driven tests:

1. In Silk Central, create a project from which the Silk Test tests will be executed.
2. Under **Tests** > **Details View**, create a new test container for the new project.

   For additional information about Silk Central, refer to the *Silk Central Help*.

   The test container is required to specify the source control profile for the Silk Test tests.
   a) In the **Tests** tree, right-click on the node below which you want to add the new test container.
   b) Click **New Test Container**. The **New Test Container** dialog box opens.
   c) Type a name for the new test container into the **Name** field.
   For example, type `Keyword-Driven Tests`
   d) In the **Source control profile** field, select the source control profile in which the JAR file, which contains the Silk Test tests, is located.
   e) Click **OK**.
3. Create a new JUnit test in the new test container.

   For additional information about Silk Central, refer to the *Silk Central Help*.
   a) In the **Test class** field of the **JUnit Test Properties** dialog box, type the name of the test class.
   Specify the fully-qualified name of the test suite class. For additional information, see *Replaying Keyword-Driven Tests from the Command Line*.
   b) In the **Classpath** field, specify the name of the JAR file that contains the tests.
   c) For keyword-driven testing, also specify the paths to the following files, separated by semicolons.
      - `com.borland.silk.keyworddriven.engine.jar`
      - `com.borland.silk.keyworddriven.jar`
      - `silktest-jtf-nodeps.jar`

   These files are located in the Silk Test installation directory. For example, the **Classpath** field for the keyword-driven tests in the JAR file `tests.jar` might look like the following:

   ```
   tests.jar;C:\Program Files
   (x86)\Silk\SilkTest\ng\KeywordDrivenTesting
   ```

```
\com.borland.silk.keyworddriven.engine.jar;C:\Program Files
(x86)\Silk\SilkTest\ng\KeywordDrivenTesting
\com.borland.silk.keyworddriven.jar;C:\Program Files
(x86)\Silk\SilkTest\ng\JTF\silktest-jtf-nodeps.jar
```

4. Click **Finish**.
5. Execute the tests.

   For additional information about executing tests in Silk Central, refer to the *Silk Central Help*.

Replaying Keyword-Driven Tests from the Command Line

You must update the PATH variable to reference your JDK location before performing this task. For details, reference the Sun documentation at: *http://java.sun.com/j2se/1.5.0/install-windows.html*.

To replay keyword-driven tests from the command line, for example when replaying the tests from a CI server, use the `KeywordTestSuite` class.

1. To execute a keyword-driven test from the command line, create a JUnit test suite with the `@KeywordTests` annotation. For example, if you want to execute the keyword-driven test *My Keyword-Driven Test*, create the JUnit test suite `MyTestSuite` as follows:

   ```
   @RunWith(KeywordTestSuite.class)
   @KeywordTests({ "My Keyword-Driven Test" })
   public class MyTestSuite {

   }
   ```

2. Include the following in the CLASSPATH:

   - `junit.jar`.
   - The `org.hamcrest.core` JAR file.
   - `silktest-jtf-nodeps.jar`.
   - `com.borland.silk.keyworddriven.engine.jar`.
   - The JAR of folder that contains your keyword-driven tests.

   ```
   set CLASSPATH=<eclipse_install_directory>\plugins
   \org.junit_4.11.0.v201303080030\junit.jar;<eclipse_install_directory>
   \plugins\org.hamcrest.core_1.3.0.v201303031735.jar;%OPEN_AGENT_HOME%\JTF
   \silktest-jtf-nodeps.jar;%OPEN_AGENT_HOME%\KeywordDrivenTesting
   \com.borland.silk.keyworddriven.engine.jar;C:\myTests.jar
   ```

3. *Optional:* Add a new *.properties* file to the project to set the values of any variables that are used for the execution of the keyword-driven test.

   For additional information, see *Replaying a Keyword-Driven Test with Specific Variables*.

4. Run the JUnit test method by typing `java org.junit.runner.JUnitCore <Name>`, where the *Name* is the name of the JUnit test suite that you have created in the first step.

   Note: For troubleshooting information, reference the JUnit documentation at: *http://junit.sourceforge.net/doc/faq/faq.htm#running_1*.

   Example

   For example, to run the two keyword driven tests *My Keyword Driven Test 1* and *My Keyword Driven Test 2*, create the following class:

   ```
   package demo;

   import org.junit.runner.RunWith;

   import com.borland.silktest.jtf.keyworddriven.KeywordTestSuite;
   import com.borland.silktest.jtf.keyworddriven.KeywordTests;
   ```

```
@RunWith(KeywordTestSuite.class)
@KeywordTests({ "My Keyword Driven Test 1", "My Keyword Driven
Test 2" })
public class MyTestSuite {

}
```

To run the class from the command line, type the following:

```
java org.junit.runner.JUnitCore demo.KeywordTestSuite
```

To run the class from the command line, using global variables stored in the file *c:\temp\globalvariables.properties*, type the following:

```
java -Dsilk.keyworddriven.engine.globalVariablesFile=c:\temp
\globalvariables.properties org.junit.runner.JUnitCore
demo.KeywordTestSuite
```

For additional information, see *Replaying a Keyword-Driven Test with Specific Variables*.

Replaying Keyword-Driven Tests with Apache Ant

To perform the actions described in this topic, ensure that Apache Ant is installed on your machine.

To replay keyword-driven tests with Apache Ant, for example to generate HTML reports of the test runs, use the `KeywordTestSuite` class.

1. To execute a keyword-driven test with Apache Ant, create a JUnit test suite with the `@KeywordTests` annotation. For example, if you want to execute the keyword-driven test *My Keyword-Driven Test*, create the JUnit test suite `MyTestSuite` as follows:

   ```
   @RunWith(KeywordTestSuite.class)
   @KeywordTests({ "My Keyword-Driven Test" })
   public class MyTestSuite {

   }
   ```

2. Open the `build.xml` file of the Silk Test project, which includes the keyword-driven test.
3. To execute the keyword-driven test, add the following target to the `build.xml` file:

   ```
   <target name="runTests" depends="compile">
     <mkdir dir="./reports"/>
     <junit printsummary="true" showoutput="true" fork="true">
       <classpath>
         <fileset dir="${output}">
           <include name="**/*.jar" />
         </fileset>
         <fileset dir="${buildlib}">
           <include name="**/*.jar" />
         </fileset>
         <fileset dir="C:/Program Files (x86)/Silk/SilkTest/ng/
   KeywordDrivenTesting">
           <include name="**/*.jar" />
         </fileset>
       </classpath>

       <test name="MyTestSuite" todir="./reports"/>
     </junit>
   </target>
   ```

   For additional information about the `JUnit` task, see *https://ant.apache.org/manual/Tasks/junit.html*.

4. *Optional:* To create XML reports for all tests, add the following code to the target:

   ```
   <formatter type="xml" />
   ```

5. *Optional:* To create HTML reports out of the XML reports, add the following code to the target:

```
<junitreport todir="./reports">
  <fileset dir="./reports">
    <include name="TEST-*.xml" />
  </fileset>
  <report format="noframes" todir="./report/html" />
</junitreport>
```

   For additional information about the `JUnitReport` task, see *https://ant.apache.org/manual/Tasks/junitreport.html*.

   The complete target should now look like the following:

```
<target name="runTests" depends="compile">
  <mkdir dir="./reports"/>
  <junit printsummary="true" showoutput="true" fork="true">
    <classpath>
      <fileset dir="${output}">
        <include name="**/*.jar" />
      </fileset>
      <fileset dir="${buildlib}">
        <include name="**/*.jar" />
      </fileset>
      <fileset dir="C:/Program Files (x86)/Silk/SilkTest/ng/
KeywordDrivenTesting">
        <include name="**/*.jar" />
      </fileset>
    </classpath>

    <formatter type="xml" />

    <test name="MyTestSuite" todir="./reports"/>
  </junit>
  <junitreport todir="./reports">
    <fileset dir="./reports">
      <include name="TEST-*.xml" />
    </fileset>
    <report format="noframes" todir="./report/html" />
  </junitreport>
</target>
```

6. To run the tests from Eclipse, perform the following actions:
   a) In the **Package Explorer**, right-click the `build.xml` file.
   b) Select **Run As** > **Ant Build ...**.
   c) In the **Targets** tab of the **Edit Configuration** dialog box, check **runTests**.
   d) Click **Run**.

You can also execute the tests from the command line or from a CI server. For additional information, see *https://ant.apache.org/manual/running.html* and *Replaying Tests from a Continuous Integration Server* in the *Silk Test Help*.

Replaying a Keyword-Driven Test with Specific Variables

Before you can set the values of variables for the execution of a keyword-driven test, you have to create the project.

To set the values of global variables for all executions of a keyword-driven test asset, where these executions are triggered by you, use the **Global variables** grid of the **Run Configurations** dialog box. For additional information, see *Replaying Keyword-Driven Tests from Eclipse*.

When executing keyword-driven tests that are part of an automation framework and that are managed in a test management tool, for example Silk Central, you can set the values of any variables that are used for

the execution of the keyword-driven test in Silk Test. To set the values of global variables for the entire project, which means that these values are used whenever a Silk Test user executes the keyword-driven test assets in this project, perform the following actions:

1. In the **Package Explorer**, expand the project which includes the keyword-driven tests that you want to execute based on the variables.
2. Right-click the folder **Keyword Driven Tests** of the project and select **New** > **File**. The **New File** dialog box opens.
3. Type `globalvariables.properties` into the **File name** field.
4. Click **Finish**. The new properties file opens.
5. Add new lines to the file to specify the variables.

   The format for a new variable is:

   ```
   name=value
   ```

   For example, to specify the two variables *user* and *password*, type the following:

   ```
   user=John
   password=john5673
   ```

   For information about the format of a properties file and how you can enter UNICODE characters, for example a space, see *Properties File Format*.
6. Save the `globalvariables.properties` file.

Whenever a keyword-driven test in the project is executed from Silk Test, the variables are used.

Grouping Keywords

To better structure the keywords in a library, you can group them.

This topic shows how you can add a keyword to a specific group in Silk4J. The steps for Silk4NET and Silk Test Workbench are similar. For additional information on performing keyword-driven testing with a specific Silk Test client, refer to the documentation of the Silk Test client. These group names are also used by Silk Central and your keywords are grouped accordingly.

To add a keyword to a specific group:

1. Open the implementation of the keyword.
   a) Open the project in which the keyword is implemented.
   b) Open the **Keywords** window.
   c) In the **Keywords** window, select the keyword.
   d) Click **Go to implementation**.
2. To add all methods in a class to the keyword group, add the keyword group before the start of the class.

   For example, to add the group calculator to the keywords, type:

   ```
   @KeywordGroup("Calculator")
   ```

In the **Keywords** window, the displayed keyword name now includes the group. For example, the keyword *Addition* in the group *Calculator* is displayed as `Calculator.Addition`.

Index

N

O

P

R

S

T

U

V