Mobile Interface Testing

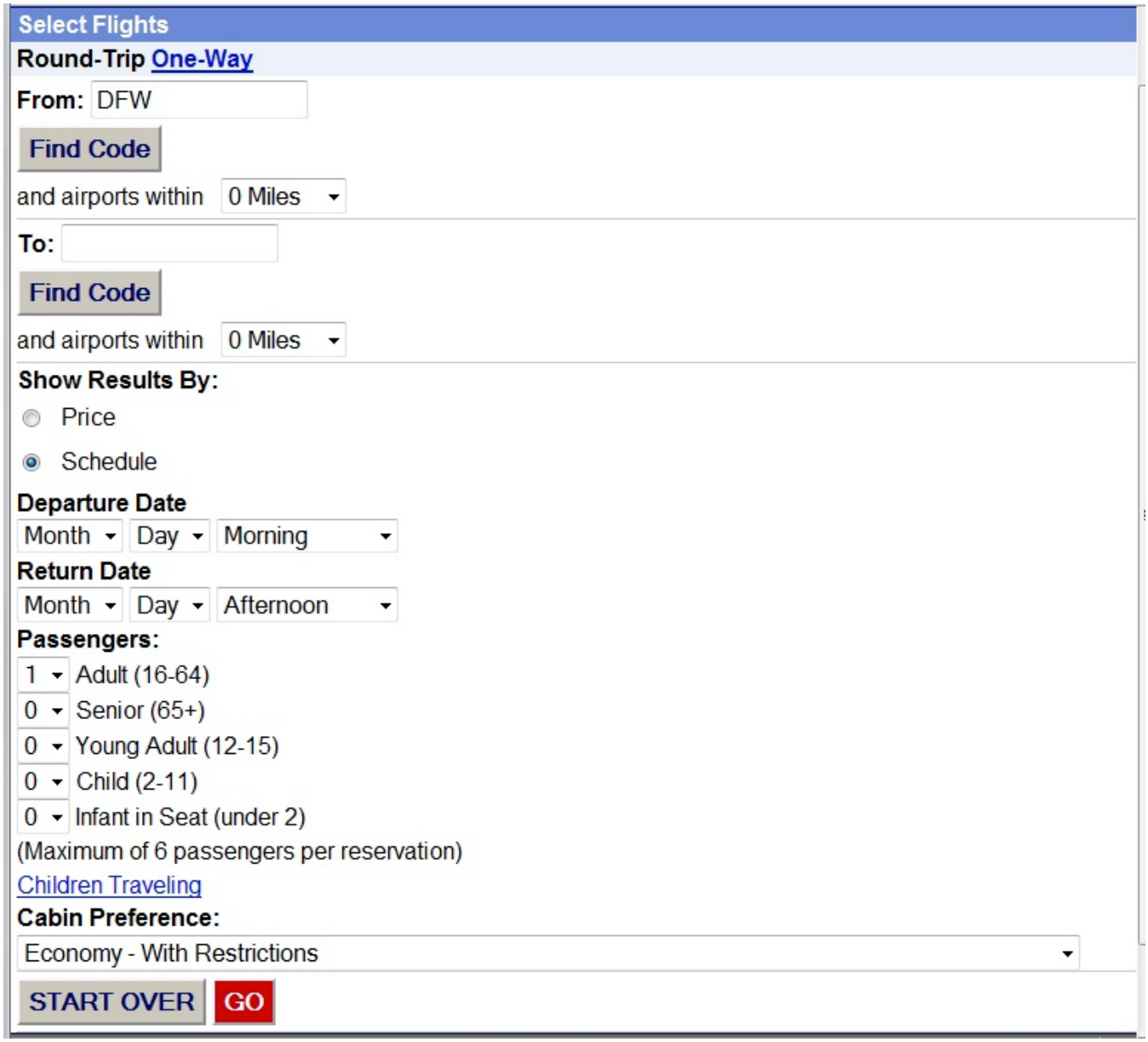
This topic explains how to use SOAtest to record, play back, and validate web pages as they are displayed on specific mobile devices.

In this section:

- About SOAtest's Mobile Interface Testing
- Configuring User Agents
- Setting User Agents
- Tips
- Sample Configurations

About SOAtest's Mobile Interface Testing

You can use SOAtest to record, play back, and validate web pages as they are displayed on specific mobile devices. This is done by having SOAtest pretend to be the desired mobile device (by altering SOAtest's user agent) so that the server sends the desktop browser the appropriate mobile version of the web application. For instance, if SOAtest identifies itself as an Android device, it might access a travel site's web page as follows:



You can emulate any browser by entering the appropriate user agent, but the browser used for playback must be either Firefox or Chrome browser using the legacy engine.

Configuring User Agents

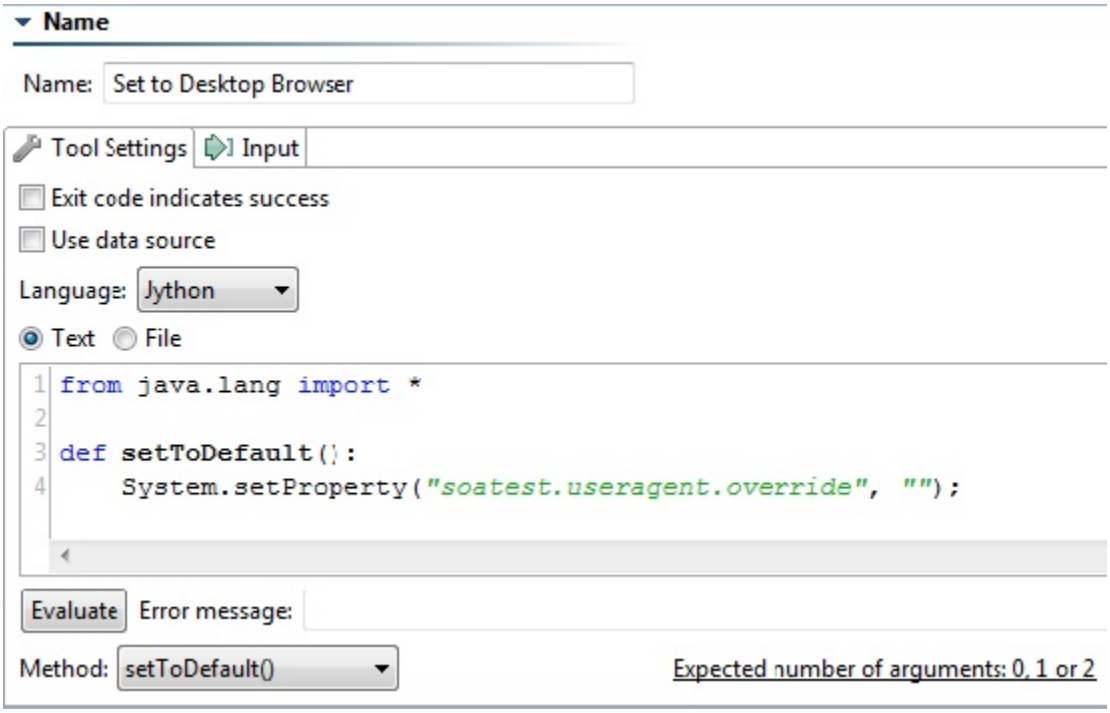
To configure SOAtest to access mobile interfaces:

1. For each mobile interface you want to test, configure one Extension tool to use the appropriate user agent. This is done by setting the system property soatest.useragent.override to the appropriate user agent. See Sample Configurations for details.
2. Configure one Extension tool to clear any mobile device user agent settings. This is done by setting the system property soatest.useragent. override to an empty string. For example, here is a Jython script that clears mobile device user agent settings:

```
from java.lang import *

def setToDefault():

     System.setProperty("soatest.useragent.override", "");
```



Setting User Agents

After you have these Extension tools configured, use them as follows:

- Before performing the desired record/playback actions for a mobile interface, run the Extension tool that sets the appropriate user agent.
- When you want to stop mobile interface recording/playback, run the Extension tool that clears mobile device user agent settings.

Note that when you update the user agent by running an Extension tool, the user agent settings will persist as long as SOAtest is open or until you run an Extension tool that sets it back to default. These settings will not persist when you close SOAtest.

Tips

- You can create a .tst file that sets the user agent for each mobile device, then reference that .tst file across multiple scenarios (see Reusing /Modularizing Test Suites and Creating Reusable (Modular) Test Suites for details on referencing .tst files). This way, you can create the Extension tool once, and reuse it across multiple projects and scenarios. Be sure that the referenced test suite is positioned before the tests invoking the mobile interface.
- If your test scenarios combine device-independent steps with device-specific steps, you might want to create one .tst for the common steps, then reference that .tst in device-specific scenarios. Thus, you might end up with a test suite that starts by referencing a .tst that sets the user agent, then moves to another .tst that steps through device-independent functionality, then finally performs device-specific navigation and/or validations (e.g., checking that a link says "**Download our app for Android**.")
- When you run command line tests, it's best to have each cli session use a single user agent.

    - If each scenario contains a call to set the appropriate user agent (including setting back to the desktop browser), then you can run a set of scenarios that all use a different user agent.
    - However, if you assume that your scenarios vs. a desktop browser don't explicitly set the user agent, then you cannot run them in a cli session with other scenarios that do set the user agent. Otherwise, any desktop scenarios that run after mobile scenarios will use the mobile user agents.
- You can use these functional tests for load testing with Parasoft Load Test as long as you con-figure and validate them in the normal manner (described in Preparing Web Functional Tests for Load Testing).
- When you run load tests with these devices, ensure that each load test uses a single user agent. The option to set the user agent is a global product option that cannot be set on a per-session or per-useragent basis. Thus, if you have a load test with different test scenarios that set different user agents, when one scenario changes the user agent, the user agent will be changed for all the virtual users.

Sample Configurations

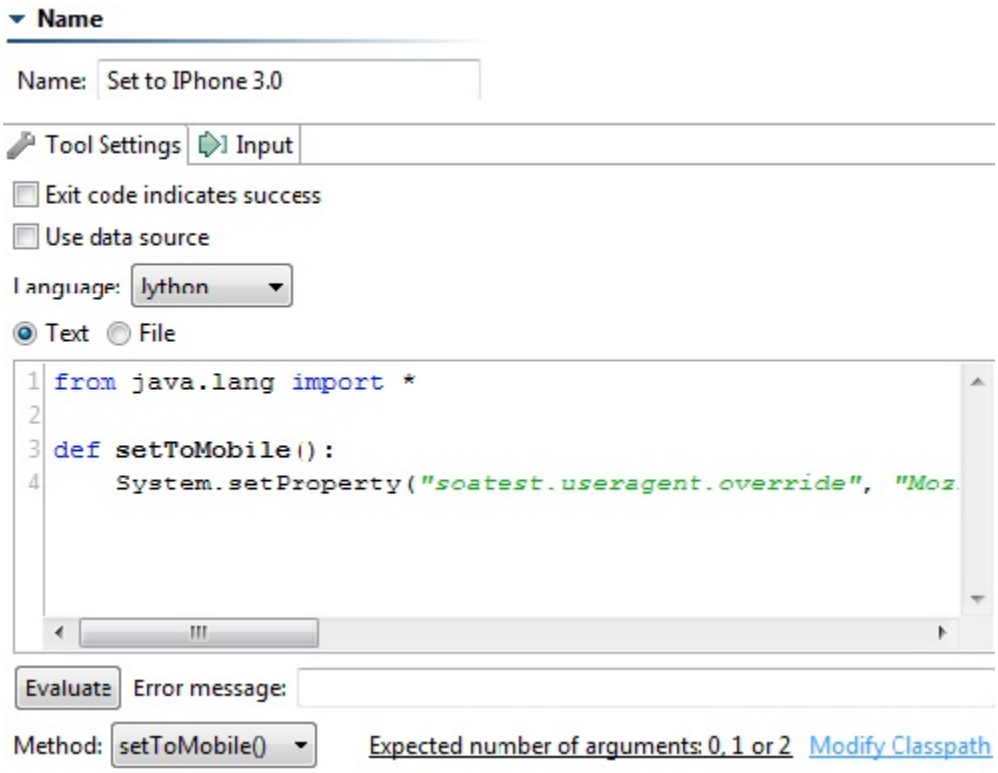
The following are sample Jython scripts for some common mobile devices. You access the interface shown to *any* mobile device; just use the appropriate user agent.

IPhone

```
from java.lang import *

def setToMobile():

    System.setProperty("soatest.useragent.override", "Mozilla/5.0 (iPhone; U; CPU iPhone OS 3_0 like Mac OS X; en-
us) AppleWebKit/528.18 (KHTML, like Gecko) Version/4.0 Mobile/7A341 Safari/ 528.16");
```



iPad

```
from java.lang import *

def setToDefault():

    System.setProperty("soatest.useragent.override", "Mozilla/5.0 (iPad; U; CPU OS 3_2 like Mac OS X; en-us)
AppleWebKit/531.21.10 (KHTML, like Gecko) Version/4.0.4 Mobile/7B334b Safari/ 531.21.10");
```

Android

```
from java.lang import *

def setToDefault():
```

```
    System.setProperty("soatest.useragent.override", "Mozilla/5.0 (Linux; U; Android 2.1-update1; de-de; HTC
Desire 1.19.161.5 Build/ERE27) AppleWebKit/530.17 (KHTML, like Gecko) Version/4.0 Mobile Safari/530.17");
```

Blackberry

```
from java.lang import *

def setToDefault():

    System.setProperty("soatest.useragent.override", "BlackBerry8700/4.1.0 Profile/MIDP-2.0 Configuration/CLDC-
1.1");
```

Windows Phone 7

```
from java.lang import *

def setToDefault():

    System.setProperty("soatest.useragent.override", "Mozilla/4.0 (compatible; MSIE 7.0; Windows Phone OS 7.0;
Trident/3.1; IEMobile/7.0; SAMSUNG; Taylor)");
```