Table of Contents

Sakuli E2E Testing and Monitoring - Documentation

Version

`v1.0.2`

TODO TS update to github pages structure

- Stable version (latest tagged `vx.x.x` documentation) docs stable
- Latest verison (documentation on `master` branch, with corrections) docs latest
- Dev version (current development version on `dev` branch) docs dev

Contributors

At this point we want to thank all contributors, which helped to move this great project by submitting code, writing documentation, or adapting other tools to play well together with Sakuli.

- Tobias Schneck - Sakuli Team / Project Leader, Core Development
- Simon Meggle - Sakuli Team / Project Leader, Monitoring Integration
- Christoph Deppisch - Sakuli Team / Core Development
- Georgi Todorov - Sakuli Team / Core Development
- Lukas Höfer - Sakuli Team / Consultant
- Sven Nierlein
- Philip Griesbacher - Sakuli Go Starter
- Thomas Rothenbacher

(did we forget you? Please poke us):

Valued supporters

Very special thanks to all customers which always nourish this project with new ideas and impulses and make it possible in the first place to give it back to the community as Open Source Software. Thumbs up!

- LIDL Stiftung & Co. KG
- Deutsche Pfandbriefbank AG
- Siemens AG, Global Services Information Technology

Copyright © 2017 ConSol Software GmbH

www.sakuli.org

TODO Links zu

- Installer **->** **Download**
- Maven Decencies
- GH-Pages index.html
- PDF-Docu
- REMOVE Readthedocs state

branch	build state	docs state	docker images state
master	build passing	docs stable	consol/sakuli-ubuntu-xfce version latest 763.5MB 67 layers consol/sakuli-centos-xfce version latest 733.9MB 64 layers consol/sakuli-ubuntu-xfce-java version latest 762.3MB 72 layers consol/sakuli-centos-xfce-java version latest 773.9MB 69 layers
dev	build failing	docs dev docs latest	consol/sakuli-ubuntu-xfce:dev version dev 763.6MB 67 layers consol/sakuli-centos-xfce:dev version dev 735.7MB 64 layers consol/sakuli-ubuntu-xfce-java:dev version dev 827.6MB 72 layers consol/sakuli-centos-xfce-java:dev version dev 841MB 69 layers

Why Sakuli?

There are already a variety of free end2end/automation tools on the market (Sahi, Selenium, WebInject, Sikuli, CasperJS, AutoIT , ...), but each of them has at least one of the following drawbacks:

- **Too specifically**: *pure* web testing tools can only deal with *pure* web content. Applets, Flash, dialogues generated by OS, browser dialogues etc. are invisible and a insurmountable hurdle for such tools.
- **Too generic**: screen-based testing tools "see" everything the user sees. They are the best choice for GUI tests, but inappropriate for web tests, because each browser type has its own understanding of how to render and display a web page.
- **Far from reality**: There are tools to test web applications on protocol level - but to know whether a web application is working or not requires a test from the user's perspective.
- **Unflexible**: Hardly one of these tools brings the ability to integrate into other systems (like Nagios or Jenkins).

TODO something missing?

Key Features

- Platform-independent UI testing tool
- End-2-End monitoring of application functionality and quality
- Combines two automation technologies:
    - DOM based web testing
    - image pattern based UI automation (for non-web content)
- Scalable from single-client up to multi-node container setup for parallel execution
- Integration of test results into
    - monitoring systems (e.g. Nagios/Icinga)
    - CI builds (e.g. Jenkins)

Concept

TODO aktualisieren und abstrachieren, https://jaxenter.de/sakuli-testing-monitoring-container-54374

TODO Grobes Kontzept danach tiefere Themen wie Detail-Testdefiniton verlinken

TODO Liste der Forwarder hinzufügen
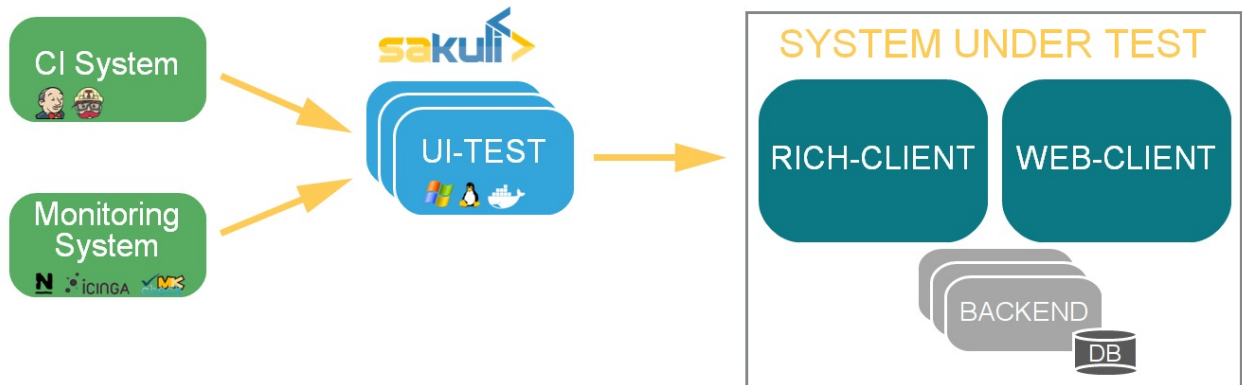
Sakuli simulates user actions on graphical user interfaces (web, fat client, citrix, …), and provides the obtained information (runtime, result, screenshots) to third party (e.g. Nagios compatible monitoring) systems.

Sakuli depends on **Java** and should run on most OS; **Windows**, **Ubuntu Linux (14.04 LTS)** and **OpenSUSE 13.2** are currently proved.

The **Sakuli** project brings together two Open-Source end-to-end testing tools which perfectly fit together: **Sahi** (http://www.sahi.co.in/) for **web-based tests** (by injecting JavaScript code into the browser), as well as the screenshot-based testing tool **Sikuli** (http://sikulix.com/), which allows the execution of **keyboard and mouse actions** on screen areas that have been defined by previously recorded screenshots and are recognized using the OpenCV engine.

Sakuli accesses both tools via its **Java API** and makes it possible to **use them simultaneously**. For example, web tests can be done very performant with Sahi (where a screenshot-based approach would be at best the second choice), whereas "off-DOM"-content can be catched with Sikuli. Whenever a web test comes into a situation which Sahi can't handle (e.g. a PIN dialogue for a smartcard login), use a Sikuli command. On the other hand, pure tests of fat client applications can be easily be setup by using only the Sikuli functions of Sakuli.
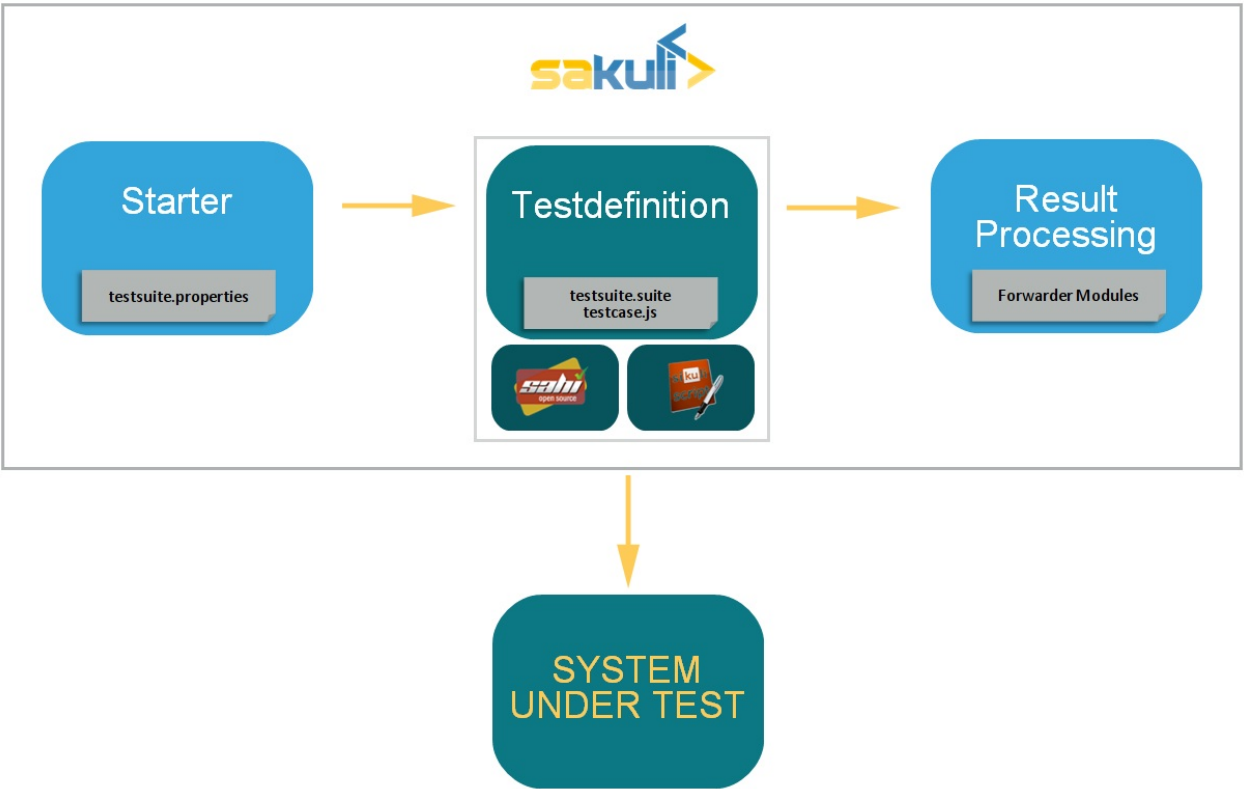
TODO TS Why different executions are possible

TODO extract from installation-client.md

If you have some errors with your Windows installation, you can check the following points:

Change Windows theme and title bar colors

Windows 7 comes by default with an "aero" theme, which is quite awkward for Sikuli, because there are many transparency effects which cause window elements to change their appearance dependend on the elements below. For that, change the theme to "Windows Classic".

Furthermore, change the colors of **active** and **inactive** title bars to **non gradient**:

RDP related settings

The following steps have only to be done if you are accessing the Sakuli Client via RDP.

Disable Clipboard Sharing

The "paste" function of Sakuli uses the clipboard at runtime to decrypt and paste passwords. For this reason, the clipboard exchange of the Sakuli client and the RDP client should be suppressed in the settings tab of your **local Remote Desktop client**:

This can be set globally in the registry **of your local host**:

- "regedit"
- [HKEY_CURRENT_USER\Software\Microsoft\Terminal Server Client]
- "DisableDriveRedirection" (DWORD) => "1"

Disable the "GUI-less" mode

If you minimize the Remote Desktop window (the window that display the remote computer's desktop), the operating system switches the remote session to a "GUI-less mode" which does not transfer any window data anymore. As a result, Sakuli is unable to interact with the tested application's GUI, as the whole screen is not visible.

To disable the "GUI-less" mode **on your local host**:

- "regedit"
- [HKEY_CURRENT_USER\Software\Microsoft\Terminal Server Client]
- "RemoteDesktop_SuppressWhenMinimized" (DWORD) => "2"

RDP peculiarities

things to know

There are four ways to connect to and work on a Sakuli client machine:

1. **VNC**
2. **Console** of a virtualization platform (ESX, Virtualbox, etc.)
3. **Remote Desktop** (Windows)
4. **local screen**

For case 1. and 2. there is nothing special to watch out for, except that the screen must not be locked (otherwise Sikuli will also see a locked screen). The screen content will be the same as displays on a local screen (4.).

For RDP on Windows there are some special things to know. Connecting to the Sakuli test client via RDP **locks any existing local console session of that user** and **attaches (="moves") it to a RDP session**.

Sakuli will just as well run within that RDP session. But closing/disconnecting/logging of that RDP session will not unlock the local console session again. Sakuli will see the same as a regular user: nothing but a locked screen. Read the next paragraph to learn how to avoid this.

LOGOFF.bat

To log off a RDP session, right-click `%SAKULI_HOME%\bin\helper\LOGOFF.bat` and execute the script with administrator privileges. The script then

- determines the current RDP session ID
- redirects this session back to the local console
- terminates the RDP session.

check_logon_session.ps1

In `%SAKULI_HOME%\setup\nagios` you can find **check_logon_session.ps1** which can be used as a client-side Nagios check to ensure that the Sakuli user is always logged on, either via RDP or on the local console. Instructions for the implementation of this check can be found in the script header.

Define a service dependency of all Sakuli checks to this logon check; this will ensure that a locked session will not raise false alarms.

TODO extract from installation-client.md

TODO extract from installation-client.md

If you have some errors with your Linux installation, you can check the following points:

install GNOME session fallback theme

Sakuli can test on Unity, of course - but gnome-session-fallback is more than sufficient…

```
sudo apt-get install gnome-session-fallback
```

After the installation, relogin and select the desktop envirionment **GNOME Flashback (Metacity)**:

The Ubuntu menu bar should have changed now to the "classical" one:

Restore gsettings key bindings

In headless checks you will encounter problems using the TAB key as well as "s":

- The TAB key will switch applications (like Alt+tab)
- "s" will open the applications menu

For some reason (?), gsettings binds "s" and TAB to the Super key by default. Open a terminal as the Sakuli user and execute the following commands to restore that to the default:

```
gsettings set org.gnome.desktop.wm.keybindings switch-applications "['<Alt>Tab']"
gsettings set org.gnome.desktop.wm.keybindings panel-main-menu "['<Alt>F1']"
```

Headless Sakuli checks on Linux

Running Sakuli on the user desktop is nice, but has the drawback that the session gets highjacked on each Sakuli run. Moving the mouse e.g. can have negative effects on the test execution.

For that reason we advise to run Linux based Sakuli checks in in one of the following modes:

- in a virtual display (="headless"), which is straightforward in Linux (documented on this page)
- in a Docker container (more scalable)

Installing and configuring the VNC server

On **Ubuntu**, first **install** vnc4server:

```
sudo apt-get install vnc4server
```

Start vncserver for the first time to create a **session password**:

```
~$ vncserver

You will require a password to access your desktops.
Password:
Verify:

New 'sakulidemo:1 (sakuli)' desktop is sakulidemo:1

Creating default startup script __HOME__/.vnc/xstartup
Starting applications specified in __HOME__/.vnc/xstartup
Log file is __HOME__/.vnc/sakulidemo:1.log
```

`.vnc/xstartup` controls what to start within a xvnc session. Do not touch this file on OpenSUSE; on **Ubuntu** you have to replace its content with the following lines (because you are using **gnome-session-fallback**, aren't you…?):

```
~$ vim .vnc/xstartup

#!/bin/sh
export XKL_XMODMAP_DISABLE=1
unset SESSION_MANAGER
unset DBUS_SESSION_BUS_ADDRESS

gnome-panel &
gnome-settings-daemon &
metacity &
```

Restart the current vnc sesssion:

```
~$ vncserver -kill :1 && vncserver
```

Now open a RDP client (on Ubuntu: *Applications - Internet - Remmina Remote Desktop Client*) and enter the connection data:

- Protocol: VNC
- Server: localhost:5901
- Password: `__VNC_SESSION_PASSWORD__`

You should see now an empty GNOME/KDE desktop - started headless!

Test

You are now ready to run the **minimal Sakuli check** in **headless (=VNC)** mode. Sakuli provides for this task a pre- and postHook script, which can be used like follow:

On the **Ubuntu** desktop, open a terminal window and execute

- on **Ubuntu**: `sakuli run __INST_DIR__/example_test_suites/example_ubuntu/ --vnc`
- on **openSUSE**: `__SAKULI_HOME__/bin/sakuli.sh --run __INST_DIR__/example_test_suites/example_opensuse/ --vnc`

You should see that Sakuli

1. opens **Firefox**
2. opens the **calculator** and calculates *525+100=625*
3. opens an **editor** and writes a **status message**

Scheduling by cron

Add the following line to Sakuli's crontab:

```
SAKULI_HOME=__SAKULI_HOME__
DISPLAY=:1

*/2 * * * * $SAKULI_HOME/bin/sakuli run $SAKULI_HOME/../example_test_suites/example_ubuntu -preHook $SAKULI_HOME/bin/hel
```

Upgrade process

Before you upgrade your current Sakuli installation, please ensure that you have read the Changelog.

Execute `__SAKULI_HOME__/../Uninstaller/uninstaller.jar` to remove the current installed version. This will only affect the files in `__SAKULI_HOME__` . The uninstaller removes all installed files, and resets the environment configuration.

After that just install the new version.

Sakuli Java DSL

Sakuli provides a Java DSL for writing test cases in pure Java. The DSL is designed as fluent API and provides the exact same capabilities as the Javascript API.

Usage

The Sakuli Java DSL comes to you as Maven module JAR file. You can add the module to your Sakuli project as Maven dependency. Currently the Java tests have to be written with the [TestNG] unit framework, so also provide the [TestNG] Maven dependency in your project POM:

```
org.sakuli
java-dsl
${sakuli.version}
test


${project.groupId}
java-dsl
${sakuli.version}
test-jar
test


org.testng
testng
${testng.version}
test
```

In the last step, we have to provide also some local resources for our Sakuli test setup. Therefore we use the [maven-dependency-plugin] to unpack all needed Sakuli resources to our local `project.build.outputDirectory` :

```
org.apache.maven.plugins
maven-dependency-plugin


    unpack
    generate-resources

        unpack



                org.sakuli
                java-dsl
                ${sakuli.version}
                jar
                true


    ${project.build.outputDirectory}
```

Now we are ready to write Java tests. See the following sample test which uses TestNG unit testing library in combination with Sakuli Java DSL:

```java
public class FirstExampleTest extends AbstractSakuliTest {

    Environment env;

    @BeforeClass
    @Override
    public void initTC() throws Exception {
        super.initTC();
        env = new Environment();
    }

    @Override
    protected TestCaseInitParameter getTestCaseInitParameter() throws Exception {
        return new TestCaseInitParameter("test1");
    }

    @Test
    public void testCitrus() throws Exception {
        browser.open();
        browser.navigateTo("http://www.citrusframework.org/");

        ElementStub heading1 = browser.paragraph("Citrus Integration Testing");
        heading1.highlight();
        assertTrue(heading1.isVisible());

        ElementStub download = browser.link("/Download v.*/");
        download.highlight();
        assertTrue(download.isVisible());
        download.click();

        ElementStub downloadLink = browser.cell("2.6.1");
        downloadLink.highlight();
        assertTrue(downloadLink.isVisible());
    }
}
```

All people that are familiar with TestNG unit testing will notice that a Sakuli Java test is nothing but a normal TestNG unit test. Sakuli just adds the end-to-end testing capabilities. The test class extends an abstract class coming from Sakuli. This `AbstractSakuliTest` provides convenient access to the Sakuli Java DSL API.

These are methods such as `initTC()` and `getTestCaseInitParameter()` that are customizable in test classes. Just override the methods and add custom logic. In addition to that the abstract super class in Sakuli provides access to the `browser` field that represents the Sahi web browser capabilities. This browser object is used in the test cases to trigger Sahi related actions such as opening a website and highlighting links or buttons.

In the example above we open a website **http://www.citrusframework.org/** and assert the content on that page.

Now lets add some testing logic that works with content other than HTML dom related content. We add a test step that puts focus to the web browser task bar. In detail we click into the browser search input field, fill in a word and perform the search.

```java
new Region().find("search").click().type("Citrus");
env.sleep(1);
new Region().takeScreenshot("test.png");
```

The `region` object provides access to Sakuli screen related actions such as finding a component on the screen. We can click that region and fill in some characters (e.g. in the search input field). After that we sleep some time to give the search operation some time to perform its actions. As a last step we take a screenshot of the result page and the test is finished.

This little example demonstrates the basic usage of the Sakuli Java API. We write normal Java unit tests with TestNG and add Sakuli specific actions on HTML dom content in a browser or any native application operations by screen related access.

Next we will setup a complete sample project for Sakuli Java.

Sakuli Java Example

The next section describes how to get started with the Sakuli Java DSL by example. The Java example is a fully runnable Java sample test case. So at the end of this chapter you should be able to start writing Sakuli test in pure Java.

An example how to use Java DSL and setup Maven you will find at: **github.com/ConSol/sakuli-examples**

Installation

Preparation

1. Install Java Development Kit version 8.
2. Install Maven (Version 3.2.5 or higher).
3. Download `java-example` directory from github.com/ConSol/sakuli-examples.

Project setup and compilation

1. Import `java-example` to IDE (IntelliJ or Eclipse...) as Maven project: **Example for IntelliJ:**
   - Choose `Project from Existing Sources...` in File menu.
   - Choose `pom.xml` and click `next` button till the project is imported.
2. Try to **compile** the new Sakuli Maven project. If an **ERROR** is reported please check your `pom.xml` first. The following section has to be present in your Maven POM:

```
<repository>
    <id>labs-consol-snapshots</id>
    <name>ConSol* Labs Repository</name>
    <url>http://labs.consol.de/maven/snapshots-repository</url>
    <snapshots>
        <enabled>true</enabled>
    </snapshots>
    <releases>
        <enabled>false</enabled>
    </releases>
</repository>
```

The ConSol labs Maven repository should be placed to the **repositories** section in your POM. After this is done please execute the Maven **compile** phase.

Test execute

Once compilation has been **SUCCESS** try to execute **test** phase as a next step.

Configuration

For customized browser detection create your own `browser_types.xml` file. This file should be located in `main/resources/sahi/userdata/config` package in `src` folder. The content of this file looks like follows:

```
<browserTypes>
    <browserType>
        <name>firefox</name>
        <displayName>Firefox</displayName>
        <icon>firefox.png</icon>
        <path>$ProgramFiles (x86)\Mozilla Firefox\firefox.exe</path>
        <options>-profile "$userDir/browser/ff/profiles/sahi$threadNo" -no-remote</options>
        <processName>firefox.exe</processName>
        <capacity>5</capacity>
    </browserType>

    <browserType>
```

```
            <name>firefox4</name>
            <displayName>Firefox 4</displayName>
            <icon>firefox.png</icon>
            <path>$ProgramFiles (x86)\Mozilla Firefox 4.0 Beta 7\firefox.exe</path>
            <options>-profile "$userDir/browser/ff/profiles/sahi$threadNo" -no-remote</options>
            <processName>firefox.exe</processName>
            <capacity>5</capacity>
        </browserType>


        <browserType>
            <name>ie</name>
            <displayName>IE</displayName>
            <icon>ie.png</icon>
            <path>$ProgramFiles\Internet Explorer\iexplore.exe</path>
            <options>-noframemerging</options>
            <processName>iexplore.exe</processName>
            <useSystemProxy>true</useSystemProxy>
            <capacity>5</capacity>
        </browserType>


        <!-- <browserType>
            <name>ie</name>
            <displayName>IE 8</displayName>
            <icon>ie.png</icon>
            <path>$ProgramFiles (x86)\Internet Explorer\iexplore.exe</path>
            <options>-noframemerging</options>
            <processName>iexplore.exe</processName>
            <useSystemProxy>true</useSystemProxy>
            <capacity>5</capacity>
        </browserType> -->
        <!--
        <browserType>
            <name>ie</name>
            <displayName>IE 7</displayName>
            <icon>ie.png</icon>
            <path>$ProgramFiles (x86)\Internet Explorer\iexplore.exe</path>
            <options></options>
            <processName>iexplore.exe</processName>
            <useSystemProxy>true</useSystemProxy>
            <capacity>5</capacity>
        </browserType>
         -->


        <!--
        <browserType>
            <name>ie</name>
            <displayName>IE 6</displayName>
            <icon>ie.png</icon>
            <path>$ProgramFiles (x86)\Internet Explorer\iexplore.exe</path>
            <options></options>
            <processName>iexplore.exe</processName>
            <useSystemProxy>true</useSystemProxy>
            <capacity>5</capacity>
        </browserType>
         -->


        <browserType>
            <name>chrome</name>
            <displayName>Chrome</displayName>
            <icon>chrome.png</icon>
            <path>C:\Program Files (x86)\Google\Chrome\Application\chrome.exe</path>
            <options>--incognito --user-data-dir=$userDir\browser\chrome\profiles\sahi$threadNo --proxy-server=localhost
            <processName>chrome.exe</processName>
            <capacity>5</capacity>
        </browserType>


        <browserType>
            <name>safari</name>
            <displayName>Safari</displayName>
            <icon>safari.png</icon>
            <path>$ProgramFiles (x86)\Safari\Safari.exe</path>
            <options> </options>
            <processName>safari.exe</processName>
            <useSystemProxy>true</useSystemProxy>
            <capacity>1</capacity>
        </browserType>
```

```
        <browserType>
            <name>opera</name>
            <displayName>Opera</displayName>
            <icon>opera.png</icon>
            <path>$ProgramFiles (x86)\Opera\opera.exe</path>
            <options> </options>
            <processName>opera.exe</processName>
            <useSystemProxy>true</useSystemProxy>
            <capacity>1</capacity>
        </browserType>

    </browserTypes>
```

Note: **If needed change the for your own locations of each browser!**

Now you can execute **test** phase and enjoy the successful execution of the test.

```
        <browserType>
            <name>opera</name>
            <displayName>Opera</displayName>
            <icon>opera.png</icon>
            <path>$ProgramFiles (x86)\Opera\opera.exe</path>
            <options> </options>
            <processName>opera.exe</processName>
            <useSystemProxy>true</useSystemProxy>
            <capacity>1</capacity>
        </browserType>
```

Usage of Sakuli Docker Images

Docker allows you to run a Sakuli test in an isolated environment, called "container", which is always started off from the same base image. This ensures that tests always run under equal conditions.

Image OS types

The repository's subfolder ./docker contains all source files Sakuli docker images are made of. Currently we provide images on DockerHub for:

- CentOS 7 (consol/sakuli-centos-xfce, consol/sakuli-centos-xfce-java)
- Ubuntu 14.04 LTS (consol/sakuli-ubuntu-xfce,consol/sakuli-ubuntu-xfce-java)

Image tags

The build process on DockerHub is triggered by Github hooks; that means that you are always getting the current version for the two branches

- **master** -> image tag "**latest**": contains the latest stable release of Sakuli
- **dev** -> image tag "**dev**": contains the latest snapshot version of Sakuli

Architecture of Sakuli containers

Each Sakuli docker image is installed with the following components:

- Desktop environment (**Xfce4**)
- VNC-Server (default VNC port `5901`)
- **noVNC** - HTML5 VNC client (default http port `6901`)
- Java JRE 8
- Browsers:
  - Mozilla Firefox + Java Plugin
  - Google Chrome (Java-Plugin is no longer supported)
- **Sahi OS 5**
- **Sakuli** in the latest stable version

The running containers are accessible with VNC (default password: *sakuli*) by:

- **VNC viewer**:
  - `DOCKER_HOST:5901`
- **noVNC HTML5 client**:
  - http://localhost:6901/vnc_auto.html?password=sakuli

Get Sakuli Docker Images

The following example command pulls the CentOS7 image from DockerHub:

```
docker pull consol/sakuli-centos-xfce
```

Alternatively, you can build this image from the sources:

```
git clone https://github.com/ConSol/sakuli.git
docker build -t consol/sakuli-centos-xfce docker/sakuli-centos-xfce .
```

Start/test a Sakuli container

Once you have pulled/built the image, you can start a container on top of it which binds port 5901/tcp and 6901/tcp to localhost (on native docker installations; $DOCKER_IP on boot2docker):

```
# default tag "latest" = Sakuli stable
docker run -it -p 5901:5901 -p 6901:6901 consol/sakuli-centos-xfce
# tag "dev" = Sakuli Snapshot version of dev branch
docker run -it -p 5901:5901 -p 6901:6901 consol/sakuli-centos-xfce:dev
```

The container will execute a small headless self-test and exit afterwards. Read on to learn how to execute your own JavaScript or Java based tests within this containers.

1) Run your test suite in a Sakuli container - JavaScript based test

There are three important lines in the *Dockerfile* of each Sakuli image which define what has to be done on a container start:

```
ENV SAKULI_TEST_SUITE /root/sakuli/example_test_suites/example_xfce
ENTRYPOINT ["/root/scripts/sakuli_startup.sh"]
CMD ["run $SAKULI_TEST_SUITE"]
```

- `ENTRYPOINT` is the command which is executed once the container is started with `docker run`.
- `CMD` is the default argument for `ENTRYPOINT`, that is, to run a test suite set by a variable.
- `ENV SAKULI_TEST_SUITE` is set to the path of a test suite which has to run when the container starts. By default, this is set to the built-in folder `/root/sakuli/example_test_suites/example_xfce`.

There is more than one way to integrate a custom testsuite in a container, discussed in the following. Assume you want to run a suite called `suite_1` located on your host at the path `/home/myuser/my-sakuli-testsuites` - use one of the following ways:

1.1) `run` command

Mount the suite folder on your host into the container and override `CMD` from Dockerfile (=argument for `ENTRYPOINT`) with custom parameters for the Sakuli starter `sakuli`. In this way you can also give further parameters to Sakuli e.g. to use another browser (`-browser chrome`).

```
# running tests in chrome
~$ docker run -it -p 5901:5901 -p 6901:6901 consol/sakuli-centos-xfce 'run "$SAKULI_TEST_SUITE" -browser chrome'
```

To get all possible command line parameters call `docker run consol/sakuli-ubuntu-xfce -help`.

CMD can be overrideen in two ways:

1.1.1) Using the command line

```
~$ docker run -it -p 5901:5901 -p 6901:6901 -v "/home/myuser/my-sakuli-testsuites:/my-sakuli-testsuites" consol/sakuli-
```

This command will

- mount the test suites folder to `/my-sakuli-testsuites` within the container
- execute the suite `suite_1`

1.1.2) Using docker-compose

A more elegant way is to pack all parameters into a Docker Compose file. Create `docker-compose.yml` :

```
sakuli-example-ubuntu:
  image: consol/sakuli-centos-xfce
  ports:
  - 5901:5901
  - 6901:6901
  volumes:
  - /home/myuser/my-sakuli-testsuites:/my-sakuli-testsuites
  command: "'run /my-sakuli-testsuites/suite_1'"
```

When executed in the same directory as `docker-compose.yml` , a simple `docker-compose up` will bring up all containers. (Intentionally, `docker-compose` is made to bring up environments with *multiple* containers which are linked together; but even with one container it eases the parameter handling.)

`docker-compose rm -f` in contrast removes all currently stopped and running containers, which defined in the `docker-compose.yml` . Otherwise, if `docker-compose up` will called again, the test execution will reattach the instance and the start the test execution again in the same container instance.

1.2) Environment variable `SAKULI_TEST_SUITE`

Mount a folder on your host into the container and override the environment variable `SAKULI_TEST_SUITE` .

1.2.1) Using the command line

```
~$ docker run -it -p 5901:5901 -p 6901:6901   \\
    -v "/home/myuser/my-sakuli-testsuites:/my-sakuli-testsuites"   \\
    -e "SAKULI_TEST_SUITE=/my-sakuli-testsuites/suite_1"   \\
    consol/sakuli-ubuntu-xfce
```

1.2.2) Using docker-compose

Similar to #12-using-docker-compose, the file `docker-compose.yml` would look like this:

```
sakuli-example-ubuntu:
  image: consol/sakuli-ubuntu-xfce
  ports:
  - 5901:5901
  - 6901:6901
  volumes:
  - /home/myuser/my-sakuli-testsuites:/my-sakuli-testsuites
  environment:
  - SAKULI_TEST_SUITE=/my-sakuli-testsuites/suite_1
```

2) Run your test suite in a Sakuli container - Java based test

Also for Sakuli test writen in Java and executed through Maven, we provide to preconfigured docker images: consol/sakuli-centos-xfce-java, consol/sakuli-ubuntu-xfce-java. More information about how to write a Java based Sakuli test, see Sakuli JavaDSL setup. Now take a look at the important lines in the *Dockerfile* which define how the container will start:

```
WORKDIR /opt/maven
ENTRYPOINT ["/root/scripts/start_hook.sh"]
CMD ["mvn clean test"]
```

- `WORKDIR` is set to the path, where the maven build will be executed. By default, this is set to the built-in example folder `/opt/maven` .
- `ENTRYPOINT` is the command which is executed once the container is started with `docker run` .
- `CMD` is the default argument for `ENTRYPOINT` , what will trigger the Maven build, which contains the Java based Sakuli tests.

Assume you want to run the Sakuli end-2-end test from your Maven project located at the path `/home/myuser/my-sakuli-maven-project` you can executed the Maven build in the inside of the Sakuli container like follow:

2.1) Using the command line

```
~$ docker run -it -p 5901:5901 -p 6901:6901 -v "/home/myuser/my-sakuli-maven-project:/opt/maven" consol/sakuli-ubuntu-x
```

This command will

- mount the test suites folder to `/home/myuser/my-sakuli-maven-project` within the container
- execute the maven build with default command `mvn clean test`

If you want to for example also build youre maven artifacts over `mvn install` override the default command like follow:

```
~$ docker run -it -p 5901:5901 -p 6901:6901 -v "/home/myuser/my-sakuli-maven-project:/opt/maven" consol/sakuli-ubuntu-x
```

2.2) Using docker-compose

A more elegant way is to pack all parameters into a Docker Compose file. Create `docker-compose.yml` :

```
sakuli-example-ubuntu:
  image: consol/sakuli-ubuntu-xfce-java
  ports:
  - 5901:5901
  - 6901:6901
  volumes:
  - /home/myuser/my-sakuli-maven-project:/opt/maven
```

When executed in the same directory as `docker-compose.yml` , a simple `docker-compose up` will bring up all containers. (Intentionally, `docker-compose` is made to bring up environments with *multiple* containers which are linked together; but even with one container it eases the parameter handling.)

`docker-compose rm -f` in contrast removes all currently stopped and running containers, which defined in the `docker-compose.yml` . Otherwise, if `docker-compose up` will called again, the test execution will reattach the instance and the start the test execution again in the same container instance.

Like above you can for example also override the default mvn command and use a additional persistent volume for caching the maven dependencies:

```
version: '2'

services:
  sakuli_java_test:
    image: consol/sakuli-ubuntu-xfce-java
    volumes:
    - /home/myuser/my-sakuli-maven-project:/opt/maven
    - data:/root/.m2
    network_mode: "bridge"
    ports:
    - 5911:5901
    - 6911:6901
    command: "'mvn clean install'"
    # to keep container running and login via `docker exec -it javaexample_sakuli_java_test_1 bash`
    # command: "'--tail-log'"

volumes:
  data:
    driver: local
```

3) Override VNC environment variables

The following VNC environment variables can be overwritten at the `docker run` phase to customize your desktop environment inside the container:

- `VNC_COL_DEPTH` , default: `24`
- `VNC_RESOLUTION` , default: `1280x1024`
- `VNC_PW` , default: `sakuli`

For example, the password for VNC could be set like this:

```
~$ docker run -it -p 5901:5901 -p 6901:6901 -e "VNC_PW=my-new-password"  \\
    consol/sakuli-ubuntu-xfce
```

4) Further Information

Further information about the usage of Sakuli docker containers can be found at:

- Publications:
  - **Containerized UI-Tests in Java with Sakuli and Docker**
- Presentations:
  - **Containerized End-2-End-Testing - ContainerDays 2016 Hamburg**
  - **Containerized End-2-End-Testing - ConSol CM Testing**
- Example projects on GitHub:
  - **ConSol/sakuli-examples**
  - **toschneck/sakuli-example-bakery-testing**
  - **ConSol/sakuli-example-testautomation-day**

Contact

For questions or maybe some hints, feel free to contact us via **sakuli@consol.de** or open an issue.

The guys behind Sakuli:

ConSol Consulting & Solutions Software GmbH
Franziskanerstr. 38, D-81669 München
Tel. +49-89-45841-100, Fax +49-89-45841-111
Homepage: http://www.consol.de E-Mail: info@consol.de </tr>

TODO extract from docker-images.md

TODO extract from docker-images.md

TODO fill out, see also JaxEnter article

- Why two different test languages?
    - What purpose?
    - Advantages
- API is for both the same (in JS just not type-safe)
- Who the other modules (Starter, Forwarder) are affected

TODO see jaxenter article

Link to tutorials at https://github.com/ConSol/sakuli-examples

TODO see jaxenter article

Link to tutorials at https://github.com/ConSol/sakuli-examples

TODO TS check path in api generator and if this can be added automatically

Sakuli API

Sakuli provides methods of three different types:

- JS methods of **Sahi**, which can be used 100% *natively*
- JS methods which encapsulate a subset of the **Sikuli** Java API
- JS methods of **Sakuli** framework itself (testcase stuff, exception handling, …)

For documentation see

- Sahi-API
- TestCase
- Application
- Environment
- Region
- Keys
- MouseButton
- Logger

Index

Namespaces

- Sahi-API
- TestCase
  - TestCase.addImagePaths(imagePaths)
  - TestCase.endOfStep(stepName, optWarningTime)
  - TestCase.handleException(e)
  - TestCase.saveResult()
  - TestCase.getID()
  - TestCase.getLastURL()
  - TestCase.getTestCaseFolderPath()
  - TestCase.getTestSuiteFolderPath()
  - TestCase.throwException(message, screenshot)
- Application
  - Application.open()
  - Application.focus()
  - Application.focusWindow(windowNumber)
  - Application.close(optSilent)
  - Application.kill(optSilent)
  - Application.setSleepTime(seconds)
  - Application.getRegion()
  - Application.getRegionForWindow(windowNumber)
  - Application.getName()
- Environment
  - Environment.setSimilarity(similarity)
  - Environment.resetSimilarity()
  - Environment.getRegionFromFocusedWindow()
  - Environment.takeScreenshot(pathName)
  - Environment.sleep(seconds)
  - Environment.getClipboard()
  - Environment.setClipboard(text)
  - Environment.pasteClipboard()
  - Environment.copyIntoClipboard()
  - Environment.cleanClipboard()

- Region.setW(width)
  - Region.getW()
  - Region.setX(x)
  - Region.getX()
  - Region.setY(y)
  - Region.getY()
  - Region.highlight(seconds)
  - Region.takeScreenshot(filename)
  - Region.sleep(seconds)
  - Region.extractText()
- RegionRectangle

Sahi-API

All Sahi-API functions are natively usable in Sakuli. For a complete documentation, see Sahi-API.

Members

- Sahi-API

TestCase

TestCase - initializes the Sakuli object and sets the warning and critical time for this test case.

Params

- warningTime `number` - threshold in seconds. If the threshold is set to 0,

```
        the execution time will never exceed, so the state will be always OK!
```

- criticalTime `number` - threshold in seconds. If the threshold is set to 0,

```
        the execution time will never exceed, so the state will be always OK!
```

- optImagePathArray `Array.<String>` - (optional) Path or Array of Paths to the folder containing the image patterns

```
                for these test cases.
```

Returns: - an initialized Sakuli object.

Example

```
var testCase = new TestCase(20,30, "path-to/image-folder-name");
```

Members

- TestCase
  - TestCase.addImagePaths(imagePaths)
  - TestCase.endOfStep(stepName, optWarningTime)
  - TestCase.handleException(e)
  - TestCase.saveResult()
  - TestCase.getID()
  - TestCase.getLastURL()
  - TestCase.getTestCaseFolderPath()
  - TestCase.getTestSuiteFolderPath()
  - TestCase.throwException(message, screenshot)

TestCase.addImagePaths(imagePaths)

Adds the additional paths to the current image library of the TestCase. If a relative path is assigned, the current testcase folder will be used as current directory.

Params

- imagePaths `string` - one or more path strings

TestCase.endOfStep(stepName, optWarningTime)

A step allows to sub-divide a case to measure logical units, such as "login", "load report" etc. in its particular runtime. When a case starts, Sakuli starts a "step" timer. It gets read out, stored with the step name, and resetted each time endOfStep() is called. If the step runtime exceeds the step threshold (second parameter, optional), the step is saved with state "WARNING" (there is no CRITICAL state).

Params

- stepName `String`
- optWarningTime `number` - (optional) threshold in seconds, default = 0. If the threshold is set to 0,

```
the execution time will never exceed, so the state will be always OK!
```

TestCase.handleException(e)

Handles any Exception or Error. The handleException function calls the Java backend and stores the Exception for further processing.

Use it at the end of a catch-block.

Params

- e `Error` - any Exception or Error

Example

```
try {
  ... do something
} catch (e) {
    sakuli.handleException(e);
}
```

TestCase.saveResult()

Saves the results of the current test case for further processing.

Should be called in finally-block of the test case:

Example

```
try {
    ... do something
} catch (e) {
    sakuli.handleException(e);
} finally {
    sakuli.saveResult();
}
```

TestCase.getID()

Returns the **current** id of this test case.

Returns: `String` - id

TestCase.getLastURL()

Updates and returns the URL of the last visited URL

Returns: `String` - last visited URL

TestCase.getTestCaseFolderPath()

Returns: `String` - the folder path of the current testcase.

TestCase.getTestSuiteFolderPath()

Returns: `String` - the folder path of the current testcase.

TestCase.throwException(message, screenshot)

Creates a new test case based exception with an optional screenshot at the calling time. Will be called from sakuli.js or in side of 'org.sakuli.javaDSL.AbstractSakuliTest'.

Params

- message `String` - error message
- screenshot `Boolean` - enable / disable screenshot functionality

Application

Application Class - Represents an application.

Params

- applicationNameOrPath `String` - Path to the application file. Example: `C:\Windows\system32\notepad.exe`
- optResumeOnException `Boolean` - Determines whether to ignore exceptions from this class. If this parameter is undefined, it will be false.

Returns: Application - an initialized object.

Example

```
//windows
var editor = new Application("notepad.exe");
//linux
var editor = new Application("gedit");
```

Members

- Application
    - Application.open()
    - Application.focus()
    - Application.focusWindow(windowNumber)
    - Application.close(optSilent)
    - Application.kill(optSilent)
    - Application.setSleepTime(seconds)
    - Application.getRegion()
    - Application.getRegionForWindow(windowNumber)
    - Application.getName()

Application.open()

Opens the created application. For application with a long load time you may need to change the default sleep time with setSleepTime(...).

Returns: - this Application object.

Application.focus()

Focuses the current application, if the application is in the background.

Returns: - this Application object.

Application.focusWindow(windowNumber)

Focuses a specific window of the application.

Params

- windowNumber `number` - identifies the window

Returns: - this Application object.

Application.close(optSilent)

Closes the already existing application.

Params

- optSilent `boolean` - (optional) if true, no exception will be thrown on errors and stop the test execution.

Returns: - this Application object.

Application.kill(optSilent)

Kill the already existing application hardly.

Params

- optSilent `boolean` - (optional) if true, no exception will be thrown on errors.

Returns: - this Application object.

Application.setSleepTime(seconds)

Sets the sleep time in seconds of the application actions to handle with long loading times. The default sleep time is set to 1 seconds.

Params

- seconds `number` - sleep time in seconds

Returns: - this Application object.

Application.getRegion()

Creates and returns a Region object from the application.

Returns: - a Region object.

Application.getRegionForWindow(windowNumber)

Creates and returns a Region object from a specific window of the application.

Params

- windowNumber `number` - identifies the window

Returns: - a Region object.

Application.getName()

Returns: - the name of the current application.

Environment

Environment - Represents the environment of the current test host.

Params

- optResumeOnException `Boolean` - (optional) if this parameter is undefined, it will be false.

Members

- Environment
  - Environment.setSimilarity(similarity)
  - Environment.resetSimilarity()
  - Environment.getRegionFromFocusedWindow()
  - Environment.takeScreenshot(pathName)
  - Environment.sleep(seconds)
  - Environment.getClipboard()
  - Environment.setClipboard(text)
  - Environment.pasteClipboard()
  - Environment.copyIntoClipboard()
  - Environment.cleanClipboard()
  - Environment.paste(text)
  - Environment.pasteMasked(text)
  - Environment.pasteAndDecrypt(text)
  - Environment.type(text, optModifiers)
  - Environment.typeMasked(text, optModifiers)
  - Environment.typeAndDecrypt(text, optModifiers)
  - Environment.decryptSecret(secret)
  - Environment.keyDown(keys)
  - Environment.keyUp(keys)
  - Environment.write(text)
  - Environment.mouseWheelDown(steps)
  - Environment.mouseWheelUp(steps)
  - Environment.isWindows()
  - Environment.isLinux()
  - Environment.getOsIdentifier()
  - Environment.runCommand(command, optThrowException)

Environment.setSimilarity(similarity)

Set a new default similarity for the screen capturing methods.

Params

- similarity `number` - value between 0 and 1, default = 0.8

Returns: - this Environment or NULL on errors.

Environment.resetSimilarity()

Resets the current similarty of the screen capturing methods to the original default value of 0.8.

Returns: - this Environment or NULL on errors.

Environment.getRegionFromFocusedWindow()

Get a Region object from the current focused window

Returns: - a Region object from the current focused window or NULL on errors.

Environment.takeScreenshot(pathName)

Takes a screenshot of the current screen and saves it to the overgiven path. If there ist just a file name, the screenshot will be saved in your testsuite log folder.

Params

- pathName `String` - `pathname/filname.format` or just `filename.format`

Example

```
environment.takeScreenshot("test.jpg");
```

Environment.sleep(seconds)

Blocks the current testcase execution for x seconds

Params

- seconds `number` - to sleep

Returns: - this Environment or NULL on errors.

Environment.getClipboard()

Returns: - the current content of the clipboard as String or NULL on errors

Environment.setClipboard(text)

sets the String paramter to the system clipboard

Params

- text `String` - text as string

Returns: - this Environment.

Environment.pasteClipboard()

pastes the current clipboard content into the focused area. Will do the same as "STRG + V".

Returns: - this Environment.

Environment.copyIntoClipboard()

copy the current selected item or text to the clipboard. Will do the same as "STRG + C".

Returns: - this Environment.

Environment.cleanClipboard()

Clean the content of the clipboard.

Environment.paste(text)

pastes the text at the current position of the focus/carret
using the clipboard and strg/ctrl/cmd-v (paste keyboard shortcut)

Params

- text `String` - a string, which might contain unicode characters

Returns: - this Environment or NULL on errors.

Environment.pasteMasked(text)

makes a masked paste(String) without any logging.

Params

- text `String` - a string, which might contain unicode characters

Returns: - this Environment or NULL on errors.

Environment.pasteAndDecrypt(text)

combines pasteMasked(String) and decryptSecret(String).

Params

- text `String` - encrypted secret

Returns: - this Environment or NULL on errors.

Environment.type(text, optModifiers)

Enters the given text one character/key after another using keyDown/keyUp.

About the usable Key constants see documentation of Key. The function could also type UTF-8 unicode characters, if the OS supports it. The text is entered at the current position of the focus.

Params

- text `String` - containing characters and/or Key constants
- optModifiers `string` - (optional) an String with only Key constants.

Returns: - this Environment or NULL on errors.

Environment.typeMasked(text, optModifiers)

Enters the given text one character/key after another using keyDown/keyUp. The entered text will be masked at the logging.

About the usable Key constants see documentation of Key. The function could also type UTF-8 unicode characters, if the OS supports it. The text is entered at the current position of the focus.

Params

- text `String` - containing characters and/or Key constants
- optModifiers `string` - (optional) an String with only Key constants.

Returns: - this Environment or NULL on errors.

Environment.typeAndDecrypt(text, optModifiers)

Decrypt and enters the given text one character/key after another using keyDown/keyUp. The entered text will be masked at the logging. For the details of the decryption see decryptSecret(String).

About the usable Key constants see documentation of Key. The function could also type UTF-8 unicode characters, if the OS supports it. The text is entered at the current position of the focus.

Params

- text `String` - containing characters and/or Key constants
- optModifiers `string` - (optional) an String with only Key constants.

Returns: - this Environment or NULL on errors.

Environment.decryptSecret(secret)

Decrypt a encrypted secret and returns the value at runtime. The decryption will only work if the encryption and decryption happen on the same physical machine. There will be no logging with the decrypted secret during this step.

To create a encrypted secret see "sakuli-manual.md".

Params

- secret `String` - encrypted secret as String

Returns: - decrypted String

Environment.keyDown(keys)

Press and hold the given keys including modifier keys
use the key constants defined in class Key,
which only provides a subset of a US-QWERTY PC keyboard layout
might be mixed with simple characters
use + to concatenate Key constants

Params

- keys `String` - valid keys

Returns: - this Environment or NULL on errors.

Environment.keyUp(keys)

release the given keys (see Environment.keyDown(...)).

Params

- keys `String` - valid keys

Returns: - this Environment or NULL on errors.

Environment.write(text)

Compact alternative for type() with more options

- special keys and options are coded as #XN. or #X+ or #X-
  where X is a refrence for a special key and N is an optional repeat factor
  A modifier key as #X. modifies the next following key
  the trailing . ends the special key, the + (press and hold) or - (release) does the same,
  but signals press-and-hold or release additionally.
  except #W / #w all special keys are not case-sensitive
  a #wn. inserts a wait of n millisecs or n secs if n less than 60
  a #Wn. sets the type delay for the following keys (must be > 60 and denotes millisecs)
- otherwise taken as normal wait
  Example: wait 2 secs then type CMD/CTRL - N then wait 1 sec then type DOWN 3 times
  Windows/Linux: write("#w2.#C.n#W1.#d3.")
  Mac: write("#w2.#M.n#W1.#D3.")
  for more details about the special key codes and examples consult the sikuliX docs

Params

- text `String` - a coded text interpreted as a series of key actions (press/hold/release)

Returns: - this Environment or NULL on errors.

Environment.mouseWheelDown(steps)

move the mouse pointer to the given target location and move the wheel the given steps down.

Params

- steps `number` - the number of steps

Environment.mouseWheelUp(steps)

move the mouse pointer to the given target location and move the wheel the given steps up.

Params

- steps `number` - the number of steps

Environment.isWindows()

Returns: `boolean` - true, if the OS is any instance of an Windows based OS

Environment.isLinux()

Returns: `boolean` - true, if the OS is any instance of an Linux based OS

Environment.getOsIdentifier()

Returns: `string` - identifier of the current OS

Environment.runCommand(command, optThrowException)

Runs the assigned command on the host and returns the result. **Attention:** this is OS depended feature! So be aware which os you are running, maybe us to check `Environment#isLinux()` or `Environment#isWindows()` .

Params

- command `string` - OS depended command as `String`
- optThrowException `boolean` - defines if an exception should be thrown, if the exit code != 0

Returns: - the result as `CommandLineResult` object, you can use the methods `result.getOutput()` and `result.getExitCode()`

Example

```
var app;
if(environmen.runCommand('uname --machine') == 'x86_64'){
    //open app from other path
    app = new Application('/lib64/appname');
} else {
    app = new Application('/lib/appname');
}
```

Key

Key - representing some Key constants which can be used in type functions as input text and as modifier keys.

The following **Key** values are possible:

`SPACE` , `ENTER` , `BACKSPACE` , `TAB` , `ESC` , `UP` , `RIGHT` , `DOWN` , `LEFT` , `PAGE_UP` , `PAGE_DOWN` , `DELETE` , `END` , `HOME` , `INSERT` , `F1` , `F2` , `F3` , `F4` , `F5` , `F6` , `F7` , `F8` , `F9` , `F10` , `F11` , `F12` , `F13` , `F14` , `F15` , `SHIFT` , `CTRL` , `ALT` , `ALTGR` , `META` , `CMD` , `WIN` , `PRINTSCREEN` , `SCROLL_LOCK` , `PAUSE` , `CAPS_LOCK` , `NUM0` , `NUM1` , `NUM2` , `NUM3` , `NUM4` , `NUM5` , `NUM6` , `NUM7` , `NUM8` , `NUM9` , `SEPARATOR` , `NUM_LOCK` , `ADD` , `MINUS` , `MULTIPLY` , `DIVIDE` , `DECIMAL` , `CONTEXT`

Using `Key.ALTGR` **on Unix:** To enable the key command `ALTGR` for unix systems please bind it to `CTRL` + `ALT` , for more information see stackexchange.com - how-to-bind-altgr-to-ctrl-alt.

Example

Closing an window over typing the short cut `ALT` + `F4` :

```
env.type(Key.F4, Key.ALT);
```

Members

- Key

Logger

Logger - Logging functions to do 'debug, 'info', 'warning' and 'error' log entries.

Members

- Logger
  - Logger.logError(message)
  - Logger.logWarning(message)
  - Logger.logInfo(message)
  - Logger.logDebug(message)

Logger.logError(message)

make a error-log over Java backend into the log file. This won't stop the execution of the test case.

Params

- message `String` - as a String

Logger.logWarning(message)

make a debug-log over Java backend into the log file.

Params

- message `String` - as a String

Logger.logInfo(message)

make a info-log over Java backend into the log file.

Params

- message `String` - as a String

Logger.logDebug(message)

make a debug-log over Java backend into the log file.

Params

- message `String` - as a String

MouseButton

MouseButton - representing the possible mouse action button.

The following **MouseButton** values are possible:

`LEFT` , `RIGHT` , `MIDDLE`

Example

Press and release the right mouse button vor 3 seconds on a specified region:

```
var region = new Region().find("your-pattern.png");
region.mouseDown(MouseButton.RIGHT).sleep(3).mouseUp(MouseButton.RIGHT);
```

Members

- MouseButton

Region

Region - Represents a region as a part of or the hole screen.

Params

- optResumeOnException `Boolean` - if true, the test execution won't stop on an occurring error. Default: false.

Example

```
var screen = new Region(); //represents the hole screen
```

Members

- Region
    - Region.find(imageName)
    - Region.findRegion()

- ○ Region.exists(imageName, optWaitSeconds)
- ○ Region.click()
- ○ Region.doubleClick()
- ○ Region.rightClick()
- ○ Region.mouseMove()
- ○ Region.mouseDown(mouseButton)
- ○ Region.mouseUp(mouseButton)
- ○ Region.dragAndDropTo(targetRegion)
- ○ Region.waitForImage(imageName, seconds)
- ○ Region.paste(text)
- ○ Region.pasteMasked(text)
- ○ Region.pasteAndDecrypt(text)
- ○ Region.type(text, optModifiers)
- ○ Region.typeMasked(text, optModifiers)
- ○ Region.typeAndDecrypt(text, optModifiers)
- ○ Region.keyDown(keys)
- ○ Region.keyUp(keys)
- ○ Region.write(text)
- ○ Region.deleteChars(amountOfChars)
- ○ Region.mouseWheelDown(steps)
- ○ Region.mouseWheelUp(steps)
- ○ Region.move(offsetX, offsetY)
- ○ Region.grow(range)
- ○ Region.grow(width, height)
- ○ Region.above(range)
- ○ Region.below(range)
- ○ Region.left(range)
- ○ Region.right(range)
- ○ Region.setH(height)
- ○ Region.getH()
- ○ Region.setW(width)
- ○ Region.getW()
- ○ Region.setX(x)
- ○ Region.getX()
- ○ Region.setY(y)
- ○ Region.getY()
- ○ Region.highlight(seconds)
- ○ Region.takeScreenshot(filename)
- ○ Region.sleep(seconds)
- ○ Region.extractText()

Region.find(imageName)

Finds an image inside this region immediately.

Params

- imageName `String` - name of the preloaded picture

      (if not set, the find operation will take place on the predefined region object.)

Returns: - the found Region or if the target can't be found null.

Region.findRegion()

Finds a target in this Region immediately;

Returns: - the found Region or if the target can't be found null.

Region.exists(imageName, optWaitSeconds)

Check whether the give pattern is visible on the screen.

Params

- imageName `string` - if set, the function search inside the given region for the image
- optWaitSeconds `number` - if set, the function search for x seconds for the pattern.

Returns: - this Region or null

Region.click()

makes a mouse click on the center of the Region.

Returns: - the Region or NULL on errors.

Region.doubleClick()

makes a double click on the center of the Region.

Returns: - the Region or NULL on errors.

Region.rightClick()

makes a right click on the center of the Region.

Returns: - the Region or NULL on errors.

Region.mouseMove()

Move the mouse pointer to the center of the Region and "hovers" it.

Returns: - the Region or NULL on errors.

Region.mouseDown(mouseButton)

Low-level mouse action to press the assigned MouseButton on the current position.

Params

- mouseButton - on of MouseButton values

Returns: - the Region or NULL on errors.

Example

Press and release the right mouse button vor 3 seconds on a specified region:

```
var region = new Region().find("your-pattern.png");
region.mouseDown(MouseButton.RIGHT).sleep(3).mouseUp(MouseButton.RIGHT);
```

Region.mouseUp(mouseButton)

Low-level mouse action to release the assigned MouseButton.

Params

- mouseButton - on of [MouseButton](#) values

Returns: - the [Region](#) or NULL on errors.
Example
Press and release the right mouse button vor 3 seconds on a specified region:

```
var region = new Region().find("your-pattern.png");
region.mouseDown(MouseButton.RIGHT).sleep(3).mouseUp(MouseButton.RIGHT);
```

Region.dragAndDropTo(targetRegion)

Drag from region's current position and drop at given targetRegion and using the left mouse.

Params

- targetRegion `Region` - target where to drop

Returns: - the Region or NULL on failure
Example
move the bubble button 20px to the rigth:

```
var bubble = screen.find("bubble.png");
bubble.dragAndDropTo(bubble.right(20));
```

Region.waitForImage(imageName, seconds)

Blocks and waits until a target which is specified by the optImageName is found in the hole Screen within a given time period in seconds.

Params

- imageName `String` - name of the image pattern
- seconds `number` - the maximum time to waitFor in seconds

Returns: - a Region object representing the region occupied by the found target, or null if the target can not be found within the given time.

Region.paste(text)

pastes the text at the current position of the focus/carret
using the clipboard and strg/ctrl/cmd-v (paste keyboard shortcut)

Params

- text `String` - as a string, which might contain unicode characters

Returns: - this Region or NULL on errors.

Region.pasteMasked(text)

makes a masked paste(String) without any logging.

Params

- text `String` - a string, which might contain unicode characters

Returns: - this Region or NULL on errors.

Region.pasteAndDecrypt(text)

combines pasteMasked(String) and decryptSecret(String).

Params

- text `String` - encrypted secret

Returns: - this Region or NULL on errors.

Region.type(text, optModifiers)

Enters the given text one character/key after another using keyDown/keyUp.

About the usable Key constants see documentation of Key. The function could also type UTF-8 unicode characters, if the OS supports it. The text is entered at the current position of the focus.

Params

- text `String` - containing characters and/or Key constants
- optModifiers `String` - (optional) an String with only Key constants.

Returns: - this Region or NULL on errors.

Region.typeMasked(text, optModifiers)

Enters the given text one character/key after another using keyDown/keyUp. The entered text will be masked at the logging.

About the usable Key constants see documentation of Key. The function could also type UTF-8 unicode characters, if the OS supports it. The text is entered at the current position of the focus.

Params

- text `String` - containing characters and/or Key constants
- optModifiers `String` - (optional) an String with only Key constants.

Returns: - this Region or NULL on errors.

Region.typeAndDecrypt(text, optModifiers)

Decrypt and enters the given text one character/key after another using keyDown/keyUp. The entered text will be masked at the logging. For the deatails of the decryption see decryptSecret(String).

About the usable Key constants see documentation of Key. The function could also type UTF-8 unicode characters, if the OS supports it. The text is entered at the current position of the focus.

Params

- text `String` - containing characters and/or Key constants
- optModifiers `String` - (optional) an String with only Key constants.

Returns: - this Region or NULL on errors.

Region.keyDown(keys)

Press and hold the given keys including modifier keys
use the key constants defined in class Key,
which only provides a subset of a US-QWERTY PC keyboard layout

might be mixed with simple characters
use + to concatenate Key constants

Params

- keys `String` - valid keys

Returns: - this Region or NULL on errors.

Region.keyUp(keys)

release the given keys (see Region.keyDown(...)).

Params

- keys `String` - valid keys

Returns: - this Region or NULL on errors.

Region.write(text)

Compact alternative for type() with more options

- special keys and options are coded as #XN. or #X+ or #X-
  where X is a refrence for a special key and N is an optional repeat factor
  A modifier key as #X. modifies the next following key
  the trailing . ends the special key, the + (press and hold) or - (release) does the same,
  but signals press-and-hold or release additionally.
  except #W / #w all special keys are not case-sensitive
  a #wn. inserts a wait of n millisecs or n secs if n less than 60
  a #Wn. sets the type delay for the following keys (must be > 60 and denotes millisecs)
- otherwise taken as normal wait
  Example: wait 2 secs then type CMD/CTRL - N then wait 1 sec then type DOWN 3 times
  Windows/Linux: write("#w2.#C.n#W1.#d3.")
  Mac: write("#w2.#M.n#W1.#D3.")
  for more details about the special key codes and examples consult the sikuliX docs

Params

- text `String` - a coded text interpreted as a series of key actions (press/hold/release)

Returns: - this Region or NULL on errors.

Region.deleteChars(amountOfChars)

delete a amount of chars in a field

Params

- amountOfChars `number` - number of chars to delete

Returns: - this Region or null on errors

Region.mouseWheelDown(steps)

move the mouse pointer to the given target location and move the wheel the given steps down.

Params

- steps `number` - the number of steps

Region.mouseWheelUp(steps)

move the mouse pointer to the given target location and move the wheel the given steps up.

Params

- steps `number` - the number of steps

Region.move(offsetX, offsetY)

Set a offset to a specific Region and returns the new Region object. The offset function will move the Region's rectangle x pixels to the right and y pixels down. The size of the rectangle will be the same.

Params

- offsetX `number` - x-value for the offset action
- offsetY `number` - y-value for the offset action

Returns: - a Region with the new coordinates

Region.grow(range)

create a region enlarged range pixels on each side

Params

- range `number` - of pixels

Returns: - a new Region

Region.grow(width, height)

create a region with enlarged range pixels

Params

- width `number` - in pixels to grow in both directions
- height `number` - in pixels to grow in both directions

Returns: - a new Region

Region.above(range)

Params

- range `number` - of pixels

Returns: - a new Region that is defined above the current region's top border with a height of range number of pixels.

Region.below(range)

Params

- range `number` - of pixels

Returns: - a new Region that is defined below the current region's bottom border with a height of range number of pixels.

Region.left(range)

Params

- range `number` - of pixels

Returns: - a new Region that is defined on the left the current region's left border with a width of range number of pixels.

Region.right(range)

Params

- range `number` - of pixels

Returns: - a new Region that is defined on the right the current region's right border with a width of range number of pixels.

Region.setH(height)

set the height, based form the upper left corner downsides

Params

- height `number` - in pixels

Region.getH()

Returns: - height as int value

Region.setW(width)

set the width, based form the upper left corner to the right

Params

- width `number`

Region.getW()

Returns: - width as int value

Region.setX(x)

set the X coordinate of the upper left corner.

Params

- X `number`

Region.getX()

Returns: - width as int value

Region.setY(y)

set the Y coordinate of the upper left corner.

Params

- y `number`

Region.getY()

Returns: - Y coordinate of the upper left corner

Region.highlight(seconds)

Params

- seconds `number` - highlights this Region for x seconds or the default time

Region.takeScreenshot(filename)

Takes a screenshot of the current Region in the screen and saves it the current testcase folder with the assigned filename. If an absolute Path is assigned like e.g. `/home/user/test.jpg` , the screenshot will be saved at that place.

Params

- filename `String` - name of the screenshot, e.g. `region_screenshot` .

        Default: screenshot

Returns: `String` - file path to the created screenshot OR null on errors

Region.sleep(seconds)

Blocks the current testcase execution for x seconds

Params

- seconds `number` - to sleep

Returns: - this Region or NULL on errors.

Region.extractText()

Returns: - from this region a extracted Text as String

RegionRectangle

RegionRectangle (extends Region) - Represents a region specified by the x and y coordinates, width and height as a part of the screen.

Params

- x `number` - — x position of a rectangle on the screen.
- y `number` - — y position of a rectangle on the screen.
- w `number` - — width of a rectangle in pixel.
- h `number` - — height of a rectangle in pixel.
- optResumeOnException `Boolean` - (optional) if true, the test execution won't stop on an occurring error. Default: false.

Example

```
var notepadRegion = new RegionRectangle(0,0,100,100);
//represents a region which start at x=0, y=o (left upper corner) and have a size of 100px * 100px.
```

Members

- RegionRectangle

TODO see jaxenter article

Additional settings

This page contains different topics regarding the configuration of both **Sakuli** and its components, **Sahi** and **Sikuli**.

Sakuli settings

General note on property settings

Sakuli properties are predefined in `__SAKULI_HOME__/config/sakuli-default.properties` ; these values should/can be **overridden** in the following order (last match wins):

1. as a **global testsuite property** in `test-suites-folder/sakuli.properties` -> valid for **all test suites** within this folder
2. as a **testsuite property** in `test-suites-folder/test-suite/testsuite.properties` -> valid for **all test cases** within a test suite
3. as a **Java VM option** like `-D log.level.sakuli=DEBUG` , as option of the Sakuli starter -> valid for only one run

We do not recommend to change any values in `__SAKULI_HOME__/config/sakuli-default.properties` as a new version of Sakuli will have its own default property file; your changes will not be preserved.

Exception handling

Some objects (*Region, Application, Environment*) allow on their creation to specify the optional boolean argument `resumeOnException` , which controls whether the script should resume on an exception which is related to the object or one of its method (default: false).

Setting this to *true* can be useful if you want to raise a custom exception or no exception at all.

Set property `sakuli.exception.suppressResumedExceptions` to

- `true` = the exception will be logged and appear in the test result
- `false` = the exception will NEITHER be logged NOR appear in the test result.

Example:

```
// create region "foo"
var foo = new Region("bar.png",true);
// if "image" is not found, the script will resume
var baz = foo.find("image");
// throw your "own" exception.
// If you do not, and suppressResumedExceptions=true, the exception will be suppressed.
if (baz == null){
    throw "Sorry, I could not find image 'image'.";
}
```

Logging

Verbosity

The logging verbosity of all compoments *Sakuli, Sahi, Sikuli, Spring* - and *Java* in general can be changed individually.

("Verbosity" means one of the levels `DEBUG - INFO - WARN - ERROR`)

- `log.level.sakuli` `=__LEVEL__` - logging level for **Sakuli** output
- `log.level.sikuli` `=__LEVEL__` - logging level for **Sikuli** output
- `log.level.sahi` `=__LEVEL__` - logging level for **Sahi** output
- `log.level.spring` `=__LEVEL__` - logging level for the **Spring** framework (only used internally)
- `log.level.root` `=__LEVEL__` - logging level for all other **Java classes and libraries**

Log file folder / pattern

- `sakuli.log.pattern` `=%-5level [%d{YYYY-MM-dd HH:mm:ss.SSS}] - %msg%n` - Format string
- `sakuli.log.folder` `=${sakuli.testsuite.folder}/_logs` - Log folder

In general it is also possible to add your own Logback configuration under `__SAKULI_HOME__/config/sakuli-log-config.xml` . For more information about the Logback syntax please refer to the Logback manual.

Log file rotation

- `sakuli.log.maxAge` `14` - Deletes all files that are older than (default) 14 days in the defined `sakuli.log.folder` .

On Linux you can additional configure **logrotate** to tidy up old log files:

```
vim /etc/logrotate.d/sakuli

__SUITE_FOLDER__/*/_logs/* __SUITE_FOLDER__/*/_logs/_screenshots/* {
    size 1k
    missingok
    maxage 2
}
```

Secret De-/Encryption

Interface selection

Neither Sahi nor Sikuli have a way to prevent **sensible data** (passwords, PINs, etc.) from being logged and stored in the script in **clear text**.

That's the reason why Sakuli is able to **encrypt** them on the command line, and to **decrypt** them again on runtime just for the moment when they are needed. There is no (quick) way to decrypt those secrets again on the command line, so this is rather a way to obscure things not everybody should see than high-secure a encryption mechanism.

Among other parameters, Sakuli uses the MAC address of a local network interface card as a encryption salt. Hence no virtual adapters can be choosen.

You can decide whether Sakuli should automatically select an adapter...

- `sakuli.encryption.interface.autodetect` `=true`

..or a specific one should be used:

- `sakuli.encryption.interface.autodetect` `=false`
- `sakuli.encryption.interface` `=eth0`

Encrypt a secret

To encrypt secrets on the command line, Sakuli uses the MAC address of a NIC on the local machine (Windows/Linux). The following command lets Sakuli decide which NIC will be used:

```
sakuli encrypt foo
=========== Calling Sakuli JAR: java -classpath C:\Program Files (x86)\sakuli\sakuli-v0.9.3-SNAPSHOT\libs\jav\sakuli.ja

String to Encrypt: foo
...
Encrypted secret with interface 'eth3': CKXIAZmO7rSoBVMGgJZPDQ==

... now copy the secret to your testcase!
```

Add `-interface eth0` to select eth0 as salt interface. Add `-interface list` to get a list of all available adapters.

Decrypt a secret

To decrypt and use a secret in Sakuli test cases, use one of the following methods:

- pasteAndDecrypt
- typeAndDecrypt
- decryptSecret

Screenshot settings

To set the format and destination folder for screenshots taken by Sakuli:

- `sakuli.screenshot.onError = true` - take a screenshot in case of an exception
- `sakuli.screenshot.dir =${sakuli.log.folder}/_screenshots` - folder for screenshot files (if activated)
- `sakuli.screenshot.format =jpg` - screenshot file format (Possible values: jpg, png)
- `sakuli.forwarder.gearman.nagios.template.screenshotDivWidth = 640px` - Screenshot dimensions for results sent to Gearmand

Sikuli settings

Highlighting

- `sakuli.highlight.seconds = 1.1f` - duration for auto-highlighting and `highlight()` method
- `sakuli.autoHighlight.enabled = false` - If true, every region gets highlighted automatically for `sakuli.highlight.seconds`

Sahi settings

Browser configuration

If the Sahi dashboard does not show any browser or if you want to add another browser to the dashboard…

…you have to edit `__SAHI_DIR__/userdata/config/browser_types.xml`. Each browser is defined within a **browserType** block. Please refer to the Sahi Documentation, "Configure Browsers" to see the *browserType* Nodes for popular browsers.

For **PhantomJS** please save sahi.js into the folder `__SAHI_DIR__\phantomjs\` and use this option line:

```
<options>--proxy=localhost:9999 __SAHI_DIR__\phantomjs\sahi.js</options>
```

Attention: PhantomJS 2 is currently unsupported. Use version 1.9.x

Browser selection

You may want to change the browser due to the following reasons:

- to check if a web test (made with Sahi methods) for browser A is also running properly on browser B
- to run a headless browser
  - just for curiosity :-)
  - to keep the browser in background while Sakuli tests a non-web application (e.g. fat client)

In addition to the possibilities described in General note on property settings, the generic Sakuli starter `sakuli/sakuli.exe` can also be given the parameter `-browser`:

```
sakuli run /path/to/suite -browser chrome
```

Sahi behind a proxy

Set the following properties (as global in the sakuli.properties) to define a proxy Sahi should connect to.

```
### HTTP/HTTPS proxy Settings
### Set these properties, to enable the test execution behind company proxies
# Use external proxy server for HTTP*
ext.http.proxy.enable=true
ext.http.proxy.host=proxy.yourcompany.com
ext.http.proxy.port=8080
ext.http.proxy.auth.enable=false
ext.http.proxy.auth.name=user
ext.http.proxy.auth.password=password

# Use external proxy server for HTTPS
ext.https.proxy.enable=true
ext.https.proxy.host=proxy.server.com
ext.https.proxy.port=8080
ext.https.proxy.auth.enable=false
ext.https.proxy.auth.name=user
ext.https.proxy.auth.password=password

# There is only one bypass list for both secure and insecure.
ext.http.both.proxy.bypass_hosts=localhost|127.0.0.1|*.internaldomain.com|www.verisign.com
```

```
# Use external proxy server for
```

HTTPS support in Sahi

This documentation is not complete yet. Any help is appreciated.

General

Solution 1: accept self-signed certificates

This will work on most systems, especially the ones you have full control over.

- Open Sahi Dashboard, start the browser.
- On the Sahi start page, navigate to the https site you want Sahi to trust (e.g. https://github.com)
- You will be presented a message that this page "connection is not secure/trusted".
  - On **Firefox**, click **"I understand the risks."**
  - On **Chrome**, click **"Proceed anyway (unsecure)"**
  - On **IE**, click **"Continue to the website (not recommended)"**.
- Once the page has loaded, click **"SSL Manager"** from the Sahi start page
- You should be presented a green check mark right of the https URL, indicating that Chrome accepted the site's certificate, signed by Sahi:

Solution 2: manual certificate import (browser-specific)

Google Chrome (Windows)

- Open Sahi Dashboard, start *Chrome*
- On the Sahi start page, navigate to the https site you want Sahi to trust (e.g. https://github.com)
- You will be presented a message that this page "connection is not secure". Click on the **red-crossed locker symbol** left of the URL in the address bar and then **"Certificate information"**
- Change to tab **"Details"** and press **"Copy to File…"**
- In the following export assistent, save the certificate as **"Cryptographic Message Syntax Standard PKCS #7 Certificates (.P7B)""**
- Go to Chrome preferences, **"Manage certificates"**
- **"Import…"** -> select the exported **.p7b** certificate (in the "Open" dialogue, chnange the filename extension filter to **"PKCS#1"**)
- Choose **"Trusted Root Certification Authorities"** as certificate store
- Accept the Import confirmation.
- Restart Chrome from the Sahi dashboard.
- From the Sahi start page (step #2), click **"SSL Manager"**
- You should be presented a green check mark right of the https URL, indicating that Chrome accepted the site's certificate, signed by Sahi:

Mozilla Firefox

- Open Sahi Dashboard, start *Firefox/Firefox portable*
- On the Sahi start page, navigate to the https site you want Sahi to trust (e.g. https://github.com)
- You will be presented a message that this page "connection is not secure". Click on I understand the risks" and add the certificate exception. Now navigate back to the sahi start page and click on the Link "SSL Manager":

You will be promptet again a security warning for connecting to "sahi.example.com" (a local dummy domain), add the certificate here also as an exception.

The following page contains a list of all so far trusted and untrusted domains and their certificates. To import the red marked into the browser's certificate store, open the links by clicking them with the middle mouse key. This opens the page within a new tab; only in this way you are allowed to add the certificate exception.

You can close every opened tab when it is displaying "Successfully connected" for the domain:

Now go back to the SSL Manager overview and click the "refresh" button. Every previously untrusted domain should now be marked as OK:

Internet Explorer

See Sahi description Configer SSL in Sahi.

GUI-only tests

If you want to run tests which do not include any web technology, you can use phantomJS instead of firefox/chrome/IE and use the Sahi default start URL `http://sahi.example.com/_s_/dyn/Driver_initialized` .

(Reason: Sakuli depends on Sahi running, which in turn needs a running browser instance. Using PhantomJS for this, hides the browser window completely.)

Troubleshooting

Growing Firefox profile folder

If you experience that the Sahi profile folders `sahi0-9` of Mozilla Firefox (located in `sahi\userdata\browser\ff\profiles`) are getting bigger and bigger: this is caused by two bugs:

- https://bugzilla.mozilla.org/show_bug.cgi?id=85788
- https://bugzilla.mozilla.org/show_bug.cgi?id=686237

We do not know any Firefox settings which can prevent the creation of SQLITE writeahead-logs (if *you* do, please let us know). The only pragmatic solution at the moment is to delete all SQLITE files periodically or before each Sakuli run (e.g. by running as a preHook). `%SAKULI_HOME%\bin\helper\ff_purge_profile.bat` contains an example for windows.

Hanging applications

If you are testing applications which tend to hang/freeze, you can run a "tidy-up"-Script by using the `-pre-Hook` option of Sakuli:

```
sakuli run __INST_DIR__/example_test_suites/example_ubuntu/ -preHook 'cscript.exe %SAKULI_HOME%\bin\helper\killproc.vbs
```

In `procs_to_kill.txt` you can define which processes should be killed before Sakuli starts a new check:

```
# Full path:
C:\Program Files\Mozilla Firefox\firefox.exe
C:\Program Files\Internet Explorer\iexplore.exe
# Using wildcards (%):
java%sakuli.jar
```

Sikuli does not recognize images

If Sikuli does not recognize regions on the screen or does recognize the wrong ones, check the following list of possible reasons:

- **Run the client's OS on a fixed resolution:** Some applications/OS scale window elements slightly depending on the resolution. for example, if you are running Sakuli within Virtualbox, the guest OS changes its resolution as soon as you resize the VM window. The dimensions of window elements can then slightly diverge by 1-2 pixels from the screenshots taken before. This difference is small for human's eyes, but a big one for Sikuli. Make sure to disable features like "Auto-Adjust Guest Display" and set the Client's desktop to a common resolution (e.g. 1024x768). Side note: the smaller you set the resolution, the less work has to be done by Sikuli.
- **Disable any image compression algorithms** in your screenshot capturing program (Greenshot, Shutter, …). Otherwise Sikuli will compare *compressed* pattern images with *umcompressed* image data on the screen, which will fail for sure.
- Sikuli uses a **similarity** value of **0.99 by default**. That value (range: 0-0.99) determines that more than (X *100) % of the region pixels have to match. If Sikuli does not recognize anything or the wrong regions, try to **slightly** decrease the similarity* by changing 'sakuli.region.similarity.default' globally or inside a test e.g. with `env.setSimilarity(0.8)` . This should only be neccessary if the pattern images are of poor quality or the screen always differs slightly from the pattern images (e.g. due to compression of remote sessions like vnc). Please note that a similarity of "1" would mean that "more than 100%" of the region pixels would have to match - which is completely wrong.

Missing keystrokes on `type("...")` or failing `paste("...")`

Sikuli keyboard events (`type()` and `paste()`) on a Sahi-controlled browser instance can get lost if they are executed at the same time when Sahi internal status requests are sent from the browser to the Sahi proxy (default: 10x per sec).

For this reason, Sikuli type/paste methods first extend the Sahi status interval to the value of
`sahi.proxy.onSikuliInput.delayPerKey` (in ms) which is long enough to execute *one* keyboard action. For the method `type` (which is "press/release character by character""), a multiple if this value is chosen. Before leaving the paste/type method, the interval gets reset by Sakuli to the default Sahi status interval.

This setting is not needed if Sikuli does keyboard actions on GUIs not controlled by Sahi.

`Application("...").getRegion()` returns `NULL` or an `NullPointerException`

On **Ubuntu** or other **Linux** based OS check if the packe `wmctrl` is installed. If not install it via:

```
sudo apt-get install wmctrl
```

TODO

What integration possibilities do we have?

- Monitoring Systems
- Continuous Integration Systems
- Weitere Integration möglich, contact us ;-)

Forwarder

TODO Liste der Forwarder hinzufügen

Sakuli can hand over test result to "**Forwarder**", which can be currently **GearmanD** servers (such as Nagios monitoring systems with mod-gearman) and **JDBC databases**. If no forwarder is defined, a result summary is printed out in the end of a suite.

For the configuration of forwarders on the OMD server side, see Forwarder in OMD

- Setting up Sakuli client to send results to the **Database**
- Setting up Sakuli client to submit results to the **Gearman Forwarder**

Gearman Forwarder

This page describes how the results of the Sakuli tests **example_windows7/ubuntu/opensuse** can be transmitted directly into the **Gearman result queue** of the monitoring system.

- Nagios **host**: property `sakuli.forwarder.gearman.nagios.hostname` (define globally or per suite)
- Nagios **service**: property `testsuite.id` in `testsuite.properties`

OMD Configuration

Enable the site's mod-gearman server

Stop the OMD site:

```
OMD[sakuli]:~$ omd stop
```

Start the OMD configuration menu

```
OMD[sakuli]:~$ omd config
```

Select *Distributed Monitoring* and

- -> `GEARMAND` -> `ON`
- -> `GEARMAND_PORT` -> `__OMD_IP__:__GEARMAN_PORT__` (default: 4730)
- -> `GEARMAN_NEB` -> `ON`
- -> `GEARMAN_WORKER` -> `OFF` # only if workers aren't still used
- -> `MOD_GEARMAN` -> `ON`

As Sakuli only uses the *result queue* of gearmand, you can disable all other queues unless you are using mod-gearman for regular checks:

```
OMD[sakuli]:~$ vim ~/etc/mod-gearman/server.cfg
eventhandler=no
services=no
hosts=no
do_hostchecks=no
```

At the time of the creation of this documents, Sakuli does not encrypt any gearman results by default. Therefore, set `accept_clear_results` in `server.cfg`:

```
OMD[sakuli]:~$ vim ~/etc/mod-gearman/server.cfg
accept_clear_results=yes
```

Restart OMD:

```
OMD[sakuli]:~$ omd start
```

Create a Nagios service

Create a check_command, which will be executed only if Nagios did not receive a Sakuli result within the last 30 minutes. This ensures that you get a notification even if no passive check results arrive in Nagios at all:

```
vim ~/etc/nagios/conf.d/commands.cfg

define command {
  command_name                    check_dummy
  command_line                    $USER1$/check_dummy $ARG1$ $ARG2$
}
```

Create a host object for the Sakuli client:

```
OMD[sakuli]:~$ vim etc/nagios/conf.d/hosts.cfg

define host {
  host_name                       sakuli_client
  alias                           Sakuli Client
  address                         [IP]
  use                             generic-host
}
```

Create the following service object for the first test case. *freshness_threshold* should be slightly higher than the interval Sakuli tests are planned (see also RRD heartbeat)

```
vim ~/etc/nagios/conf.d/services.cfg

OMD[sakuli]:~$ vim etc/nagios/conf.d/services.cfg

define service {
  # service_description          example_windows7
  # service_description          example_opensuse
  service_description            example_ubuntu
  host_name                      sakuli_client
  use                            generic-service,srv-pnp
  active_checks_enabled          0
  check_command                  check_dummy!3!'Did not receive any Sakuli result within 3 minutes.'
  check_freshness                1
  freshness_threshold            180
  passive_checks_enabled         1
}
```

Reload OMD:

```
omd reload
```

Now open Thruk; you should see now the Sakuli host with one service attached:

The check is waiting now for check results from a Sakuli client.

Sakuli gearman forwarder configuration

On the Sakuli client you must set the global properties for the gearman receiver. For this, edit `sakuli.properties` in the folder containing the test suites (you can copy the lines from `__SAKULI_HOME__/conf/sakuli-default.properties`):.

```
__INST_DIR__/example_test_suites/sakuli.properties:

sakuli.forwarder.gearman.enabled=true
sakuli.forwarder.gearman.encryption=false
sakuli.forwarder.gearman.secret.key=secret_password
sakuli.forwarder.gearman.server.host=__GEARMAN_IP__
sakuli.forwarder.gearman.server.port=[Gearman Port defined in "omd config" (default:4730)]
sakuli.forwarder.gearman.server.queue=check_results

# Nagios host where all Sakuli services are defined on. If neccessary, override this value per test suite.
# (Nagios service name is defined by testsuite.properties -> suiteID)
sakuli.forwarder.gearman.nagios.hostname=sakuli_client
sakuli.forwarder.gearman.nagios.check_command=check_sakuli
```

Using AES encryption (optional)

The gearman forwarder supports AES encryption when sending checked results to the OMD server. The AES encryption uses a 32 byte (256 bit) secret key that has to be given in the properties.

```
sakuli.forwarder.gearman.encryption=true
sakuli.forwarder.gearman.secret.key=secret_password
```

In case you get a `java.lang.security.InvalidKeyException` with error message *"Illegal key size or default parameters"* you probably need to enable unlimited strength security policies in your Java JRE. This is done by adding a special security policy JAR to the Java JRE lib directory. For the Java JRE 8, take a look at Oracle - Java Cryptography Extension 8.

On the server side (OMD) you have to enable the encryption feature of mod-gearman. Therefore the following two steps are necessary:

1) Set the server side encryption password:

```
OMD[sakuli]:~$ echo "secret_password" > ~/etc/mod-gearman/secret.key
```

2) Enable the `encryption` and disable `accept_clear_results` in the config file:

```
OMD[sakuli]:~$ vim ~/etc/mod-gearman/server.cfg

encryption=yes
accept_clear_results=yes
```

Test result transmission to OMD

Execute the example test case again:

- **Ubuntu**: `sakuli run __INST_DIR__/example_test_suites/example_ubuntu/`
- **openSUSE**: `sakuli run __INST_DIR__/example_test_suites/example_opensuse/`
- **Windows7**: `sakuli run __INST_DIR__\example_test_suites\example_windows7\`
- **Windows8**: `sakuli run __INST_DIR__\example_test_suites\example_windows8\`

The service should change its status to:

TODO: link zu (manual/integration/forwarder-sql-databse) TODO: describe OMD integration trough SQL Database

Database Forwarder

This page describes how the results of the Sakuli tests **example_windows7/ubuntu/opensuse** can be written into a **MySQL database** which is then checked asynchronously by the monitoring system with *check_mysql_health*.

OMD Configuration

Enabling the site's MySQL Database

If not already done for other reasons, a site-specific MySQL instance has to be started. That's the place where all Sakuli clients will store their check results.

Stop the OMD site and start the OMD configuration menu:

```
OMD[sakuli]:~$ omd stop
 ...
OMD[sakuli]:~$ omd config
```

Select *Addons -> MYSQL -> ON* and exit the menu.

Open `~/.my.cnf` and set the following values:

```
OMD[sakuli]:~$ vim .my.cnf
# make sure to choose a free port, e.g. 3007
port = __DB_PORT__
# bind on all interfaces
bind-address = 0.0.0.0
# enable network access
#skip-networking
```

Create the system tables for the new database and start OMD afterwards. You should see now OMD coming up with a **dedicated MySQL instance**:

```
OMD[sakuli]:~$ mysql_install_db
OMD[sakuli]:~$ omd start
Starting dedicated Apache for site sakuli...OK
Starting MySQL... .OK.
Starting rrdcached...OK
Starting npcd...OK
Starting nagios...OK
Initializing Crontab...
```

create Sakuli DB and user

Create the **Sakuli database**:

```
OMD[sakuli]:~$ mysql < __TEMP__/sakuli-vx.x.x-SNAPSHOT/setup/database/create_sakuli_database.sql
```

Create the **database user**:

```
OMD[sakuli]:~$ mysql
  grant ALL on sakuli.* to '__DB_USER__'@'%' identified by '__DB_PASSWORD__';
  flush privileges;
  quit
```

Check the connection with *check_mysql_health*:

```
OMD[sakuli]:~$    lib/nagios/plugins/check_mysql_health -H __DB_IP__ --username __DB_USER__ --password __DB_PASSWORD__
  OK - 0.24 seconds to connect as sakuli | connection_time=0.2366s;1;5
```

create Nagios check

Nagios fetches Sakuli check results using the plugin check_mysql_health, which is already contained in OMD.

CheckMySQLHealthSakuli.pm

The Perl module `CheckMySQLHealthSakuli.pm` enhances the functionality of *check_mysql_health* by introducing the mode `--my-sakuli-suite`.

Create a config directory for *check_mysql_health* and **copy the module** there:

```
OMD[sakuli]:~$ mkdir etc/check_mysql_health
OMD[sakuli]:~$ cp __TEMP__/setup/nagios/CheckMySQLHealthSakuli.pm etc/check_mysql_health/
```

resource.cfg

Set **USER macros** for static vars in `resource.cfg`, which makes it easy to use them in all nagios checks:

```
OMD[sakuli]:~$ vim ~/etc/nagios/resource.cfg
  # database name
  $USER10$=sakuli
  # database user
  $USER11$=__DB_USER__
  # database password
  $USER12$=__DB_PASSWORD__
  # database port
  $USER13$=__DB_PORT__
  # check_mysql_health module dir
  $USER15$=/opt/omd/sites/sakuli/etc/check_mysql_health/
  # database IP
  $USER16$=__MySQL_Database_IP__
```

Nagios configuration

Create a new **check_command**:

```
OMD[sakuli]:~$ vim etc/nagios/conf.d/commands.cfg

# check_command for Sakuli
# --name = Suite ID
# --name2 = number of seconds the last result is allowed to be old
define command {
  command_name              check_sakuli
  command_line              $USER1$/check_mysql_health \
      --hostname=$USER16$ \
    --database=$USER10$ \
    --username=$USER11$ \
    --password=$USER12$ \
    --mode=my-sakuli-suite \
    --name='$ARG1$' \
    --name2=$ARG2$ \
    --report=html \
    --port=$USER13$ \
    --with-mymodules-dyn-dir=$USER15$
}
```

Create a **host object** for Sakuli database checks (the checks are executed on the *local* machine, but belong logically to *sakuli_client*):

```
OMD[sakuli]:~$ vim etc/nagios/conf.d/hosts.cfg

define host {
  host_name                   sakuli_client
  alias                       sakuli_client
  address                     __SAKULI_CLIENT_IP__
  use                         generic-host
}
```

Create the following **service object** for the first test case. Note the ARG2 in check_command: the database check will only evaluate the last result if it is max. 180 seconds old. If older, the check will return UNKNOWN. (For comparison: this is equivalent to "freshness_threshold" if you would use the gearman forwarder. In any case, you should set the RRD heartbeat to the same value to get a gap in the graph if recent client results are missing.)

```
OMD[sakuli]:~$ vim etc/nagios/conf.d/services.cfg

define service {
  # service_description       example_windows7
  # service_description       example_opensuse
  service_description         example_ubuntu
  host_name                   sakuli_client
  use                         generic-service,srv-pnp
  check_command               check_sakuli!sakuli_demo!180
}
```

Reload OMD:

```
omd reload
```

Now open Thruk; you should see now the Sakuli host with one service attached:

Re-scheduling this service should display the UNKNOWN message that the requested suite could not be found. For the moment, this is ok:

Sakuli database forwarder parameter

On the Sakuli client you must set the global properties for the database receiver. For this, edit `sakuli.properties` in the folder containing the test suites (you can copy the lines from `__SAKULI_HOME__/conf/sakuli-default.properties`):

```
__INST_DIR__/example_test_suites/sakuli.properties:

# DEFAULT: false
sakuli.forwarder.database.enabled=true

#DEFAULT: mysql
sakuli.forwarder.database.jdbc.driverClass=com.mysql.jdbc.Driver
sakuli.forwarder.database.host=__DB_IP__
sakuli.forwarder.database.port=__DB_PORT__
sakuli.forwarder.database=sakuli
sakuli.forwarder.database.user=__DB_USER__
sakuli.forwarder.database.password=__DB_PW__
```

Test result transmission to OMD

Execute the example test case again:

- **Ubuntu**: `sakuli run __INST_DIR__/example_test_suites/example_ubuntu/`
- **openSUSE**: `sakuli run __INST_DIR__/example_test_suites/example_opensuse/`

- **Windows 7**: `sakuli run __INST_DIR__\example_test_suites\example_windows7\`
- **Windows 8**: `sakuli run __INST_DIR__\example_test_suites\example_windows8\`

The service should change its status to:

Database cleanup (optional)

Sakuli's database can get very large over time. Use the following database maintenance script to keep only the most recent data.

```
OMD[sakuli]:~$ cp `__SAKULI_HOME__/bin/helper/mysql_purge.sh local/bin/`
```

Create a OMD crontab entry for a automatic database cleanup of data older than 90 days:

```
OMD[sakuli]:~$ vim etc/cron.d/sakuli
00 12 * * * $OMD_ROOT/local/bin/mysql_purge.sh 90 > /dev/null 2>&1
```

After that, reload the OMD crontab:

```
OMD[sakuli]:~$ omd reload crontab
Removing Crontab...OK
Initializing Crontab...OK
```

Icinga2 API Forwarder

This page describes how the results of Sakuli tests can be sent to the REST API of an Icinga2 monitoring instance.

Icinga2 configuration

Enable the Icinga2 API

The steps to enable the Icinga2 API are described in the REST API documentation.

Create a Icinga2 service

Create a **check_command**, which will be executed only if Icinga did not receive a Sakuli result within a certain time. This ensures that you get a notification even if no passive check results arrive in Icinga at all:

```
vim /etc/icinga2/conf.d/commands.conf

object CheckCommand "check_dummy" {
   import "plugin-check-command"
   command = [
     PluginDir + "/check_dummy","$dummy_state$","$dummy_text$"
   ]
   vars.dummy_state = 0
   vars.dummy_text = "Check was successful."
}

object CheckCommand "check_sakuli" {
   import "check_dummy"
   vars.dummy_state = 3
   vars.dummy_text = "No passive Sakuli check result received."
}
```

Create a **host** object for the Sakuli client:

```
vim /etc/icinga2/conf.d/hosts.conf

object Host "sakuliclient01" {
   import "generic-host"
   address = [IP]
}
```

Create the following **service** object for the first test case. *freshness_threshold* should be slightly higher than the interval Sakuli tests are planned (if you are using PNP4Nagios, see also RRD heartbeat)

```
object Service "sakuli_demo" {
  import "generic-service"
  host_name = "sakuliclient01"
  check_command = "check_sakuli"
  enable_active_checks = 0
  enable_passive_checks = 1
  enable_flapping = 0
  volatile = 1
  enable_perfdata = 1
}
```

Reload Icinga2:

```
service icinga2 reload
```

Now open Icingaweb2; you should see the Sakuli host with the service *"sakuli_demo"* attached:

The check is waiting now for check results from a Sakuli client.

Sakuli Icinga2 forwarder parameter

On the Sakuli client you must set the global properties for the Icinga2 receiver. For this, edit `sakuli.properties` in the folder containing the test suites (you can copy the lines from `__SAKULI_HOME__/conf/sakuli-default.properties`):
sakuli.forwarder.icinga2.enabled=false

```
sakuli.forwarder.icinga2.api.host=__ICINGA_IP__
sakuli.forwarder.icinga2.api.port=5665
sakuli.forwarder.icinga2.api.username=icingasakuli
sakuli.forwarder.icinga2.api.password=icingasakuli
sakuli.forwarder.icinga2.hostname=sakuliclient01
```

Graph settings

Icinga2 integration is very new; we did not yet dive into the configuration of Graphite or Grafana graphs. The only supported graphing solution is PNP4Nagios. Nevertheless you are welcome to contribute graph templates for Grafana and/or Graphite!

PNP4Nagios

Set the RRD storage type of PNP to MULTIPLE, which creates one RRD file per perfdata label:

```
echo "RRD_STORAGE_TYPE = MULTIPLE" > /etc/pnp4nagios/check_commands/check_sakuli.cfg
```

Copy the PNP graph template `check_sakuli.php` from `%SAKULI_HOME%/setup/nagios/` on the client to `/usr/share/nagios/html/pnp4nagios/templates/` on the Icinga2 server.

Grafana

tbd

Graphite

tbd

Test result transmission to Icinga2

Execute the example test case again:

- **Ubuntu**: `sakuli run __INST_DIR__/example_test_suites/example_ubuntu/`
- **openSUSE**: `sakuli run __INST_DIR__/example_test_suites/example_opensuse/`
- **Windows 7**: `sakuli run __INST_DIR__\example_test_suites\example_windows7\`
- **Windows 8**: `sakuli run __INST_DIR__\example_test_suites\example_windows8\`

The service in Icnga2 should change its status to:

Sakuli: Integration in OMD

This chapter describes all neccessary steps to configure a **Nagios** compatible monitoring system to receive Sakuli test results. OMD with Thruk as web frontend is highly recommended, but any other Nagios based system will do also.

Requirements

- **OMD** **1.x** installed on a Linux operating system
- a running **OMD site** (here: **sakuli**)

Download

General note: you should download the **same package version of Sakuli** as you did on the clients. **Do not mix versions**.

- Download **Sakuli** from http://labs.consol.de/sakuli/install into a temporary folder `__TEMP__` .

```
cd __TEMP__
wget http://labs.consol.de/sakuli/install/sakuli-vx.x.x-SNAPSHOT.zip
```

- decompress it:

```
unzip sakuli-vx.x.x-SNAPSHOT.zip
```

General preparations

All following steps should be done as the **OMD site user** (here: "sakuli"):

```
su - sakuli
```

Nagios

Sakuli will produce HTML formatted output. **HTML escaping** in Nagios must be turned off:

```
OMD[sakuli]:~$ vim etc/nagios/cgi.cfg
    escape_html_tags=0
```

Choose a forwarder

Depending on your environment, you can set up on of these two possible forwarder types. Each of them is documented on a single page.

- Setting up Nagios to **receive Gearman results** from Sakuli clients (**recommended**)
    - passive check (via Gearmand result queue)
    - + get results immediately
    - + PNP graphs
    - - no performance value history to do further reporting (despite reading the RRDs again; not recommended)
- Setting up the Sakuli **result database** in OMD and let Nagios check the database
    - active check (against MySQL result database)
    - - results in Nagios always lag behind
    - + PNP graphs
    - + all suite/case/step performance values are stored in database and can be used for further reporting

TODO mention PNP + deliverd default templates

PNP4Nagios

RRD Storage Type

In PNP4Nagios RRD Storage type "MULTIPLE" is of great importance for Sakuli checks, because the number of steps can change by time (=if you are adding/deleting some).

Verify `RRD_STORAGE_TYPE` in `process_perfdata.cfg` :

```
OMD[sakuli]:~$ vim ~/etc/pnp4nagios/process_perfdata.cfg
    RRD_STORAGE_TYPE = MULTIPLE
```

If this value is *"SINGLE"* on your system and you do not want to change it globally, use the *custom check_command* cfg file. PNP4Nagios will then use storage type *"MULTIPLE"* only for this check_command then:

```
OMD[sakuli]:~$ cp __TEMP__/sakuli-vx.x.x-SNAPSHOT/setup/nagios/check_sakuli.cfg ~/etc/pnp4nagios/check_commands/
```

RRD heartbeat

Each RRD file contains a heartbeat value, which determines how much time must pass without any new update, before RRDtool writes an UNKNOWN value (nan) into the data slot (the graph will have a gap then). In PNP4nagios heartbeat is defined at approx. 2 1/2 hours. If your Sakuli check runs only every 2 hours, this value will be fine. But for a 5 minute interval, this is way too long. As a consequence, the graph line will be continuously drawed even Sakuli did no check for two hours. Hence, always make sure to adapt the heartbeat to a value which is slightly higher than the interval of Sakuli checks (and indeally the same as freshness_threshold, if you use the gearman receiver):

```
OMD[sakuli]:~$ cd ~/var/pnp4nagios/perfdata/sakulihost/
# Sakuli check interval: 2 minutes --> RRD heartbeat 3 minutes
OMD[sakuli]:~$ for file in sakuli_e2e_webshop*.rrd; do rrdtool tune $file --heartbeat 1:180; done
```

install PNP graph template

Copy the PNP4nagios graph template into the templates folder:

```
OMD[sakuli]:~$ cp __TEMP__/sakuli-vx.x.x-SNAPSHOT/setup/nagios/check_sakuli.php ~/etc/pnp4nagios/templates/
```

Include CPU/Memory metrics in Sakuli graphs (optional)

If Sakuli reports a long check runtime, it is good to know the CPU/Memory metrics on the Sakuli client machine, because CPU/IO bottlenecks affect Sakuli tests, too.

The following optional enhancement displays the **CPU/Memory graph** lines of the Sakuli test client in the suite/case graph. By setting **custom host macros**, the graph template knows where to fetch these data from.

 (picture shows a Linux client with CPU check, displayed by a yellow line)

add CPU load check (for Linux Sakuli clients)

Add this **command** to `commands.cfg` :

```
define command{
    command_name    check_local_load
    command_line    $USER1$/check_load -w $ARG1$ -c $ARG2$
}
```

Add this **service** to `services.cfg` :

```
define service {
  service_description           CPU_Load
  host_name                 sakuli_client
  use                           generic-service,srv-pnp
  # if Sakuli checks are running on the same machine (as in the demo VM)
  check_command                 check_local_load!2.5,1.5,1!5,3.5,2
  # if Sakuli checks are running on another host than OMD
  check_command                 check_by_ssh!check_load!2.5,1.5,1!5,3.5,2
}
```

Add this **custom host macros** to every Sakuli host in `hosts.cfg` :

```
define host {
    …
    _E2E_CPU_HOST                 sakuli_client
    _E2E_CPU_SVC                  CPU_Load_load5
}
```

Now reload OMD:

```
omd reload
```

You should see now the following service on `sakuli_client` :

Note: The value of `_E2E_CPU_SVC` and `_E2E_MEM_SVC` refer to the file name of the corresponding RRD file. `CPU_Usage_5` for example means to get the the CPU usage data from `$OMD_ROOT/var/pnp4nagios/perfdata/[_E2E_CPU_HOST]/CPU_Usage_5.rrd` .

add CPU/Memory usage check (for Windows Sakuli clients)

Install **NSClient++** on the Windows client. Then add this **command check_nrpe_arg:**

```
vim ~/etc/nagios/conf.d/commands.cfg

define command {
  command_name                  check_nrpe_arg
  command_line                  $USER1$/check_nrpe -H $HOSTADDRESS$ -c $ARG1$ -a $ARG2$
}
```

Then add these **services** to Nagios:

```
define service {
  service_description           CPU_Usage
  host_name                  win7sakuli
  use                           generic-service,srv-pnp
  check_command                 check_nrpe_arg!CheckCPU!warn=80% crit=90% time=15 time=5 time=1 ShowAll
}

define service {
  service_description           Mem_Usage
  host_name                  win7sakuli
  use                           generic-service,srv-pnp
  check_command                 check_nrpe_arg!CheckMem!MaxWarn=80% MaxCrit=90% ShowAll type=page type=paged type=phy
}
```

Add these host macros to every Nagios host where Sakuli checks are defined:

```
    _E2E_CPU_HOST                 win7sakuli
    _E2E_CPU_SVC                  CPU_Usage_5
    _E2E_MEM_HOST                 win7sakuli
    _E2E_MEM_SVC                  Mem_Usage
```

Now reload OMD:

```
omd reload
```

Note: The value of `_E2E_CPU_SVC` and `_E2E_MEM_SVC` refer to the file name of the corresponding RRD file. `CPU_Usage_5` for example means to get the the CPU usage data from `$OMD_ROOT/var/pnp4nagios/perfdata/[_E2E_CPU_HOST]/CPU_Usage_5.rrd`.

XML update delay

As soon as the created services produce perfdata for the first time, their XML file created by PNP4Nagios will also contain the host macros created in the step before. If not, check if `XML_UPDATE_DELAY` in `etc/pnp4nagios/process_perfdata.cfg` is set too high.

Change PNP working mode

OMD runs PNP by default in **Bulk Mode with NPCD and npcdmod.o**. In this mode the Nagios broker module `npcdmod.o` reads the performance directly from the monitoring core and writes them in *var/spool/perfdata*. This data are not expandable with **custom macros** - therefore the mode has to be changed to **Bulk Mode with NPCD**. (the performance of both modes will be equal).

In this mode the monitoring core itself writes perfdata to the spool directory (instead of *npcdmod.o*). The format of this data can be freely defined by adapting `service_perfdata_file_template`. In the following code block you can see that the four **custom host macros** were added to this template string. Perfdata files are then moved to *var/spool/perfdata* every 15 seconds by the monitoring core.

Make sure to replace the OMD site name placeholder `__OMD_SITE__` with your site name! (in *vim* type `:%s/__OMD_SITE__/yoursitename/g`)

```
vim ~/etc/nagios/nagios.d/pnp4nagios.cfg

process_performance_data=1

# COMMENT THE LINE BELOW
# broker_module=/omd/sites/__OMD_SITE__/lib/npcdmod.o config_file=/omd/sites/__OMD_SITE__/etc/pnp4nagios/npcd.cfg

# services
service_perfdata_file=/omd/sites/__OMD_SITE__/var/pnp4nagios/service-perfdata
service_perfdata_file_template=DATATYPE::SERVICEPERFDATA\tTIMET::$TIMET$\tHOSTNAME::$HOSTNAME$\tSERVICEDESC::$SERVICEDES
service_perfdata_file_mode=a
service_perfdata_file_processing_interval=15
service_perfdata_file_processing_command=omd-process-service-perfdata-file

# hosts
host_perfdata_file=/omd/sites/__OMD_SITE__/var/pnp4nagios/host-perfdata
host_perfdata_file_template=DATATYPE::HOSTPERFDATA\tTIMET::$TIMET$\tHOSTNAME::$HOSTNAME$\tHOSTPERFDATA::$HOSTPERFDATA$\t
host_perfdata_file_mode=a
host_perfdata_file_processing_interval=15
host_perfdata_file_processing_command=omd-process-host-perfdata-file
```

Check if the perfdata processing commands are present:

```
vim ~/etc/nagios/conf.d/pnp4nagios.cfg

define command{
    command_name    omd-process-service-perfdata-file
    command_line    /bin/mv /omd/sites/__OMD_SITE__/var/pnp4nagios/service-perfdata /omd/sites/__OMD_SITE__/var/pnp4nag:
}

define command{
    command_name    omd-process-host-perfdata-file
    command_line    /bin/mv /omd/sites/__OMD_SITE__/var/pnp4nagios/host-perfdata /omd/sites/__OMD_SITE__/var/pnp4nagios,
}
```

Restart the OMD site to unload the *npcdmod.o* module:

```
omd restart
```

Test

First reschedule the CPU/Mem check on the sakuli client. It can take several minutes to store the values in the RRD database. As soon as you can see "real" values in the PNP4Nagios graph of "CPU Load" (instead of " `-nan` "), restart the Sakuli check. The Sakui graph should now contain also CPU/Memory values.

TODO SM how to use!

TODO TS: write how to execute in Jenkins

TODO TS: write how to execute in Jenkins

TODO: remove OMD stuff and described sql schema and how results will be placed there.

Database Forwarder

This page describes how the results of the Sakuli tests **example_windows7/ubuntu/opensuse** can be written into a
MySQL database which is then checked asynchronously by the monitoring system with *check_mysql_health*.

OMD Configuration

Enabling the site's MySQL Database

If not already done for other reasons, a site-specific MySQL instance has to be started. That's the place where all Sakuli
clients will store their check results.

Stop the OMD site and start the OMD configuration menu:

```
OMD[sakuli]:~$ omd stop
  ...
OMD[sakuli]:~$ omd config
```

Select *Addons -> MYSQL -> ON* and exit the menu.

Open `~/.my.cnf` and set the following values:

```
OMD[sakuli]:~$ vim .my.cnf
# make sure to choose a free port, e.g. 3007
port = __DB_PORT__
# bind on all interfaces
bind-address = 0.0.0.0
# enable network access
#skip-networking
```

Create the system tables for the new database and start OMD afterwards. You should see now OMD coming up with a
dedicated MySQL instance:

```
OMD[sakuli]:~$ mysql_install_db
OMD[sakuli]:~$ omd start
Starting dedicated Apache for site sakuli...OK
Starting MySQL... .OK.
Starting rrdcached...OK
Starting npcd...OK
Starting nagios...OK
Initializing Crontab...
```

create Sakuli DB and user

Create the **Sakuli database**:

```
OMD[sakuli]:~$ mysql < __TEMP__/sakuli-vx.x.x-SNAPSHOT/setup/database/create_sakuli_database.sql
```

Create the **database user**:

```
OMD[sakuli]:~$ mysql
   grant ALL on sakuli.* to '__DB_USER__'@'%' identified by '__DB_PASSWORD__';
   flush privileges;
   quit
```

Check the connection with *check_mysql_health*:

```
OMD[sakuli]:~$    lib/nagios/plugins/check_mysql_health -H __DB_IP__ --username __DB_USER__ --password __DB_PASSWORD__
  OK - 0.24 seconds to connect as sakuli | connection_time=0.2366s;1;5
```

◀ ▬▬ ▶

create Nagios check

Nagios fetches Sakuli check results using the plugin check_mysql_health, which is already contained in OMD.

CheckMySQLHealthSakuli.pm

The Perl module `CheckMySQLHealthSakuli.pm` enhances the functionality of *check_mysql_health* by introducing the mode `--my-sakuli-suite`.

Create a config directory for *check_mysql_health* and **copy the module** there:

```
OMD[sakuli]:~$ mkdir etc/check_mysql_health
OMD[sakuli]:~$ cp __TEMP__/setup/nagios/CheckMySQLHealthSakuli.pm etc/check_mysql_health/
```

resource.cfg

Set **USER macros** for static vars in `resource.cfg`, which makes it easy to use them in all nagios checks:

```
OMD[sakuli]:~$ vim ~/etc/nagios/resource.cfg
  # database name
  $USER10$=sakuli
  # database user
  $USER11$=__DB_USER__
  # database password
  $USER12$=__DB_PASSWORD__
  # database port
  $USER13$=__DB_PORT__
  # check_mysql_health module dir
  $USER15$=/opt/omd/sites/sakuli/etc/check_mysql_health/
  # database IP
  $USER16$=__MySQL_Database_IP__
```

Nagios configuration

Create a new **check_command**:

```
OMD[sakuli]:~$ vim etc/nagios/conf.d/commands.cfg

# check_command for Sakuli
# --name = Suite ID
# --name2 = number of seconds the last result is allowed to be old
define command {
  command_name                    check_sakuli
  command_line                    $USER1$/check_mysql_health \
      --hostname=$USER16$ \
    --database=$USER10$ \
    --username=$USER11$ \
    --password=$USER12$ \
    --mode=my-sakuli-suite \
    --name='$ARG1$' \
    --name2=$ARG2$ \
    --report=html \
    --port=$USER13$ \
    --with-mymodules-dyn-dir=$USER15$
}
```

Create a **host object** for Sakuli database checks (the checks are executed on the *local* machine, but belong logically to *sakuli_client*):

```
OMD[sakuli]:~$ vim etc/nagios/conf.d/hosts.cfg

define host {
  host_name                     sakuli_client
  alias                         sakuli_client
  address                       __SAKULI_CLIENT_IP__
  use                           generic-host
}
```

Create the following **service object** for the first test case. Note the ARG2 in check_command: the database check will only evaluate the last result if it is max. 180 seconds old. If older, the check will return UNKNOWN. (For comparison: this is equivalent to "freshness_threshold" if you would use the gearman forwarder. In any case, you should set the RRD heartbeat to the same value to get a gap in the graph if recent client results are missing.)

```
OMD[sakuli]:~$ vim etc/nagios/conf.d/services.cfg

define service {
  # service_description         example_windows7
  # service_description         example_opensuse
  service_description           example_ubuntu
  host_name                     sakuli_client
  use                           generic-service,srv-pnp
  check_command                 check_sakuli!sakuli_demo!180
}
```

Reload OMD:

```
omd reload
```

Now open Thruk; you should see now the Sakuli host with one service attached:

Re-scheduling this service should display the UNKNOWN message that the requested suite could not be found. For the moment, this is ok:

Sakuli database forwarder parameter

On the Sakuli client you must set the global properties for the database receiver. For this, edit `sakuli.properties` in the folder containing the test suites (you can copy the lines from `__SAKULI_HOME__/conf/sakuli-default.properties`):

```
__INST_DIR__/example_test_suites/sakuli.properties:

# DEFAULT: false
sakuli.forwarder.database.enabled=true

#DEFAULT: mysql
sakuli.forwarder.database.jdbc.driverClass=com.mysql.jdbc.Driver
sakuli.forwarder.database.host=__DB_IP__
sakuli.forwarder.database.port=__DB_PORT__
sakuli.forwarder.database=sakuli
sakuli.forwarder.database.user=__DB_USER__
sakuli.forwarder.database.password=__DB_PW__
```

Test result transmission to OMD

Execute the example test case again:

- **Ubuntu**: `sakuli run __INST_DIR__/example_test_suites/example_ubuntu/`
- **openSUSE**: `sakuli run __INST_DIR__/example_test_suites/example_opensuse/`

- **Windows 7**: `sakuli run __INST_DIR__\example_test_suites\example_windows7\`
- **Windows 8**: `sakuli run __INST_DIR__\example_test_suites\example_windows8\`

The service should change its status to:

Database cleanup (optional)

Sakuli's database can get very large over time. Use the following database maintenance script to keep only the most recent data.

```
OMD[sakuli]:~$ cp `__SAKULI_HOME__/bin/helper/mysql_purge.sh local/bin/`
```

Create a OMD crontab entry for a automatic database cleanup of data older than 90 days:

```
OMD[sakuli]:~$ vim etc/cron.d/sakuli
00 12 * * * $OMD_ROOT/local/bin/mysql_purge.sh 90 > /dev/null 2>&1
```

After that, reload the OMD crontab:

```
OMD[sakuli]:~$ omd reload crontab
Removing Crontab...OK
Initializing Crontab...OK
```

Developer Documentation

In this section you will find a few helpful documentation for developers who want to contribute.

- Basic Installation for Developers
- Jenkins-CI http://labs-build.consol.de/view/Sakuli/
- How-To-Release
- JavaFX installation for integration testing
- Some additional Maven aspects
- (only for Linux) Usage of the sikuliX supplemental-package

Example projects on GitHub

- **ConSol/sakuli-examples**
- **toschneck/sakuli-example-bakery-testing**
- **ConSol/sakuli-example-testautomation-day**

Change Log of Sakuli Releases

TODO ensure that file is included at installer package

Version 1.0.2 (Bugfix + some small features)

- issue #210: upgrade Sahi to version `5.1` due to Sahi compatibility issue with Chrome 53+ - _click
- fix docker images
  - Chrome don't startup under CentOS Docker image, see also https://github.com/ConSol/docker-headless-vnc-container/issues/2
  - XFCE window manager don't startup under CentOS Docker image, see also https://github.com/ConSol/docker-headless-vnc-container/issues/4
  - use `SAKULI_VERSION` ARG in Dockerfiles, to have more flexible to build images
- issue #215 add java-based Sakuli Docker images
- issue #91: add AES encryption option for Gearman forwarder module
  - add Java JCE extension to Docker images
- fix #216: set `dom.storage.enabled` to true in firefox pref.js
- add Sakuli-Example page https://github.com/ConSol/sakuli-examples
- fix #177 add description for the javaDSL and update the documentation
- issue #205: use maven-jgitflow for releases and branching

Version 1.0.1 (Bugfix)

- fix #190: fix Docker centos image: use tagged version `consol/centos-xfce-vnc:1.0.1`
- Use consistent file naming and fix broken links in docs
- fix example_xfce for new centos 7 version

Version 1.0.0

- First step tutorial and https documentation. Fixes #161, fixes #53 partially.
- fix #32 highlight function on linux does not work (in underlying SikuliX library)
- close #102 add method `dragAndDropTo` to the `Region` object
- Changed order of properties.
- Improve example_xfce:
  - Replaced calculator screenshot by a small one.
  - add mouse move action to example_xfce
- close #139 remove PDF als download content type, to enable to use the browser PDF viewer
- close #139 add start chrome + firefox maximised (firefox have to hold the file `localstore.rdf` in his profile folder)
- close #168 add reboot hint if user install the package `Windows environement changes`
- update the installer translation to the recommend one from https://github.com/izpack/izpack/tree/master/izpack-core/src/main/resources/com/izforge/izpack/bin/langpacks/installer
- fix maven snapshot repository path of the `labs.consol.de` maven-repository

Version 0.9.3

- Move to a new binary starter for Windows and Linux (sakuli.exe / sakuli), #150:
  - Change syntax from the new starter to:

```
Usage: sakuli[.exe] COMMAND ARGUMENT [OPTIONS]

      sakuli -help
      sakuli -version
      sakuli run <sakuli suite path> [OPTIONS]
      sakuli encrypt <secret> [OPTIONS]


Commands:
      run        <sakuli suite path>
      encrypt        <secret>


Options:
      -loop      <seconds>     Loop this suite, wait n seconds between
                              executions, 0 means no loops (default: 0)
      -javaHome  <folder>      Java bin dir (overrides PATH)
      -javaOption <java option> JVM option parameter, e.g. '-agentlib:...'
      -preHook   <programpath> A program which will be executed before a
                              suite run (can be added multiple times)
      -postHook  <programpath> A program which will be executed after a
                              suite run (can be added multiple times)
      -D        <JVM option>   JVM option to set a property at runtime,
                              overrides file based properties
      -browser   <browser>     Browser for the test execution
                              (default: Firefox)
      -interface <interface>   Network interface card name, used by
                              command 'encrypt' as salt
      -sahiHome  <folder>      Sahi installation folder
      -version                 Version info
      -help                    This help text
```

- modify VNC dokumentation to flag `-preHook` and `postHook`

- change documentation and docker scripts to new starter syntax `sakuli run TEST_SUITE [OPTION]`
- add binaries `sakuli` and `sakuli.exe` from repo https://github.com/ConSol/sakuli-go-wrapper
- remove `sakuli.sh/sakuli.bat`
- Add new forwarder module `Icinga2`, see #145:
  - Rest client to send the results to Icinga2 API
  - new property `sakuli.forwarder.gearman.nagios.template.suite.summary.maxLength` to cut to long output due to error messages
  - introduce `sakuli.forwarder.icinga2` properties // consolidate `sakuli.forwarder.database` properties
- Separate examples for Window 7 and Windows 8
- close #118 improved output of nagios messages
- close #151 add a bunch of Windows registry settings to the installer, to improve the test stability. Disables graphical effects, screen saver and error reporting.
- fix #135 Environment similarity:
  - Extract constant Environment#DEFAULT_SIMILARITY to `sakuli-default.properties` as `sakuli.environment.similarity.default`:
  - Set default similarity to `0.99`
- close #163: add clean up method, which release all modifier keys before a test case will startet and at the teardown phase
- fix #162: release keys didn't work correctly => update to sikulix version 1.1.998 and add function "run as admin" to dev suites
- add an Around Aspect to `net.sf.sahi.util.Utils.getCommandTokens` to fix parsing errors during calling native commands, see http://community.sahipro.com/forums/discussion/8552/sahi-os-5-0-and-chrome-user-data-dir-containing-spaces-not-working
- Documentation how to solve increasing sahi profile folders. Closes #164.
- reduce wait times for example test suites
- fix `firefox_portable` executable path in `browser.xml`: replace it with $userDir relativ path
- consolidate forwarder properties: adjust `jdbc.*` properties to `sakuli.forwarder.database.*` properties
- improve logging of database receiver
- fix #153 `sakuli.log.maxAge` error, is smaller then 1
- check_sakuli.php: added wrapper for function declarations to fix errors in PNP basket (cannot redefine...)
- cl: update installer with special cl installer preselected options

- close #155: add environment variables to --version output
- fix for #158: linux installer correct firefox var to `MOZ_DISABLE_OOP_PLUGINS`
- Added ff_purge_profile.bat to helper scripts (delete sqlite file before each run)
- close #155: add -version parameter to Sakuli starter (sakuli / sakuli.exe)
- close #153 log data rotation: *add a property sakuli.log.maxAge in days (default 14 days)* deletes all files that are older than the defined days in the folder `sakuli.log.folder`

Version 0.9.2

- add setting some firefox variables (`MOZ_DISABLE_OOP_PLUGINS` , `MOZ_DISABLE_AUTO_SAFE_MODE` , `MOZ_DISABLE_SAFE_MODE_KEY`) for UI testing to the installer, see #158.
- Executable JAR installer `sakuli-vX.X.X-installer.jar` , downloadable via , see #24.
  - The installer contains a complete Sakuli setup and the following options:
    - **1)** will set/update the environment variable `__SAKULI_HOME__` to this version.
    - **2)** will set/update the environment to a recommend UI testing configuration. In examples disables the Firefox safe.
    - **3)** will install one example test suite per OS which will help you to test and understand Sakuli.
    - **4)** will install Firefox Portable, which can be used exclusively for Sakuli Tests.
    - **5)** will install QRes, a open source screen mode changer (Windows only)
  - modify docker images to new headless linux installer
  - custom sahi `browser_types.xml` for firefox, firefox_portable, chrome
- Property `sikuli.typeDelay` now also set the `RobotDesktop.stdAutoDelay` to make the delay more effective, default is `0.0` , #154.
- issue #149 add `Application.kill()` to force closeting an app without "save prompts"
- issue #94: disable highlighting in case of an exception
- docker container: modify test suite permissions after test run in `sakuli_startup.sh`
- Improve typing handling #154:
  - typing all special characters via unicode keyboard shortcuts
  - keyboard mapping only contains alphanumeric characters, so region.type now will work with all local keyboards, because of typing special characters via UFT-8
  - Mac's currently not supports directly typing UFT-8 keys, by default see https://en.wikipedia.org/wiki/Unicode_input#In_Mac_OS . Unicode typing will only be used if correct keyboard is activated.
- improve takeScreenshot: now also can handle with absolute paths
- rename `Region.takeScreenShot` to `Region.takeScreenshot`
- fix #107: Fix Sikuli `LinuxUtil` Implementation of:
  - Focus application
  - Closing application
  - Make output and control of native commands more flexible
- include Sahi base installation to java DSL, #24
- modify properties, so that all default values are included
  - add check that `testsuite.id` will be defined at least
  - allow to write test without explicit image folder path, to enable java tests
- added docker-container `sakuli-omd-labs-ubuntu` for a Sakuli preconfigered OMD
- sakuli.bat: added exitcode, fixes bug #128 (Errors with paths containing spaces)
- fix #142: remove some not valid windows escape chars like `'` or `"` to prevent a InvalidPathException in SakuliStarter arguments
- docker-containers.md: Added hint for boot2docker users.
- check_sakuli.php: fixed #132 (suite runtime)
- close #103: make docker-container able to override the running testsuite in `docker run` `CMD` arguments
  - make the `sakuli.sh` command line parameters usable in `docker run` `CMD` arguments, like for example `docker run consol/sakuli-centos-xfce '--run $SAKULI_TEST_SUITE --browser chrome'`
- Added documentation how to configure HTTPS in Sahi. #53
- Rename README.md to index.md in case of https://readthedocs.org/projects/sakuli/
- headless-linux.md: Added crontab documentation.

Version 0.9.1

- fix #116 wrong comma in gearman output
- sakuli.bat: added SAKULI_HOME fallback if env var not set #124
- sakuli.bat: added javahome parameter, added JVM option passing #122
- update sikuliX version to 1.1.993
- Merge branch 'dev-v0.4' into dev
- sakuli.sh: JVM options (-D) allowed. #75
- Improve Nagios monitoring integration:
    - check_sakuli.json: added first JSON template for grafana
    - check_sakuli.php: logfile destination now contains hostname and service description
    - check_sakuli.php: removed UNIT var. Everything is in seconds.
    - CheckMySQLHealthSakuli.pm: no perfdata on stale result (fixes #120), small output improvements
    - check_sakuli.php: fixed bug, Suite graph did not have correct value.
    - CheckMySQLHealthSakuli.pm: adjust perfdata output as on gearman output (closes #106)
    - Adapted mysql_purge.sh to new table names, added parameter. #10
- Merge pull request #108 from sgbeal/master
- Add Docker container `consol/sakuli-ubuntu-xfce` and `consol/sakuli-centos-xfce`, see #103:
    - add return value to sakuli_startup.sh and add exit state to sakuli.sh script
    - HTML5-VNC client (noVNC) enabled containers
    - `docker-compose.yml` example for parallel test-execution
    - add example_xfce_test for the docker-containers
- fix PrioritizedServiceComparator so now 2 service with the same priority will also accepted
- close #49 add Environment#runCommand method
- add `takeScreenshot()` method to Region object to get the current region as screenshot
- Merge pull request #99 from c-kr/dev
- close #46 - add read the docs links and badge
- improve the way to include additional image folders, #96:
    - add method `addImagePaths` to the TestCase functions
    - add variable '$testSuiteFolder' as global variable to JavaScript testcase.js for a more strait filepath handling
- add optional parameter 'silent' to Application.close function to suppress exceptions
- add OS identification functions 'isWindows', 'isLinux', 'getOsIdentifier' to Environment class
- close #98 add source and javadoc files to maven build artifacts (on release-builds)
- close #92 exclude Base64 String in log output
- fix #95 state of suite/case/step is always 'OK', if no error occurs and warning + critical time is 0
- close #81 Expanded color array in PHP template and change numbering scheme for cases and steps to 3-digits

Version 0.9.0

- close #74 extract logging functions to separate javascript class Logger
- close #70 rename sakuli.screenbased. *properties to sikuli.*
- close #42 rename Application#closeApp to Application#close in javascript
- close #27 modify 'non screenshot exception handling' // add TestCaseAction#throwException(message,screenshot)
- add mysql Dockefile for sakuli database forwarder setup, see #10
- close #10 rename table name from sahi to sakuli
- rewritten documentation for sahi delay.
- close #79 rename property `sahi.proxy.requestDelayOnSikuliInput.delayTime` -> `sahi.proxy.onSikuliInput.delayPerKey` and `sahi.proxy.requestDelayOnSikuliInput.refreshTime` -> `sahi.proxy.onSikuliInput.delayBeforeInput`
- finish JavaDSL to be fully supported of all Sakuli features also in Java
- fix #11 custom sahi-port handling (use different ports for sahi-proxy)
- close #7 update sahi-jar verison to sahi 5.0

Version 0.5.0

- rename sakuli.autoHighlight.seconds to sakuli.highlight.seconds

- Documentation
- fix #72 modfy testsuite.suite file writer - just overwrite the file if any blank lines are inside
- add Environment#resetSimilarity()
- fix api generation script
- improve error message for invalid paths in 'testsuite.suite' file
- add support for more screenshot paterns - .jpg, .JPG, .png, .PNG
- .#52 rename sakuli.receiver properties to sakuli.forwarder
- fix #71 add the resumeOnException flag to some missing handleException calls
- refactor exception handling // improve exception handling for javaDSL
- refactor dependency management // extract bin, config, libs to new common 'module'
- .#13 rename screeshot property to 'sakuli.screenshot.onError'
- .#20 enable testCase.endOfStep("name")
- .#66 add -b, --browser into sakuli.jar/sakuli.sh
- .#64 Added Linux (sakuli.sh) and Windows (sakuli.bat) starter.
- .#55 low-level-mouse functions, add mouseMove(), mouseUp(mouseButton), mouseDown(mouseButton)
- .#60 refactor command line options
- .#62 move log-level settings to sakuli.properties
- .#60 introduce a 'sakuli-default.properties' file to move the sakuli.properties to the test suite root
- .#60 introduce new file system structure

Version 0.4.9 (Bugfix Release)

- add #106 add warn/crit thresholds as perfdata values for the Gearman results
  - Adaption for 3-digit case/step ids
  - PNP template with unknown perfdata support
  - added TICKer for incomplete data, warn/crit states
  - Changed color scheme
- add #77 separate error state to identify the affected TestCaseStep on errors:
  - modify SakuliExceptionHandler to find current Step and enable adding exception to the current step
  - add error message output for exceptions in TestCaseSteps
- add #31 determine all not executed TestCaseSteps, to secure that the nagios performance graphs are displayed correctly:
  - introduce new TestCaseStepState INIT
  - modify nagios RRD performance data output for initialized and not started steps to typ 'unknown'
  - add caching mechanism the step information for not started steps implementation
  - call write cached steps information on every 'not error' result
  - gearman forward: write unknown values to every result line if a suite, case or step entity has finished with errors or have even not been called
  - database forwarder: write NULL instead of '0' at warning and critical times
- add `takeScreenshot()` method to Region object to get the current region as screenshot
- add troubleshooting for Nullpointer at `new Application("..").getRegion()` to documentation
- fix PrioritizedServiceComparator so now 2 service with the same priority will also accepted
- add jenkins-build badge
- add #46 add dev-v0.4 read-the-docs & read-the-docs badge
- add #96 add variable '$testSuiteFolder' fore more strait forward import handling
- fix dependency path of javafx for java7
- close #92 exclude Base64 String in log output
- modify documentation of warning / critical times
- add testcase.endOfStep function without warning time
- add #81 change numbering scheme for cases and steps to always three digits to expanded color array in PHP template

Version 0.4.8

- fix bug: test suite has stat 'OK' instead of 'RUNNING' during the execution

- improve logging for more information, see Sakuli - Manual
- clearify the sakuli encryption functionality - modify documentation and improve the implementation, see #5
- refactor data structur, see #60
- exctract `sakuli.properties` to the test suits folder and introduce a `sakuli-default.properties` file.

Version 0.4.7

- add function `getLastUrl()` to the `TestCase` functions, to enable URL based test case handling.
- uncomment some receiver properties in `sakuli.properties` to make the property overriding more generic.
- fix bug that `new Region("image_pattern.png").click();` always clicks on the center of the screen
- introduce experimental JAVA-DSL as new module

Version 0.4.6

- add `sleep()` method to Region
- `keyUp(...)` , `keyDown(...)` and `write(...)` method to the Region and Environment functions to have more control over the typing.

Version 0.4.5

- add method to set an delay for the sahi-status-requests, so that no key or click events will be lost by the JavaScript engine of the Browser, see new entry in `sakuli.properties` :

```
# Specifies the interval in milliseconds, what should be applied when sikuli based input
# (like typing or clicking) is interacting with a Browser website.
# This setting only make sense, if your test does NOT use Sahi functions for controlling the
# testing website. This setting will prevent the test for losing some key or click events
# in case of blocking, synchronous sahi-interal state requests.
#
#sahi.proxy.requestDelayOnSikuliInput.delayTime=500
#
### refresh time for the sahi proxy to set the delay time
#sahi.proxy.requestDelayOnSikuliInput.refreshTime
```

Version 0.4.1

- update release build so that the zipped-release files can be downloaded from http://labs.consol.de/sakuli/install.
- remove zipped-release files from git repository
- documentation update
- build automatic sakuli-api documentation
- clean up repository
- introduce some maven-profiles for individual usage
- change `.inc` and `.sah` file ending to `.js`
- fixed some typos
- set up jenkins build

Version 0.4.0

- centralized the configuration of properties files:
  - `_include/sakuli.properties` now contains all possible configuration options for Sakuli. These are the *default values* for all tests
  - `<test-suite>/testsuite.properties` contains the *test suite specific configuration options*. The only mandatory property here is the test suite identifier `testsuite.id` . All other properties are optional.
  - Options set in `testsuite.properties` will override the default settings in `sakuli.properties`
- Proxy configuration options can now be set in `sakuli.properties` (defaults) or `testsuite.properties` (suite specific)
- Re-organized the folder structure of `sakuli-zipped-release-vX.X.X.zip` and source code directory.
- Extended logging with more configuration possibilities (SLF4J with underlying logback-Logging)
- Consolidation of the applicationContext files
- Remove the program-based setting of system properties.

- The possibility to disable the "encryption interface" with new property `sakuli.encryption.interface.testmode=true`
- Added a separate module for integration testing
- Bugfixing and extended unit tests
- documentation update
- Added a separate module for integration testing.
- Bugfixing and extended unit tests.
- Update the documentation

Version 0.4.2

- Introducing receiver concept: For each receiver the results will be sent. Currently Supported JDBC-Databases and the Gearman receiver.

- Gearman receiver: sent all data directly to your OMD/Nagios distribution. Currently it is missing that the screenshots will also be transferred. This will be fixed in the next version

- Bufixing in maven build, exception handling, testcase ids and added some more unit tests

Publications

01/2017: Java aktuell: Automatisiertes Testen in Zeiten von Microservices (Christoph Deppisch / Tobias Schneck)

10/2016: Informatik Aktuell: Software-Test im Container: So können Sie Graphical User Interfaces mit Docker und Sakuli testen (Tobias Schneck)

10/2016: ConSol Labs: Containerized UI-Tests in Java with Sakuli and Docker (Tobias Schneck)

10/2016: ConSol Labs: Sakuli EndToEnd Tests mit Android (Philip Griesbacher)

8/2016: heise Developer: Testautomatisierung in Zeiten von Continuous Delivery (Christoph Deppisch / Tobias Schneck)

8/2016: Pressemitteilung: Testautomatisierung darf nicht bei Unit-Tests Halt machen

2/2016: Success Story: pbb Deutsche Pfandbriefbank: Monitoring with Sakuli

2/2016: IT Administrator: "Den Anwender simuliert" (Simon Meggle)

5/2015: heise Developer: End-to-End Application Monitoring mit Sakuli

2/2015: IT Administrator: End2End-Monitoring mit dem Open Source-Tool Sakuli (Simon Meggle)

Events

April 27, 2017: Stuttgarter Test-Tage, Stuttgart: Testing - Selenium? Rich-Clients? Containers? (Tobias Schneck)

January 30 - February 3, 2017: OOP 2017, Munich (Tobias Schneck)

January 19, 2017: Agile Testing Meetup - Testen im Zeitalter von Containern, Munich (Tobias Schneck)

November 16, 2016: ContainerConf 2016, Mannheim (Tobias Schneck)

November 3, 2016: Software QS-Tag 2016, Nuremberg (Tobias Schneck)

September 30, 2016: JUG Saxony Day, Dresden (Tobias Schneck)

August 31, 2016: Herbstcampus 2016, Nuremberg (Tobias Schneck)

July 25, 2016: JUG München (Tobias Schneck)

June 27, 2016 Meetup during the ContainerDays, Hamburg (Tobias Schneck)

March 8-10, 2016: JavaLand, Brühl (Tobias Schneck)

March 3, 2016: Allianz Arena München (Simon Meggle)

March 1, 2016: Icinga Camp, Berlin (Simon Meggle)

January 26, 2016: Linux-Stamtisch München (Tobias Schneck)

October 24, 2015: Ubucon Berlin (Simon Meggle)

October 13, 2015: Testing & Integration Day, Allianz Arena Munich (Tobias Schneck)

June 22, 2015: Agile Testing Munich (Tobias Schneck)

May 14, 2015: OpenTechSummit (Simon Meggle)

March 28, 2015: LinuxTag Augsburg (Simon Meggle)

TODO add links to event like, (Slides)

Slides

2017: Containerized End-2-End-Testing - Agile Testing Meetup Munich (Tobias Schneck)

2016: Containerized End-2-End-Testing (German) - JUG Saxony Days Dresden (Tobias Schneck)

2016: Containerized End-2-End-Testing - ContainerDays 2016 Hamburg (Tobias Schneck)

2014: End-to-end testing for web sites and common UIs with full Nagios integration (Tobias Schneck)

Media

Monitoring Minutes

Episode 9 of the ***ConSol Monitoring Minutes*** shows the main features of Sakuli in 15 minutes. The machine used in this video is an instance of our demo appliance (see above).

[End2End Monitoring mit Sakuli und Nagios - ConSol Monitoring Minutes 9](#)

End-2-End-Testing

[Sakuli End-2-End-Testing - Lightning Talk - Agile Testing Meetup (June 2015)](#)

Support

TODO update like http://www.citrusframework.org/docs/consulting/

In case you have any questions or requests for help, feel free to get in contact with us! The Sakuli team members are reachable on the email address **sakuli@consol.de**.

The guys behind Sakuli:

ConSol Software GmbH
Franziskanerstr. 38, D-81669 München
Tel. +49-89-45841-100, Fax +49-89-45841-111
Homepage: http://www.consol.de E-Mail: info@consol.de </tr>