Automated Exploratory Testing - 1.3

Following section contains complete AET System documentation.

1. Introduction - 1.3

Cognifide has developed AET in direct response to the growing internal QA demand for an online tool that could support regression testing of web services to improve the quality with fast feedback.

This document details the technical aspects of version 1.0.0. It has been commissioned to assist

- the AET administrator who may wish to understand the detail about system architecture and installation procedure.
- the end-users who will configure test suites and analyze reports generated by the tool.

This section provides some basic information about the

- application,
- scope of the document,
- document structure and conventions used,

all of which are helpful in guiding the user.

1.1. Application overview - 1.3

AET is an on-line tool designed as a flexible application that can be adapted and tailored to the regression requirements of a given project.

Definition of Regression testing : This is a type of software testing that seeks to uncover new software bugs, or regressions, in existing functional and non-functional areas of a system. It is especially useful after changes such as enhancements, patches or configuration changes, have been made to them.

Definition of Baseline: The act of taking a snap shot of the url/page and saving it to a file for future comparison in a number of ways to find differences.

Its name is an acronym formed from **A**utomated **E**xploratory **T**esting. The tool has been developed to aid front end client side layout regression testing of websites or portfolios using common components in a CMS. In essence assessing the impact or change of a website from one baseline to the next.

The following is a basic scenario of use:

1. The user *baselines* a set of components or pages with URLs as an input to the tool.
2. At some point in the future, the CMS user may change the page component or content.
3. At this point the ‘current baseline’ is used to compare with the ‘new version’ and the change assessed for one of the 3 possibilities:
   a. there are no changes - no involvement is required,
   b. there is a change but the user accepts it, which means she/he re-baselines,
   c. there is a change and the user does not accept it, so she/he has to fix it.
4. The basic report provided will show at a minimum:
   a. layout comparison,
   b. W3C source compliance,
   c. HTTP status code verification,
   d. responsiveness - using different resolutions,
   e. JS errors check.

Software teams can use the tool for the following as well:

- at the end of a hourly/daily/weekly/per sprint completion,
- as a part of an upgrade of infrastructure or a platform project.

The tool is predominately used to conduct regression testing for,

- software development of a web site
- content authoring.

1.2. Scope of the document - 1.3

The documentation covers the following:

- The technical specification of AET - which includes a brief description of the services provided as well as a system overview.
- The system requirements - an outline of the components for the AET application with a technical specification of system requirements and minimum hardware configuration.
- An installation guide - which provides details on how to install the platform on a set of servers.
- A User Guide - That enables the reader to configure test suites and how to interpret reports generated by AET.
- Known issues with workaround - To enable the user to take advantage of the functionality in the best way.

1.3. Document structure - 1.3

This document is divided into several parts:

Chapter	Description
Introduction	General technical information outlining design decisions, architecture of the AET solution as well as deployment information and delivery methods.
System overview	This chapter outlines the AET application together with its components, modules and technologies used.
Environment setup	This chapter details steps to undertake the configuration of the environment in an Amazon Web Services (Cloud Computing Services).
Project setup and configuration	This chapter details the installation process of the AET tool (including its configuration).
Test setup and run	This chapter describes how to set up the test suite, how to run them and then how to verify the results.
Report	This chapter is concerned with how to generate and interpret the report.
Known bugs and workarounds	This chapter details known issues together with workarounds that assist in getting the best out of AET.

1.4. Conventions used - 1.3

The following section lists conventions used for text-formatting styles:

Font	Application
italic	• file names • brand names • proper names • links to other sections/chapters
bold	• important information • table heading
`courier`	• configuration variables and values • paths

Important information

Such boxes contain important information e.g. requirements, assumptions.

Warnings

Such boxes contain notices of potential problems.

Sample chunks of code are enclosed in a code block:

```
<example>
<info>content</info>
<version>5.3.5</version>
```

```
<module>core</module>
</example>
```

1.5. Dictionary - 1.3

In this section the acronyms and terms used most commonly in this document are explained.

Active MQ

a *JMS* (*Java Message Service*) Server which is a basic communication channel between AET System components.

AET

an acronym for **A**utomatic **E**xploratory **T**esting, an online testing tool developed by Cognifide.

AET Core

a set of system modules that are crucial to whole system work. The AET system will not work properly without all core modules configured and running properly.

AET Jobs

a set of system modules that can perform a particular task (e.g. collect screenshots, compare sources, validate a page against *W3C*).

AET Maven Plugin

a default client application for the AET system that is used to trigger the execution of the *Test Suite*.

Amazon Web Services

Cloud Computing Services where AET environment is setup.

Apache Karaf

see *Karaf*.

Artifact

usually used in the context of a small piece of data, the result of some operation (e.g. a collected screenshot or a list of *W3C* validation errors).

AWS

see Amazon Web Services.

Browsermob

a proxy server used by AET to collect some kinds of data from tested pages.

Cleaner

a module responsible for removing old and unused artefacts from the database.

Collector

a module responsible for gathering data necessary for its further processing (e.g. validation, comparison).

Collection

the first phase of the AET service during which all specified data is collected (e.g. screenshots, page source, js errors). Once they are collected successfully, all collection results are saved in the database.

Comparator

a module responsible for comparing data currently collected to its existing pattern or validating it against a set of defined rules.

Comparison

the second phase of the AET service that performs the operation on the data. collected during the first phase In some cases the collected data is compared to patterns, in others special validation is performed (e.g. *W3C*). The second phase starts before the collection finishes - just the moment when required artefacts are collected and become ready to be compared (e.g. to compare two screenshots system does not have to wait until the source of a page is collected).

Cookie Collector

a collector responsible for collecting cookies.

Cookie Comparator

a comparator responsible for processing collected cookies.

Cookie Modifier

a modifier that allows to modify cookies for a given page, i.e. to add or remove cookies.

Baseline

The act of taking a snap shot of the url/page and saving it to a file for future comparison in a number of ways to find differences.

Data Filter

a module responsible for filtering the collected data before performing comparison e.g. filtering uninteresting js errors before the js errors check takes place.

Extract Element Modifier

a modifier that allows to extract an element from the html source (collected by the *Screen Collector*) by providing the id attribute or the class attribute.

Feature

a part of the AET system which covers full testing case e.g. layout - this feature consists of the *Screen Collector*, the screen comparator and the layout reporter module.

Firefox

a browser the AET tool makes use of, currently the version that is used is 30 en-US.

Header Modifier

a modifier responsible for adding additional headers to a page.

Hide Modifiers

a modifier responsible for hiding an element on a page that is unnecessary for a given test.

Html-report

a basic report in a form of a HTML file.

Java

a programming language that is used to develop the AET tool.

Java Development Kit

see *JDK.*

Java Management Extensions

see *JMX*

Java Message Service

see *JMS*

JavaScript

see *JS*.

JDK

the *Java Development Kit* is a program development environment for developing Java applications.

Jenkins

a continuous Integration (CI) server which is used as the user interface wrapper for the *AET Maven Plugin*.

Jetty

a simple Http Server, used as a container for web applications.

JMS

an acronym for the *Java Message Service*, simple message standard that allows application components to communicate with one another.

JMX

Java Management Extensions (JMX) is a technology that is used to manage and monitor advanced interfaces of Java applications. In the AET tool it is used to manage *ActiveMQ*.

JS

a dynamic programming language.

JS Error

a JavaScript error that occurs in a script during its execution.

JS Errors Collector

a collector responsible for collecting JavaScript errors occurring on a given page.

JS Errors Comparator

a comparator responsible for processing the collected JavaScript error resource.

JS Errors Filter

a filter that filters the results returned by the JS Errors Collector. It removes matched JavaScript errors from reports.

JUnit

a simple framework allowing to develop repeatable tests. It is an instance of the xUnit architecture for unit testing frameworks. More information about it can be found at: http://junit.org/.

Karaf

in fact *Apache Karaf* is an OSGi container that provides a basic configuration for existing OSGi implementations (e.g. Apache Felix).

Layout Comparator

a comparator responsible for comparing a collected screenshot of page to its pattern.

Login Modifier

a modifier that allows to log in into the application and access secured sites.

Maven

a software project management and comprehension tool. It used as a base for the *AET Maven Plugin*.

Modifier

a module responsible for converting the target before the data collection process is performed e.g. modifying a requested header, adding a new cookie, hiding a visible element.

MongoDB

an open-source cross-platform document-oriented database that the AET tool makes use of for data storage and management. MongoDB is developed by MongoDB Inc.

Open

A module that is a special operand for the Collect Phase.

OSGi

a modular system and services platform for Java. It is used as an application environment for AET Java components.

Pattern

a sample model of data. Collection results are compared to their patterns to discover potential differences.

pom.xml

a Maven tool configuration file that contains information about the project and configuration details used by Maven to build the project.

Rebasing

am operation changing the existing pattern to the current result.

Regression testing

This is a type of software testing that seeks to uncover new software bugs, or regressions, in existing functional and non-functional areas of a system. It is especially useful after changes such as enhancements, patches or configuration changes, have been made .

Remove Lines Data Modifier

a modifier that allows to remove lines from the source (data or pattern) that a given page is compared to.

Remove Nodes Data Modifier

a modifier that allows to delete some node(s) from a html tree. Node(s) are defined by the xpath selector.

Report

a summary of the AET test process.

Reporter

a module responsible for generating reports.

Representational State Transfer API

see *Rest API*.

Resolution Modifier

a modifier responsible for changing the size of the browser screen.

Resource type

a unique name for the resource produced by the collector and consumed by the comparator.

Rest API

a Representational State Transfer API for the data stored in the AET Database. It enables the user to browse the data and artifacts stored after a run of the *Test Suite* was completed.

Runner

a unit responsible for the communication with the client and dispatching processing among workers.

SCM repository

a data structure storing metadata for a set of files that is managed by a source control management (SCM) system responsible for managing changes in files. The most popular examples of SCM systems are Git (http://git-scm.com/) and SVN (https://subversion.apache.org/).

Screen Collector

a collector responsible for collecting a screenshot of the page under a given URL.

Selenium

a portable software testing framework for web applications.

Selenium Driver

a test tool that allows to perform specific actions in a browser environment (e.g. take a screenshot of a page).

Sleep Modifier

a modifier responsible for ceasing the execution of a given test temporarily. It causes a current thread to sleep.

Source Collector

a collector responsible for collecting the source of a page under a given URL. Unlike other collectors the *Source Collector* does not use *Web Driver*. It connects directly to a web server.

Source Comparator

a comparator responsible for comparing a collected page source with its pattern.

Status Code

a response code for the resource request. For a detailed list of codes please refer to the Hypertext Transfer Protocol documentation at: http://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html.

Status Codes Collector

a collector responsible for collecting status codes for links to resources on a page under a given URL.

Status Codes Comparator

a comparator responsible for processing collected *Status Codes*.

Test

a definition of logical set of *Test Cases* performed on a set of URLs.

Test Suite

a set of *Tests* (at least one) finished with the *Report*.

Test Case

a single URL *Test* against a feature, e.g. a *W3C* page test, a screenshot for the resolution 800x600 test.

Thresholds

a feature allowing to declare a Jenkins build as ‘success’, ‘unstable’ or ‘failed’ depending on the number of *Tests* that failed or were skipped.

Version Storage

a name of a database abstraction layer which contains versioned data (data grid).

Wait For Page Loaded Modifier

a modifier that waits until a page is loaded or a fixed amount of time is up.

Web Console

the OSGi console installed on *Apache Karaf*. By default it is accessible via a browser: http://localhost:8181/system/console/configMgr. The default user/password are as follows: *karaf/karaf.*

Worker

a single processing unit that can perform a defined amount of tasks (e.g. collect a screenshot, compare a source).

W3C Comparator

a comparator responsible for validating a collected page source against *W3C* standards.

W3C Validator

a third party standalone software used to validate pages against the *W3C* standard.

xunit-report

a *Report* that visualizes risks on the Jenkins job board and that contains information about the number of performed tests and the number of failures (potential threats).

2. System overview - 1.3

This chapter describes the system architecture and technologies.

2.1. Architecture overview - 1.3

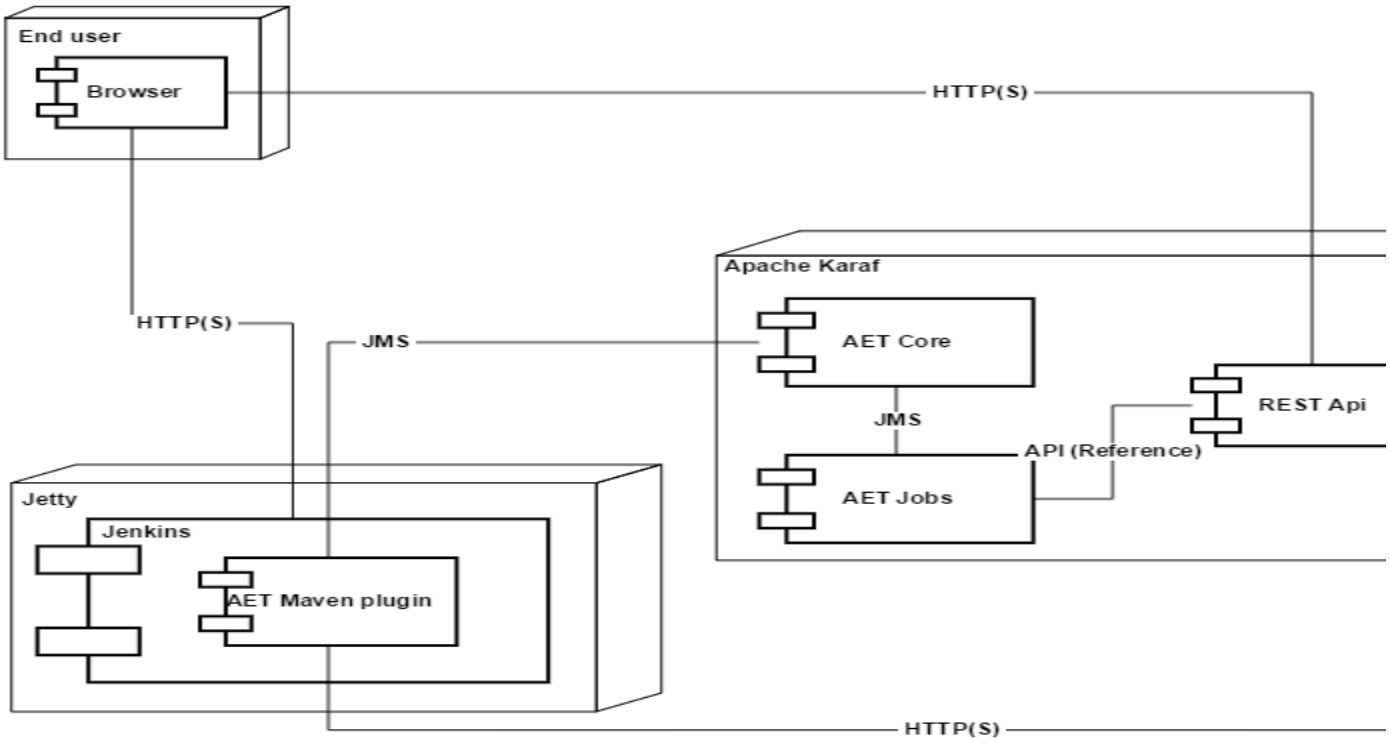
Architecture

The core AET system consists of:

- the *OSGi* container with the Runner, Worker and REST API deployed to it,
- the *Database (Version Storage),*
- the *JMS Server*
- the *Client* (*AET Maven plugin*)

Communication between system components

Sample communication in a small system is presented on the diagram below:



Third-party software used by system

AET uses the following third party software as parts of the system:

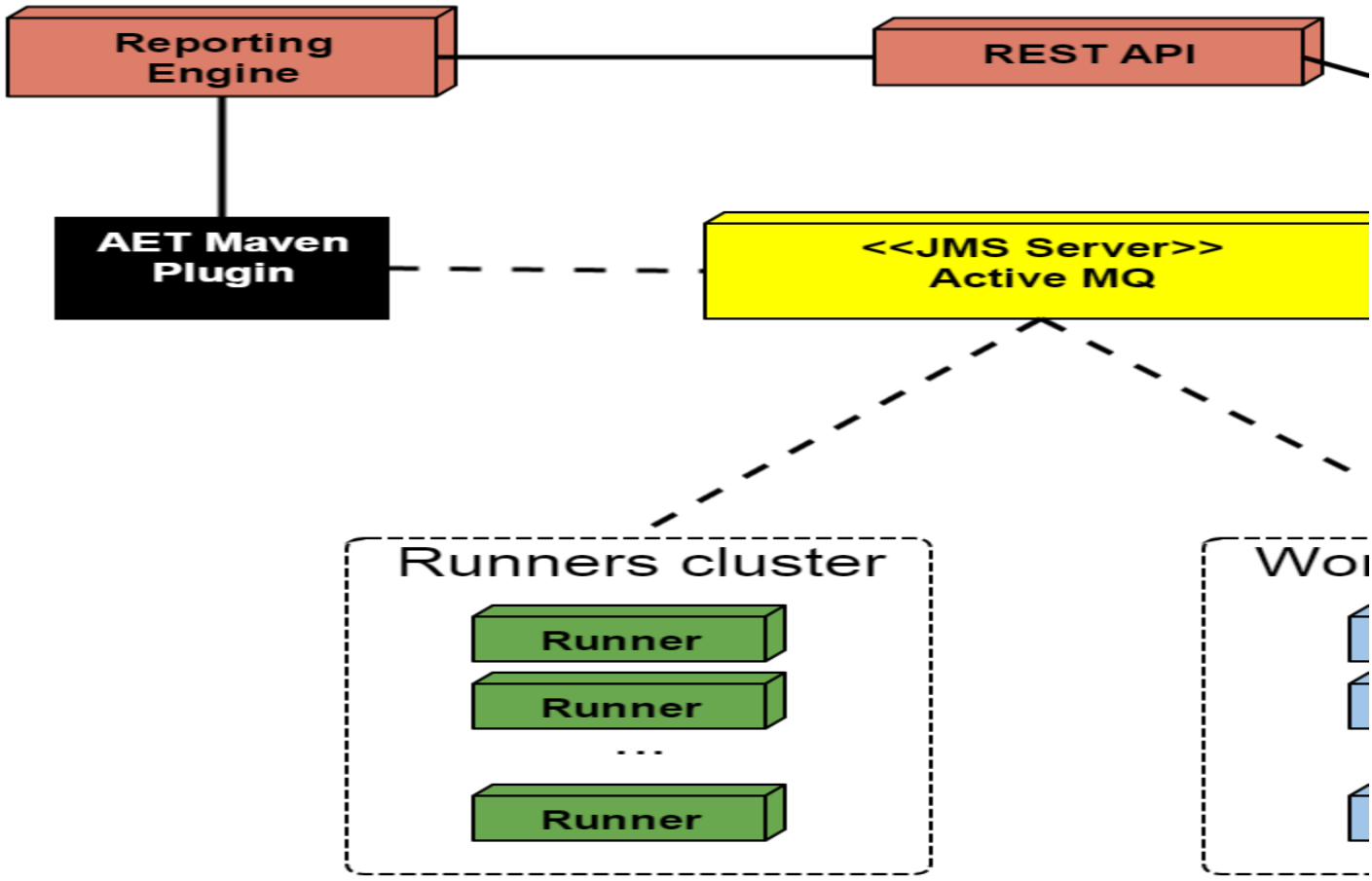
Software	Used version	Function
Apache Karaf	2.3.9 with 4.2.1 Apache Felix Framework.	OSGi container for AET bundles and REST API.
Apache ActiveMQ	5.13.1	JMS Sever used for communication between system components.
MongoDB	3.2.3	System database.
Browsermob	2.0.0	Proxy server.
W3c validator	1.2	W3C validation service.
Firefox	38.6.0 ESR (en-US)	Browser with Selenium (2.50.1).

2.2. System components - 1.3

The AET System consists of 7 core units:

- The Client (AET Maven Plugin)
- The Runner cluster
- The Worker cluster
- The JMS Server
- The Database
- The Rest API
- The Reporting Engine

AET System architecture



Client

The Client component is has the following functions

- used to send request to the AET System.
- the Maven Plugin parses the input Test Suite xml file.
- after the Test Suite run is finished, the Client downloads the Report.

Runner

The Runner is the heart of the system. It is responsible for consuming Client's request and dispatching it to Workers. It works similar to the Map-Reduce algorithm. During the execution of the suite, the Runner checks if the next phase can begin and when all the phases are finished the Runner informs the client about it.

Worker

The Worker is a single processing unit that can perform a specific task e.g. collect a screenshot using the *Firefox* browser in the *Windows 7* environment, collect a page source, compare two screenshots, check if the source of a page is W3C-complaint and many others.

JMS Server

The JMS Server is a communication hub for the whole system. Workers, runners and the client communicate with one another using JMS messages.

Database

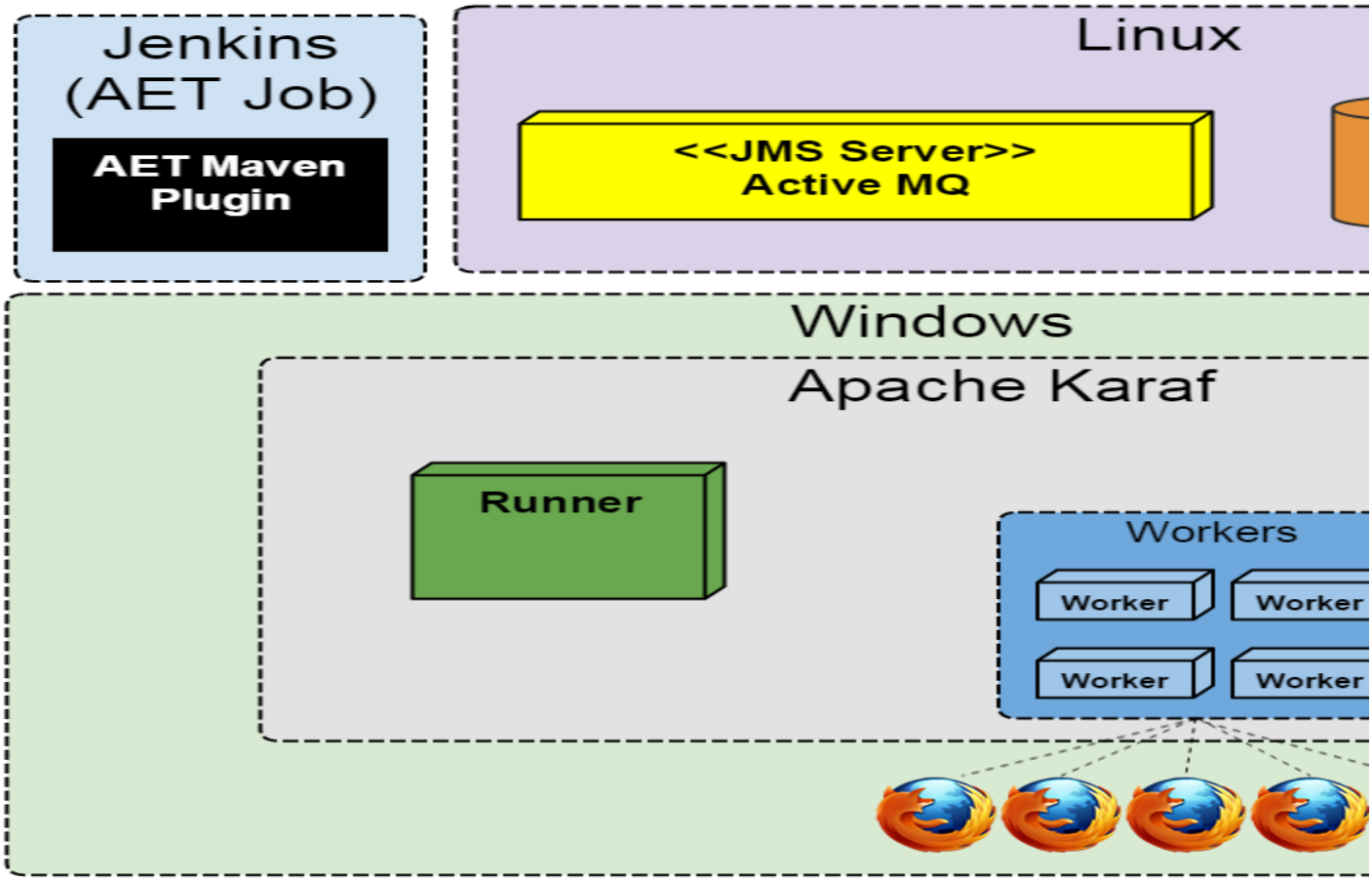
The Database serves as the system storage. It stores all the results, reports and patterns.

Rest API

The Rest API for the stored data; the user can download the Report, collected sources, view screenshots and comparison results via the Rest API.

AET System architecture in practice

The following architecture is for a small capacity instance (consisting of one Windows instance and one Linux machine) is presented below:

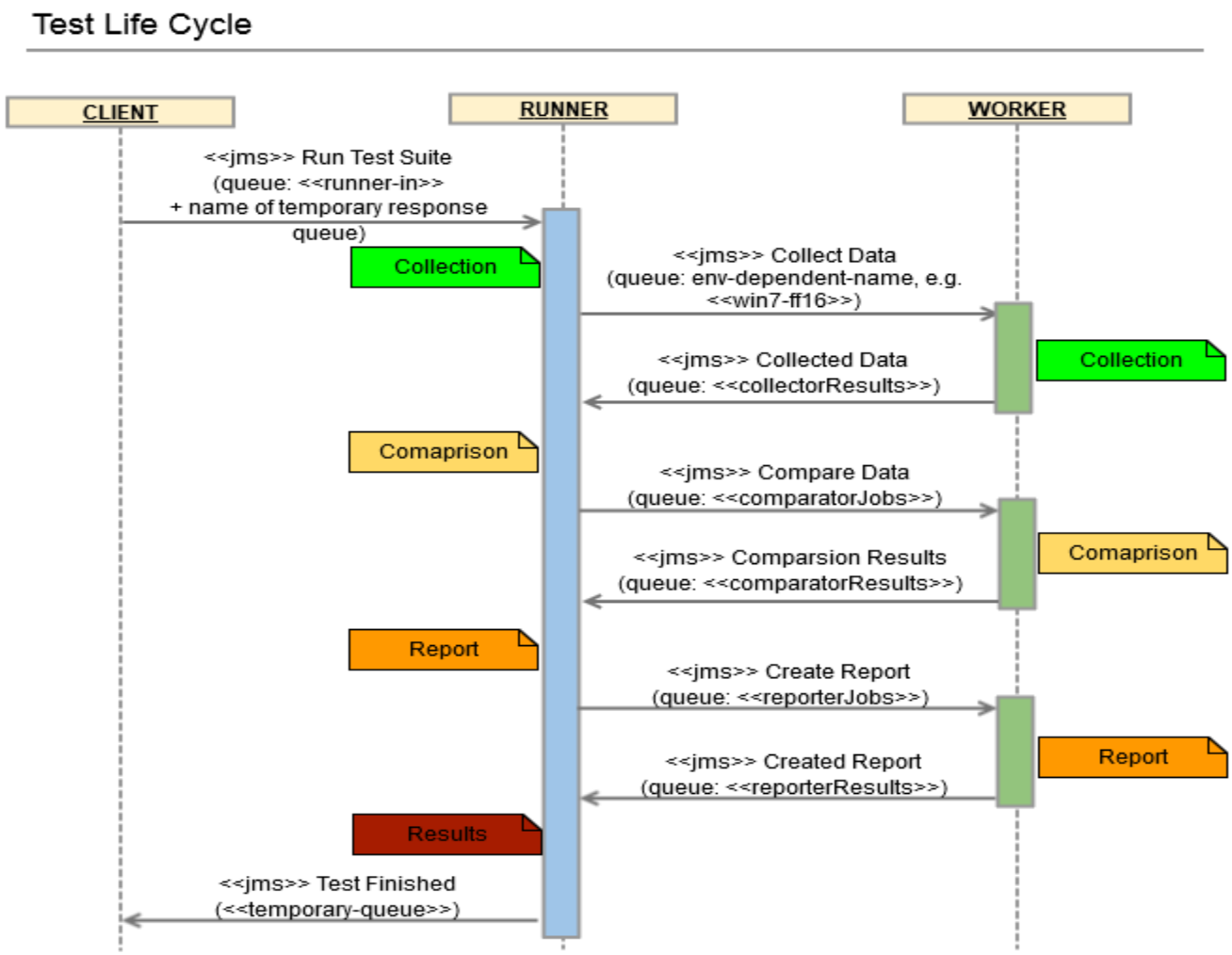


2.3. Test processing - 1.3

Each AET test consists of three phases:

- collection,
- comparison,
- report generation.

After report is generated, client is notified about the finished test run.



Collection

This is the first phase during which all specified data will be collected (e.g. screenshots, page source, js errors). All collection results are saved in database after successful collection.

Comparison

The second phase is operation on collected data. In some cases collected data is compared to patterns, in other special validation is performed (e.g. w3c). The second phase starts before collection finishes - just at the moment when required artefacts are collected and ready to compare (e.g. to compare two screenshots, system does not have to wait until source of page is collected).

Report metadata generation

Report metadata generation phase is based on comparing results and can be started after all comparisons are finished. The metadata is generated in json format and it is stored at MongoDB. The report can be generated by accessing the Report Generator Service.

2.4. System modules - 1.3

AET System uses OSGi to run its modules. Following table contains description of each module (bundle) in system:

Name	Description
`communication-api`	Contains API for Communication module.
`datastorage-api`	Contains API for Datastorage module.
`jobs-api`	Contains API for Jobs module.
`validation-api`	Contains API for Validation module.
`worker-api`	Contains API for Worker module.
`communication`	Module is responsible for communication between other modules.
`datastorage-gridfs-impl`	Datastorage GridFs Implementation, responsible for storing artifacts in database.
`selenium`	OSGi wrapper for Selenium.
`proxy`	Proxy layer.
`browsermob`	OSGi wrapper for Browsermob.
`runner`	Heart of the whole system, responsible for communication with client and dispatching requests to workers.
`validation`	Responsible for validation of defined suite.
`versionstorage`	Abstract layer over datastorage system, contains Rest API.
`worker`	Contains worker unit logic.
`job-common`	Contains all basic features implementations.
`report-engine`	Responsible for creating reports for the end user.

Additionally, system requires `aet-maven-plugin` to work properly.

2.5. Database architecture - 1.3

Database

All results of collected, compared and reported are stored in MongoDB, with use of GridFs implementation.

A MongoDB instance hosts one or more databases. Each database is related to **`company`** attribute of `suite` element in test suite xml definition.

Collection

A database holds a set of collections and GridFS specification assumes storing files in two related collections:

- chunks for the binary chunks,
- files for the file's metadata.

Thus every AET collection is placed in a common bucket by prefixing each with the bucket name. A bucket name is related to **`project`** attribute of `suite` element in test suite xml definition.

For example, test suite definition:

TestSuite

```
<?xml version="1.0" encoding="UTF-8" ?>
<suite name="test-suite" company="cognifide" project="aet-sanity" environment="win7-ff16">
 <test name="cookie-test">
  ...
 </test>
</suite>
```

artefacts will be stored in *cognifide* database (which will be created if non existent) with two *aet-sanity* collections:

- `aet-sanity.files`
- `aet-sanity.chunks.`

Document

Each collection holds documents as BSON objects in form of key-value pairs. A document is a basic unit of data. For every single AET artefact processing result in two documents to be stored - exact result file (i.e. *screenshot.png*, *source.html*) and additional *result.json* metadata file.

AET metadata document (one that belong to PROJECT_NAME.files collection) - contains among others common key-value pairs:

- `_id` - the unique ID for this document,
- `filename` - the name of the document (for example screen collecting phase will produce *screenshot.png*, source collecting phase - *source.html*),
- `uploadDate` - the date the document was first stored. It is used by GridFSCleaner module to search for old artifacts to remove,
- `metadata` - key-value map of additional information which allows uniquely identify given artifact. Following paragraph covers metadata details.

Metadata

There are four types of artefacts in AET system:

- PATTERNS - results of collect phase set as pattern for next comparisons. PATTERNS can be set explicitly by user by rebase action or automatically during first test execution if no patterns exists for given artifact. Not all types of data have PATTERNS artifacts.
- DATA - result of collect phase - data collected during current test execution.
- RESULTS - results of compare phase - differences between DATA collected during current test execution and previously set PATTERNS is saved as RESULTS artefact.
- REPORTS - metadata useful for generating report service

Metadata part of the document differs depending on the artefact type.

PATTERNS:

For PATTERNS metadata consist of following mappings:

Artefact name	Description	XML attribute
project	Name of the project. The same as MongoDB collection name.	suite.project
testName	Name of the test case.	test.name
artifactType	Type of the artefact - in this case PATTERNS.	(N/A)
environment	Environment where test is run.	suite.environment
company	Name of the company. The same as MongoDB database name	suite.company
urlName	Url name. If not provided encoded `href` property value from `url` is saved.	url.name (or url.href)
testSuiteName	Name of the test suite.	suite.name
version	Artefact version.	(N/A)
collectorModule	Resource type of the collector.	*collector tag*
collectorModuleName	Name of the collector module. If not provided, default collector name is chosen for given collector type (collectorModule).	*collector.*name
currentPattern	Specifies if artefact is current pattern.	(N/A)
rebaseSource	Metadata information of related DATA artefact that the PATTERN was created from.	(N/A)

DATA

For DATA metadata consist of following mappings:

Artifact name	Description	XML attribute
project	Name of the project. The same as MongoDB collection name.	suite.project
testName	Name of the test case.	test.name
correlationId	Unique identifier for each AET test execution.	(N/A)
artifactType	Type of the artefact - in this case DATA.	(N/A)
environment	Environment where test is run.	suite.environment
company	Name of the company. The same as MongoDB database name	suite.company
domain	General domain name consistent for all considered urls.	suite.domain
urlName	Url name. If not provided encoded `href` property value from `url` is saved.	url.name (or url.href)
testSuiteName	Name of the test suite.	suite.name
url	Url href.	url.href
version	Artifact version.	(N/A)
collectorModule	Type of the collector.	*collector tag*
collectorModuleName	Name of the collector. If not provided, default collector name is chosen for given collector type (collectorModule).	*collector.*name

RESULTS

For RESULTS metadata consist of following mappings:

Artifact name	Description	XML attribute
project	Name of the project. The same as MongoDB collection name.	suite.project
testName	Name of the test case.	test.name
correlationId	Unique identifier for each AET test execution.	(N/A)
artifactType	Type of the artefact - in this case RESULTS.	(N/A)
environment	Environment where test is run.	suite.environment
company	Name of the company. The same as MongoDB database name	suite.company
domain	General domain name consistent for all considered urls.	suite.domain
urlName	Url name. If not provided encoded `href` property value from `url` is saved.	url.name (or url.href)
testSuiteName	Name of the test suite.	suite.name
url	Url href.	url.href
version	Artifact version.	(N/A)
collectorModule	Type of the collector.	*collector tag*
collectorModuleName	Name of the collector. If not provided, default collector name is chosen for given collector type (collectorModule).	*collector*.name
comparatorModule	Type of the comparator. If not provided, default comparator type is chosen.	*comparator.* comparator
comparatorModuleName	Name of the comparator. If not provided, default comparator name is chosen for given comparator type (comparatorModule).	*comparator*.name

REPORTS

For REPORTS metadata consist of following mappings:

Artifact name	Description	XML attribute
project	Name of the project. The same as MongoDB collection name.	suite.project
correlationId	Unique identifier for each AET test execution.	(N/A)
artifactType	Type of the artifact - in this case REPORTS.	(N/A)
environment	Environment where test is run.	suite.environment
company	Name of the company. The same as MongoDB database name	suite.company
domain	General domain name consistent for all considered urls.	suite.domain
testSuiteName	Name of the test suite.	suite.name
version	Artefact version.	(N/A)
reporterModule	Type of the reporter.	*reporter tag*

Example

Test suite definition:

Test suite definition

```
<?xml version="1.0" encoding="UTF-8" ?>
<suite name="test-suite" company="cognifide" project="aet-sanity" environment="win7-ff16">
 <test name="w3c-test">
  <collect>
   <open/>
   <sleep duration="2000" />
   <source/>
  </collect>
```

```
  <compare>
   <source comparator="w3c"/>
  </compare>
  <urls>
   <url href="http://www.cognifide.com"/>
  </urls>
 </test>
 <reports>
  <html-report/>
  <xunit-report/>
 </reports>
</suite>
```

Examples of artifacts saved in MongoDB:

Patterns artifact example


```
{
    "_id" : ObjectId("551135174cc4f3e0bb8f9268"),
    "chunkSize" : NumberLong(261120),
    "length" : NumberLong(37104),
    "md5" : "625085ce1bd2bba128bdf900ce3c37be",
    "filename" : "source.html",
    "contentType" : null,
    "uploadDate" : ISODate("2015-03-24T09:57:43.220Z"),
    "aliases" : null,
    "metadata" : {
        "project" : "aet-sanity",
        "testName" : "w3c-test",
        "artifactType" : "PATTERNS",
        "environment" : "win7-ff16",
        "collectorModuleName" : "source",
        "company" : "cognifide",
        "urlName" : "http://www.cognifide.com",
        "testSuiteName" : "test-suite",
        "collectorModule" : "source",
        "version" : NumberLong(1),
        "currentPattern" : true
    }
}
```


Data artifact example


```
{
    "_id" : ObjectId("551135174cc4f3e0bb8f9264"),
    "chunkSize" : NumberLong(261120),
    "length" : NumberLong(37104),
    "md5" : "625085ce1bd2bba128bdf900ce3c37be",
    "filename" : "source.html",
    "contentType" : null,
    "uploadDate" : ISODate("2015-03-24T09:57:43.158Z"),
    "aliases" : null,
    "metadata" : {
        "project" : "aet-sanity",
        "testName" : "w3c-test",
        "correlationId" : "cognifide-aet-sanity-test-suite-1427191056352",
        "artifactType" : "DATA",
        "environment" : "win7-ff16",
        "collectorModuleName" : "source",
        "company" : "cognifide",
        "urlName" : "http://www.cognifide.com",
        "testSuiteName" : "test-suite",
        "url" : "http://www.cognifide.com",
        "collectorModule" : "source",
```

```
        "version" : NumberLong(1)
    }
}
```

Results artifact example

```
{
    "_id" : ObjectId("551135184cc4f3e0bb8f926a"),
    "chunkSize" : NumberLong(261120),
    "length" : NumberLong(2698),
    "md5" : "e4c8163b17e04ae0ba663f63eceea50b",
    "filename" : "result.json",
    "contentType" : null,
    "uploadDate" : ISODate("2015-03-24T09:57:44.236Z"),
    "aliases" : null,
    "metadata" : {
        "artifactType" : "RESULTS",
        "correlationId" : "cognifide-aet-sanity-test-suite-1427191056352",
        "testName" : "w3c-test",
        "urlName" : "http://www.cognifide.com",
        "testSuiteName" : "test-suite",
        "url" : "http://www.cognifide.com",
        "comparatorModule" : "w3c",
        "version" : NumberLong(1),
        "collectorModule" : "source",
        "project" : "aet-sanity",
        "environment" : "win7-ff16",
        "collectorModuleName" : "source",
        "company" : "cognifide",
        "comparatorModuleName" : "w3c"
    }
}
```

2.6. REST API - 1.3

Representational State Transfer API for accessing and modifying data stored in AET Database. Rest API is part of AET System and is the interface between system database, user and application.

Its methods are used by *AET Maven Plugin* to download reports, *HTML Report* uses it to load images and to perform rebase action. Additionally, Rest API enables user to browse resources and rebase patterns using *Metadata*.

Rest API is part of system core and it is by default installed on the *Apache Karaf* instance.

REST API HTTP methods

Note for Urls/UrlNames

`For REST API purpose url's in query must be passed in url-escaped form e.g. http://localho`

`'http%3A%2F%2Fwww.cognifide.com' part.`

`Response from REST API by default returns urls in url-escaped form.`

BASE_PATH: `http://${WINDOWS_MACHINE_PUBLIC_IP}:8181/cxf/aet.`

Resource's path can consist of *Metadata* parameters.

GET Methods

Table below presents Rest API request and result that is returned in response.

Resource (Rest API Request)

/

/{company}

/{company}/{project}

/{company}/{project}/{testSuite}

/{company}/{project}/{testSuite}/{environment}

/{company}/{project}/{testSuite}/{environment}/{test}

/{company}/{project}/{testSuite}/{environment}/{test}/urls

/{company}/{project}/{testSuite}/{environment}/{test}/urls/{urlName}

/{company}/{project}/{testSuite}/{environment}/{test}/urls/{urlName}/artifactTypes

/{company}/{project}/{testSuite}/{environment}/{test}/urls/{urlName}/artifactTypes/{artifa

/{company}/{project}/{testSuite}/{environment}/{test}/urls/{urlName}/artifactTypes/{artifa

/{company}/{project}/{testSuite}/{environment}/{test}/urls/{urlName}/artifactTypes/data/{c

/{company}/{project}/{testSuite}/{environment}/{test}/urls/{urlName}/artifactTypes/data/{c

/{company}/{project}/{testSuite}/{environment}/{test}/urls/{urlName}/artifactTypes/data/{c

/{company}/{project}/{testSuite}/{environment}/{test}/urls/{urlName}/artifactTypes/data/{c

/{company}/{project}/{testSuite}/{environment}/{test}/urls/{urlName}/artifactTypes/results

/{company}/{project}/{testSuite}/{environment}/{test}/urls/{urlName}/artifactTypes/results

/{company}/{project}/{testSuite}/{environment}/{test}/urls/{urlName}/artifactTypes/results

/{company}/{project}/{testSuite}/{environment}/{test}/urls/{urlName}/artifactTypes/results

/{company}/{project}/{testSuite}/{environment}/{test}/urls/{urlName}/artifactTypes/results

/{company}/{project}/{testSuite}/{environment}/{test}/urls/{urlName}/artifactTypes/results

/{company}/{project}/{testSuite}/{environment}/{test}/urls/{urlName}/artifactTypes/patterr

/{company}/{project}/{testSuite}/{environment}/{test}/urls/{urlName}/artifactTypes/patterr

/{company}/{project}/{testSuite}/{environment}/{test}/urls/{urlName}/artifactTypes/patterr

/{company}/{project}/{testSuite}/{environment}/reports

/{company}/{project}/{testSuite}/{environment}/reports/{reporterModule}

/{company}/{project}/{testSuite}/{environment}/reports/{reporterModule}/{correlationId}

/{company}/{project}/{testSuite}/{environment}/reports/{reporterModule}/{correlationId}/{a

POST Methods

POST Methods are used for rebase. Form parameter *rebaseCorrelationId* with specified correlationId is needed.

Resource

```
/{company}/{project}/{testSuite}/{environment}
```

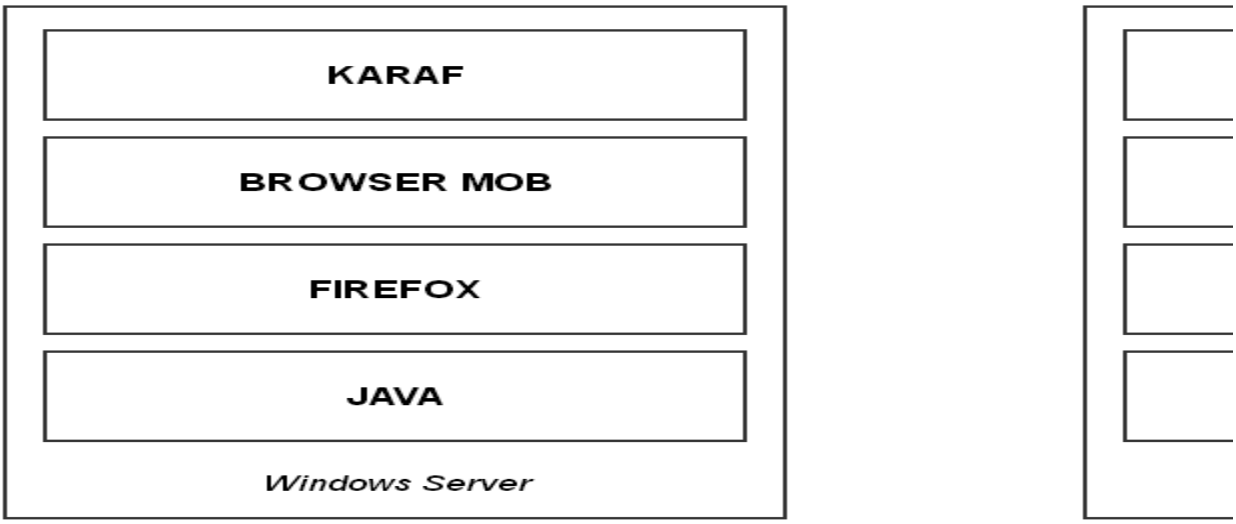
```
/{company}/{project}/{testSuite}/{environment}/{test}
```

```
/{company}/{project}/{testSuite}/{environment}/{test}/urls/{urlName}
```

```
/{company}/{project}/{testSuite}/{environment}/{test}/urls/{urlName}/artifactTypes/data/{c
```

3. Environment setup - 1.3

Environment setup intro.



3.1. AWS configuration - 1.3

System configuration

Basic AET System configuration requires two AWS t2.medium(setups) to give a working capability:

- 1 x Windows t2.medium (2x vCPU, 4GB RAM) (Karaf + Browsermob) + 40GB Storage for Windows OS
- 1 x Linux t2.medium (2x vCPU, 4GB RAM) (MongoDB + ActiveMQ + W3C Validator) + 10GB Storage for Linux OS + 50 GB for Mongo

Prerequisites

AWS account with proper rights to proceed following steps.

Network

Please find an example network used in this manual as 172.16.240.0/28 and that can be changed to another CIDR (but no smaller than /28) for it to work.

1. Logon to AWS panel

- - Create VPC in selected AWS Region
  - Set Name - AET
  - Set CIDR block - 172.16.240.0/28

2. Create IGW

- - Set Name - AET GW
  - Assign to VPC - AET

3. Create Subnet assigned to VPC AET

- - Create it in selected Availability Zone
  - Set Name - AET-subnet
  - Set CIDR block - 172.16.240.0/28

4. Modify route table

  - Associate created subnet
  - Assign public route - 0.0.0.0/0, target: created IGW

5. Create NACL(s) - allow ALL egress and ingress traffic.

6. Create SG (separate for each VM instance)

- - Set Name - Linux_SG
  - Set Inbound Rules - All traffic / all / all / S_IP: 91.202.100.0/22
  - Set Inbound Rules - All traffic / all / all / S_IP: 172.16.240.0/28
  - Set Outbound Rules - All traffic / all / all / 0.0.0.0

- - Set Name - Windows_SG
  - Set Inbound Rules - All traffic / all / all / S_IP: 91.202.100.0/22
  - Set Inbound Rules - All traffic / all / all / S_IP: 172.16.240.0/28
  - Set Outbound Rules - All traffic / all / all / 0.0.0.0

7. Allocate 2 EIPs.

8. Create Instances

- Choose AMI - Windows_Server-2008-R2_SP1-English-64Bit-Base-2015.02.11 (ami-d47341c9)
  - Choose type - t2.medium
  - Assign IP - 172.16.240.9
  - Assign VPC - AET
  - Assign subnet - AET-subnet
  - Assign EIP
  - Assign Security Group - Windows_SG
  - Set size 40 GiB and type General Purpose (SSD) with Name - Windows
  - Set Name - Windows
  - Assign SG - Windows_SG
  - Turn on Termination Protection
  - Create Keypair for Instances (and save it!).

- Choose AMI - CentOS6.5 - ami-f82a1ce5
  - Choose type - t2.medium
  - Assign IP - 172.16.240.8
  - Assign VPC - AET
  - Assign subnet - AET-subnet
  - Assign EIP
  - Assign Security Group - Linux_SG
  - Set size 16 GiB and type General Purpose (SSD) with Name - Linux_root
  - Assign additional storage 50 GiB and type General Purpose (SSD) with Name - Linux_content
  - Set Name - Linux
  - Assign SG - Linux_SG
  - Turn on Termination Protection
  - Create Keypair for Instances (and save it!).

Run instances, wait until deployment finishes, check access.

3.2. Linux setup - 1.3

Linux OS Configuration

Configuration procedure and software versions are based on CentOS 6.5 (with latest packages available).

Networking

Please find an example network used in this manual as 172.16.240.0/28 and that can be changed to another CIDR (but no smaller than /28) for it to work.

OS preparation

Basic OS Configuration consist of following steps:

1. Disable `selinux`,
2. Disable `iptables`,

3. Check if hostname is mapped to IP:

   /etc/sysconfig/network

```
NETWORKING=yes
HOSTNAME=centos6
```

/etc/hosts

```
172.16.240.8    centos6
```

4. Assuming that larger disk: is `/dev/xvdb`, is empty (no partitions), has size of 50 GB, please execute following commands:

```
echo 'n
p
1


t
8e
w' | fdisk -cu /dev/xvdb
pvcreate /dev/xvdb1
vgcreate vg_content /dev/xvdb1
lvcreate -n lv_mongod -L 40G vg_content
lvcreate -n lv_activemq -L 8G vg_content
lvcreate -n lv_w3c -l +100%FREE vg_content
mkfs.ext4 /dev/mapper/vg_content-lv_mongod
mkfs.ext4 /dev/mapper/vg_content-lv_activemq
mkfs.ext4 /dev/mapper/vg_content-lv_w3c
echo "/dev/mapper/vg_content-lv_mongod /content/mongod ext4 defaults 0 0" >> /etc/fstab
echo "/dev/mapper/vg_content-lv_activemq /content/activemq ext4 defaults 0 0" >> /etc/fstab
echo "/dev/mapper/vg_content-lv_w3c /content/w3c ext4 defaults 0 0" >> /etc/fstab
mkdir -p /content/mongod /content/activemq /content/w3c
mount -a
```

Following commands will:

1. Create partition on disk
2. Create LVM structure on disk, including
    1. VG: vg_content
    2. LV: lv_mongod, ext4, 40GB, mounted on /content/mongod
    3. LB: lv_activemq, ext4, 8 GB, mounted on /content/activemq
    4. LV: lv_w3c, ext4, all free space, mounted on /content/w3c

MongoDB installation

1. Create file `/etc/yum.repos.d/mongodb-org-3.2.repo` with following body:

```
[mongodb-org-3.2]
name=MongoDB Repository
baseurl=https://repo.mongodb.org/yum/redhat/$releasever/mongodb-org/3.2/x86_64/
gpgcheck=0
enabled=1
gpgkey=https://www.mongodb.org/static/pgp/server-3.2.asc
```

2. Execute following commands:

```
yum install -y mongodb-org-3.2.3
```

3. Modify `/etc/mongod.conf` to meet following requirements:

```
logpath=/content/mongod/log/mongod.log
dbpath=/content/mongod/db
#bind_ip=127.0.0.1
```

4. Execute following commands:

```
mkdir -p /content/mongod/db /content/mongod/log
chown -R mongod:mongod /content/mongod
chkconfig mongod on
/etc/init.d/mongod start
```

Java

Install java *JDK* from Oracle (1.7) from **rpm**. Please visit http://www.oracle.com/technetwork/java/javase/downloads/jdk7-downloads-1880260.html to download latest 1.7 binaries.

To use html5 validator you need JDK 1.8. Please download tar.gz version and decompress to /usr/java/ (for example jdk1.8.0_51)

W3C validator installation

1. Execute following commands:

```
yum install wget perl-CPAN httpd mod_ssl opensp opensp-devel libxml2-devel perl-CGI perl-Confi
yum groupinstall 'Development Tools'
cpan
o conf commit
echo 'o conf prerequisites_policy follow
install Bundle::W3C::Validator
install Encode::HanExtra
install HTML::Encoding
install SGML::Parser::OpenSP
install XML::LibXML
exit' | perl -MCPAN -e shell
chkconfig httpd on
```

2. Download and extract validator:

```
mkdir -p /content/w3c/validator-src
cd /content/w3c/validator-src

# for validator 1.3
wget https://github.com/w3c/markup-validator/archive/validator-1_3-release.tar.gz
tar xf validator-1_3-release.tar.gz
```

3. Prepare validator execution stack:

```
mkdir -p /etc/w3c
cd /content/w3c/validator-src

mv markup-validator-validator-1_3-release/httpd/cgi-bin /content/w3c
mv markup-validator-validator-1_3-release/{htdocs,share,httpd} /content/w3c
cp /content/w3c/htdocs/config/* /etc/w3c
cp /content/w3c/httpd/conf/httpd.conf /etc/w3c
ln -fs /etc/w3c/httpd.conf /etc/httpd/conf.d/w3c-validator.conf
ln -s /content/w3c /usr/local/validator
```

4. Edit `/etc/w3c/httpd.conf` - add following lines to `<Directory>` section (just inside opening block is fine):

```
Order Allow,Deny
Allow from all
```

5. Edit `/etc/w3c/validator.conf` file to meet following requirements:

```
Base = /usr/local/validator
HTML5 = http://localhost:8888/
```

6. Make sure you have git, python, and **JDK installed**

```
yum install -y git python
```

7. Prepare HTML5 validator environment

```
export JAVA_HOME=/usr/java/jdk1.8.0_51
export PATH=$JAVA_HOME/bin:$PATH
mkdir -p /content/html5
cd /content/html5
git clone https://github.com/validator/validator.git
cd validator
python ./build/build.py all
```

The first time you run the build script, you'll need to be online and the build will need time to download several megabytes of dependencies.

8. Create startup script */content/html5/html5_start.sh* with content

```
#!/bin/bash
export JAVA_HOME=/usr/java/jdk1.8.0_51
export PATH=$JAVA_HOME/bin:$PATH
cd /content/html5/validator
python ./build/build.py run
```

9. Create init script /etc/init.d/html5 with content

```
#!/bin/bash
#
# html5       Starts HTML5 validator.
#
#
# chkconfig: 345 88 12
# description: NU HTML5 validator.
### BEGIN INIT INFO
# Provides: $html5
### END INIT INFO
# Source function library.
. /etc/init.d/functions
[ -f /content/html5/html5_start.sh ] || exit 0
USER=html5
PIDFILE=/content/html5/html5.pid
RETVAL=0
umask 077
start() {
       echo -n $"Starting HTML5: "
    PID=`daemon --user $USER /content/html5/html5_start.sh >>/dev/null 2>&1 & echo $!`
       echo $PID>$PIDFILE
}
stop() {
       echo -n $"Shutting down HTML5: "
       killproc -p $PIDFILE
       rm -f $PIDFILE
}
restart() {
       stop
       start
}
case "$1" in
    start)
```

```
        start
        ;;
    stop)
        stop
        ;;
    restart|reload)
        restart
        ;;
    *)
        echo $"Usage: $0 {start|stop|restart}"
        exit 1
esac
exit $?
```

Execute commands

```
useradd html5
chown -R html5:html5 /content/html5
chmod +x /content/html5/html5_start.sh /etc/init.d/html5
chkconfig --add html5
chkconfig html5 on
```

ActiveMQ installation

1. Goto http://archive.apache.org/dist/activemq/5.13.1/ and download ActiveMQ in version 5.13.1. Save it to `/content/activemq`.

2. Execute following commands:

```
cd /content/activemq
unzip apache-activemq-5.13.1-bin.zip
ln -s apache-activemq-5.13.1 apache-activemq
useradd -r activemq
```

3. Create file `/etc/init.d/activemq` with following body:

```
#!/bin/bash
      #
      # activemq       Starts ActiveMQ.
      #
      #
      # chkconfig: 345 88 12
      # description: ActiveMQ is a JMS Messaging Queue Server.
      ### BEGIN INIT INFO
      # Provides: $activemq
      ### END INIT INFO

      # Source function library.
      . /etc/init.d/functions

      [ -f /content/activemq/apache-activemq/bin/activemqstart.sh ] || exit 0
      [ -f /content/activemq/apache-activemq/bin/activemqstop.sh ] || exit 0

      RETVAL=0

      umask 077

      start() {
             echo -n $"Starting ActiveMQ: "
             daemon su -c /content/activemq/apache-activemq/bin/activemqstart.sh activemq
             echo
             return $RETVAL
      }
      stop() {
```

```
        echo -n $"Shutting down ActiveMQ: "
        daemon su -c /content/activemq/apache-activemq/bin/activemqstop.sh activemq
        echo
        return $RETVAL
}
restart() {
        stop
        start
}
case "$1" in
    start)
        start
        ;;
    stop)
        stop
        ;;
    restart|reload)
        restart
        ;;
    *)
        echo $"Usage: $0 {start|stop|restart}"
        exit 1
esac
exit $?
```

4. Create file `/content/activemq/apache-activemq/bin/activemqstart.sh` with following body:

```
#!/bin/bash
export ACTIVEMQ_HOME=/content/activemq/apache-activemq
/content/activemq/apache-activemq/bin/activemq start
```

5. Create file `/content/activemq/apache-activemq/bin/activemqstop.sh` with following body:

```
#!/bin/bash
export ACTIVEMQ_HOME=/content/activemq/apache-activemq
/content/activemq/apache-activemq/bin/activemq stop
```

6. Go with manuals below:

    1. Enable JMX for ActiveMQ

    2. Switch Persistence for ActiveMQ

7. Execute following commands:

```
chmod a+x /etc/init.d/activemq /content/activemq/apache-activemq/bin/activemqstop.sh /content/a
chown -R activemq:activemq /content/activemq
chkconfig --add activemq
chkconfig activemq on
/etc/init.d/activemq start
```

Enable JMX for *ActiveMQ*

In order to do it, following steps should be proceeded:

1. In ActiveMQ config file `/content/activemq/apache-activemq/conf/activemq.xml` in `<broker>` markup, set `useJmx="true"` property:

```
<broker xmlns="http://activemq.apache.org/schema/core" useJmx="true" brokerName="localhost" da
```

2. In ActiveMQ config file `/content/activemq/apache-activemq/conf/activemq.xml` set `<managementContext>` to create connector and use port `11199` for JMX communication:

```
<managementContext>
    <managementContext createConnector="true" connectorPort="11199"/>
</managementContext>
```

3. Ensure `advisorySupport` is on. In order to do that check config file `/content/activemq/apache-activemq/conf/activemq.xml, <broker>` section should have set `advisorySupport` property to true or not set (true by default).

   ```
   <broker advisorySupport="true" ...
   ```

4. In ActiveMQ file `/content/activemq/apache-activemq/bin/activemq` set java `JMX` parameters

   ```
   ACTIVEMQ_SUNJMX_START="$ACTIVEMQ_SUNJMX_START -Dcom.sun.management.jmxremote -Dcom.sun.manageme
   ```

More details about JMX configuration for *ActiveMQ* can be found here: http://activemq.apache.org/jmx.html.

Switch Persistence for ActiveMQ

In config file `/content/activemq/apache-activemq/conf/activemq.xml, <broker>` definition set `persistent` property to **false**:

```
<broker ... persistent="false">
```

Enable cleaning unused topic for ActiveMQ

In order to do it, following steps should be proceeded:

1. In config file `/content/activemq/apache-activemq/conf/activemq.xml, <broker>` definition set `schedulePeriodForDestinationPurge` property to **300000**:

   ```
   <broker ... schedulePeriodForDestinationPurge="300000">
   ```

2. In config file `/content/activemq/apache-activemq/conf/activemq.xml inside <broker>, <destinationPolicy>, <policyMap>, <policyEntries> add following destination policy entry:`

   ```
   <policyEntry topic=">" gcInactiveDestinations="true" inactiveTimoutBeforeGC="600000"/>
   ```

With this configuration Active MQ will check for inactive destination every 5 minutes. And it will delete all topics that are inactive for 10 minutes.

3.3. Windows setup - 1.3

Graphic settings

Change default console resolution - install VNC server (e.g. http://www.tightvnc.com/), connect by VNC client to console and change resolution (min. 1024x768).

Windows configuration

Turn off Windows Firewall (both, private and public network location settings).

Software setup

Requirements

Installed *Java 7* jdk (jdk-7u55-windows-x64) with proper `JAVA_HOME` environment variable.

Prerequisites

1. Create directory `C:\content` .

3.3.1. Karaf setup - 1.3

Setup

1. Extract *karaf.zip* into `C:\content` .
2. Run *karaf.bat* from `C:\content\karaf\bin`. *Karaf* should install all features from `C:\content\karaf\deploy\aet-feature.xml` by itself, but it might take some time as it has to collect all dependencies (2-3 minutes). The more advisable approach is to explicitly tell *Karaf* to install *aet-features* in *Karaf* console:

   ```
   features:install aet-features
   ```

3. Try to access *Karaf Web Console* - (default: http://localhost:8181/system/console/, default: login/password is `karaf/karaf`).

Install Karaf as a service

Important information

By default maximum memory allocation pool size is setted to 3072M. I can be changed in `C:\content\karaf\bin\service-install.bat` (Line 3: `SET MAX_MEMORY_ALLOCATION_POOL_SIZE=3072M`)

1. Close *Karaf* console.
2. Run *service-install.bat* from `C:\content\karaf\bin`.
3. Restart *Windows and* try to access *Karaf Web Console*.
4. Karaf should be also visible in Services panel.

3.3.2. Browsermob proxy setup - 1.3

Setup

1. Extract *browsermob.zip* into `C:\content` .
2. Run *service-install.bat* from `C:\content\browsermob\bin` .
3. Try to access http://localhost:9272/proxy and check if service is running.
4. *Browsermob* service should be visible in Services panel.

Known issue

There is known issue with running Browsermob as Service on Windows 10.

Browsermob service works fine on Windows 7.

3.3.3. Firefox setup - 1.3

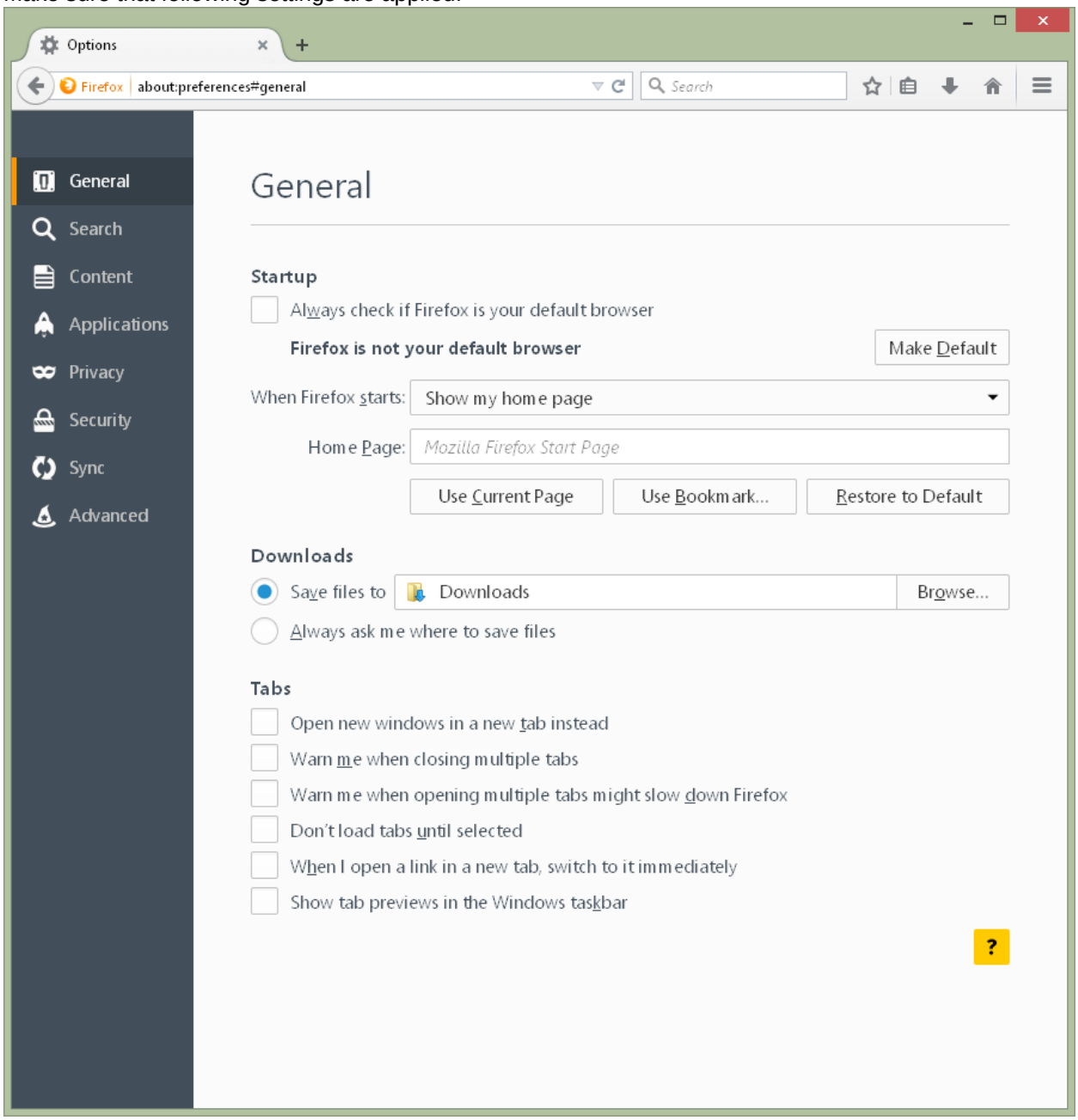
Firefox

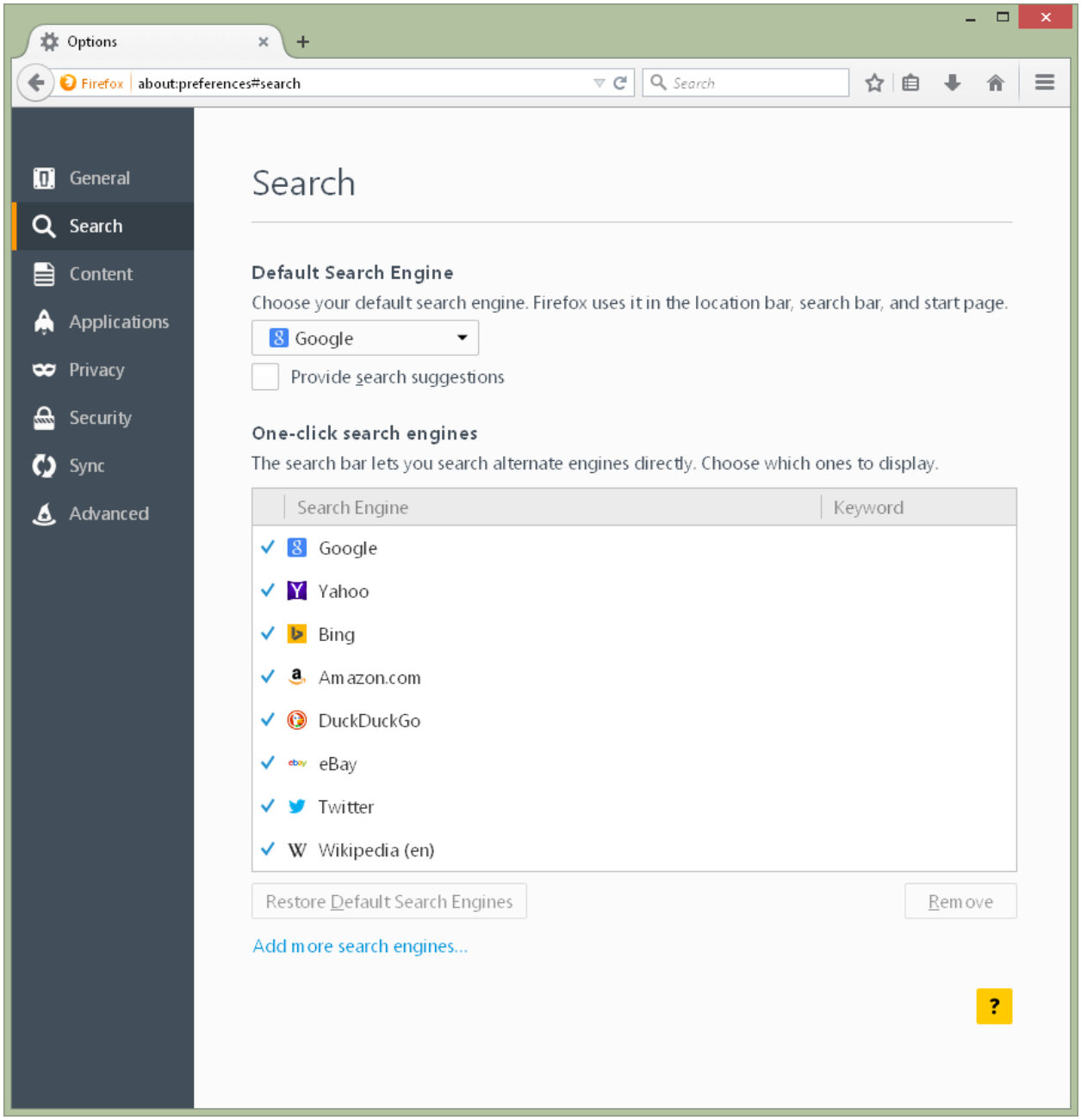
Currently used version of Firefox is 38.6.0 ESR (en-US) together with Selenium 2.50.1.

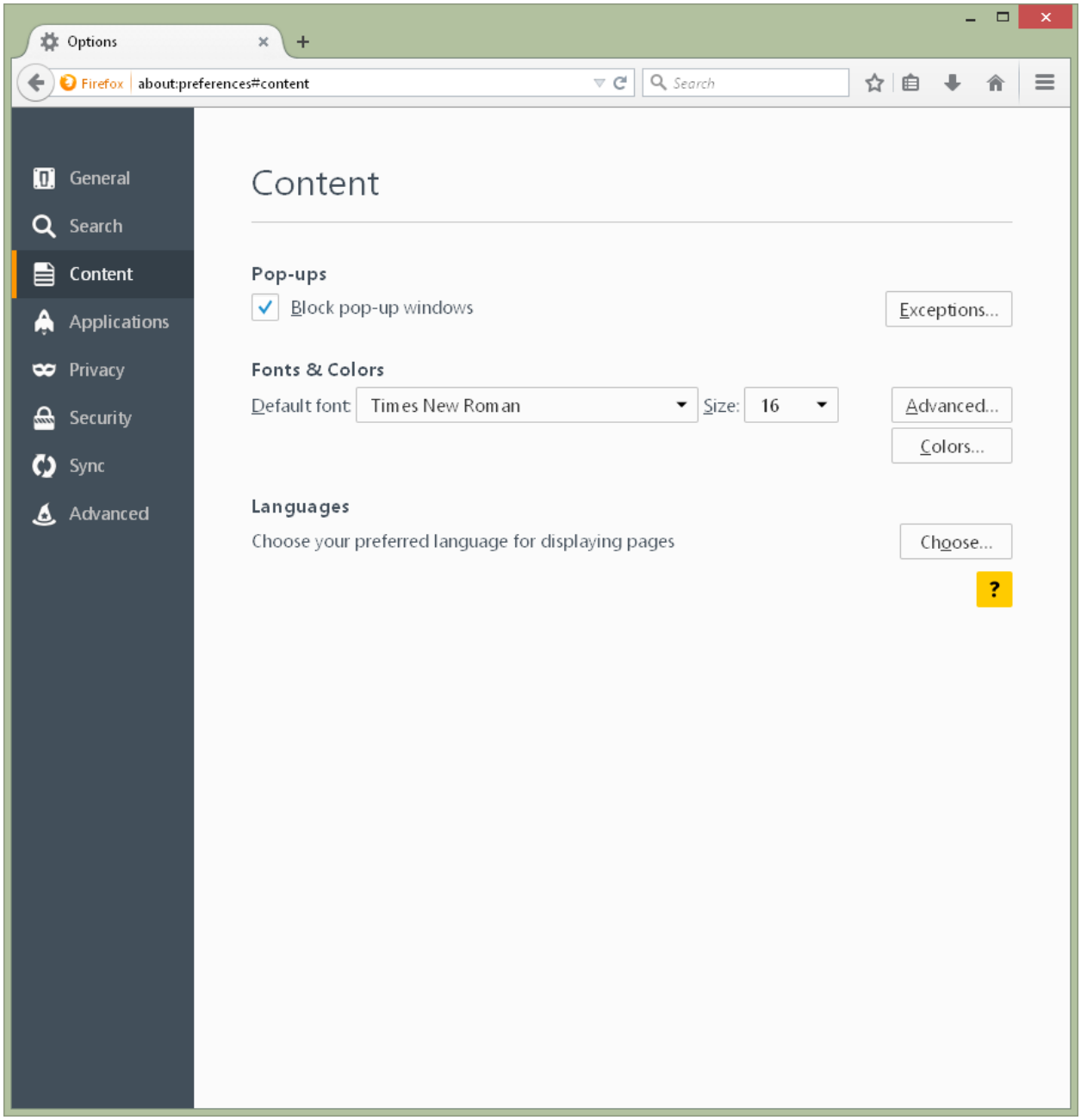
Setup

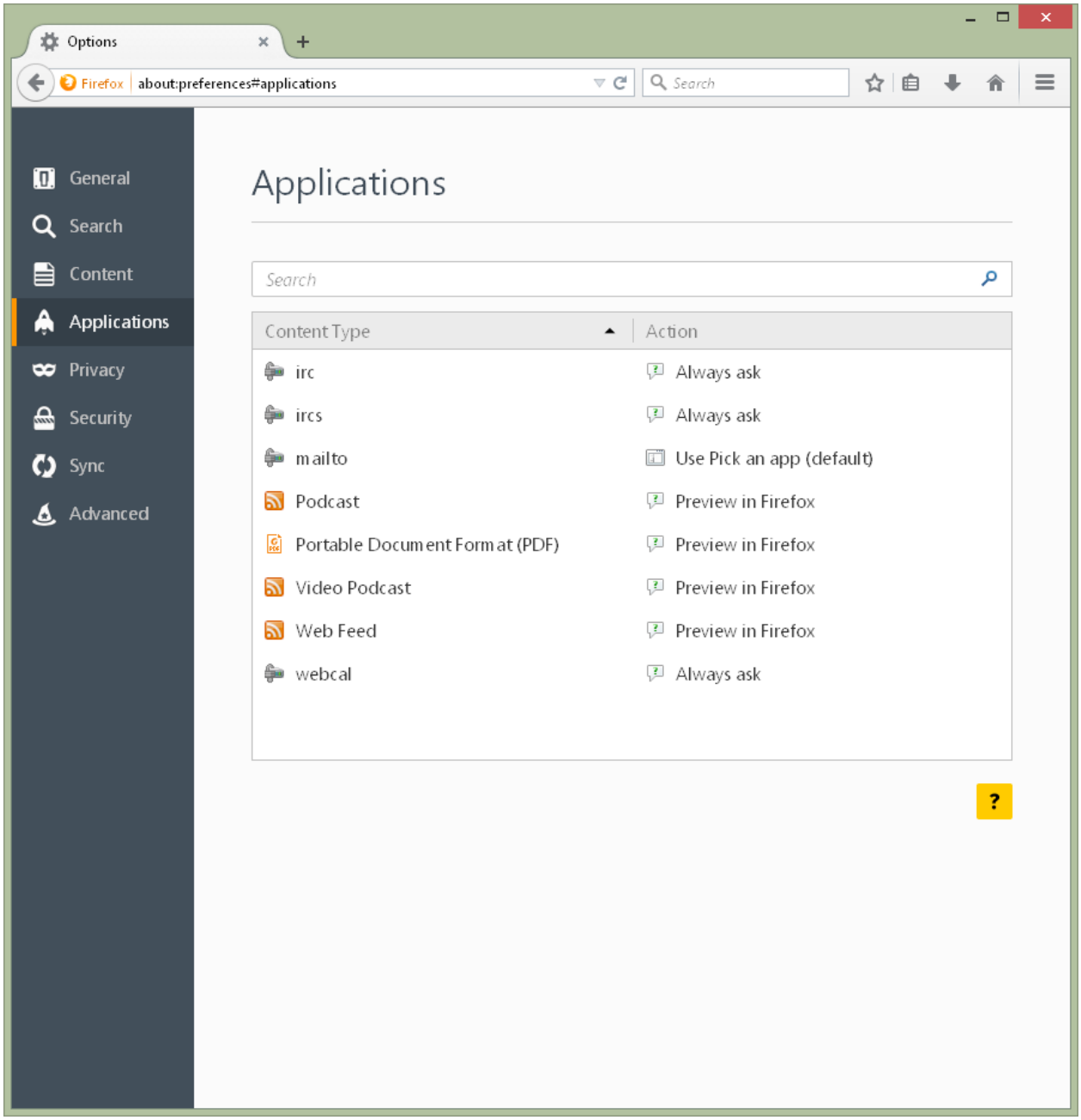
1. Download `FireFox 38.6.0 ESR` from https://ftp.mozilla.org/pub/firefox/releases/38.6.0esr/ (e.g. win32/en-US version).
2. Install Firefox with custom installation, without update agent.
3. Run firefox after installing and open Options.

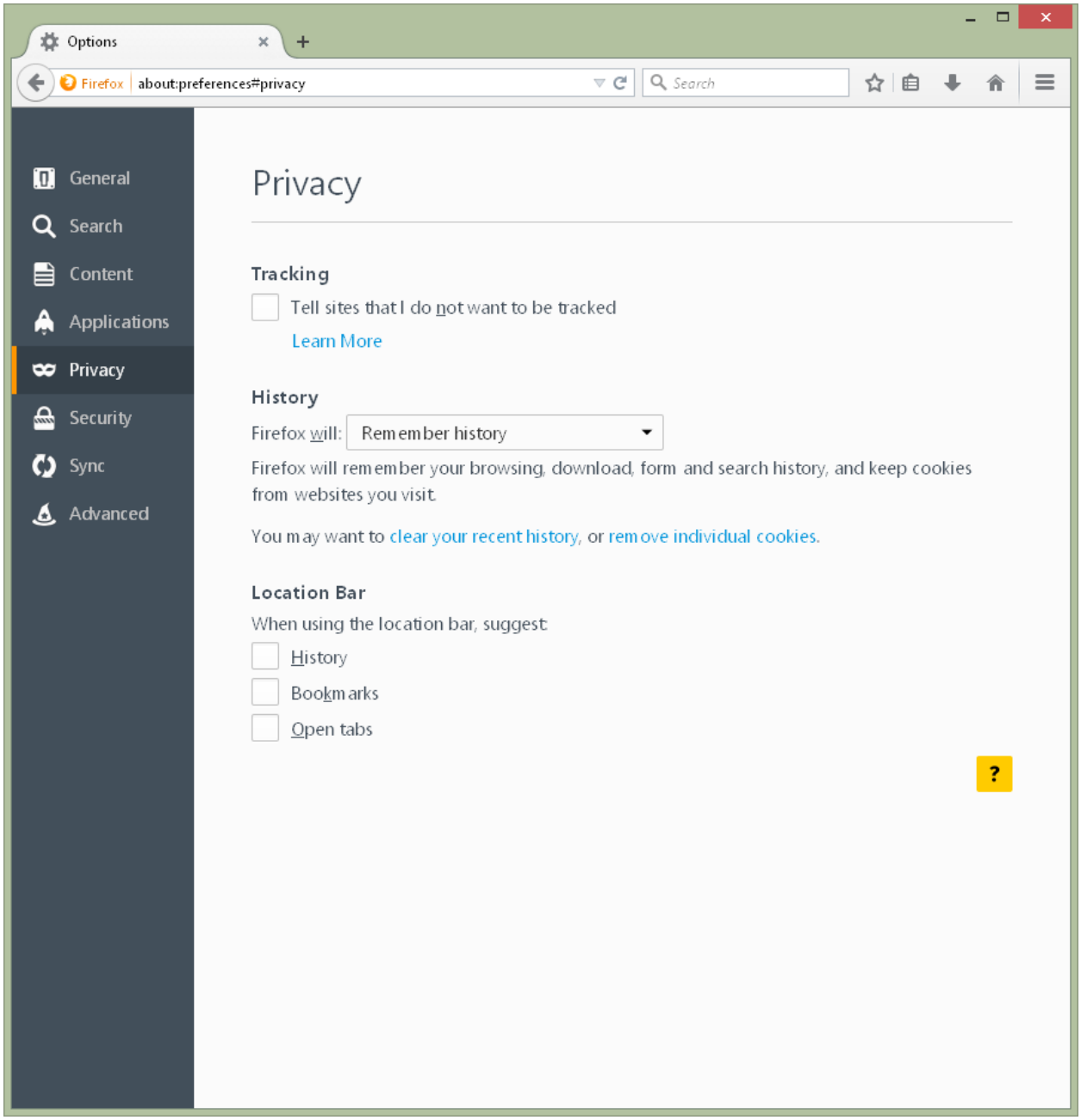
4. Make sure that following settings are applied:

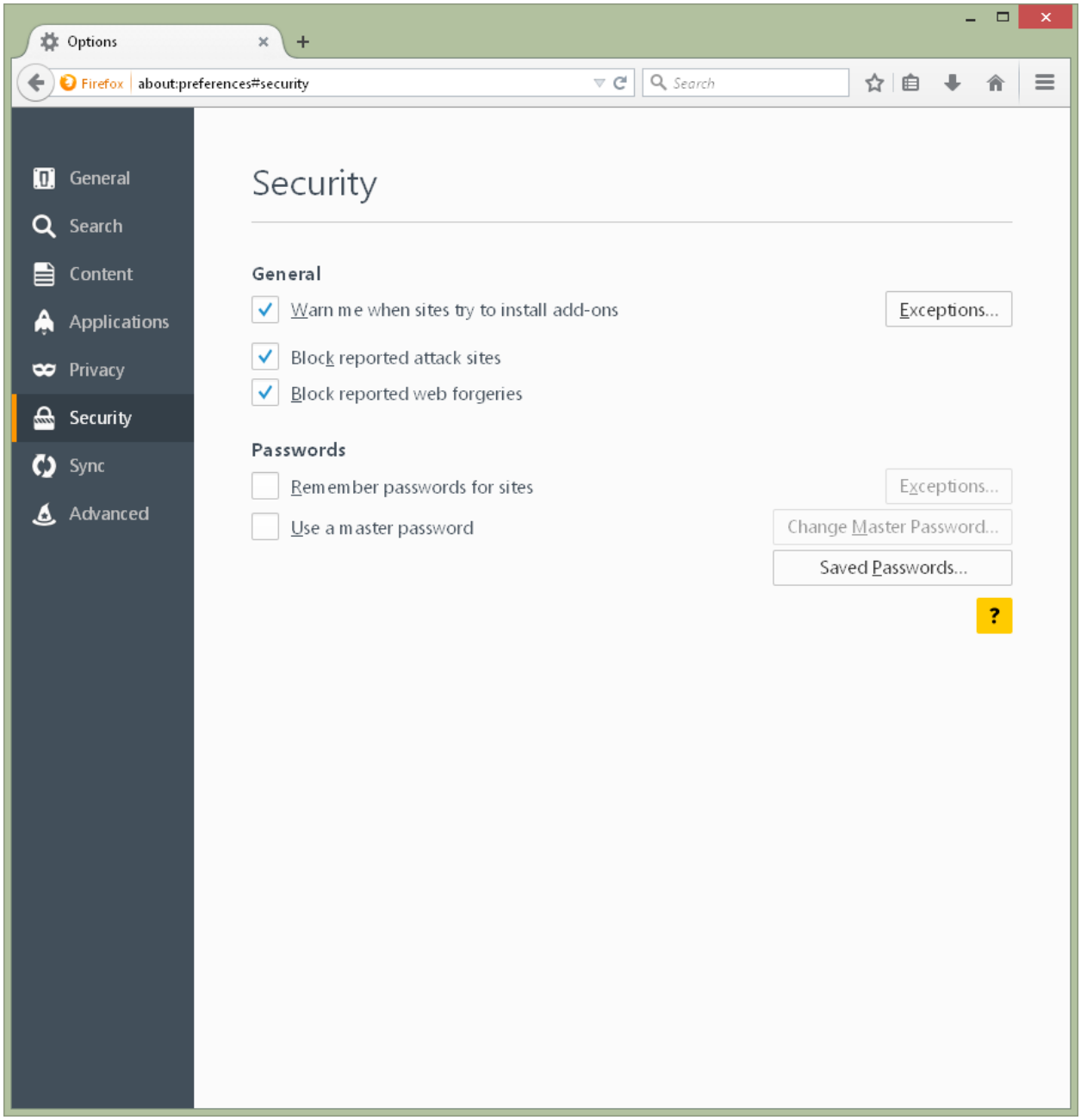


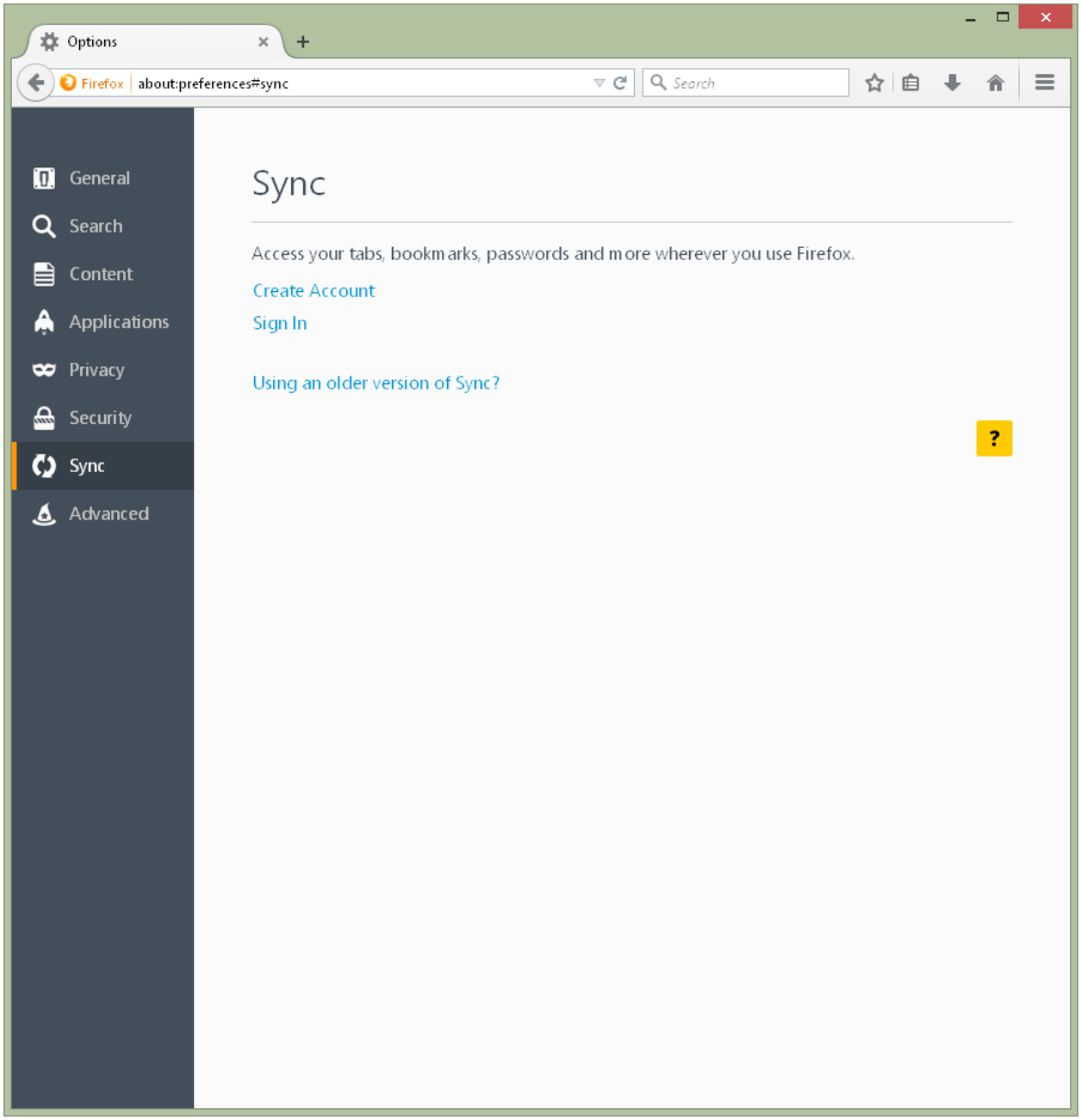

Options
Firefox
about:preferences#search
Search
General
Search
Content
Applications
Privacy
Security
Sync
Advanced
Search
Default Search Engine
Choose your default search engine. Firefox uses it in the location bar, search bar, and start page.
Google
Provide search suggestions
One-click search engines
The search bar lets you search alternate engines directly. Choose which ones to display.
Search Engine
Keyword
Google
Yahoo
Bing
Amazon.com
DuckDuckGo
eBay
Twitter
Wikipedia (en)
Restore Default Search Engines
Remove
Add more search engines...
?

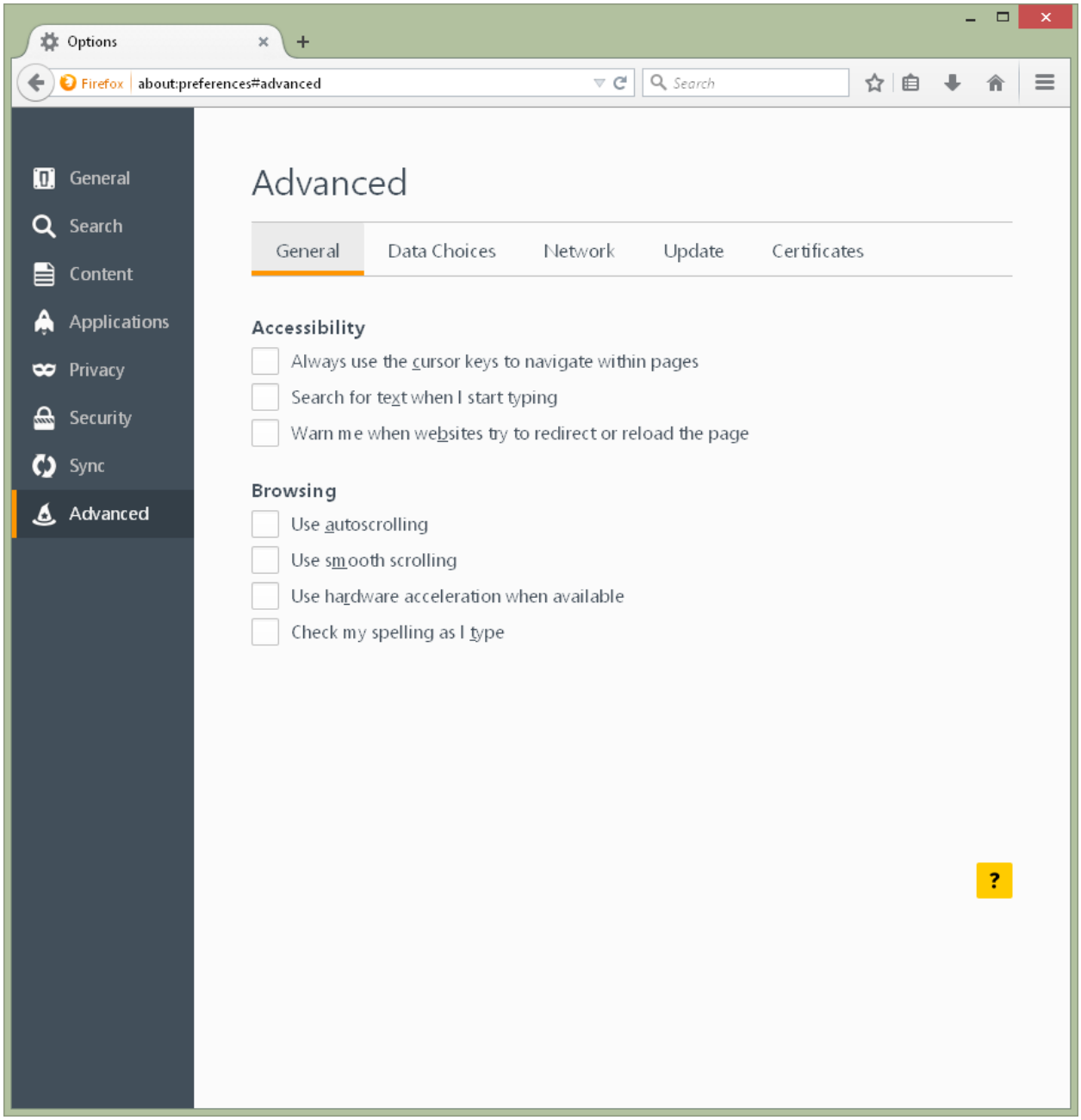

Options
Firefox
about:preferences#content
Search
General
Search
Content
Applications
Privacy
Security
Sync
Advanced
Content
Pop-ups
Block pop-up windows
Exceptions...
Fonts & Colors
Default font
Times New Roman
Size:
16
Advanced...
Colors...
Languages
Choose your preferred language for displaying pages
Choose...
?

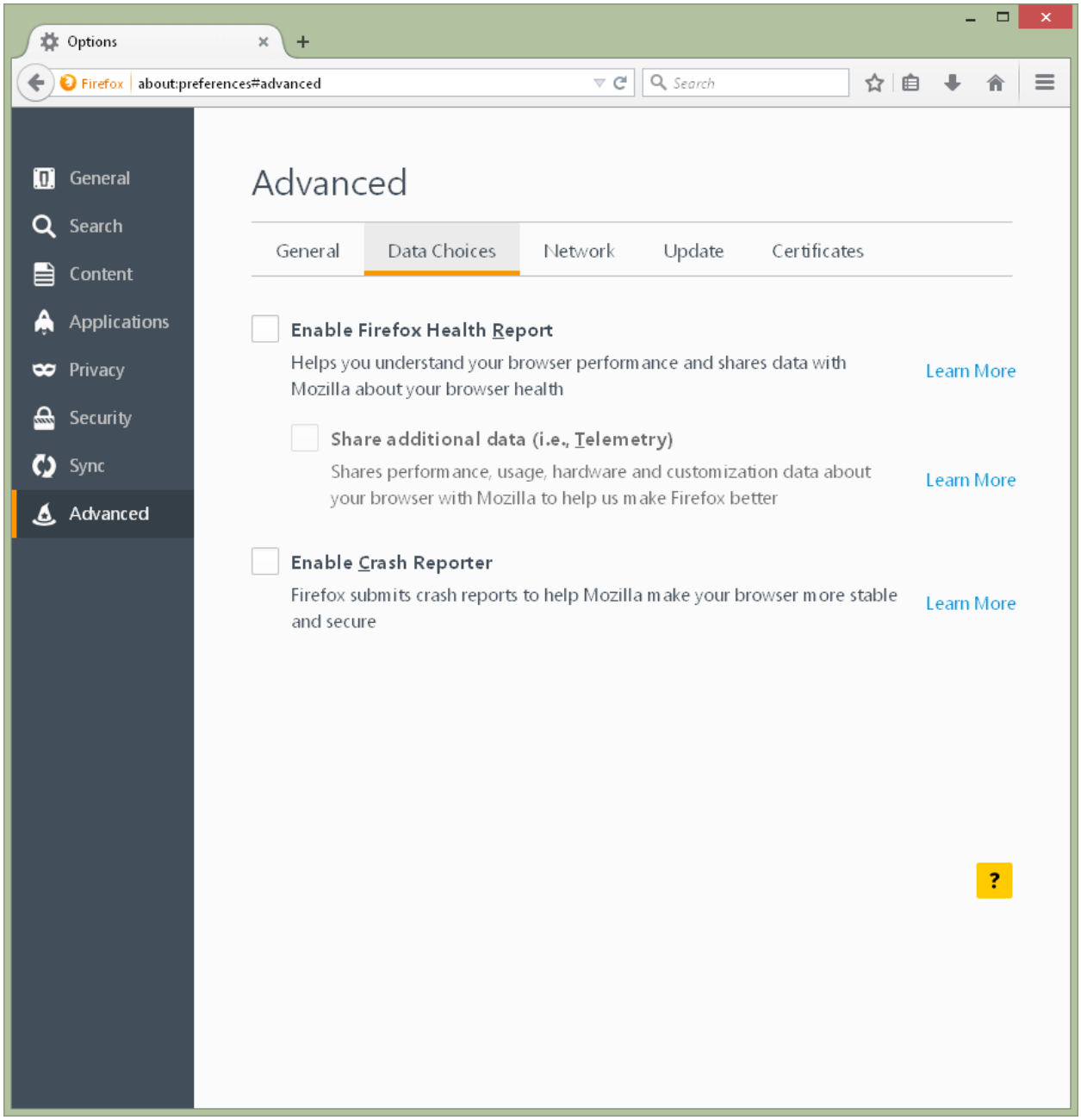

Options
Firefox
about:preferences#applications
Search
General
Search
Content
Applications
Privacy
Security
Sync
Advanced
Applications
Search
Content Type
Action
irc
Always ask
ircs
Always ask
mailto
Use Pick an app (default)
Podcast
Preview in Firefox
Portable Document Format (PDF)
Preview in Firefox
Video Podcast
Preview in Firefox
Web Feed
Preview in Firefox
webcal
Always ask
?

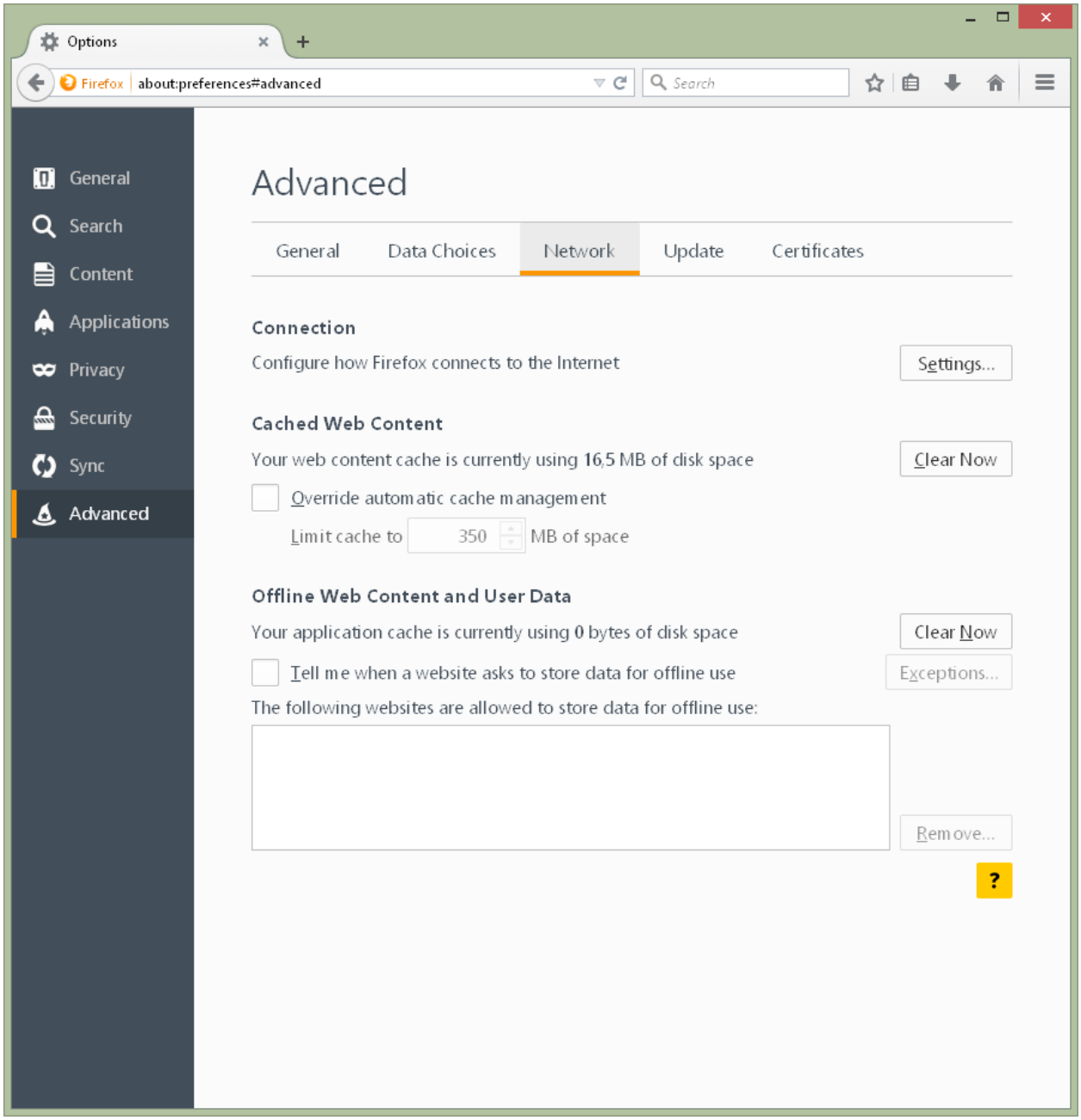

Options
Firefox
about:preferences#privacy
Search
General
Search
Content
Applications
Privacy
Security
Sync
Advanced
Privacy
Tracking
Tell sites that I do not want to be tracked
Learn More
History
Firefox will:
Remember history
Firefox will remember your browsing, download, form and search history, and keep cookies from websites you visit.
You may want to clear your recent history, or remove individual cookies.
Location Bar
When using the location bar, suggest:
History
Bookmarks
Open tabs
?

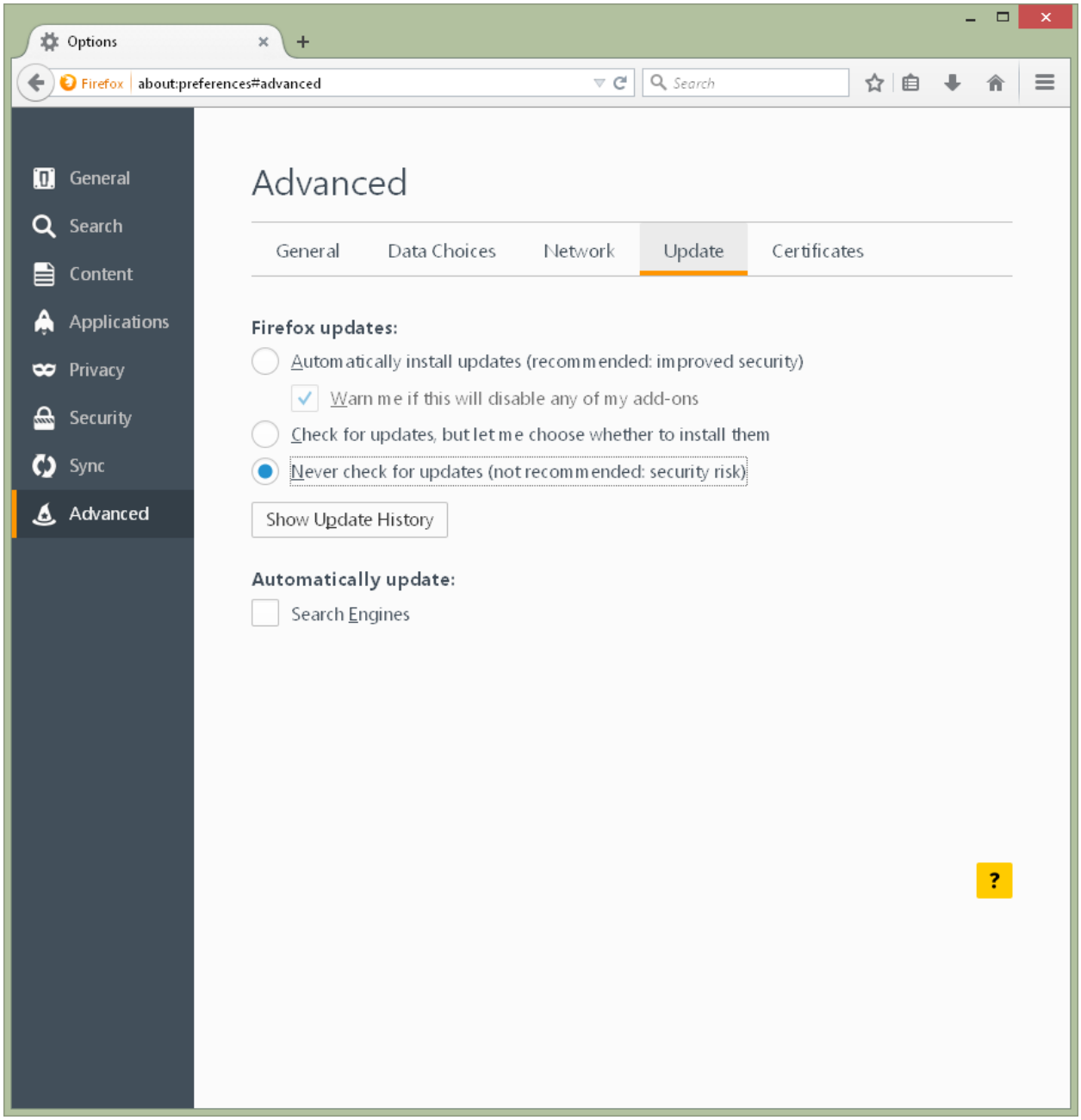

Options
Firefox
about:preferences#security
Search
General
Search
Content
Applications
Privacy
Security
Sync
Advanced
Security
General
Warn me when sites try to install add-ons
Exceptions...
Block reported attack sites
Block reported web forgeries
Passwords
Remember passwords for sites
Exceptions...
Use a master password
Change Master Password...
Saved Passwords...
?

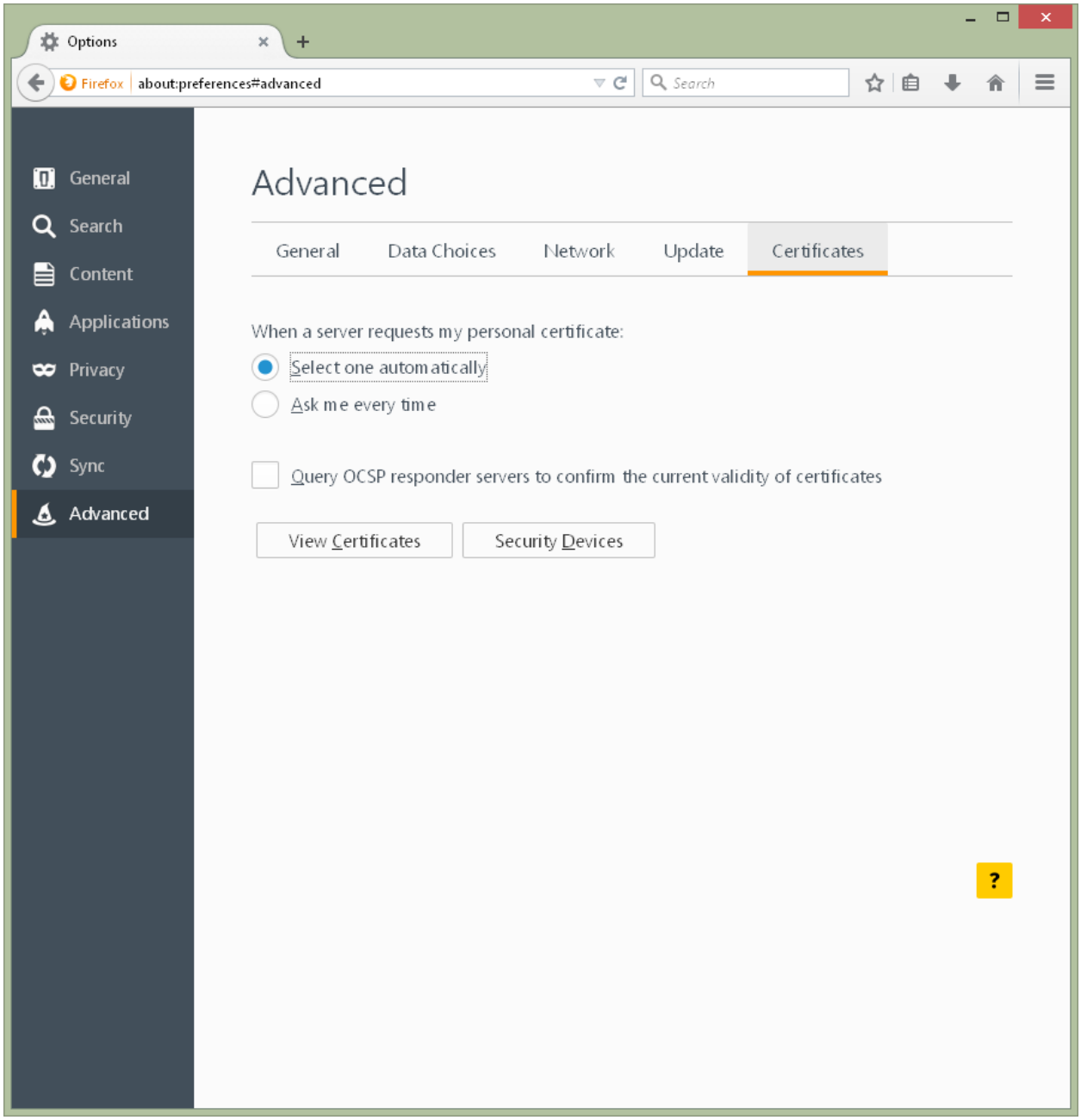

Options
Firefox
about:preferences#sync
Search
General
Search
Content
Applications
Privacy
Security
Sync
Advanced
Sync
Access your tabs, bookmarks, passwords and more wherever you use Firefox.
Create Account
Sign In
Using an older version of Sync?
?


Options
Firefox
about:preferences#advanced
Search
General
Search
Content
Applications
Privacy
Security
Sync
Advanced
Advanced
General
Data Choices
Network
Update
Certificates
Accessibility
Always use the cursor keys to navigate within pages
Search for text when I start typing
Warn me when websites try to redirect or reload the page
Browsing
Use autoscrolling
Use smooth scrolling
Use hardware acceleration when available
Check my spelling as I type
?


Options
Firefox
about:preferences#advanced
Search
General
Search
Content
Applications
Privacy
Security
Sync
Advanced
Advanced
General
Data Choices
Network
Update
Certificates
Enable Firefox Health Report
Helps you understand your browser performance and shares data with Mozilla about your browser health
Learn More
Share additional data (i.e., Telemetry)
Shares performance, usage, hardware and customization data about your browser with Mozilla to help us make Firefox better
Learn More
Enable Crash Reporter
Firefox submits crash reports to help Mozilla make your browser more stable and secure
Learn More
?


Options
Firefox
about:preferences#advanced
Search
General
Search
Content
Applications
Privacy
Security
Sync
Advanced
Advanced
General
Data Choices
Network
Update
Certificates
Connection
Configure how Firefox connects to the Internet
Settings...
Cached Web Content
Your web content cache is currently using 16,5 MB of disk space
Clear Now
Override automatic cache management
Limit cache to
350
MB of space
Offline Web Content and User Data
Your application cache is currently using 0 bytes of disk space
Clear Now
Tell me when a website asks to store data for offline use
Exceptions...
The following websites are allowed to store data for offline use:
Remove...
?


Options
Firefox
about:preferences#advanced
Search
General
Search
Content
Applications
Privacy
Security
Sync
Advanced
Advanced
General
Data Choices
Network
Update
Certificates
Firefox updates:
Automatically install updates (recommended: improved security)
Warn me if this will disable any of my add-ons
Check for updates, but let me choose whether to install them
Never check for updates (not recommended: security risk)
Show Update History
Automatically update:
Search Engines
?



5. Shutdown firefox to save settings.

Firefox

Previously used version of Firefox was 30.0 (en-US) - together with Selenium 2.44.0.

3.3.4. Check connections - 1.3

Requirements

To check connections *Telnet Client* is needed.

Tests presented below verifies whether the connection is possible. Responses from *Telnet*'s connections are irrelevant.

All tests should be done from *Windows* machine using private IP address.

MongoDB

1. Connect to MongoDB using *Telnet Client*:

   ```
   telnet ${LINUX_MACHINE_PRIVATE_IP} 27017
   ```

ActiveMQ

1. Connect to ActiveMQ using *Telnet Client*:

   ```
   telnet ${LINUX_MACHINE_PRIVATE_IP} 61616
   ```

2. Connect to ActiveMQ's JMX using JConsole:

   ```
   jconsole.exe ${LINUX_MACHINE_PRIVATE_IP}:11199
   ```

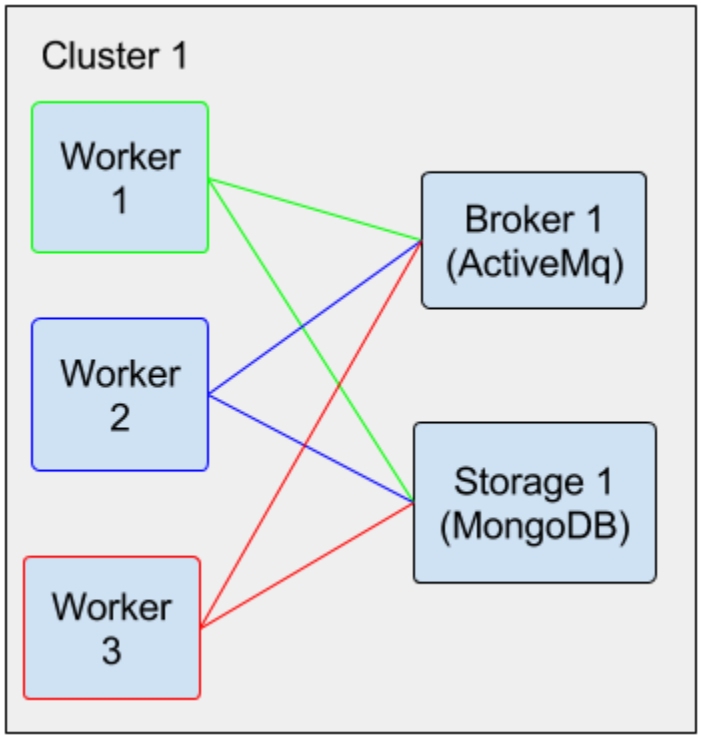
W3C Validator

1. Verify if `${LINUX_MACHINE_PRIVATE_IP}`/w3c-validator/ is available.

3.4. Cluster configuration - 1.3

AET Cluster Configuration

To setup AET cluster, Workers must be 'connected' to same broker (ActiveMq) and storage server (MongoDB).

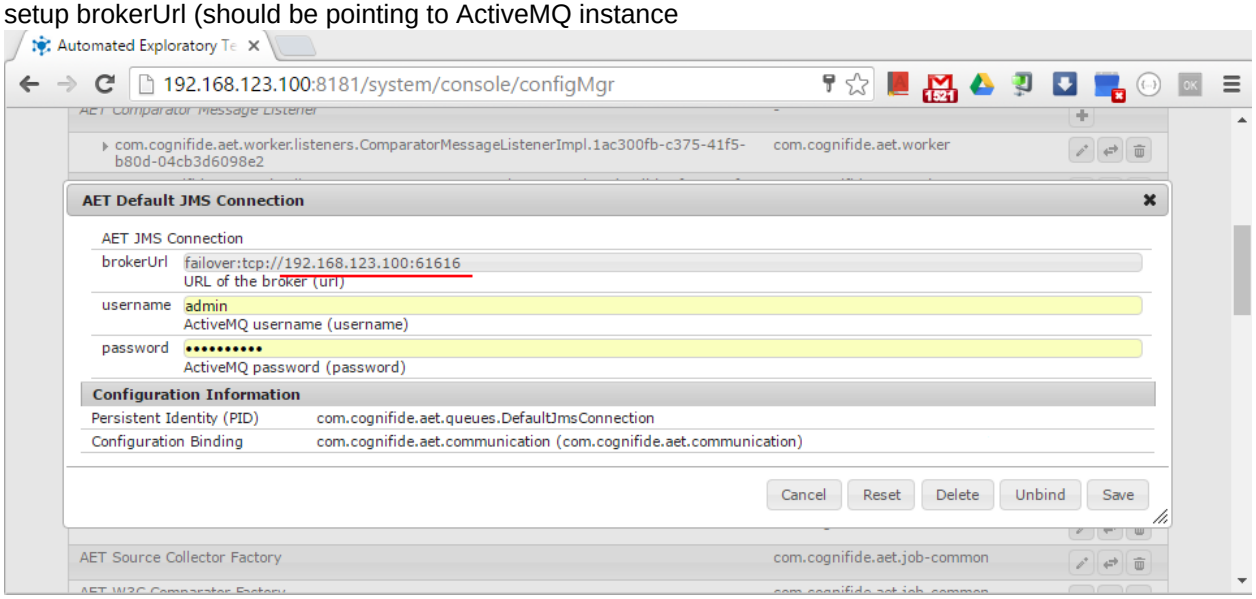


Adding single worker to a Cluster:

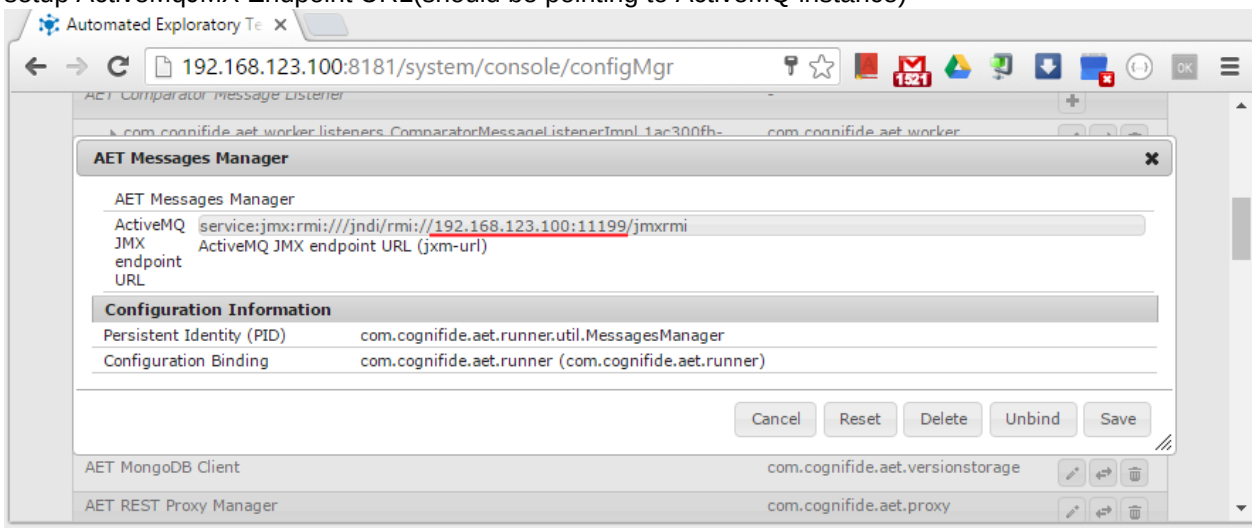
*assuming that Broker and Storage setup is done:

1. Open worker configuration manager in you browser
   http://<worker-ip-addres:port|domain>/system/console/configMgr

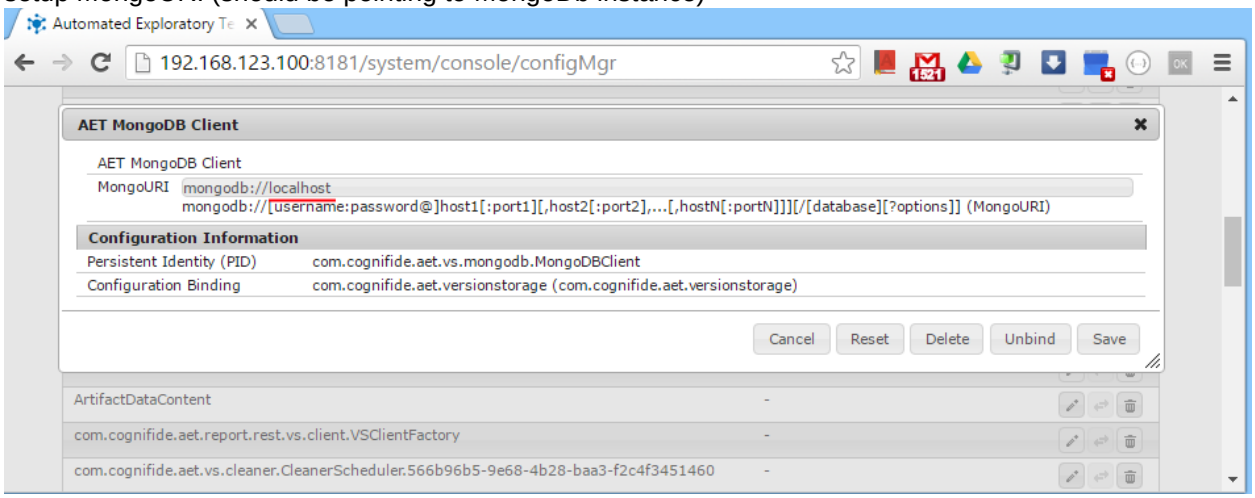
2. Find and setup "AET Default JMS Connection" configuration.
   setup brokerUrl (should be pointing to ActiveMQ instance



3. Find and setup "AET Messages Manager" configuration.
   setup ActiveMqJMX Endpoint URL(should be pointing to ActiveMQ instance)



4. Find and setup "AET MongoDB Client" configuration
   setup MongoURI (should be pointing to MongoDb instance)



*Don't forget to save changes after changing each configuration.

Repeat those steps for each worker you want to add to Cluster.

4. Project setup and configuration - 1.3

This section covers AET System setup and configuration.

4.1. Application installation - 1.3

AET installation on Windows

1. Login into *Windows* machine.
2. From *aet.zip* extract the content from the *deploy* directory into `C:\content\karaf\deploy`.
   - After this step, the `deployed` directory should contain a total of 20 files:
     - 16 AET bundles (see System modules for more details),
     - `aet-features.xml` file (configuration file with set of AET libraries dependencies automatically installed by *Karaf* on startup),
     - 3 third-party libaries: `diff_match_patch-current`, `JSErrorCollector-0.5-atlassian-2`, `org.apache.karaf.webconsole.branding-2.3.9.`
3. Using *Web Console* (http://localhost:8181/system/console/bundles) check if all bundles starting with "*Cognifide AET*" are active - there should be a total of 16 of them (see System modules for more details). If not then try to start them manually using *Web Console*.

Problems and troubleshooting

Problem: After deploying bundles into *Karaf* `deploy` directory and waiting several minutes bundles *Cognifide AET* in *Web Console* are not active. Starting them manually does not work.

Solution: Restart *Karaf* and *Windows* machine.

4.2. Application configuration - 1.3

Basic configuration

1. Login into *Windows* machine.
2. From *aet.zip* extract content of `etc` directory into `C:\content\karaf\etc`. Overwrite all files if necessary.
3. Using *Web Console Configuration* (http://localhost:8181/system/console/configMgr) configure following components:

    1. AET Default JMS Connection

Property	Value	Description
`brokerUrl`	`failover:tcp://${LINUX_VM_PRIVATE_IP_ADDRESS}:61616`	Url to JMS Server. Application will be waiting for messages from this sytem.
`username`	`admin`	JMS Server authentication username. In ActiveMQ default value is *admin*.
`password`	`admin`	JMS Server authentication password . In ActiveMQ default value is *admin*.

    2. AET GridFs Storage

Property	Value	Description
`Url`	`https://${REST_API_ADDRES_WITH_SSL_SUPPORT}/cxf/aet`	Url to REST Api. TODO This part will be updated when solution for SSL support will be ready.
`MongoURI`	`mongodb://${LINUX_VM_PRIVATE_IP_ADDRESS}`	Url to *MongoDB* which in this configuration is installed on *Linux* machine. This url will be used to create TCP connection between AET bundles and database.

    3. AET Messages Manager

Property	Value	Description
ActiveMQ JMX endpoint URL	`service:jmx:rmi:///jndi/rmi://${LINUX_VM_PRIVATE_IP_ADDRESS}:1199/jm`	This is url for JMX connection to ActiveMQ JMS Server. This connection is used to manage queues and topics from AET Application level.

Custom configuration

Following steps are optional. All described below configurations were set with default values in step 2. from *Basic configuration* part.

AET Collector Message Listener

Number of `AET Collector Message Listeners` specifies number of available collectors. The number of all registered collectors for specified `Consumer queue name` is visible at *ActiveMQ Web Console* at *Queues* view as the *Number of Consumers* for this specific consumer queue.

Configuration provides the consumer queue name that the collector will listen for messages and producer queue name that the collector results will be sent to. The `Consumer queue name` should describe configuration and environment that the test is run with (example: `AET.win7-ff16`). Also specifies the name of *Web Driver* that will be used - this name should match the `AET Firefox Web Driver Factory`'s `Name` property (see below).

Each collector must have unique `Collector name` and `Embedded Proxy Server Port` properties.

Property	Default value	Description
`Collector name`	`Collector`	Name of collector. Used in logs only.
`Prefetch size`	`1`	See http://activemq.apache.org/what-is-the-prefetch-limit-for.html.
`Consumer queue name`	`AET.win7-ff16`	Consumer queue name that describes the test run environment (machine + browser). Collector uses this queue to read collection tasks from the system.
`Producer queue name`	`AET.collectorResults`	Producer queue name. Collector uses this queue to send collection results and inform about finished collection.
`Web Driver name`	`ff`	Name of used Web Driver. Should be the same as name defined in *AET Firefox WebDriver Factory*.
`Embedded Proxy Server Port`	`4501`	Proxy Server Port. Should be different for each collector.

AET Comparator Message Listener

Number of `AET Comparator Message Listeners` specifies number of available comparators. The number of all registered comparators for specified `Consumer queue name` is visible at *ActiveMQ Web Console* at *Queues* view as the *Number of Consumers* for this specific consumer queue (`AET.comparatorJobs` by default).

Specifies the consumer queue name that the comparator will listen for messages and producer queue name that the comparator results will be sent to.

Property	Default value	Description
`Comparator name`	`Comparator`	Name of comparator. Used in logs only. Should be unique.
`Prefetch size`	`1`	See http://activemq.apache.org/what-is-the-prefetch-limit-for.html .
`Consumer queue name`	`AET.comparatorJobs`	Consumer queue name. Comparator uses this queue to read comparison tasks from the system.
`Producer queue name`	`AET.comparatorResults`	Producer queue name. Comparator uses this queue to inform about finished comparison.

AET Reporter Message Listener

Number of `AET Reporter Message Listeners` specifies number of available report generators. The number of all registered reporters is visible at *ActiveMQ Web Console* at *Queues* view as the *Number of Consumers* for this specific consumer queue (`AET.reporterJobs` by default).

Property	Default value	Description
`Reporter name`	`Reporter`	Name of reporter. Used in logs only. Should be unique.
`Prefetch size`	`1`	See http://activemq.apache.org/what-is-the-prefetch-limit-for.html.
`Consumer queue name`	`AET.reporterJobs`	Consumer queue name. Reporter uses this queue to read report generation tasks from the system.
`Producer queue name`	`AET.reporterResults`	Producer queue name. Reporter uses this queue to inform about finished report generation.

AET REST Proxy Manager

Property	Default value	Description
`Server`	`localhost`	*BrowserMob* server address. According to this guide tutorial, *BrowserMob* is installed on *Windows* machine and is accessible locally.
`Port`	`9272`	*BrowserMob* API port.

AET Runner

Property	Default value	Description
`Failure timeout`	`360`	Time in seconds after which test run will be interrupted if no progress over task was recorded in the system in duration of this parameter.
`Message ttl`	`420`	Time in seconds after which messages will be dequeued if message was not consumed.
`URL Package Size`	`5`	Defines how many links are being sent in one collection task message. Each message is being processed by single *Collector Messages Listener*.
`Max Messages in Collector Queue`	`5`	Defines the maximum amount of messages in the collector queue at one time.
`Runner mode`	`online`	Defines mode in which Runner can work. Possible options are: ▪ **`online`** - Runner works in this mode by default. Only requests on queue `AET.runner-in` are processed normally. Runner does not listen on `AET.maintenance-in` queue. ▪ **`maintenance`** - Requests form queue `AET.runner-in` are rejected with proper message. Requests on queue `AET.maintenance-in` are processed. ▪ **`offline`** - Only requests on queue `AET.maintenance-in` are processed. Runner does not listen on `AET.runner-in` queue. This parameters is used during deployments to prevent system from processing normal tests. Only tests that are used to check sanity (maintenance tests) can be accepted and processed by the system.

AET Source Collector Factory

Property	Default value	Description
`Timeout value for source collector`	`20000`	Timeout value in ms after which source collection that didn't finished will be aborted.

AET Rebase Service

Property	Default value	Description
`nRebaseThreads`	`5`	The number of threads available for rebase.
`timeout`	`2000`	Defines how long (ms) rebase would wait for operation status before returning "Rebase in progress".
`cacheTimeout`	`15000`	Defines how long (ms) operation status will be available after last access.

AET Firefox WebDriver Factory

Property	Default value	Description
`Name`	`ff`	Name of Web Driver. Used in *AET Collector Message Listener*.
`Custom path to Firefox binary`		Path to *Firefox* binary installed in system. This path is used by *WebDriver* to localize *Firefox* browser. When left empty, *WebDriver* will try to localize *Firefox* binary by itself.

AET Chrome WebDriver Factory

Warning

Not used by the system in current version.

Property	Default value	Description
`Name`	`chrome`	Name of Web Driver. Used in *AET Collector Message Listener*.
`Custom path to Chrome binary`		Path to *Chrome* binary installed in system.

AET Cleaning Scheduler Service

This service is responsible for running *Cleaner* service. *Cleaner* is responsible for removing old data artifacts that are not used any more by the system in order to save used database disk space.

Property	Default value	Description
`Schedule`		CRON notation of when the cleaner job will be fired. [example: '`0 0 21 ? * *`' will trigger job daily at 21:00].
`Last versions to keep`	`1`	Defines number of artifacts versions that will be left after clean operation. If left empty, only one version will be kept after cleaning operation.
`Remove artifacts older than`	`5`	Defines how old files should be removed. Works as conjunction with *Last versions to keep*.
`Artifact Types`		Comma-separated list of artifact types that are to be removed in scheduled event. Available values: `DATA, RESULTS, REPORTS, PATTERNS`.
`Company Name`	`*`	Name of the company for which cleaning will be performed. Use '`*`' to trigger this job for each company on database.
`Project Name`	`*`	Name of the project for which cleaning will be performed. Use '`*`' to trigger this job for each project on database.
`Dry run`	`true`	Flag that says if operation should be run in '*dry run*' mode. When checked, no changes will be performed on database.

AET W3C Comparator Factory

Property	Default value	Description
`Validator url`	`http://w3c.qa.cognifide.com/w3c-validator/`	Url of the W3c validator that is used to check W3C using W3C Comparator.

4.3. Application sanity check - 1.3

Log files check

Check if log files were created in `C:\content\karaf\data\log` directory. There should be at least four files:

- `karaf.log`,
- `runner.log`,
- `worker.log`
- `cleaner.log`.

More information about these files in Logs section.

Configuration sanity check

Log into *Web Console Components* (http://localhost:8181/system/console/components) and check if all components are *active*. Only component *CleanerScheduler* should remain *unsatisfied*.

Rest API sanity check

Open Rest API in browser (`${WINDOWS_VM_PRIVATE_IP}:8181/cxf/aet`). Result:

```
{
message: "Resource not found",
success: false
}
```

should be visible.

Then check if Rest API is accessible via `${WINDOWS_VM_PUBLIC_IP}`.

ActiveMQ sanity check

Open ActiveMq Console in browser (*${LINUX_VM_PRIVATE_IP}:8161/admin*) and verify if all necessary queues are created (*AET.collectorResults, AET.comparatorJobs, AET.comparatorResults, AET.reporterJobs, AET.reporterResults, AET.runner-in, AET.win7-ff16*).

Problems and troubleshooting

Problem: After configuring all components, some of them are in status `registred` or `unsatisfied`. Manual start does not work.

Solution: Check if all bundles are active. If not and manual start of bundles does not work - restart *Karaf* and *Windows* instance.

Problem: Log files under directory `C:\content\karaf\data\log` were not created.

Solution: Check in *Web Console Bundles* if *OPS4J Pax Logging* bundles are active. Try to restart them. If files are not present after restarting bundles, check if *org.ops4j.pax.logging* configuration contains *aet.runner* and *aet.worker* appenders defined. If not, shutdown *Karaf* and copy once again `org.ops4j.pax.logging.cfg` configuration file from `etc` directory in *aet.zip* to `C:\content\karaf\etc`. Start *Karaf* and check log files.

Problem: REST Api projects page (*${WINDOWS_VM_PUBLIC_IP}:8081/cxf/aet*) returns 404.

Solution: Check in *Web Console Bundles* if all *Apache CXF* bundles are active. Try to restart them. If page still responds with 404, check *CXF Servlet Transport* configuration in *Web Console Configuration* (http://localhost:8181/system/console/configMgr) . `Servlet context path` property should be set to `/cxf`. Restart *Karaf* instance.

5. Test setup and run - 1.3

Running AET test consists of few easy steps:

- defining testsuite,
- running testsuite,
- checking testsuite report.

All those steps are described in this section with details.

5.1. Suite setup - 1.3

Test suite

In general the test suite is an XML document that defines tests performed over collection of web pages. This chapter covers test suite API, with description of each element.

Example of test suite

TestSuite

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Each test suite consists of one suite -->
<suite name="test-suite" company="cognifide" project="project" environment="win7-ff16">
    <!-- The First test of Test Suite -->
    <!-- The flow is [collect] [compare] [urls] -->
    <test name="first-test" useProxy="true">
        <!-- Description of the collect phase -->
        <collect>
            <open/>
            <!-- sleep 1500 ms before next steps - used on every url defined in urls -->
            <sleep duration="1500"/>
            <screen maximize="true"/>
            <source/>
            <status-codes/>
            <js-errors/>
        </collect>
        <!-- Description of compare phase, says what collected data should be compared to the patter
        <compare xmlns="http://www.cognifide.com/aet/compare/">
            <screen comparator="layout"/>
            <source comparator="w3c"/>
            <status-codes filterRange="400,600"/>
```

```
            <js-errors>
    <js-errors-filter source="http://w.iplsc.com/external/jquery/jquery-1.8.3.js" line="2" />
   </js-errors>
        </compare>
        <!-- List of urls which will be taken into tests -->
        <urls>
            <url href="http://www.cognifide.com" description="Cognifide"/>
        </urls>
    </test>
    <reports>
        <html-report/>
        <xunit-report/>
    </reports>
</suite>
```

Root element of test suite definition is **suite** element.

suite

Important!

When defining a suite a user should think of three mandatory parameters properly: `name, company, project`. Those parameters (together with constant parameter `environment`) are used by the AET System to identify the suite.

Any change in one of those parameters values in the future will occur in treating the suite as a completely new one, which will in effect gather all the patterns from scratch.

Root element for xml definition, each test suite definition consists of exactly one **suite** tag.

Attribute name	Description	Mandatory
`name`	Name of the test suite. Should consist only of lowercase letters, digits and/or characters: '-', '_' .	✔
`company`	Name of the company. Should consist only of lowercase letters, digits and/or characters: '-'.	✔
`environment`	Name of an execution environment where particular test suite will be executed (represented by name of consumer queue defined for AET Collector Message Listener). In the current system version, it should be always exactly the following string: win7-ff16.	✔
`project`	Name of the project. Should consist only of lowercase letters, digits and/or characters: '-'.	✔
`domain`	General domain name consistent for all considered urls. Every url link is built as a concatenation of *domain* name and *href* attribute of it. If `domain` property is not set, then `href` value in `url` definition should contain full valid url. See more in Urls section.	✘

suite element contains:

1. one or more **test** elements,
2. one **reports** element.

5.1.1. Test - 1.3

test

This tag is definition of the single test in test suite. Test suite can contain many tests.

Attribute name	Description	Mandatory
`name`	Name of the test. Should consists only of letters, digits and/or characters: '-', '_'. This value is also presented on report (more details in Report navigation section).	✔
`useProxy`	Defines which (if any) *Proxy* should be used during collection phase. If not provided, empty or set with `"false"`, proxy won't be used. If set to `"true"`, default *Proxy Manager* will be used. Otherwise *Proxy Manager* with provided name will be used (see Proxy). Proxy is needed by Status Codes Collector and Header Modifier.	✘
`zIndex`	Specifies order of tests on *HTML Report*. A test with greater `zIndex` is always before test with lower value. Default value is `0`. This attribute accepts integers in range <-2147483648; 2147483647>.	✘

Each **test** element contains:

- **one collect and one compare element** - test execution phases,
- **one urls element** - list of urls to process.

Proxy

Proxy is provided by two separated implementations: *embedded* and *rest*.

embedded

Embedded proxy does not need standalone *Browsermob Server*, but does not support SSL. *Embedded* proxy is used as default when `useProxy` is setted to "true" (which is equivalent to setting `useProxy="embedded"`).

Example usage

TestSuite

```
<?xml version="1.0" encoding="UTF-8" ?>
<suite name="test-suite" company="cognifide" project="project" environment="win7-ff16">
 <test name="header-modify-test" useProxy="embedded">
  ...
 </test>
 ...
</suite>
```

rest

Rest proxy requires standalone *Browsermob Server*.

Example usage

TestSuite

```
<?xml version="1.0" encoding="UTF-8" ?>
<suite name="test-suite" company="cognifide" project="project" environment="win7-ff16">
 <test name="header-modify-test" useProxy="rest">
  ...
 </test>
 ...
</suite>
```

5.1.1.1. Collect - 1.3

collect

This tag contain list of collectors and modifiers which will be run. It specifies what pages' data should be collected and it allows for some data modification before collection step. All collect steps are processed in defined order.

Each collector provides some specific result of gathering current data (i.e. png, html files) and a common metadata file - `result.json`.

Following elements are available in **collect** element:

Collectors:

- cookie
- js-errors
- screen
- source
- status-codes

Modifiers:

- header
- modify-cookie
- hide
- login
- resolution
- sleep
- wait-for-page-loaded

Special module:

- open

Collectors - 1.3

Collector is module which main task is to collect data from tested pages.

Each collector presented in section below consist of two elements:

- module name (produced resource type),
- parameters.

Module name (produced resource type)

This name is unique identifier of system functionality. Each collector has its unique name, this name should be also unique for all modules in *collect* phase. This is always name of tag definition for collector.

AET System does not know what work will be performed by collector when it reads suite definition. The only thing that is known is **module name**. System will recognize which collector should be called by matching definition from *collect* phase with name registered in system. When no collector in system with defined name is found, system exception will occur and test will be not performed. This solution enables adding new features to the system without system downtime (just by installing new feature bundle).

Each collector produces resource of defined type. This type can be later recognized by comparators and data filters. Two collectors can't produce data with the same resource type. **Produced resource type is always equal to collector module name.**

Parameters

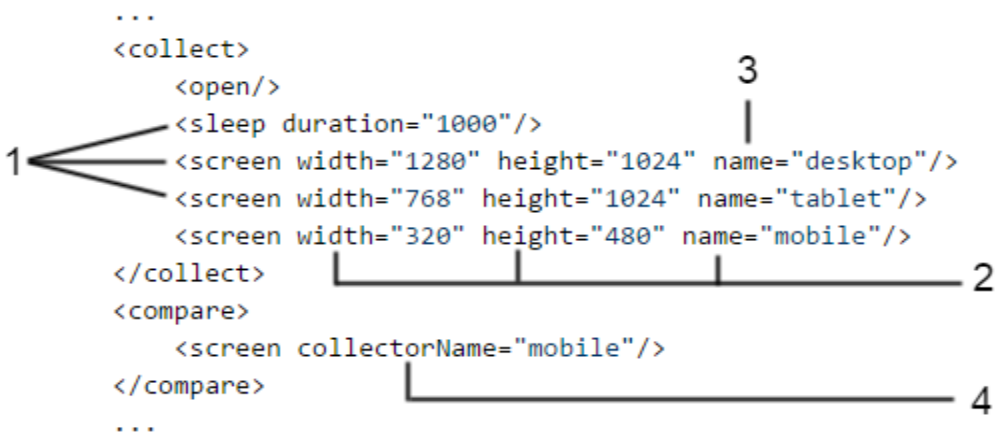
This is set of key-value pairs using which user can pass some configuration and information to collector. Parameters for collectors are usually not mandatory - passing this parameter is not obligatory, usually this is some collector functionality extension. However, there is one special property: **name**. Collector with set name can be treated in special way by comparators (some comparators may look only for collection results from specifically named collectors), example:

```
...
<collect>
 <open/>
 <sleep duration="1000"/>
 <screen width="1280" height="1024" name="desktop"/>
 <screen width="768" height="1024" name="tablet"/>
 <screen width="320" height="480" name="mobile"/>
</collect>
<compare>
 <screen collectorName="mobile"/>
</compare>
...
```

During collect phase, three screenshot with different resolutions will be taken and saved to database. However, only one of them (*mobile*) will be compared with pattern during comparison phase and presented on report (under "*Layout For Mobile*" section).

Definitions illustration

Following picture presents described earlier elements:



where:

1. Module name (produced resource type),
2. Parameters,
3. Special collector property: name,
4. Special comparator property: collectorName.

Accessibility Collector BETA - 1.3

Beta Version

This AET Plugin is currently in BETA version.

Accessibility Collector is responsible for collecting validation result containing violations of a defined coding standard found on a page. It uses HTML_CodeSniffer tool to find violations.

Module name: **accessibility**

Parameters

Parameter	Value	Description	Mandatory
standard	WCAG2A WCAG2AA (default) WCAG2AAA	Parameter specifies a standard against which the page is validated. More information on standards: WCAG2	✖

Example Usage

Example Usage

TestSuite

```
<?xml version="1.0" encoding="UTF-8" ?>
<suite name="test-suite" company="cognifide" project="project" environment="win7-ff16">
 <test name="source-test">
  <collect>
   ...
   <accessibility standard="WCAG2AAA" />
   ...
  </collect>
  <compare>
   ...
  </compare>
  <urls>
   ...
  </urls>
 </test>
 ...
 <reports>
  ...
 </reports>
</suite>
```

Cookie Collector - 1.3

Cookie collector is responsible for collecting cookies.

Module name: **cookie**

Parameters

No parameters.

Example Usage

TestSuite

```
<?xml version="1.0" encoding="UTF-8" ?>
<suite name="test-suite" company="cognifide" project="project" environment="win7-ff16">
 <test name="cookie-test">
  <collect>
   ...
   <cookie/>
   ...
  </collect>
  <compare>
   ...
  </compare>
  <urls>
   ...
  </urls>
 </test>
 ...
 <reports>
  ...
 </reports>
</suite>
```

JS Errors Collector - 1.3

JS Errors Collector is responsible for collecting javascript errors occuring on given page.

Module name: **js-errors**

Parameters

No parameters

Example Usage

TestSuite

```
<?xml version="1.0" encoding="UTF-8" ?>
<suite name="test-suite" company="cognifide" project="project" environment="win7-ff16">
 <test name="js-errors-test">
  <collect>
   ...
   <js-errors/>
   ...
  </collect>
  <compare>
   ...
  </compare>
  <urls>
   ...
  </urls>
 </test>
 ...
 <reports>
  ...
 </reports>
</suite>
```

Screen Collector - 1.3

Compare collected screenshots

Please remember that defining collector and not using it during comparison phase is configuration error. From now on suites that define screen collection and does not use it during comparison phase will be rejected during suite validation phase.

Screen Collector is responsible for collecting screenshot of the page under given URL.

Module name: **screen**

Parameters

Parameter	Value	Description	Mandatory
`maximize`	boolean (default: false)	Deprecated property This property is deprecated and may be removed in future release. Please use Resolution Modifier in order to perform browser resolution change. Using Screen collector without resolution change will not guarantee any specific screenshot resolution. Maximize browser window, sets window size to maximal system resolution.	✗
`width`	integer (1 to 65536)	Deprecated property This property is deprecated and may be removed in future release. Please use Resolution Modifier in order to perform browser resolution change. Using Screen collector without resolution change will not guarantee any specific screenshot resolution. Window width, cant be set over system resolution width size.	✗
`height`	integer (1 to 65536)	Deprecated property This property is deprecated and may be removed in future release. Please use Resolution Modifier in order to perform browser resolution change. Using Screen collector without resolution change will not guarantee any specific screenshot resolution. Window height, cant be set over system resolution height size.	✗

Note that you cannot maximize the window and specify the dimension at the same time. If no parameters provided, default browser size is set before taking screenshot.

Example Usage

TestSuite

```
<?xml version="1.0" encoding="UTF-8" ?>
<suite name="test-suite" company="cognifide" project="project" environment="win7-ff16">
 <test name="screen-test">
  <collect>
   ...
   <screen/>
   ...
  </collect>
  <compare>
   ...
  </compare>
  <urls>
   ...
  </urls>
 </test>
 ...
 <reports>
  ...
 </reports>
</suite>
```

Instead of

```
<screen width="1280" height="1024" name="desktop" />
```

please use:

```
<resolution width="1280" height="1024"/>
<sleep duration="1000" />
<screen name="desktop" />
```

Note

Before taking screenshot Hide modifier can be applied in order to hide from the screen some elements that are not necessary for comparison, i.e. Twitter feed.

Also Resolution Modifier and Wait For Page Loaded Modifier can be applied before Screen Collector usage to change expected collect result.

As in example presented above, `name` parameter can be very useful when using screen collector. More information about this parameter can be found in Collectors section.

Source Collector - 1.3

Source Collector is responsible for collecting source of the page under given URL. Unlike others collectors source collector don't use web driver, it connects directly to web server.

Module name: **source**

Note

Timeout is defined for loading time for one url (by default is set to 20 seconds). Is change is needed than please contact AET Team.

Parameters

No parameters

Example Usage

TestSuite

```
<?xml version="1.0" encoding="UTF-8" ?>
<suite name="test-suite" company="cognifide" project="project" environment="win7-ff16">
 <test name="source-test">
  <collect>
   ...
   <source />
   ...
  </collect>
  <compare>
   ...
  </compare>
  <urls>
   ...
  </urls>
 </test>
 ...
 <reports>
  ...
 </reports>
</suite>
```

Status Codes Collector - 1.3

Status Codes Collector is responsible for collecting status codes of links to resources on the page under given URL.

Module name: **status-codes**

Important information

In order to use this collector *proxy* must be used.

Parameters

No parameters needed.

Example Usage

TestSuite

```
<?xml version="1.0" encoding="UTF-8" ?>
<suite name="my-test-suite" company="cognifide" project="project" environment="win7-ff16">
 <test name="status-codes-test" useProxy="rest">
  <collect>
   ...
   <status-codes/>
   ...
  </collect>
  <compare>
   ...
  </compare>
  <urls>
   ...
  </urls>
 </test>
 ...
 <reports>
  ...
 </reports>
</suite>
```

Modifiers - 1.3

Modifier is module which performs particular modification on data before collection happens.

Each modifier consists of two elements:

- module name,
- parameters.

Module name

This name is unique identifier for each modifier (and each module in collect phase).

Parameters

This is set of key-value pairs using which user can pass some configuration and information to modifier. Parameters for modifiers can be divided into two groups:

- mandatory - parameters without which modification will be not possible,
- optional - passing this parameter is not obligatory, usually they trigger some functionality extension.

Click Modifier - 1.3

Click Modifier allows to perform click action on some element on page. When element is not found (e.g. by improper xpath value) warning will be logged but test will be passed to the next steps.

Module name: **click**

Important information

In order to use this modifier it must be declared after open module in test suite XML definition.

Remember that element that will be clicked **must be visible** in the moment of performing click action.

Parameters

Parameter	Default value	Description	Mandatory
`xpath`		xpath of element to click	✔
`timeout`		Timeout for element to appear, in miliseconds. Max value of this parameter is 15000 miliseconds (15 seconds).	✔

Example Usage

TestSuite

```
<?xml version="1.0" encoding="UTF-8" ?>
<suite name="test-suite" company="cognifide" project="project" environment="win7-ff16">
 <test name="click-test">
  <collect>
   <open/>
   ...
```

```
    <click xpath="//*[@id='header_0_container1_0_pRow']/div[1]/div/div/a/img" timeout="3000"/>
    <sleep duration="2000"/>
    ...
    <screen width="1280" height="800" name="desktop"/>
    ...
   </collect>
   <compare>
    ...
    <screen comparator="layout"/>
    ...
   </compare>
   <urls>
    ...
   </urls>
  </test>
  ...
  <reports>
   ...
  </reports>
</suite>
```

Cookie Modifier - 1.3

Cookie Modifier allows to modify cookies for given page, i.e. add or remove some cookies.

Module name: **modify-cookie**

Important information

In order to use this modifier it must be declared before open module in test suite XML definition. When declared after open module (but before Cookie Collector) it can be used as filter for Cookie Collector.

Parameters

Parameter	Value	Description	Mandatory
`action`	add remove	Specifies what action should be taken with given cookie	✅
`cookie-name`		Cookie name	✅
`cookie-value`		Cookie value	Yes, if `add` `action` is chosen

Example Usage

TestSuite

```
<?xml version="1.0" encoding="UTF-8" ?>
<suite name="test-suite" company="cognifide" project="project" environment="win7-ff16">
 <test name="cookie-modify-test">
  <collect>
   ...
   <modify-cookie action="add" cookie-name="sample-cookie" cookie-value="sample-cookie-value"/>
   <modify-cookie action="remove" cookie-name="another-cookie"/>
   ...
   <open/>
   ...
  </collect>
```

```
  <compare>
   ...
  </compare>
  <urls>
   ...
  </urls>
 </test>
 ...
 <reports>
  ...
 </reports>
</suite>
```

Header Modifier - 1.3

Header Modifier is responsible for injecting additional headers to page before it is opened to test.

Module name: **header**

Important information

In order to use this modifier it must be declared before open module in test suite XML definition and *proxy* must be used.

Parameters

Parameter	Value	Description	Mandatory
`key`	x	Key for header	✅
`value`	y	Value for header	✅

Example Usage

TestSuite

```
<?xml version="1.0" encoding="UTF-8" ?>
<suite name="test-suite" company="cognifide" project="project" environment="win7-ff16">
 <test name="header-modify-test" useProxy="rest">
  <collect>
   ...
   <header key="Authorization" value="Basic emVuT2FyZXVuOnozbkdAckQZbiE="/>
   ...
   <open/>
   ...
  </collect>
  <compare>
   ...
  </compare>
  <urls>
   ...
  </urls>
 </test>
 ...
 <reports>
  ...
 </reports>
</suite>
```

Hide Modifier - 1.3

Hide Modifier is responsible for hiding some unnecessary for test element on page. Affects Screen Collector results. Hiding is done by setting css visibility property to hidden. Works with webDriver only. You can hide many elements by defining many <hide> nodes. If xpath cover more than one element then all elements that match defined xpath will be hidden.

Module name: **hide**

Important information

In order to use this modifier it must be declared after open module in test suite XML definition.

Parameters

Parameter	Value	Description	Mandatory
`xpath`	xpath_to_element	Xpath to element(s) to hide	✔
`leaveBlankSpace`	boolean	Defines if element(s) should be invisible (effect as using display=none) or should be not displayed (effect as using visibility=hidden). When set to `true`, blank and transparent space is left in place of the hidden element, otherwise, element is completely removed from the view. When not defined, hide modifier behaves as if `leaveBlankSpace` property was set to `true`.	✘

Example Usage

TestSuite

```
<?xml version="1.0" encoding="UTF-8"?>
<suite name="test-suite" company="cognifide" environment="win7-ff16" project="project">
 <test name="hide-test" useProxy="true">
  <collect>
   ...
   <open/>
   ...
   <hide xpath="//*[@id='logo']"/>
   <hide xpath="//*[@id='primaryNavMenu']/li[2]/a/div" />
   ...
   <screen width="1200" height="760"/>
   ...
  </collect>
  <compare>
   ...
  </compare>
  <urls>
   ...
  </urls>
 </test>
 <reports>
   ...
 </reports>
 </suite>
```

Login Modifier - 1.3

Login Modifier allows to login into pages that have access secured with login form. If input element wont be available (wont be loaded yet) then Login Modifier will wait up to 10s for login input, then for password input and at the end for submit button to appear. If any element won't be ready then TimeoutException will be thrown.

Module name: **login**

Important information

In order to use this modifier it must be declared before open module in test suite XML definition.

Parameters

Parameter	Value	Mandatory	Default value
`login`	User's login	❌	admin
`password`	Password	❌	admin
`login-page`	Url to login page	❌	http://localhost:4502/libs/granite/core/content/login.html
`login-input-selector`	Xpath expression for login input	❌	//input[@name='j_username']
`password-input-selector`	Xpath expression for password input	❌	//input[@name='j_password']
`submit-button-selector`	Xpath expression for submit button	❌	//*[@type='submit']
`login-token-key`	Name for cookie we get after successfull login	❌	login-token

Example Usage

TestSuite

```
<?xml version="1.0" encoding="UTF-8" ?>
<suite name="test-suite" company="cognifide" project="project" environment="win7-ff16">
 <test name="login-test">
  <collect>
   <login login="user"
          password="password"
          login-page="http://192.168.180.19:5503/libs/cq/core/content/login.html"
                   login-input-selector="//input[@name='j_username']"
```

```
                    password-input-selector="//input[@name='j_password']"
                    submit-button-selector="//*[@type='submit']"/>
   <open/>
   ...
  </collect>
  <compare>
   ...
  </compare>
  <urls>
   ...
  </urls>
 </test>
 ...
 <reports>
  ...
 </reports>
</suite>
```

Resolution Modifier - 1.3

Resolution Modifier is responsible for changing browser screen size. Affects Screen Collector results.

⚠ Please note that final resoulution of screenshots may be different when scrollbar is dispayed.
⚠Default width of FireFox's Scrollbar is equal to 33px. (so when you want to grab viewport of size 1024, then set width parameter to 1057px)

Module name: **resolution**

Parameters

Parameter	Value	Description	Mandatory
`maximize`	true false (default)	Maximize browser window	✗
`width`	int (1 to 100000)	Window width	✗
`height`	int (1 to 100000)	Window height	✗

Important information

You cannot maximize the window and specify the dimension at the same time. If you specify height param you have to also specify width param and vice versa.

Example Usage

TestSuite

```
<?xml version="1.0" encoding="UTF-8" ?>
<suite name="test-suite" company="cognifide" project="project" environment="win7-ff16">
 <test name="resolution-modify-test">
  <collect>
   ...
   <resolution width="200" height="300"/>
   <screen/>
   ...
  </collect>
```

```
  <compare>
   ...
  </compare>
  <urls>
   ...
  </urls>
 </test>
 ...
 <reports>
  ...
 </reports>
</suite>
```

Sleep Modifier - 1.3

Sleep Modifier is responsible for temporarily ceasing execution, causes current thread to sleep. It is useful in situations when page resources have a long loading time - it suspends next collectors for some time.

Module name: **sleep**

Parameters

Parameter	Value	Description	Mandatory
`duration`	int (1 to 30000)	Sleep time, in milliseconds	✔

Important information

One sleep duration cannot be longer than 30000 milliseconds (30 seconds).

Two consecutive sleep modifiers are not allowed.

Total sleep duration (sum of all sleeps) in test collection phase cannot be longer than 120000 milliseconds (2 minutes).

Example Usage

TestSuite

```
<?xml version="1.0" encoding="UTF-8" ?>
<suite name="test-suite" company="cognifide" project="project" environment="win7-ff16">
 <test name="sleep-test">
  <collect>
   ...
   <open>
   ...
   <sleep duration="3000"/>
   ...
  </collect>
  <compare>
   ...
  </compare>
  <urls>
   ...
  </urls>
 </test>
 ...
 <reports>
  ...
 </reports>
</suite>
```

Wait For Page Loaded Modifier - 1.3

Wait For Page Loaded Modifier waits until page is loaded or fixed amount of time is up. The idea of waiting for page is counting amount of elements [by findElements(By.xpath("//*"))] on current page state in loop. If number of elements has increased since last checkout, continue loop (or break if timeout). Else if number of elements is still, assume the page is loaded and finish waiting.

Module name: **wait-for-page-loaded**

Parameters

No parameters.

Important information

Timeout for waiting is 10000 milliseconds.

Page is checked every 1000 miliseconds.

Example Usage

TestSuite

```
<?xml version="1.0" encoding="UTF-8" ?>
<suite name="test-suite" company="cognifide" project="project" environment="win7-ff16">
 <test name="wait-for-page-loaded-test">
  <collect>
   ...
   <open/>
   ...
   <wait-for-page-loaded/>
   ...
  </collect>
  <compare>
   ...
  </compare>
  <urls>
   ...
  </urls>
 </test>
 ...
 <reports>
  ...
 </reports>
</suite>
```

Open - 1.3

Open module is special operand for collect phase. It is responsible for opening web page for given url and preparing browser environment to perform chain of collections and modifications.

Second usage of this module is to allow user easily perform actions before page is being opened, such as modify headers , cookies etc.

Open module

Each collect phase **must** contain open module.

In some cases it is recommended to use Sleep Modifier or Wait For Page Loaded Modifier after open module.

Module name: **open**

Parameters

No parameters

Example Usage

TestSuite

```
<?xml version="1.0" encoding="UTF-8" ?>
<suite name="test-suite" company="cognifide" project="project" environment="win7-ff16">
 <test name="open-test">
  <collect>
   ...
   <!-- example action before page is opened -->
   <open/>
   <!-- collect page data -->
   ...
  </collect>
  <compare>
   ...
  </compare>
  <urls>
   ...
  </urls>
 </test>
 ...
 <reports>
  ...
 </reports>
</suite>
```

5.1.1.2. Compare - 1.3

compare

This tag contain list of comparators. Each comparator takes collected resource of defined type and runs it against comparator. It provides some specific result files illustrating found differences (i.e png, html files) and a common metadata file - result.json.

Each resource type has default comparator, user can use other comparators for each type by providing attribute ' comparator' with comparator name, e.g.:

```
<source comparator="my_source_comparator"/>
```

runs my_source_comparator against each source collected during collection phase. Each comparator can contain list of data modifiers which will be performed before each compare phase.

Data filters are used to modify gathered data before these data are passed to comparator. For example you may remove some node from html tree. Data filters are defined in test suite xml as subnodes of **comparator** node.

Each Data Filter has predefined type of data on which it operates.

Following elements are available in **compare** element:

- cookie
- js-errors
- screen
- **source** - two comparator types are available for **source** data:
  - source
  - w3c
- status-codes

Comparators - 1.3

Comparator is module which main task is to consume data and compare it with pattern or against defined set of rules.

Each comparator presented in section below consists of three elements:

- consumed resource type,
- module name (comparator),
- parameters.

Consumed resource type

This is name of resource type consumed by defined comparator. This is always name of tag definition for comparator.

This name says the system which **resource type** should be consumed by defined comparator. When no comparator in system can consume defined resource type, system exception will occur and test will be not performed. This solution enables adding new features to the system without system downtime (just by installing new feature bundle).

Each comparator can consume only one type of resource.

Module name (comparator)

This is special parameter, unique name for comparator type treated as interpretation of given resource type. System will recognize which implementation of comparator should be called by this name. This parameter is required for each comparator but system will assume default comparator for each resource type when no `comparator` property is defined.

Default comparators for consumed resource names

- cookie -> Cookie Comparator,
- js errors -> JS Errors Comparator,
- screen -> Layout Comparator,
- source -> Source Comparator,
- status-codes -> Status Codes Comparator.

Example of usage can be found in system for *source* comparison, where two comparators exists: W3C Comparator and Source Comparator. Example below shows sample usage:

```
...
<collect>
 <open/>
 <source/>
</collect>
<compare>
 <source comparator="source"/>
 <source comparator="w3c"/>
</compare>
...
```

When test defined as above is executed, only one collection of page source is performed. But result of this collection is used twice during comparison phase. First by Source Comparator and then by W3C Comparator.

Parameters

This is set of key-value pairs using which user can pass some configuration and information to comparator. Parameters for comparators can be divided into two groups:

- mandatory - parameters without which comparison will be not possible,
- optional - passing this parameter is not obligatory, usually this is some comparator functionality extension.

collectorName

There exists special comparator property **collectorName** which is connected with collector **name** property. Using collectorName property combined with collector name property, user can control which comparator instance compares results collected by particular collector. See examples below:

```
...
<collect>
 <open/>
 <sleep duration="1000"/>
 <screen width="1280" height="1024" name="desktop"/>
 <screen width="768" height="1024" name="tablet"/>
 <screen width="320" height="480" name="mobile"/>
</collect>
<compare>
 <screen collectorName="mobile"/>
 <screen collectorName="tablet"/>
</compare>
...
```

Configuration above will trigger three screens collections (desktop, tablet and mobile) and two comparisons (mobile and tablet). Screenshot taken for *desktop* will be not compared.

```
...
<collect>
 <open/>
 <sleep duration="1000"/>
 <screen width="1280" height="1024" name="desktop"/>
 <screen width="768" height="1024" name="tablet"/>
 <screen width="320" height="480" name="mobile"/>
</collect>
<compare>
 <screen/>
</compare>
...
```

Configuration above will trigger three screens collections (desktop, tablet and mobile) and three comparisons (desktop, table, mobile).

```
...
<collect>
 <open/>
 <sleep duration="1000"/>
 <screen width="1280" height="1024" name="desktop"/>
 <screen width="768" height="1024" name="tablet"/>
 <screen width="320" height="480" name="mobile"/>
</collect>
```

```
<compare>
 <screen/>
 <screen collectorName="tablet"/>
</compare>
...
```

Configuration above will trigger three screens collections (desktop, tablet and mobile) and four comparisons (desktop, table, mobile and one additional for tablet).

Definitions illustration

Following picture presents described earlier definitions:

where:

1. Consumed resource type,
2. Special property: collectorName,
3. Special property: comparator,
4. Module name (comparator).

Accessibility Comparator BETA - 1.3

Beta Version

This AET Plugin is currently in BETA version.

Accessibility Comparator is responsible for processing of collected accessibility validation result. It uses html CodeSniffer library.

Module name: **accessibility**

Resource name: accessibility

Parameters

Parameter	Value	Description	Mandatory
`report-level`	ERROR (default)	Only violations of type ERROR are displayed on report.	
	WARN	Violations of type WARN and ERROR are displayed on report.	✖
	NOTICE	All violations are displayed on report.	
`ignore-notice`	boolean (default: `true`)	If `ignore-notice=true` test status does not depend on the notices amount If `ignore-notice=false` notices are treated as warnings in calculating test status. Enforces report-level = NOTICE.	✖
`showExcluded`	boolean (default: `true`)	Flag that says if excluded issues (see Accessibility Data Filter) should be displayed in report. By default set to `true`.	✖

Example Usage

TestSuite

```
<?xml version="1.0" encoding="UTF-8" ?>
<suite name="test-suite" company="cognifide" project="project" environment="win7-ff16">
 <test name="cookie-test">
  <collect>
   ...
   <accessibility/>
   ...
  </collect>
  <compare>
   ...
   <accessibility report-level="WARN" />
   ...
  </compare>
  <urls>
   ...
  </urls>
 </test>
 ...
 <reports>
  ...
 </reports>
</suite>
```

Cookie Comparator - 1.3

Cookie Comparator is responsible for processing of collected cookies. This can be simply listing of collected cookies, verifying if cookie exists or comparing collected cookie with pattern.

Cookie feature allows to collect patterns and can be rebased from report only in compare action mode.

Module name: **cookie**

Resource name: cookie

Parameters

Parameter	Value	Description	Mandatory
`action`	list	Displays the list of cookies	❌
	test	Tests if cookie with the given name and value exists	If `action` parameter is not provided, default `list` action is performed
	compare	Compares the current data with the pattern (compares only cookie names, values are ignored)	
`cookie-name`		Name of the cookie to test, applicable only for test action	Yes, if `action` set to `test`
`cookie-value`		Value of the cookie to test, applicable only for test action	❌
`showMatched`	boolean (default: `true`)	Works only in compare mode. Flag that says if matched cookies should be displayed in report. By default set to `true`.	❌

Example Usage

TestSuite

```
<?xml version="1.0" encoding="UTF-8" ?>
<suite name="test-suite" company="cognifide" project="project" environment="win7-ff16">
 <test name="cookie-test">
  <collect>
   ...
   <cookie/>
   ...
  </collect>
  <compare>
   ...
   <cookie/>
   ...
  </compare>
  <urls>
   ...
  </urls>
 </test>
 ...
 <reports>
  ...
 </reports>
</suite>
```

JS Errors Comparator - 1.3

JS Errors Comparator is responsible for processing of collected javascript errors resource. In this case it is simply displaying list of javascript errors.

JS Errors feature do not allow to collect patterns, so it does not compare results with any patterns - rebase action is also not avaliable.

Module name: **js-errors**

Resource name: js-errors

Parameters

No parameters

Example Usage

TestSuite

```
<?xml version="1.0" encoding="UTF-8" ?>
<suite name="test-suite" company="cognifide" project="project" environment="win7-ff16">
 <test name="js-errors-test">
  <collect>
   ...
   <js-errors/>
   ...
  </collect>
  <compare>
   ...
   <js-errors/>
   ...
  </compare>
  <urls>
   ...
  </urls>
 </test>
 ...
 <reports>
  ...
 </reports>
</suite>
```

Important information

JS Errors Filter can be applied to collected javascript errors result before comparison to modify data that is to be processed.

Layout Comparator - 1.3

Layout Comparator is responsible for comparing collected screenshot of page with pattern. This is default comparator for `screen` resource.

Can be rebased from report.

Module name: **layout**

Resource name: screen

Parameters

No parameters

Example Usage

TestSuite

```
<?xml version="1.0" encoding="UTF-8" ?>
<suite name="test-suite" company="cognifide" project="project" environment="win7-ff16">
 <test name="layout-compare-test">
  <collect>
   ...
   <screen/>
   ...
  </collect>
  <compare>
   ...
   <screen comparator="layout"/>
   ...
  </compare>
  <urls>
   ...
  </urls>
 </test>
 ...
 <reports>
  ...
 </reports>
</suite>
```

Fast pre-comparison

Since AET 1.3 fast comparison of screenshots will be implemented. Taken screenshot MD5 will be matched agains current pattern. If hashes will be the same, screenshot will be treated as one without differences and no further comparison will be performed.

Source Comparator - 1.3

Source Comparator is responsible for comparing collected page source with pattern.

Can be rebased from report.

Module name: **source**

Resource name: source

Parameters

Parameter	Value	Description	Mandatory
`compareType`	content	Compare only text, except markup.	✗ If `compareType` is not provided default `all` value is taken.
	markup	Compare only html markup.	
	allFormatted	Compare full source with formatting and whitespace ignore.	
	all	Compare all source.	

Example Usage

TestSuite

```
<?xml version="1.0" encoding="UTF-8" ?>
<suite name="test-suite" company="cognifide" project="project" environment="win7-ff16" >
 <test name="source-compare-test">
  <collect>
   ...
   <source/>
   ...
  </collect>
  <compare>
   ...
   <source comparator="source" compareType="markup" />
   ...
  </compare>
  <urls>
   ...
  </urls>
 </test>
 ...
 <reports>
  ...
 </reports>
</suite>
```

Important information

Extract Element Data Filter, Remove Lines Data Filter and Remove Nodes Data Filter can be applied to collected source before comparison to modify source data that is to be compared.

Status Codes Comparator - 1.3

Status Codes Comparator is responsible for processing collected status codes. In this case it is simply displaying the list of collected status codes from given page.

Status Codes feature do not allow to collect patterns, so it does not compare results with any patterns - rebase action is also not available.

Module name: **status-codes**

Resource name: status-codes

Parameters

Parameter	Value	Example	Description	Mandatory
`filterRange`	x,y	400,500	Defines range of status codes that should be processed	
`filterCodes`	x,y,z	400,401,404	List of status codes that should be processed	
`showExcluded`	boolean (default: `true`)	true	Flag that says if excluded codes (see Status Codes Data Filters) should be displayed in report. By default set to `true`.	

One of these parameters is required.

Example Usage

TestSuite

```
<?xml version="1.0" encoding="UTF-8" ?>
<suite name="test-suite" company="cognifide" project="project" environment="win7-ff16">
 <test name="status-codes-test" useProxy="rest">
  <collect>
   ...
   <open/>
   ...
   <status-codes/>
   ...
  </collect>
  <compare>
   ...
   <status-codes filterRange="400,404" />
   ...
  </compare>
  <urls>
   ...
  </urls>
 </test>
 ...
 <reports>
  ...
 </reports>
</suite>
```

W3C Comparator - 1.3

W3C Comparator is responsible for validating collected page source against **w3c standards (https://validator.w3.org/) and does not support html5 syntax**. When html5 syntax is required pleas use W3C HTML5 Comparator.

W3C feature do not allow to collect patterns, so it does not compare results with any patterns - rebase action is also not available.

Module name: **w3c**

Resource name: source

Parameters

Parameter	Value	Description	Mandatory
`validator`	URL address	Validator URL (url of validator instance). This property overrides value defined in AET W3C Comparator Factory (see more in Application configuration)	✗
`ignore-warnings`	boolean (default: true)	If `ignore-warnings=true` test status does not depend on the warnings amount	✗

Mandatory parameter

Please remember, that using parameter `comparator="w3c"` is mandatory while defining this comparator. More information about this parameter can be found in Comparators section.

Example Usage

TestSuite

```
<?xml version="1.0" encoding="UTF-8" ?>
<suite name="test-suite" company="cognifide" project="project" environment="win7-ff16" >
 <test name="w3c-test">
  <collect>
   ...
   <open/>
   ...
   <source/>
   ...
  </collect>
  <compare>
   ...
   <source comparator="w3c" validator="http://w3c.qa.cognifide.com/w3c-validator/"/>
   ...
  </compare>
  <urls>
   ...
  </urls>
 </test>
 ...
 <reports>
  ...
 </reports>
</suite>
```

`comparator="w3c"`

W3C HTML5 Comparator - 1.3

W3C HTML5 Comparator is responsible for validating collected page source against w3c standards using valitaor.nu (https://validator.nu/). HTML5 is supported by this library.

W3C HTML5 feature do not allow to collect patterns, so it does not compare results with any patterns - rebase action is also not available.

Module name: **w3c-html5**

Resource name: source

Parameters

Parameter	Value	Description	Mandatory
`errors-only`	boolean (default: true)	If `errors-only="true"` test status does not depend on the warnings amount, otherwise warnings counts as w3c errors when computing testcase status.	✖

Mandatory parameter

Please remember, that using parameter `comparator="w3c-html5"` is mandatory while defining this comparator. More information about this parameter can be found in Comparators section.

Example Usage

TestSuite

```
<?xml version="1.0" encoding="UTF-8" ?>
<suite name="test-suite" company="cognifide" project="project" environment="win7-ff16" >
 <test name="w3c-test">
  <collect>
   ...
   <open/>
   ...
   <source/>
   ...
  </collect>
  <compare>
   ...
   <source comparator="w3c-html5" />
   ...
  </compare>
  <urls>
   ...
  </urls>
 </test>
 ...
 <reports>
  ...
 </reports>
</suite>
```

Data Filters - 1.3

Data filters are modules which narrow area on which comparison will be performed.

They are nested in Comparators and apply only to instance of comparator in which they are defined.

Each data filter consists of two elements:

- module name,
- parameters.

Module name

This name is unique identifier for each data filter (and each module in compare phase).

Parameters

This is set of key-value pairs using which user can pass some configuration and information to data filter. Parameters can be divided into two groups:

- mandatory - parameters without which filtering will be not possible,
- optional - passing this parameter is not obligatory, usually they trigger some functionality extension.

Accessibility Data Filter - 1.3

Accessibility Data Filter filters Accessibility issues - it removes matched accessibility issues from reports.
This filter can be only applied to **accessibility** comparator tag in test case.
When more than one parameter is provided then only fully matched issues are filtered.

Module name: **accessibility-filter**

Resource name: accessibility

Parameters

Parameter	Value	Description	Mandatory
`error`	string error	Exact error message	
`principle`	string principle	Exact accessibility issue principle	
			At least one of parameter is required
`line`	integer line number	Line number in file in which issue occurred	
`column`	integer column number	Column number in file in which issue occurred	

Example Usage

In this sample exact match of accessibility issue breaking principle "WCAG2A.Principle4.Guideline4_1.4_1_2.H91.Button.Name", at line 21, column 5 with message "This button element does not have a name available to an accessibility API. Valid names are: title attribute, element content." will be totally ignored.

Test case

```
<?xml version="1.0" encoding="UTF-8" ?>
<suite name="test-suite" company="cognifide" project="project" environment="win7-ff16">
 <test name="js-errors-filter-test">
  <collect>
   ...
   <open/>
   ...
   <accessibility/>
   ...
  </collect>
  <compare>
   ...
   <accessibility>
    <accessibility-filter error="This button element does not have a name available to an accessibil
                          principle="WCAG2A.Principle4.Guideline4_1.4_1_2.H91.Button.Name"
                          line="21"
```

```
                        column="5" />
   </accessibility>
   ...
  </compare>
  <urls>
   ...
  </urls>
 </test>
 ...
 <reports>
  ...
 </reports>
</suite>
```

There can be more than one **accessibility-filter** tag in **accessibility** comparator eg:

```
<accessibility>
    <accessibility-filter principle="WCAG2A.Principle1.Guideline1_3.1_3_1.F68" />
    <accessibility-filter error="This select element does not have a name available to an accessibil
    <accessibility-filter line="270" />
    <accessibility-filter line="314" />
    <accessibility-filter column="5" />
</accessibility>
```

Extract Element Data Filter - 1.3

Extract Element Data Filter allows to extract element from html source (collected by Screen Collector) by providing id attribute or class attribute. Found element's source is processed by comparator.

Module name: **extract-element**

Resource name: source

Parameters

Parameter	Value	Description	Mandatory
`elementId`	HTML id	Id for element to extract	
`class`	HTML class	Class name for element to extract	

One of these parameters is required. Only one parameter (either id attribute or class attribute) can be provided.

Example Usage

TestSuite

```
<?xml version="1.0" encoding="UTF-8" ?>
<suite name="test-suite" company="cognifide" project="project" environment="win7-ff16">
 <test name="extract-element-test">
  <collect>
   ...
   <open/>
   ...
   <source/>
   ...
  </collect>
```

```
  <compare>
   ...
   <source comparator="source">
    <extract-element elementId="login_form"/>
   <!-- OR -->
    <extract-element class="class_form"/>
   </source>
   ...
  </compare>
  <urls>
   ...
  </urls>
 </test>
 ...
 <reports>
  ...
 </reports>
</suite>
```

JS Errors Data Filter - 1.3

Js Errors Data Filter filters JS Errors Collector result - it removes matched javascript errors from reports.
This filter can be only applied to **js-errors** comparator tag in test case.
When more than one parameter is provided then only fully matched errors are filtered.
If some XML-specific charactes (ex. &) are in parameter's value, then they must be escaped.

Module name: **js-errors-filter**

Resource name: js-errors

Parameters

Parameter	Value	Description
`error`	string error	Exact error message

`source`	string file name	Url of source file.
		Filter matches url path from its end.
		eg:
		when error will occur in file:
		
		`http://w.iplsc.com/external/jquery/jquery-1.8.3.js`
		
		any of below filters will match out file:
		
		`<js-errors-filter source="w.iplsc.com/external/jquery/jquery-1.8.3.js" />`
		
		`<js-errors-filter source="/external/jquery/jquery-1.8.3.js" />`
		
		`<js-errors-filter source="jquery/jquery-1.8.3.js" />`
		
		`<js-errors-filter source="jquery-1.8.3.js" />`
		
		Be aware that eg filter:
		
		`<js-errors-filter source="jquery-1.8.3.js" />`
		
		will also ignore all js errors from eg:

		`http://some.other.site.com/foo/jquery-1.8.3.js` Best practice is just to skip protocol and domain part of url
`line`	integer line number	Line number in file in which error occurred

Example Usage

In this sample exact match of js error from file "http://w.iplsc.com/external/jquery/jquery-1.8.3.js", line 2 with message " Error: Syntax error, unrecognized expression: .iwa_block=pasek-ding" will be totally ignored (not included in report)

Test case

```
<?xml version="1.0" encoding="UTF-8" ?>
<suite name="test-suite" company="cognifide" project="project" environment="win7-ff16">
 <test name="js-errors-filter-test">
  <collect>
   ...
   <open/>
   ...
   <js-errors/>
   ...
  </collect>
  <compare>
   ...
   <js-errors>
    <js-errors-filter error="Error: Syntax error, unrecognized expression: .iwa_block=pasek-ding"
                      line="2"
                      source="http://w.iplsc.com/external/jquery/jquery-1.8.3.js" />
   </js-errors>
   ...
  </compare>
  <urls>
   ...
  </urls>
 </test>
 ...
 <reports>
  ...
 </reports>
</suite>
```

There can be more than one **js-errors-filter** tag in **js-errors** comparator eg:

```
<js-errors>
 <js-errors-filter error="Error: Syntax error, unrecognized expression: .iwa_block=pasek-ding" />
 <js-errors-filter source="http://w.iplsc.com/external/jquery/jquery-1.8.3.js"
                   line="2" />
</js-errors>
```

Remove Lines Data Filter - 1.3

Remove Lines Data Filter allows to remove lines from compared source (data or pattern). This may be helpful when we need to compare page sources with dynamic content. We can then remove these dynamic content markup.

Line number in reports represents lines state after modification, so have in mind that marked lines have different lines number in real source.

Module name: **remove-lines**

Resource name: source

Parameters

Parameter	Value	Description	Mandatory
`dataRanges`	ranges of lines to remove from data	Ranges should be provided in form **a,b** , this will be interpreted as closed interval of integers [a,b]. Particular ranges should be separated by semicolons: **a,b;c,d;e,f** a>0, b>0	At least one of parameter is required .
`patternRanges`	ranges of lines to remove from pattern		

Examples:

Suppose we want to remove line 10: **`10,10`**

Suppose we want to remove lines from 10 to 15: **`10,15`**

Suppose we want to remove lines from 10 to 15, line 27 and lines from 30 to 38: **`10,15;27,27;30,38`**

Example Usage

TestSuite

```
<?xml version="1.0" encoding="UTF-8" ?>
<suite name="my-test-suite" company="cognifide" project="project" environment="win7-ff16">
 <test name="remove-lines-test">
  <collect>
   ...
   <source/>
   ...
  </collect>
  <compare>
   ...
   <source comparator="source">
    <remove-lines dataRanges="10,15;27,27" patternRanges="10,14;27,28"/>
   </source>
   ...
  </compare>
  <urls>
   ...
  </urls>
 </test>
 ...
 <reports>
  ...
 </reports>
</suite>
```

Remove Nodes Data Filter - 1.3

Remove Nodes Data Filter allows to delete some node(s) from html tree. Node(s) are defined by xpath selector.

Important information

Html source has to be valid xml document

Name: **remove-nodes**

Resource name: source

Parameters

Parameter	Value	Description	Mandatory
`xpath`	xpath_to_node	Xpath selector for nodes to remove	✔

Example Usage

TestSuite

```
<?xml version="1.0" encoding="UTF-8"?>
<suite name="test-suite" company="Cognifide" project="project" environment="win7-ff16">
 <test name="remove-nodes-test">
  <collect>
   ...
   <open/>
   ...
   <source/>
   ...
  </collect>
  <compare>
   ...
   <source comparator="source">
    <remove-nodes xpath="//*[@id='blueBarNAXAnchor']/div/div/div/a/i"/>
   </source>
   ...
  </compare>
  <urls>
   ...
  </urls>
 </test>
 <reports>
  ...
 </reports>
</suite>
```

Status Codes Data Filters - 1.3

Exclude Filter

Exclude Filter removes from reports Status Codes results that match specified parameters.

Name: **exclude**

Resource name: js-errors

Parameters

Parameter	Value	Description	Mandatory
`url`	String url	Exact url to be removed from results.	At least one of parameter is required.
`pattern`	String regex pattern	Regex pattern that urls should match to be removed from results.	

If both parameters are provided then result is removed when it matches at least one of the parameters.

Example Usage

In this sample match results with url http://www.cognifide.com/_cog_opt_js_f359581ea4bd3379b4c25591838a5dd8.js or url that matches pattern "^.*js$" will be ignored (not included in report).

Test case

```
<?xml version="1.0" encoding="UTF-8"?>
<suite name="test-suite" company="Cognifide" project="project" environment="win7-ff16">
    <test name="exclude-test" useProxy="rest">
        <collect>
            ...
            <open/>
            ...
            <status-codes/>
            ...
        </collect>
        <compare>
            ...
            <status-codes filterRange="200,999">
                <exclude url="http://www.cognifide.com/_cog_opt_js_f359581ea4bd3379b4c25591838a5dd8.
            </status-codes>
            ...
        </compare>
        <urls>
            ...
        </urls>
    </test>
    <reports>
        ...
    </reports>
</suite>
```

There can be more than one **exclude** tags in **status-codes** comparator. They are processed in turns. Example below is equivalent to defined above:

```
<status-codes>
 <exclude url="http://www.cognifide.com/_cog_opt_js_f359581ea4bd3379b4c25591838a5dd8.js"/>
 <exclude pattern="^.*js$"/>
</status-codes>
```

In this case both results with url http://www.cognifide.com/_cog_opt_js_f359581ea4bd3379b4c25591838a5dd8.js and urls that match pattern ".*js$" (ending with js) will not be displayed on reports.

Exclude and **include** modifiers can be both applied to **status-codes comparator**. They are processed in turns. Example:

```
<status-codes>
 <include pattern="^.*js$"/>
 <exclude url="http://www.cognifide.com/_cog_opt_js_f359581ea4bd3379b4c25591838a5dd8.js"/>
</status-codes>
```

In this case only first all urls that do not match "^.*js$" pattern are removed. Then url http://www.cognifide.com/_cog_opt_js_f359581ea4bd3379b4c25591838a5dd8.js is removed. At reports will be included urls ending with js except http://www.cognifide.com/_cog_opt_js_f359581ea4bd3379b4c25591838a5dd8.js.

Include Filter

Include Filter removes from reports Status Codes results that **do not** match specified parameters.

Name: **include**

Resource name: js-errors

Parameters

Parameter	Value	Description	Mandatory
`url`	String url	Exact url to be included in reports. Results that do not match will be removed.	At least one of parameter is required.
`pattern`	String regex pattern	Regex pattern that urls should match to be included in reports. Results that do not match will be removed.	

If both parameters are provided then result is included on report when it matches both of the parameters.

Example Usage

In example below **only** result with url http://www.cognifide.com/_cog_opt_js_f359581ea4bd3379b4c25591838a5dd8.js will be included in report.

Test case

```
<?xml version="1.0" encoding="UTF-8"?>
<suite name="test-suite" company="Cognifide" project="project" environment="win7-ff16">
    <test name="include-test" useProxy="rest">
        <collect>
            ...
            <open/>
            ...
            <status-codes/>
            ...
        </collect>
        <compare>
            ...
            <status-codes filterRange="200,999">
                <include url="http://www.cognifide.com/_cog_opt_js_f359581ea4bd3379b4c25591838a5dd8.
            </status-codes>
            ...
        </compare>
        <urls>
            ...
        </urls>
    </test>
    <reports>
        ...
```

```
    </reports>
</suite>
```

There can be more than one **include** tags in **status-codes** comparator. They are processed in turns. Example:

```
<status-codes>
 <include pattern="^.*js$"/>
 <include url="http://www.cognifide.com/_cog_opt_js_f359581ea4bd3379b4c25591838a5dd8.js"/>
</status-codes>
```

In this case only http://www.cognifide.com/_cog_opt_js_f359581ea4bd3379b4c25591838a5dd8.js url will be included on reports: first all results that do not match "^.*js$" pattern (ending with js) are removed. Then within that result all urls different that "http://www.cognifide.com/_cog_opt_js_f359581ea4bd3379b4c25591838a5dd8.js" are removed.

In example above, first <include> can be omitted and result will be the same.

Include and **exclude** modifiers can be both applied to **status-codes comparator**. They are processed in turns. Example:

```
<status-codes>
 <include pattern="^.*js$"/>
 <exclude url="http://www.cognifide.com/_cog_opt_js_f359581ea4bd3379b4c25591838a5dd8.js"/>
</status-codes>
```

In this case only first all urls that do not match "^.*js$" pattern are removed. Then url http://www.cognifide.com/_cog_opt_js_f359581ea4bd3379b4c25591838a5dd8.js is removed. At reports will be included urls ending with js except http://www.cognifide.com/_cog_opt_js_f359581ea4bd3379b4c25591838a5dd8.js .

W3c Filter - 1.3

W3c Filter allows to exclude some wc3 issues from result. Excluded issues will appear at the bottom of issues table and won't be taken into account when calculating status.

Name: **w3c-filter**

Resource name: source

Comparators: **w3c** and **w3c-html5**

Parameters

Parameter	Value	Description	Mandatory
message	string	Prefix or all message text of issue to be filter out	At least one of params should be used and all of used params should be not empty.
line	integer	Line in source file where issue appear	
column	integer	Column in source file where issue appear	

If there are some If some XML-specific charactes (ex. &) are in parameter's value, then they have to be escaped. See example below.

Example Usage for w3c comparator

TestSuite

```
<?xml version="1.0" encoding="UTF-8"?>
<suite name="test-suite" company="Cognifide" project="project" environment="win7-ff16">
 <test name="remove-nodes-test">
  <collect>
   ...
   <open/>
   ...
   <source/>
   ...
  </collect>
  <compare>
   ...
   <source comparator="w3c">
    <w3c-filter line="1" column="119"/>
    <w3c-filter message="Using Direct Input mode:"/>
   </source>
   ...
  </compare>
  <urls>
   ...
  </urls>
 </test>
 <reports>
  ...
 </reports>
</suite>
```

Example Usage for w3c-html5 comparator

TestSuite

```
<?xml version="1.0" encoding="UTF-8"?>
<suite name="test-suite" company="Cognifide" project="project" environment="win7-ff16">
 <test name="remove-nodes-test">
  <collect>
   ...
   <open/>
   ...
   <source/>
   ...
  </collect>
  <compare>
   ...
   <source comparator="w3c-html5" errors-only="false">
    <w3c-filter message = "The first occurrence of" />
    <w3c-filter message = "&#8220;&amp;&#8221; did not start a character reference"/>
    <w3c-filter line="1" column="119"/>
   </source>
   ...
  </compare>
  <urls>
   ...
  </urls>
 </test>
 <reports>
  ...
 </reports>
</suite>
```

5.1.1.3. Urls - 1.3

urls

This tag lists all urls which will be processed within current test. Contains one or more **url** elements.

url

Attribute name	Description	Mandatory
`href`	Page address (also see note under name attribute)	✅
`name`	Identifier for url. It is used to identify data for url. If provided should be unique for each test in test suite. If not provided is set to encoded `href` value. Should consists only of letters, digits and/or characters: '-', '_'. ⚠️ Note that if `href=""` with provided url `name` attribute and suite `domain` attribute is also valid	❌
`description`	Additional description for url that will be shown in html report	❌

5.1.2. Reports - 1.3

To use html-report (with enabled actions e.g. rebase) outside Cognifide network parameter `mode` must be setted to `online`. Then *redirect.html*, which redirects to proper html-report hosted by *AET RestEndpoint*, will be created.

reports

This tag is definition of reporter modules - user specifies here what reports have to be generated.

At this moment two types of reports are available

- **html-report**
- **xunit-report**

If element is empty, default report (**html-report**) will be generated.

html-report parameters

Parameter	Description	Mandatory
mode	When setted to `online` then additional *redirect.html* is created. *AET-maven-plugin* downloads it instead of *report-full.html*.	❌

Example Usage

TestSuite

```
<?xml version="1.0" encoding="UTF-8" ?>
<suite name="test-suite" company="cognifide" project="project" environment="win7-ff16">
 <test name="cookie-test">
  <collect>
   ...
  </collect>
  <compare>
```

```
    ...
  </compare>
  <urls>
    ...
  </urls>
 </test>
 ...
 <reports>
  <html-report/>
  <xunit-report/>
 </reports>
</suite>
```

For more information about reports see: Chapter 6. Report.

5.2.a Running suite using Maven - 1.3

Currently, running an AET suite requires using *aet-maven-plugin* which is an AET client application. To do so, two configuration files are mandatory:

- xml file - with defined suite (described with details in Suite setup section),
- pom.xml - which is a configuration file for *aet-maven-plugin*.

pom.xml

This file (`pom.xml`) is a *Maven* tool configuration file that contains information about the project and configuration details used by *Maven* to build the project.

Running AET suite requires creating and configuring such a file. The File presented below might be used as a template for setup AET suite runs:

pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.
    <modelVersion>4.0.0</modelVersion>

    <groupId>com.cognifide.aet</groupId>
    <artifactId>{PROJECT-NAME}</artifactId>
    <version>1.0.0</version>
    <packaging>pom</packaging>

    <name>AET :: Tests</name>
    <url>http://www.cognifide.com</url>

    <properties>
        <aet.version>{PLUGIN-VERSION}</aet.version>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    </properties>

    <pluginRepositories>
        <pluginRepository>
            <id>cognifide-maven-private-repository</id>
            <url>https://nexus.cognifide.com/content/groups/private</url>
        </pluginRepository>
        <pluginRepository>
            <id>cognifide-maven-public-repository</id>
```

```
            <url>https://nexus.cognifide.com/content/groups/public</url>
        </pluginRepository>
    </pluginRepositories>

    <build>
        <plugins>
            <plugin>
                <groupId>com.cognifide.aet</groupId>
                <artifactId>aet-maven-plugin</artifactId>
                <version>${aet.version}</version>
            </plugin>
        </plugins>
    </build>
</project>
```

User should configure two variables before proceeding to the next steps:

- `{PROJECT-NAME}` which is this build identifier for *Maven* tool. It should consist only of lowercase letters and '-',
  - example: `aet-sanity-test`
- `{PLUGIN-VERSION}` which should be set to currently used *aet-mavet-plugin* version,
  - example: `1.0.0`

  Having version as maven property (`${aet.version}`) enables defining this parameter from command line later, e.g. `-Daet.version=1.1.0` .

AET Maven plugin options

Running AET suite with *AET Maven plugin* cam be done simply by invoking a set of maven command(s):

```
mvn aet:run -DtestSuite=FULL_PATH_TO_TEST_SUITE
```

`testSuite` is the only mandatory parameter and is the path to the xmf suite configuration file. All parameters are described below:

Parameter	Description	Default Value	Mandatory
testSuite	Full path to suite definition file (at least file name with extension, e.g. *testSuite.xml*).	-	✅
brokerURL	URL to ActiveMQ broker.	tcp://localhost:61616	❌
userName	ActiveMQ user name.	karaf	❌
password	ActiveMQ password.	karaf	❌
outTopic	Active MQ outgoing queue name.	AET.runner-out	❌
inQueue	Active MQ incomming queue name.	AET.runner-in	❌

domain	Overrides *domain* parameter value from test suite definition.	-	❌
timeout	Milliseconds to detect timeout since the last message from AET. This is useful to abort test run if no activity for long time.	300000 (5 minutes)	❌

Tips and recommendations

Generally it is good idea, to create separate **SCM repository** (e.g. *GIT* or *SVN*) for AET suites. This will enable running AET suites using Jenkins with very easy steps.

5.2.1. Running using command line - 1.3

Overview

AET maven plugin is an interface for executing AET test suites (client application). There are two ways of using AET Maven plugin

- through command line
- with use of Jenkins job

This chapter covers the first option - running tests through command line.

Requirements

To execute test suites via maven command line following requirements must be fulfilled:

Requirements

1. Maven installed (recommended version - 3.0.4).
2. Proper version of AET Maven plugin installed.
3. Well-formed and valid xml test suite file available (explained in details in Suite setup section).
4. pom.xml with defined `aet-maven-plugin` configuration (explained in details in Running suite section).

Usage

Using command line invoke maven command in directory where `pom.xml` and suite file are defined:

```
mvn aet:run -DtestSuite=FULL_PATH_TO_TEST_SUITE
```

During test suite processing there will be progress information displayed on the console. It reflects how many artifacts were currently collected, compared, reported. When processing is finished information about processing status - `BUILD SUCCESS` or `BUILD FAILURE` - is displayed on the console.

More info about available options in Running suite section.

How to open report

The result of successful command line execution of AET test suite is ***report-full.html*** (or ***redirect.html*** if using html-report parameter `mode` setted to `online`) and/or ***xunit-report.xml***. They are both generated in maven run `target` folder.

5.2.2. Running using Jenkins - 1.3

Overview

AET maven plugin is an interface for executing AET test suites (client). There are two ways of using AET Maven plugin

- through command line,
- with use of Jenkins job.

This chapter covers second option - running tests with use of Jenkins job.

Requirements

To execute test suites with Jenkins job following requirements must be fulfilled:

Requirements

1. Jenkins installed with `maven-plugin.`
2. Maven installed on the same virtual machine as Jenkins (recommended version - 3.0.4).
3. Proper version of AET Maven plugin installed on the same machine as Jenkins.
4. SCM repository (i.e. Git or SVN) installed and configured to store well-formed and valid test suite configuration files (see Running suite for more details).

Configuration

1. Create new Jenkins build (preferred is *Maven project* or *MultiJob Project* - reqirement is that build should enable executing maven command).

2. Open Jenkins build configuration (click *Configure* link).

3. Find *Source Code Management* section and configure *Git*, *Subversion* or other type of repository that is in use. Repository should be checked out during each project build, before `aet:run` command is invoked.
4. Find *Build* section and set following values:

    1. *Maven Version*

        ```
        Maven 3.0.4
        ```

    2. *Root POM*

        ```
        pom.xml
        ```

    3. *Goals and options* - please remember that *IP* value for *brokerURL* depends on *IP address* used by virtual machine.

        ```
        aet:run -DtestSuite=${TEST_SUITE_FILE_NAME} -DbrokerURL=tcp://${BROKER_IP}:
        61616 -Dpassword=${ACTIVEMQ_PASSWORD}
        ```

`${TEST_SUITE_FILE_NAME}` - should be replaced with full (with `.xml` extension) suite definition file name (e.g. *testsuite.xml*).
`${ACTIVEMQ_PASSWORD}` - should be replaced with ActiveMQ broker password (if authentication required).

`${BROKER_IP}` - should be replaced with address of ActiveMQ Broker configured in Environment setup section.

4. *MAVEN_OPTS*

```
-Xms256m -Xmx2048m -XX:MaxPermSize=128m
```

5. Find *Post-build Actions* and set below value as *Files to archive*

```
target/*
```

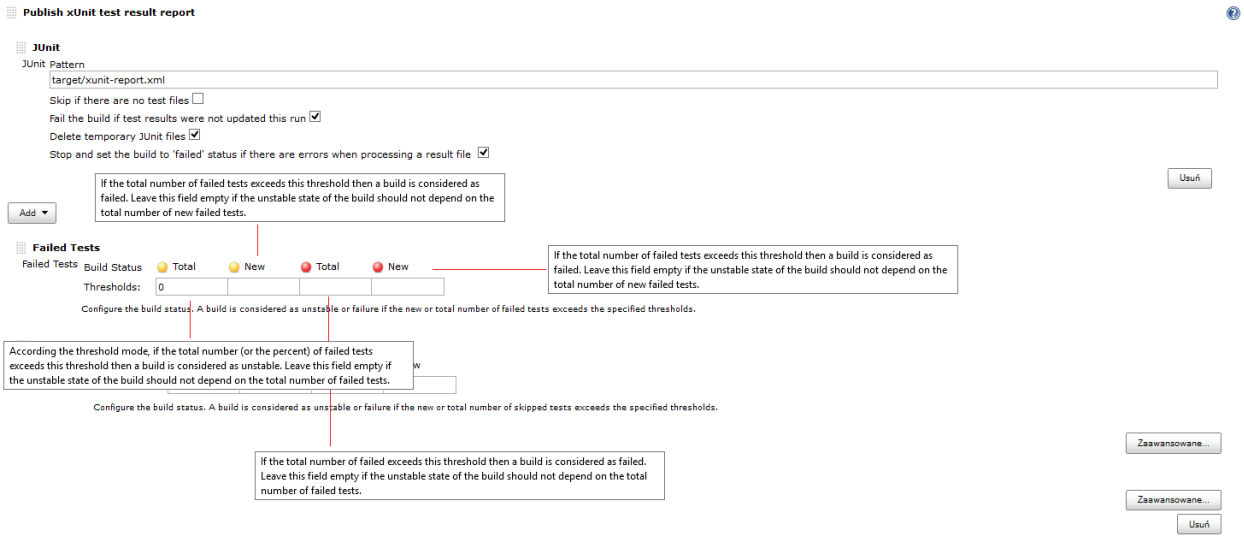
6. If xunit-report used in test suite configure job to use it. *AddPost-build Actions* and choose *Publish xUnit test result report*

    1. Add junit
    2. Set *JUnit Pattern*

    ```
    target/xunit-report.xml
    ```

    3. Check *Fail the build if test results were not updated this run* option

    4. Check *Delete temporary JUnit files* option
    5. Check *Stop and set the build to 'failed' status if there are errors when processing a result file* option
    6. Set up *Thresholds*



Usage

To run test suite open configured Jenkins job and click *Build now* link. The progress bar will appear in *Build history* panel. When the job is finished it's execution is marked with green, yellow or red color, depending on the tests' results and set thresholds.

How to open report

The result of build is ***report-full.html*** (or ***redirect.html*** if using html-report parameter `mode` setted to `online`) and/or ***xunit-report.xml*** files. To review generated reports choose newest build from *Build history* panel – reports are available in *Build Artifacts* sections. To list all failed tests click *Test results* link.

Workaround for Git checkout problems on Windows

Windows Git checkout only

This problem occurs only when Jenkins runs its build on Windows machine. Projects built on Linux are free from this problem.

In some cases for AET jenkins builds configured to be run on Windows nodes occurs problem with Git repository checkout, resulting in error:

```
stderr: Host key verification failed.
fatal: The remote end hung up unexpectedly
```

The workaround for this problem is to create two linked jenkins jobs

1. Prerequisite one for fetching Git repository artifacts - this job should be run on master node (with Linux installed).
2. Proper AET build which triggers execution of above described job, copies it's workspace and uses this copy for running AET test suites.

Settings for prerequisite (Git fetching) job

1. Create new Jenkins job (can be free-style software project).
2. Open Jenkins build configuration.
3. Find *Restrict where this project can be run* section and set *Label Expression* to *master.*
4. Find *Source Code Management* section and configure *Git* - provide *Repository URL, Credentials* and *Branches to build.*
5. In *Additional behaviors* add *Wipe out repository & force clone.*

Settings for proper AET job

1. Create new Jenkins build - follow the instruction provided above in Configuration section, omit point 4 ("Find *Source Code Management* section and configure *Git*, *Subversion* or other type of repository that is in use").
2. Find *Source Code Management* section and check *Clone workspace option*. Provide:
    1. *Parent project* - name of prerequisite (Git fetching) job created earlier.
    2. *Criteria for parent build* - Most Recent Completed Build.
3. Find *Pre-Steps* and add *Trigger / call builds on other projects* step. Provide:

1. *Projects to build* - name of prerequisite (Git fetching) job created earlier.
2. Check *Block until the triggered projects finish their builds.*

5.2.b Running suite using Gradle - 1.3

Currently, running an AET suite requires using *aet-gradle-plugin* which is an AET client application. To do so, two configuration files are mandatory:

- xml file - with defined suite (described with details in Suite setup section),
- build.gradle - which is a configuration file for *aet-gradle-plugin*.

build.gradle

This file (`build.gradle`) is a *Gradle* tool configuration file that contains information about the project and configuration details used by *Gradle* to build the project.

Running AET suite requires creating and configuring such a file. The File presented below might be used as a template for setup AET suite runs:

build.gradle

```
description = """Sample usage of gradle AET plugin"""
defaultTasks 'aet'
println cogNexusUser

buildscript {
    repositories {
        mavenLocal()
        mavenCentral()
        maven { url "https://nexus.cognifide.com/content/groups/public" }
        maven {
            url "https://nexus.cognifide.com/content/groups/private"
            credentials {
                username nexusUser
                password nexusPassword
            }
        }
        maven { url "https://repo1.maven.org/maven2" }
    }
    dependencies {
        classpath group: 'com.cognifide.aet', name: 'aet-gradle-plugin', version: "+"
    }
}

apply plugin: 'aet'

//those are default parameters
aetConfig {
    testSuite = "aet.xml"
    brokerURL = "tcp://localhost:61616"
    userName = "karaf";
    password = "karaf";
    inQueue = "AET.runner-in"
    domain;
    timeout = 300000
}
```

Note that this script has "+" sign as version value of 'aet-gradle-plugin'. This ensure that most newest version of aet-gradle-plugin will be used each time.

```
classpath group: 'com.cognifide.aet', name: 'aet-gradle-plugin', version: "+"
```

AET Gradle plugin options

Running AET suite with *AET Maven plugin* cam be done simply by invoking a set of maven command(s):

```
gradle -PaetConfig.testSuite=customSuiteFile.xml -PaetConfig.domain=http://cognifide.com -PaetConfig
```

`testSuite` is the only mandatory parameter and is the path to the xmf suite configuration file. All parameters are described below:

Parameter	Description	Default Value	Mandatory
testSuite	Full path to suite definition file (at least file name with extension, e.g. *testSuite.xml*).	aet.xml	✔
brokerURL	URL to ActiveMQ broker.	tcp://localhost:61616	✘
userName	ActiveMQ user name.	karaf	✘
password	ActiveMQ password.	karaf	✘
domain	Overrides *domain* parameter value from test suite definition.	-	✘
timeout	Milliseconds to detect timeout since the last message from AET. This is useful to abort test run if no activity for long time.	300000 (5 minutes)	✘

Tips and recommendations

Generally it is good idea, to create separate **SCM repository** (e.g. *GIT* or *SVN*) for AET suites. This will enable running AET suites using Jenkins with very easy steps.

5.3. Test run output and console - 1.3

How to read AET Reports and real time progress

AET test reports are updated on real time basis and can be viewed on the console. This progress information is accessible in using two methods:

1. as a command line and
2. with use of Jenkins job.

To see progress

- log on Jenkins
- choose proper build execution from Build history panel and
- click Console Output.

For every test suite started the execution information is provided in the progress log:

progress log

```
[INFO] ******************************************************************************
[INFO] *************** Job Setup finished. Waiting for reports...  ***************
[INFO] ******************************************************************************
```

During test processing dertail information about actual progress is displayed as in the following example:

progress log

```
...
[INFO] [06:34:20.680]: COLLECTED: [success: 0, total: 72] ::: COMPARED: [success: 0, total: 0] ::: R
[INFO] [06:34:31.686]: COLLECTED: [success: 1, total: 72] ::: COMPARED: [success: 1, total: 1] ::: R
[INFO] [06:34:35.689]: COLLECTED: [success: 2, total: 72] ::: COMPARED: [success: 1, total: 2] ::: R
[INFO] [06:34:36.691]: COLLECTED: [success: 2, total: 72] ::: COMPARED: [success: 2, total: 2] ::: R
[INFO] [06:34:43.695]: COLLECTED: [success: 3, total: 72] ::: COMPARED: [success: 2, total: 3] ::: R
[INFO] [06:34:44.698]: COLLECTED: [success: 3, total: 72] ::: COMPARED: [success: 3, total: 3] ::: R
...
```

where:

collected - shows results of collectors' work - how many artefacts have been successfully collected and what is the total number of all artefacts to be collected,

compared - shows results of comparators' work - how many artefacts have been successfully compared and what is the total number of all artefacts to be compared. The total number of artefacts to be compared depends on collectors' work progress - increases when the number of successfully collected artefacts increase,

reports - shows results of reporters' work - how many reports have been successfully generated and what is the total number of all reports to be generated.

If there are problems during processing, warning information with some description of processing step and its parameters is displayed:

progress log

```
[WARN] CollectionStep: source named source with parameters: {} thrown exception. TestName: comparato
```

In this example source collector failed to collect necessary artefacts. This information is subsequently reflected in the progress log:

progress log

```
...
[INFO] [06:36:44.832]: COLLECTED: [success: 46, failed: 1, total: 72] ::: COMPARED: [success: 46, to
[INFO] [06:36:50.837]: COLLECTED: [success: 47, failed: 1, total: 72] ::: COMPARED: [success: 47, to
[INFO] [06:36:52.840]: COLLECTED: [success: 48, failed: 1, total: 72] ::: COMPARED: [success: 47, to
...
```

In the example above one artefact has failed during collection phase.

When tests successfully finish - command line

When the AET test processing completes the information about received reports and processing status - BUILD SUCCESS or BUILD FAILURE is shown on the console - as shown below:

progress log

```
[INFO] Received report message: ReportMessage{company=aet-demo-sanity, project=demo-sanity-test, tes
[INFO] [06:38:03.549]: COLLECTED: [success: 71, failed: 1, total: 72] ::: COMPARED: [success: 71, to
[INFO] Received report message: ReportMessage{company=aet-demo-sanity, project=demo-sanity-test, tes
[INFO] Total: 2 of 2 reports received.
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 3:45.645s
[INFO] Finished at: Tue Mar 17 06:38:03 CET 2015
[INFO] Final Memory: 14M/246M
[INFO] ------------------------------------------------------------------------
```

BUILD SUCCESS - status means that test processing is successfully finished and reports are generated in target folder.

BUILD FAILURE - status means that there were some technical problem during processing for example database is not responding and it is not possible to receive reports.

When test is successfully finished - Jenkins job

Jenkins console output presents the same information as described above, but if test suite is defined to generate xunit-report additional information such as Junit processing is logged on console:

progress log

```
[xUnit] [INFO] - Starting to record.
[xUnit] [INFO] - Processing JUnit
[xUnit] [INFO] - [JUnit] - 1 test report file(s) were found with the pattern 'test-suite/target/xuni
[xUnit] [INFO] - Converting '/var/lib/jenkins/jobs/aet-sanity-test-integration/workspace/test-suite/
[xUnit] [INFO] - Check 'Failed Tests' threshold.
[xUnit] [INFO] - The new number of tests for this category exceeds the specified 'new unstable' thre
[xUnit] [INFO] - Setting the build status to UNSTABLE
[xUnit] [INFO] - Stopping recording.
Build step 'Publish xUnit test result report' changed build result to UNSTABLE
Finished: UNSTABLE
```

The meaning of '*Successful*' and '*Failed*' build is quite different here, because final build status depends mainly on tests results and thresholds configuration. The build can result with BUILD SUCCESS status (which means that all workers - collectors, comparators, reporters finish their work and proper reports were generated), but final Jenkins build status can be for example UNSTABLE becase there were some new test failures.

A Jenkins build is considered as UNSTABLE (yellow) or FAILURE (red) if the new (tests that failed now, but did not fail in previous run) or total number of failed tests exceeds the specified thresholds. For example:when "yellow total" threshold is set to 0 and one or more test cases failed, then build is mark as UNSTABLE.

5.4. Logs - 1.3

Overview

AET log files can be found on Apache Karaf directory, in `C:\content\karaf\data\log` folder.
Logs are split into four files:

`karaf.log`

Dedicated for logging Apache Karaf activities such as starting Karaf, starting AET bundles, configurations binding, e.t.c.

`runner.log`

Dedicated for logging AET runner activities, such as running new test, changes in test suite run lifecycle, communication between runner and workers, JMS messages management, e.t.c.

`worker.log`

Dedicated for logging AET workers activities, such as collecting, comparing, modifying data, saving data do Mongo DB, e.t.c.

`cleaner.log`

Dedicated for logging *Cleaner* activities (removing old Artifacts from Mongo DB, e.t.c.).

Log structue

Each log record has common structure:

worker.log

```
 1       2  3        4         5  6
2015-03-19 15:22:26,307 | INFO  | worker                   | CollectorMessageListenerImpl       86 |
2015-03-19 15:22:28,650 | INFO  | worker                   | CollectorDispatcherImpl            41 |
2015-03-19 15:22:28,651 | DEBUG | worker                   | CollectorDispatcherImpl            53 |
2015-03-19 15:22:29,351 | DEBUG | worker                   | CollectorDispatcherImpl            53 |
2015-03-19 15:22:29,351 | DEBUG | job-common               | SleepModifier                      27 |
2015-03-19 15:22:30,851 | DEBUG | worker                   | CollectorDispatcherImpl            53 |
2015-03-19 15:22:31,804 | DEBUG | datastorage-gridfs-impl  | GridFsStorage                     600 |
2015-03-19 15:22:31,813 | INFO  | datastorage-gridfs-impl  | GridFsHelper                      428 |
2015-03-19 15:22:31,814 | DEBUG | datastorage-gridfs-impl  | GridFsStorage                     600 |
...
```

where:

1. Log record date and hour,
2. Log level (INFO, DEBUG or ERROR),
3. Name of the system module where information is logged,
4. Name of the class,
5. Line of code,
6. Log message.

Logs configuration

AET logging can be configured in *org.ops4j.pax.logging.cfg* file in *C:\content\karaf\data\etc* folder.

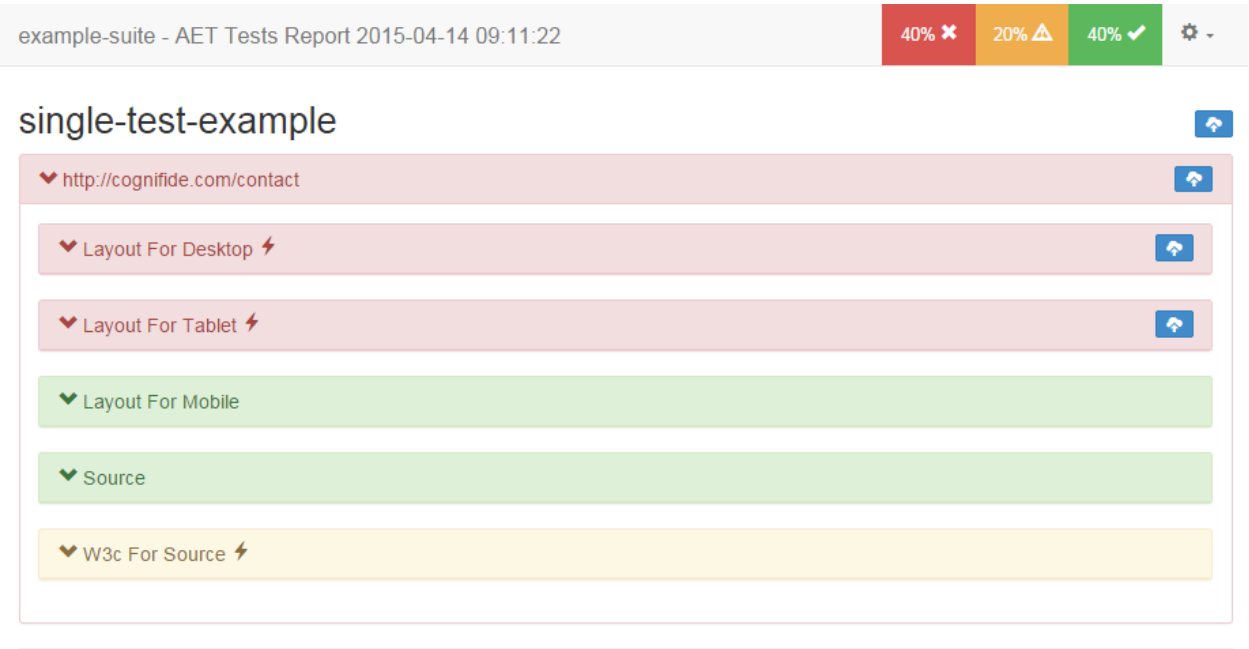
This configuration file specifies among others log files destination folder, log level and pattern, log file maximum size.

6. Report - 1.3

There are two types of reports in current version of AET System:

- html report,
- xUnit report.

HTML Report



The basic report is in form of a HTML file. HTML results presented in the report are grouped by test. Each test (section *example-test* on screenshot above) contains list of urls. On each url level (*http://cognifide.com/contact* on screenshot above), results of all tests performed on given page are displayed (*W3c For Source, Layout For Desktop, Layout For Mobile, Layout For Table, Source*).

Each result will be presented in one of three colours:

- **green** - if all group results passed and no risks was detected,
- **yellow** - if there is small risk detected, e.g. w3c warning,
- **red** - if there were some risk detected and result requires inspection.

With the example above, 2 test cases passed (*Layout for Mobile and Source comparison*), one test has been identified as having a minor risks (*W3c For Source)* and there are some major differences detected with screenshot taken for certain desktop (*Layout For Desktop*) and tablet size (*Layout For Tablet*).

The whole url group is marked with a red colour because at least one (in this case two *Layout for Desktop* and *Layout For Tablet*) test case has been identified and detected as a risk.

xUnit report

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<!-- Report generated only for Jenkins. -->
<testsuites name="example-suite" tests="5" failures="2">
    <testsuite name="single-test-example" tests="5">
        <testcase name="source /contact"/>
        <testcase name="layout for mobile /contact"/>
        <testcase name="layout for tablet /contact">
            <failure/>
        </testcase>
        <testcase name="layout for desktop /contact">
            <failure/>
        </testcase>
        <testcase name="w3c for source /contact"/>
```

```
    </testsuite>
</testsuites>
```

The second type of report is used by *Jenkins* (xUnit Plugin) to visualize risks on *Jenkins* job board.

xUnit report contains information about number of performed tests and number of failures (potential threats).

6.1. Navigation - 1.3

This section has the instructions to assist the user to navigate through an HTML report.

Report header and menu

HTML Report header contains information about:

1. suite name,
2. report generation date,
3. tests status (percentage of detected risks, warnings and passed tests from all performed tests),
4. options menu (opened by clicking on pinion icon).

Options available in menu are:

- Toggle All - opens or closes all tests results (on test level).
- Toggle Errors - opens or closes all results marked as risk (marked as red).
- Toggle Big screenshots - switch off or on displaying layout results as big screenshots.
- Set baseline TS - for each result that can be rebased (see Dictionary section), rebase action is performed. Successful action shows message with number of rebased artefacts:

  Success: artifacts rebased: 2, all artifacts: 8 ×

  In case when all results are already marked as current pattern, message:

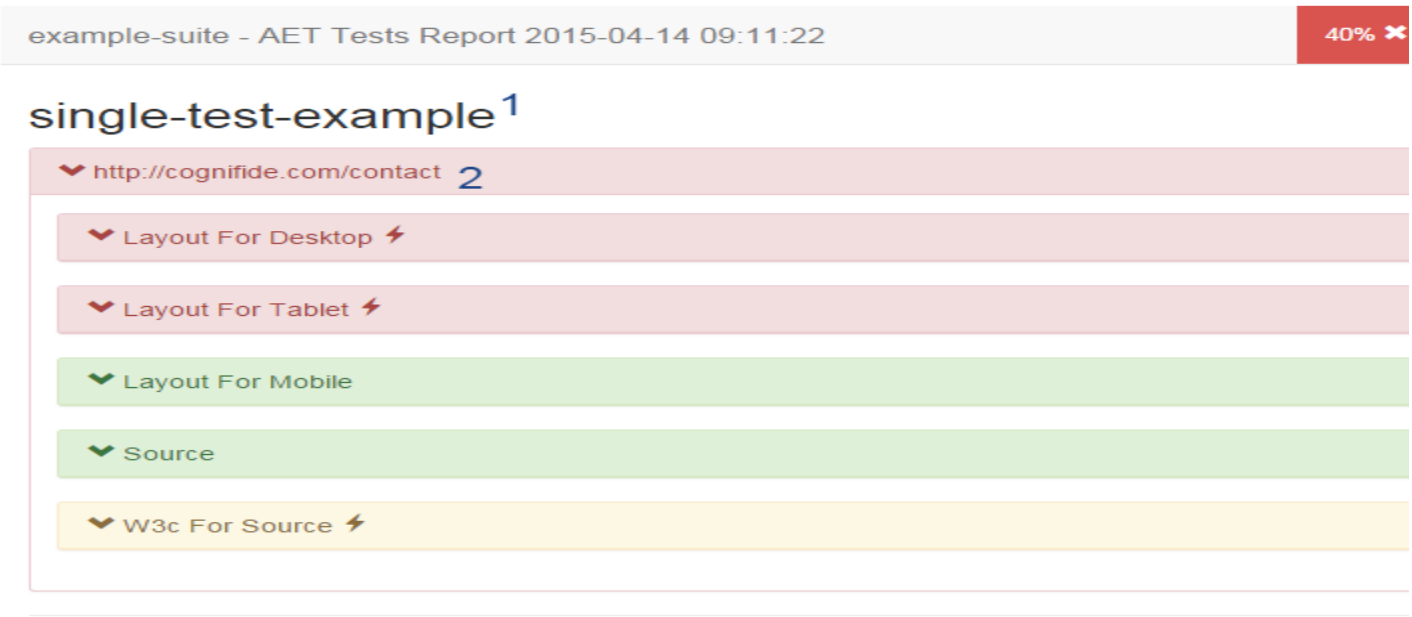
  Already rebased ×

  is presented.
  In case when rebase action was started but is still in progress:

  Rebase in progress ×

  is presented. After that every few second report checks current status of rebase action and shows one of messages presented above.
- Help - displays information about keyboard shortcuts.

Report results (tests)



Report body contains results grouped in tests section. Each section might contain:

1. Test name,
2. List of urls in test,
3. Rebase button (see Dictionary section), which performs action for all urls results in this test that can be rebased. Successful action shows message with number of rebased artefacts:

   Success: artifacts rebased: 2, all artifacts: 8

   In case when all results are already marked as current pattern, message:

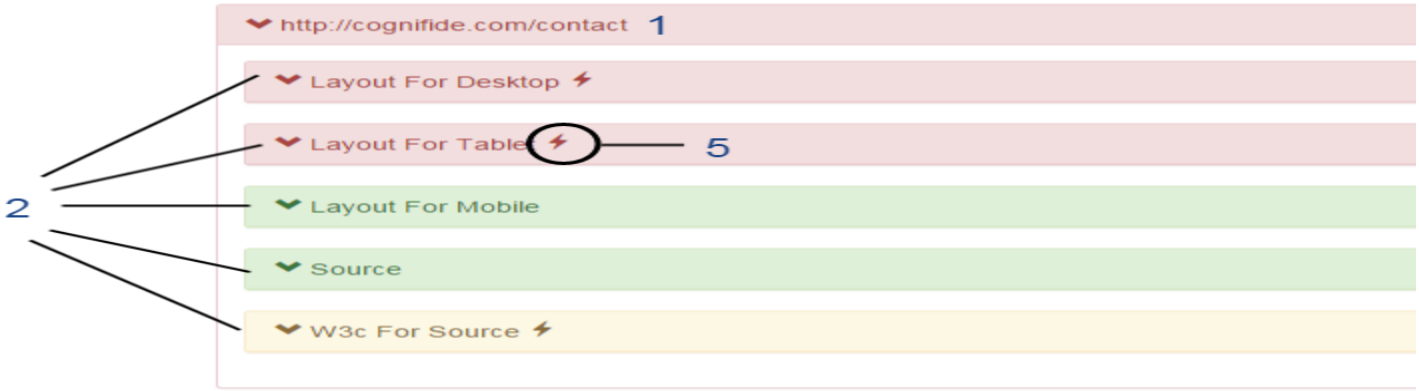
   Already rebased

   is presented.
   In case when rebase action was started but is still in progress:

   Rebase in progress

   is presented. After that every few second report checks current status of rebase action and shows one of messages presented above.

Test results (urls)



Url entry on html report contains list of test cases performed on page. To view detailed list of test cases, the user can click on url name section (1). Url section with listed tests contain:

1. Url (when clicked shows or hides details),
2. Test case name. To open result details, the user can click 'name'.
3. Rebase all url testcases button - for each testcase that can be rebased (see Dictionary section), rebase action is performed. Successful action shows message with number of rebased artefacts:

   Success: artifacts rebased: 2, all artifacts: 8 ×

   In case when all results are already marked as current pattern, message:

   Already rebased ×

   is presented.
   In case when rebase action was started but is still in progress:

   Rebase in progress ×

   is presented. After that every few second report checks current status of rebase action and shows one of messages presented above.
   Using this action is equivalent to using all three other (4) rebase testcase buttons sequentially.
4. Rebase single testcase. Successful action shows message with number of rebased artefacts:

   Success: artifacts rebased: 2, all artifacts: 8 ×

   In case when all results are already marked as current pattern, message:

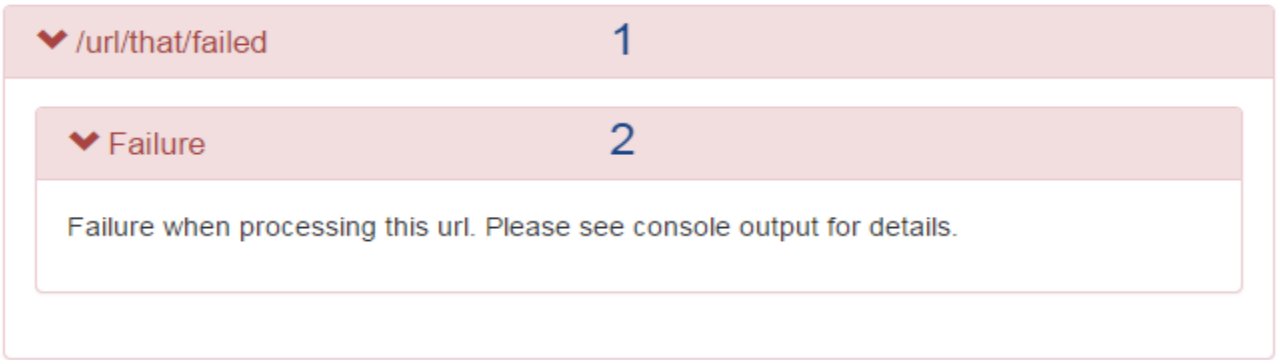
   Already rebased ×

   is presented.
   In case when rebase action was started but is still in progress:

   Rebase in progress ×

   is presented. After that every few second report checks current status of rebase action and shows one of messages presented above.
5. Risks icon. When hovering over this icon, risks that are connected with the type of testcase are displayed.

When there is problem with one of urls during the tests run then the the following section will be visible on report instead of url with testcases list:



This screen contains:

1. Failed url name (usually, when the test is successful, in this place the full page url is presented),

2. Failure details. In this case, the user should search for details in console output.

6.2. Testcase results interpretation - 1.3

This section covers all available results and the hints on how to interpret them.

Also, each section explains any **risks** connected with the change or error.

6.2.1. Cookies - 1.3

The Cookie test results can be presented in three different forms which depend on `action` parameter defined in test definition:

- list,
- test,
- compare.

List

Simply lists all cookies found on tested page. This result will always have *success* status (marked as green). Example:

Cookie

No.	Name	Value	Expiry
1.	sampleCookieName	sampleCookieValue	Mar 18, 2015 6:28:36 AM
2.	JSESSIONID	10wq4ws7ev3cu1qcan666hh3qr	
3.	authSessionCognifide	expiry=1426573421592455	Mar 17, 2015 7:28:37 AM

Table column contains:

- No. - entry ordinal number,
- Name - cookie name,
- Value - cookie value,
- Expiry - cookie expiry date if set.

Risks

The user can check all site cookies (name, value, expiry date) and find a cookie invalid expiry date or value. However, this mode (list) of cookie features is not intended to discover website issues.

This ‘list’ should be empty when a page does not intend to use cookies and The EU Cookie Law is respected.

Test

Shows result of checking if cookie with defined parameters (name and/or value) is present on tested page.

In case cookie was found, result has form like below:

Cookie

Cookie with name: 'sampleCookieName' and value: 'sampleCookieValue' found

In case when cookie was not found on page, result is marked as risk (red):

Cookie

Cookie with name: 'nonExistingCookieName' not found

Risks

- The lack of a cookie that occurred before might be caused by some website error (e.g. a bug in system functionality).
- The lack of a cookie may result in further system errors (e.g. the user will not be able to stay logged in across pages, some data about the user will be lost).
- The lack of an important cookie (e.g. a cookie with user localisation data) may cause a page to be display improperly.

Compare

Rebase

This feature result is compared with a pattern, which means it can be rebased (see Dictionary).

Cookies found on the tested page are compared with cookies that were saved in the pattern (if no pattern exists, the cookies collected during the first page entry are set as pattern). Differences are searched for only in cookies names, and values are ignored. Results will be successful if all found cookies names are identical to those in the pattern:

Cookie

Cookie names are identical.

Otherwise, result on the report will be marked as at risk (red) and differences will be presented in the form:

Cookie

Additional cookies:

No.	Name
1.	"Tue14Apr2015101414043+0200"

Not found cookies:

No.	Name
1.	"Fri27Mar2015150547314+0100"

Cookies matching the pattern:

No.	Name
1.	JSESSIONID
2.	DynamicSampleCookieName

and `rebase` action will be available. After using it, pattern value will be set to all cookies found in current result.

Risks

- The lack of a cookie that detected before might be caused by some website error (e.g. a bug in system functionality).
- The lack of a cookie may result in further system errors (e.g. the user will not be able to stay logged in across pages, some data about the user will be lost).
- An additional cookie may be caused by some unwanted content on a page, e.g. some 3rd party software adds its cookies.
- When a page does not intend to use cookies and The EU Cookie Law is respected, lists of additional and detected cookies should always be empty.

6.2.2. JS errors - 1.3

JS Errors tests results display success status when no js errors were found (all errors filtered with JS Errors Data Filter are omitted):

Js errors

No errors found

Otherwise the report is marked as at risk (red) when at least one errors has been found:

Js errors

No.	Error	Source	Line number
1.	ReferenceError: nonExistingJsFunction is not defined	http://staging-aet.cognifide.com/aet-demo/sanity/comparators/jserrors/failed.jsp	20
2.	ReferenceError: nonExistingVarible is not defined	http://staging-aet.cognifide.com/aet-demo/sanity/comparators/jserrors/failed.jsp	383

The Table report presents the following columns:

- No. - entry ordinal number,
- Error - error description,
- Source - source file from which error comes,
- Line number - line number in file from which error comes.

Risks

- JS Errors can cause improper behaviour of a page. Because of js errors, dynamic components may not work properly in some (or all) browsers.
- JS Error can also occur when good practices are not followed in javascript code.

6.2.3. Screen - Layout - 1.3

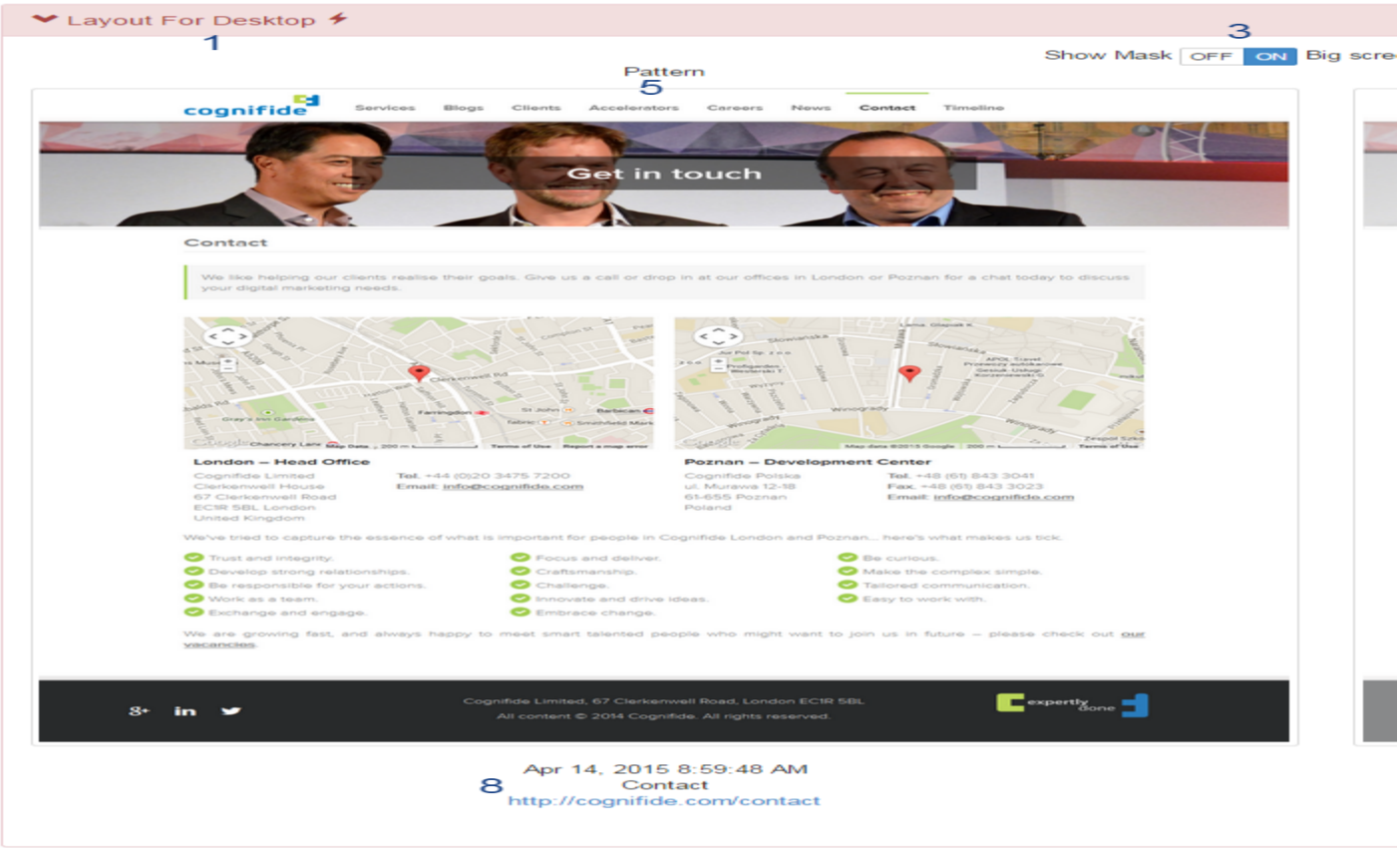
Changes in version 1.3

Starting from version 1.3:
-mask is displayed both on last collected screenshot and pattern,
-mask disappears on mouse hover (on patter and collected screenshot separately)
-screenhots identical to pattern are not saved, so there are missing from reports.(only pattern is displayed)

Rebase

This feature result is compared with pattern, which means it can be rebased (see Dictionary).

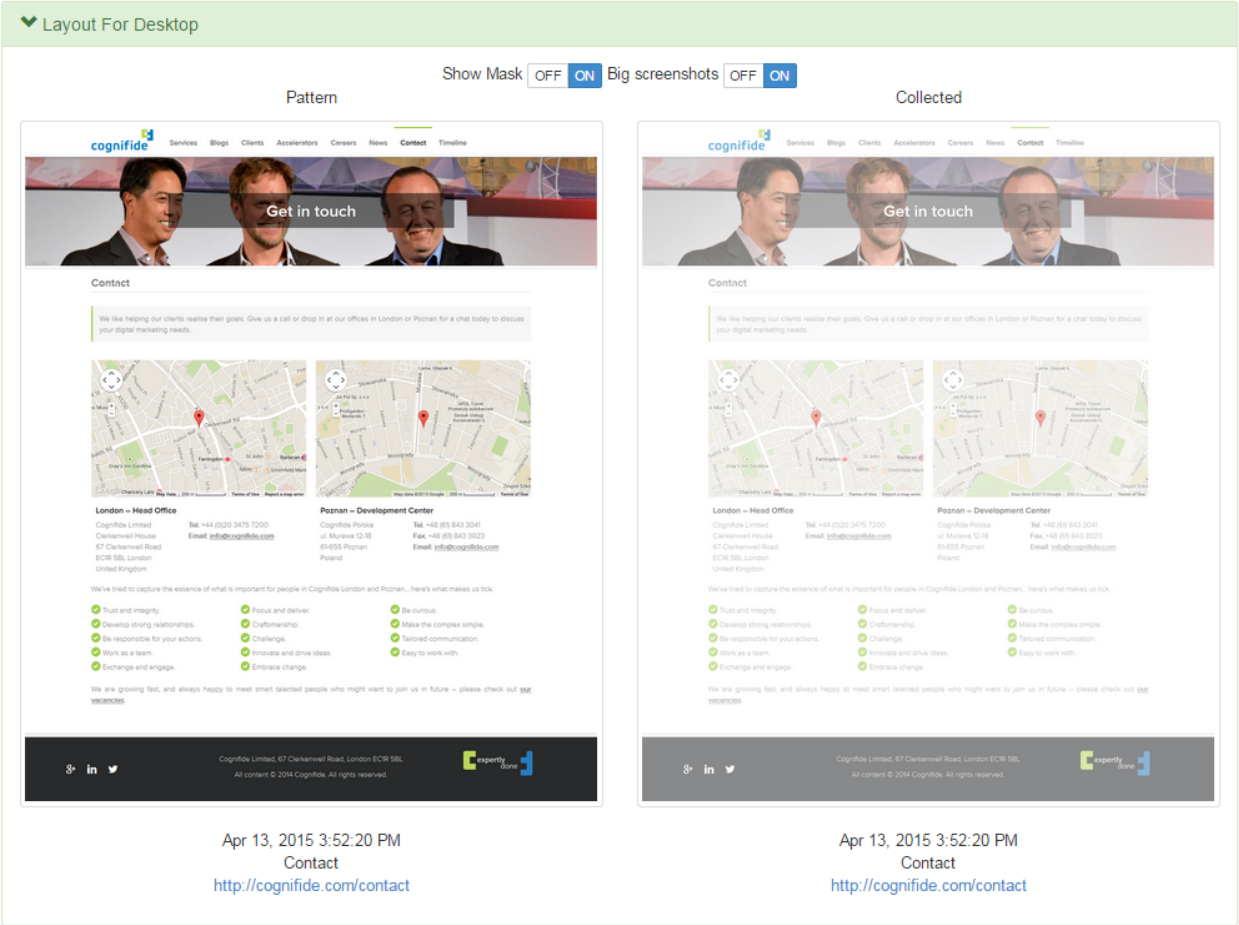
The Layout test results are presented as compared screenshots.



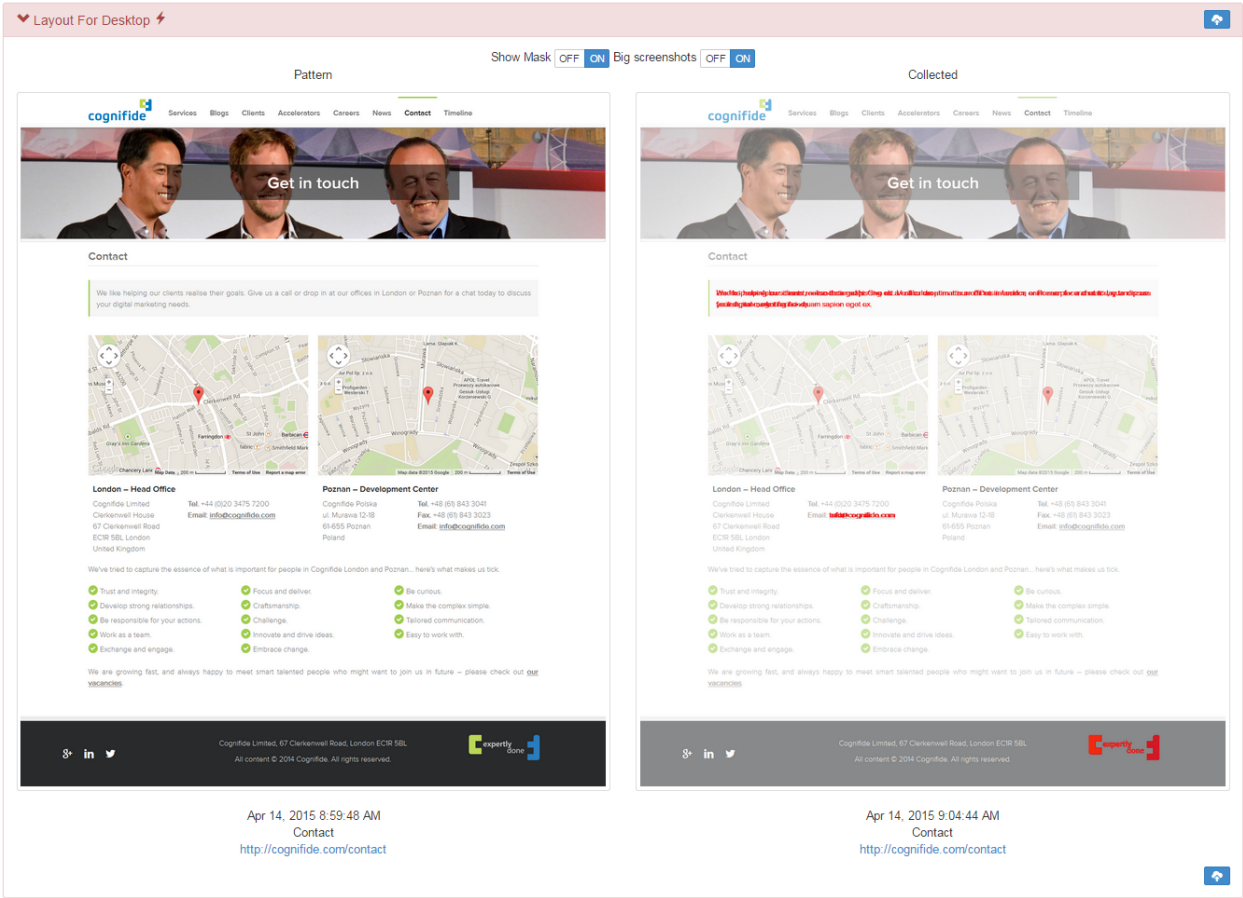
This report has additional navigation:

1. Testcase name. On the right of name - risks icon.
2. Rebase button (available only when differences were detected). Additional button is present at the bottom of report to make rebasing easier after scrolling bigger screenshots.
3. Show mask switch - changes behaviour of right *Collected* screen. When mask is on (by default) differences are marked in red colour over collected screenshot. When mask is off, raw screenshot is presented.
4. Big screenshot switch - changes size of visible screenshot.
5. Pattern - screen to which *Collected* screenshot is compared. When users clicks the screenshot it is opened in new browser tab in original size. When there is no pattern, first collected screenshot is saved as a pattern automatically.
6. Collected - screen that was taken during the test and is compared to the *Pattern*. When users clicks on the screenshot it is opened in a new browser tab with the original size.
7. Example difference area, all detected changes are marked on in red the when mask is 'ON' .
8. Pattern data - date when pattern screenshot was taken, page title and page url.
9. Collected data - date when collected screenshot was taken, page title and page url.

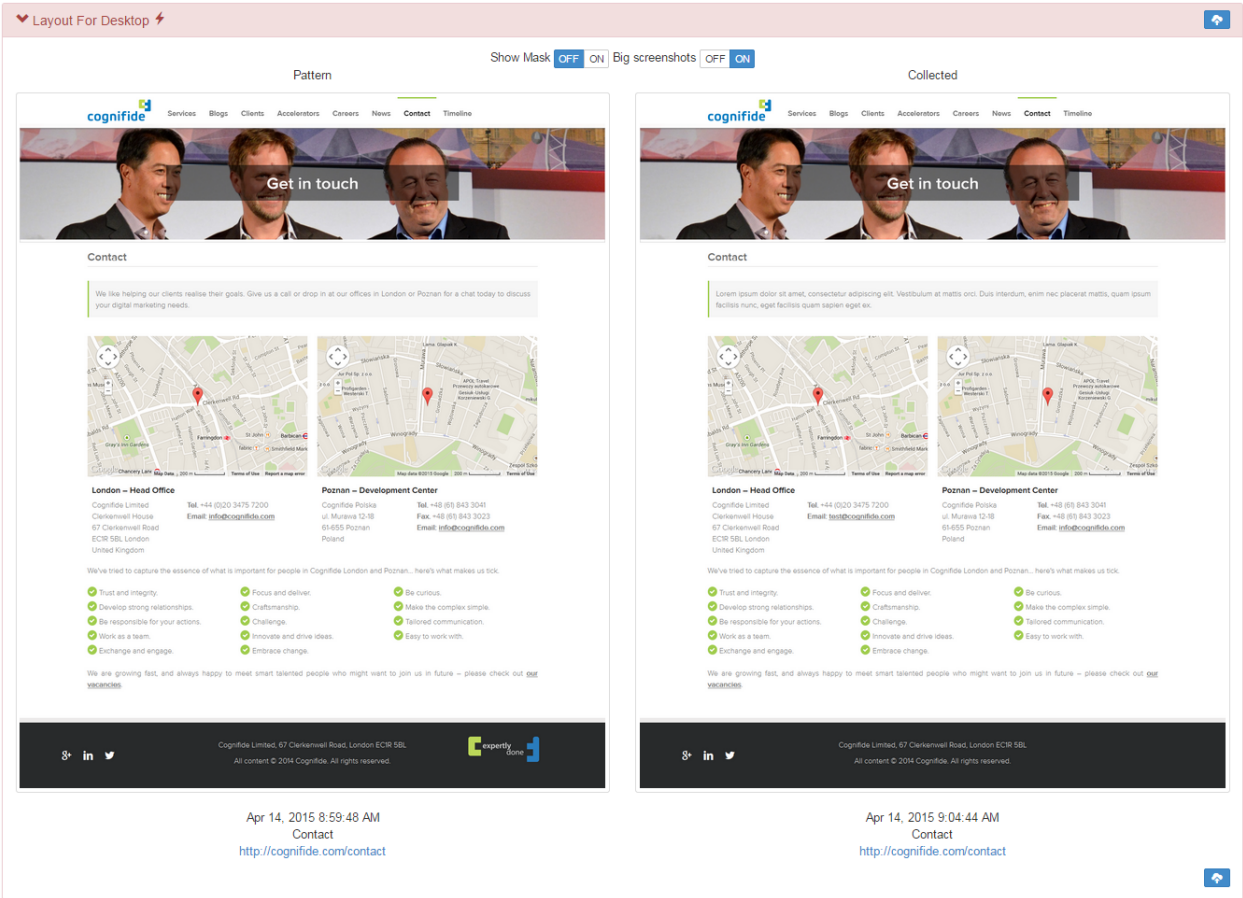
Report result is marked as successful when no difference between Pattern and Collected were found:



When differences are found the screenshots comparison (Collected with Pattern), test is marked as potentially risky. When mask is 'ON' all differences are marked with a red colour:



When mask is set to 'OFF' the user can compare visually the differences:



Risks

- Differences found on page screenshots may indicate undesired changes in the page layout (css, html structure) e.g . when a new functionality was implemented in a system it may have an impact on another system component(s). This may show itself as a changed page layout.

- Content changes can be divided into two groups: **wanted** (are intended) and **unwanted** (result of a mistake or an error). An example of a change that is not a defect (wanted) is: the *carousel component* with the latest news items displayed or the *twitter component* displaying latest tweets. In order to avoid detecting these sorts of changes in these dynamic components, the user can use the 'Hide Modifier' feature in the suite definition. Another example of a 'wanted' dynamic content is a cookies policy popup that may be hidden using the Cookie Modifier.

6.2.4. Source - 1.3

Rebase

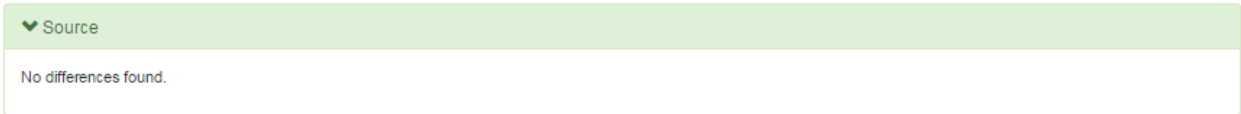
This feature result is compared with pattern, which means it can be rebased (see Dictionary).

Source tests results display compared sources with additional navigation:

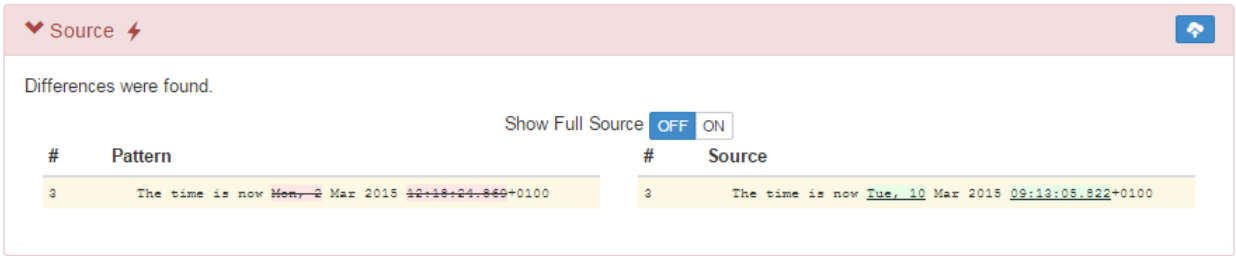
1. Testcase name. On the right of name - risks icon.
2. Rebase button (available only when differences were detected).

3. Show Full source switch - changes displaying source where differences were not found (OFF - show differences only, ON - show full source).
4. Pattern - source to which *Source* source is compared. When there is no pattern, first collected source is saved as pattern automatically.
5. Source - source which is compared with *Pattern*.
6. Example block with no differences found (light blue). Visible only when *Show Full Source* switch is ON. Starts with number of line where block begins.
7. Example block with important difference (e.g. missing block of code) marked with red color. Starts with number of line where block begins.
8. Example block with no differences found (light blue). Visible only when *Show Full Source* switch is ON. Starts with number of line where block begins.
9. Example block with change difference (e.g. changed characters) marked with yellow color. Starts with number of line where block begins.

When no differences were found, report result is marked as successful:



Otherwise report is marked as potentially risky (red) and differences are presented on report:



Risks

- Differences found by source comparison may indicate undesired changes in a page layout (html structure) and content, e.g. when a new functionality is implemented in a system it might have an impact on other system component(s). This may occur as a changed page source.
- Content changes can be divided into two groups: **wanted** (intended) and **unwanted** (result of a mistake or an error). In order to filter out wanted changes and detect changes that are a result of a mistake or an error, the user can use one of following filters in the suite definition:
  - The Extract Element Data Modifier (e.g. to find changes only in the main menu that has the parameter `id='main-menu'` set),
  - The Remove Lines Data Filter (to remove lines that changes every time - e.g. a current timestamp),
  - The Remove Nodes Data Filter (e.g. to remove content displayed by the dynamic news carousel component).

6.2.5. Status codes - 1.3

Status codes results display success when no status codes were found (all filtered codes are omitted):

Status codes

No errors found

If any status code was found, report is marked as risk (red):

Status codes

No.	Status Code	URL
1.	404	http://cognifide.com/aet-demo/assets/demo_files/NonExistingResourceFile.png
Excluded status codes:		
1.	200	http://staging-aet.cognifide.com/aet-demo/sanity/comparators/statuscodes/include-exclude.jsp
2.	200	http://staging-aet.cognifide.com/aet-demo/assets/demo_files/bootstrap.css
3.	200	http://staging-aet.cognifide.com/aet-demo/assets/demo_files/bootswatch.min.css
4.	200	http://staging-aet.cognifide.com/aet-demo/assets/demo_files/jquery.min.js
5.	200	http://staging-aet.cognifide.com/aet-demo/assets/demo_files/bootstrap.min.js
6.	200	http://cognifide.com/%7E/media/cognifide2014/avada-assets/cognifide-expertly-done.png
7.	200	http://staging-aet.cognifide.com/aet-demo/assets/demo_files/logo.png
8.	200	http://staging-aet.cognifide.com/aet-demo/assets/demo_files/ie10-viewport-bug-workaround.js
9.	200	http://fonts.googleapis.com/css?family=Lato:300,400,700
10.	200	http://staging-aet.cognifide.com/aet-demo/assets/demo_files/favicon.ico

The Table report presents the following columns:

- No. - entry ordinal number,
- Status code - status code (number),
- URL - url which caused given status code.

Additionally if Status Codes Data Filters are used, excluded codes are presented in additional section.

Risks

- All status codes with a number higher than 400 are potential errors and indicate that the resource that is used by a page is unreachable (e.g. a page logo image, a page layout css file).
- Status code errors affect SEO (e.g. google page ranking is lowered for pages with 404 status codes).

6.2.6. W3C - 1.3

Validators

When using comparator='w3c' AET uses https://validator.w3.org/ libraries and html5 will be not supported.

Please use comparator='w3c-html5' when html5 should be supported, this variant uses https://validator.nu/ libraries.

W3C report results display page source w3c validation output. If no w3c errors were found, result is marked as success (green):

W3c For Source

Validation errors count: 0.

Validation warnings count: 1.

No.	Validation Output
1	**Using Direct Input mode: UTF-8 character encoding assumed** Unlike the "by URI" and "by File Upload" modes, the "Direct Input" mode of the validator provides validated content in the form of characters pasted or typed in the validator's form field. This will automatically make the data UTF-8, and therefore the validator does not need to determine the character encoding of your document, and will ignore any charset information specified.

or with warning (yellow) if w3c warnings were present and parameter `ignore-warnings` was set to *false:*

W3c For Source

Validation errors count: 0.

Validation warnings count: 2.

No.	Validation Output
1	**Using experimental feature: *HTML5 Conformance Checker.*** The validator checked your document with an experimental feature: *HTML5 Conformance Checker*. This feature has been made available for your convenience, but be aware that it may be unreliable, or not perfectly up to date with the latest development of some cutting-edge technologies. If you find any issues with this feature, please report them. Thank you.
2	**Using Direct Input mode: UTF-8 character encoding assumed** Unlike the "by URI" and "by File Upload" modes, the "Direct Input" mode of the validator provides validated content in the form of characters pasted or typed in the validator's form field. This will automatically make the data UTF-8, and therefore the validator does not need to determine the character encoding of your document, and will ignore any charset information specified.

If at least one w3c validation error was found, report is marked as risk (red):

W3c For Source

Validation errors count: 2.

Validation warnings count: 3.

No.	Validation Output
1	*Line 381, Column 51:* **Duplicate ID themes.** `data-toggle="dropdown" href="#" id="themes"Dropdown <span`
2	*Line 383, Column 55:* **The aria-labelledby attribute must point to an element in the same document.** `<ul class="dropdown-menu" aria-labelledby="dropdown"`
3	*Line 32, Column 50:* **The first occurrence of ID themes was here.** `data-toggle="dropdown" href="#" id="themes"Dropdown <span`
4	**Using experimental feature: *HTML5 Conformance Checker.*** The validator checked your document with an experimental feature: *HTML5 Conformance Checker*. This feature has been made available for your convenience, but be aware that it may be unreliable, or not perfectly up to date with the latest development of some cutting-edge technologies. If you find any issues with this feature, please report them. Thank you.
5	**Using Direct Input mode: UTF-8 character encoding assumed** Unlike the "by URI" and "by File Upload" modes, the "Direct Input" mode of the validator provides validated content in the form of characters pasted or typed in the validator's form field. This will automatically make the data UTF-8, and therefore the validator does not need to determine the character encoding of your document, and will ignore any charset information specified.

Result shows total count of validation errors and validation warnings.

Table in report presents in columns:

- No. - entry ordinal number,
- Validation Output - result of w3c validation from w3c validation service.

Risks

- The W3C validation is important from the SEO point of view. Pages that do not comply to W3C standards are ranked low in *Google PageRank* and other rankings.
- Detected W3C errors may indicate serious html structure bugs (e.g. tags that haven't been closed) or content issues (e.g. invalid tags parameters: <a> without href).
- Maintenance of pages that follow W3C standards is much easier to carry out because pages that keep these standards are much less prone to be displayed differently in different browsers or devices.
- The W3C validation can also reveal page encoding and special characters displaying issues.

6.2.7. Accessibility BETA - 1.3

Beta Version

This AET Plugin is currently in BETA version.

Accessibility report results display validation output of page accessibility analysis. Output presented on this report comes from html CodeSniffer library.

Result shows total count of error, warning and notice type violations.

Columns in table present:

- No. - ordinal number
- Validation Output - violation position in page source code, violation description, markup associated with the violation and the name of the rule that the code was checked against.

Accessibility

Errors count: 7 .

Warnings count: 0 .

Notice count: 0 .

No.	Validation Output

1	*Line 35, column 5:* **Form does not contain a submit button (input type="submit", input type="image", or button type="submit").**
	`<form method="post" action="/" id="form">`
	`<div class="a...mit;`
	`theForm.onsubmit = WebForm_SaveScrollPositionOnSubmit;`
	`//]]>`
	`</script>`
	`</form>`
	WCAG2AA.Principle3.Guideline3_2.3_2_2.H32.2

2	*Line 362, column 13:* **This element has insufficient contrast at this conformance level. Expected a contrast ratio of at least 4.5:1, but text in this element has a contrast ratio of 4.32:1. Recommendation: change background to #fbfbfb.** `<div class="subtitle field-subtitle">We drive efficiency and agility into digital m...ital experiences while reducing time to market from months or weeks, to days or hours.</div>` *WCAG2AA.Principle1.Guideline1_4.1_4_3.G18.Fail*

3	*Line 615, column 1:* **Anchor element found with a valid href attribute, but no link content has been supplied.**
	`<a href="/newsandevents/news/zengage-wins-adobe-innovation-showcase-award/">`
	`<div style="text-align: center;"></div>`
	`</a>`
	WCAG2AA.Principle4.Guideline4_1.4_1_2.H91.A.NoContent

4	*Line 662, column 29:* **This element has insufficient contrast at this conformance level. Expected a contrast ratio of at least 4.5:1, but text in this element has a contrast ratio of 4.32:1. Recommendation: change background to #fbfbfb.** `<span style="font-family: ProximaNova-Light, Arial, Helvetica, sans-serif; font-size: 16px;...stify; background-color: rgb(246, 246, 246);">Connecting Customer Experiences</span>` *WCAG2AA.Principle1.Guideline1_4.1_4_3.G18.Fail*
5	*Line 702, column 29:* **Anchor element found with a valid href attribute, but no link content has been supplied.** `<a class="google-icon" href="http://plus.google.com/+cognifide"> </a>` *WCAG2AA.Principle4.Guideline4_1.4_1_2.H91.A.NoContent*
6	*Line 710, column 29:* **Anchor element found with a valid href attribute, but no link content has been supplied.** `<a class="linkedin-icon" href="https://www.linkedin.com/company/cognifide"> </a>` *WCAG2AA.Principle4.Guideline4_1.4_1_2.H91.A.NoContent*
7	*Line 718, column 29:* **Anchor element found with a valid href attribute, but no link content has been supplied.** `<a class="twitter-icon" href="https://twitter.com/cognifide"> </a>` *WCAG2AA.Principle4.Guideline4_1.4_1_2.H91.A.NoContent*

If no accessibility violations are found, result is marked as success (green).

Accessibility

Errors count: 0 .

Warnings count: 0 .

Notice count: 0 .

No.	Validation Output

If at least one accessibility violation of type Warning (or Notice if notices are not ignored) is found, report is marked as warning (yellow).

Accessibility

Errors count: 0.

Warnings count: 0.

Notice count: 52.

No.	Validation Output
1	*Line 7, column 1:* **Check that the title element describes the document.** `<title>AET Demo Page</title>` *WCAG2AAA.Principle2.Guideline2_4.2_4_2.H25.2*
2	*Line 18, column 5:* **Check that text of the link describes the purpose of the link.** `<a href="#" class="navbar-brand">Navbar</a>` *WCAG2AAA.Principle2.Guideline2_4.2_4_9.H30*
3	*Line 19, column 5:* **Check that a change of context does not occur when this input field receives focus.** `<button class="navbar-toggle" type="button" data-toggle="collapse"...ar"></span> <span class="icon-bar"></span> </button>` *WCAG2AAA.Principle3.Guideline3_2.3_2_1.G107*
4	*Line 25, column 27:* **Check that text of the link describes the purpose of the link.** `<a class="dropdown-toggle" data-toggle="dropdown" href="#" id="themes">Dropdown <span class="caret"></span></a>` *WCAG2AAA.Principle2.Guideline2_4.2_4_9.H30*

If at least one accessibility violation of type Error is found, report is marked as risk (red):

Accessibility

Errors count: 7.

Warnings count: 0.

Notice count: 0.

No.	Validation Output

1	*Line 35, column 5:* **Form does not contain a submit button (input type="submit", input type="image", or button type="submit").**
	`<form method="post" action="/" id="form">`
	`<div class="a...mit;`
	`theForm.onsubmit = WebForm_SaveScrollPositionOnSubmit;`
	`//]]>`
	`</script>`
	`</form>`
	WCAG2AA.Principle3.Guideline3_2.3_2_2.H32.2

2	*Line 362, column 13:* **This element has insufficient contrast at this conformance level. Expected a contrast ratio of at least 4.5:1, but text in this element has a contrast ratio of 4.32:1. Recommendation: change background to #fbfbfb.** `<div class="subtitle field-subtitle">We drive efficiency and agility into digital m...ital experiences while reducing time to market from months or weeks, to days or hours.</div>` *WCAG2AA.Principle1.Guideline1_4.1_4_3.G18.Fail*

3	*Line 615, column 1:* **Anchor element found with a valid href attribute, but no link content has been supplied.**
	`<a href="/newsandevents/news/zengage-wins-adobe-innovation-showcase-award/">`
	`<div style="text-align: center;"></div>`
	`</a>`
	WCAG2AA.Principle4.Guideline4_1.4_1_2.H91.A.NoContent

4	*Line 662, column 29:* **This element has insufficient contrast at this conformance level. Expected a contrast ratio of at least 4.5:1, but text in this element has a contrast ratio of 4.32:1. Recommendation: change background to #fbfbfb.** `<span style="font-family: ProximaNova-Light, Arial, Helvetica, sans-serif; font-size: 16px;...stify; background-color: rgb(246, 246, 246);">Connecting Customer Experiences</span>` *WCAG2AA.Principle1.Guideline1_4.1_4_3.G18.Fail*
5	*Line 702, column 29:* **Anchor element found with a valid href attribute, but no link content has been supplied.** `<a class="google-icon" href="http://plus.google.com/+cognifide"> </a>` *WCAG2AA.Principle4.Guideline4_1.4_1_2.H91.A.NoContent*
6	*Line 710, column 29:* **Anchor element found with a valid href attribute, but no link content has been supplied.** `<a class="linkedin-icon" href="https://www.linkedin.com/company/cognifide"> </a>` *WCAG2AA.Principle4.Guideline4_1.4_1_2.H91.A.NoContent*
7	*Line 718, column 29:* **Anchor element found with a valid href attribute, but no link content has been supplied.** `<a class="twitter-icon" href="https://twitter.com/cognifide"> </a>` *WCAG2AA.Principle4.Guideline4_1.4_1_2.H91.A.NoContent*

Risks

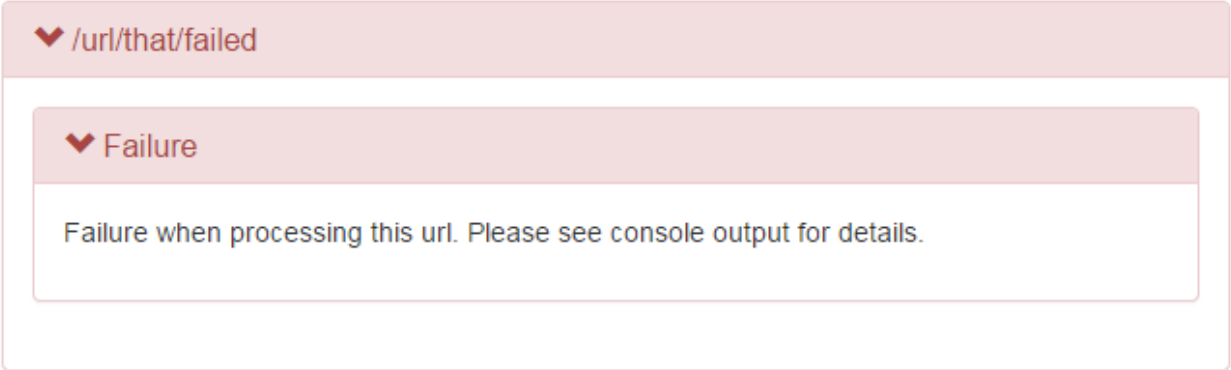
- When page fails accessibility tests it could mean that will find it difficult to access information, e.g. there are images on page without description (alt attribute), anchors elements does not have link content, page styling and design is not clear enough for people with sight disabilities.

6.3. Known issues and troubleshooting - 1.3

Problem: Rebasing from html report on Jenkis does not work.
Solution: Use html-report parameter `mode`.

Problem: Some screenshots are not visible in report preview (gray background instead).
Solution: Please refresh report page. Screenshots are not part of report and are downloaded on demand after report is opened. Downloading them may took some time.

Problem: Instead of report results for url, only part of url is visible and message about failure is present:



Solution: Please check console output for any errors that occurred. There will be hint what happened wrong with this url test.

Problem: Some screenshots on report are marked as risky (red) while when comparing screenshots without mask, no difference can be observed. Mask shows small areas marked as difference.
Solution: This problem is caused by system font-smoothing. Using bigger screen resolution than maximal system resolution (which is now 1024x768) causes rendering problems. Please refer to Known bugs and workarounds section and Screen Collector for more information.

6.4. Comments - 1.3

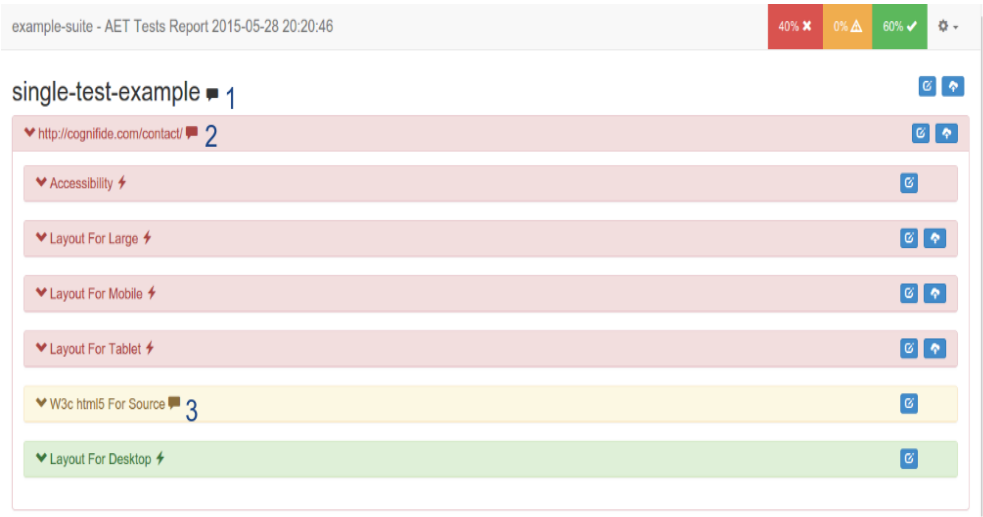
During test suite run all comments are being copied from latest report version for the same test suite.

Each comment is assigned to certain test suite run which means that editing comments on one version of report wont change comments on report from other run.

Comments are loaded at start of opening HTML report. This means that page refresh is needed to update comments changed by other user.

This section has the instructions to assist the user to use comments module on an HTML report.

Levels

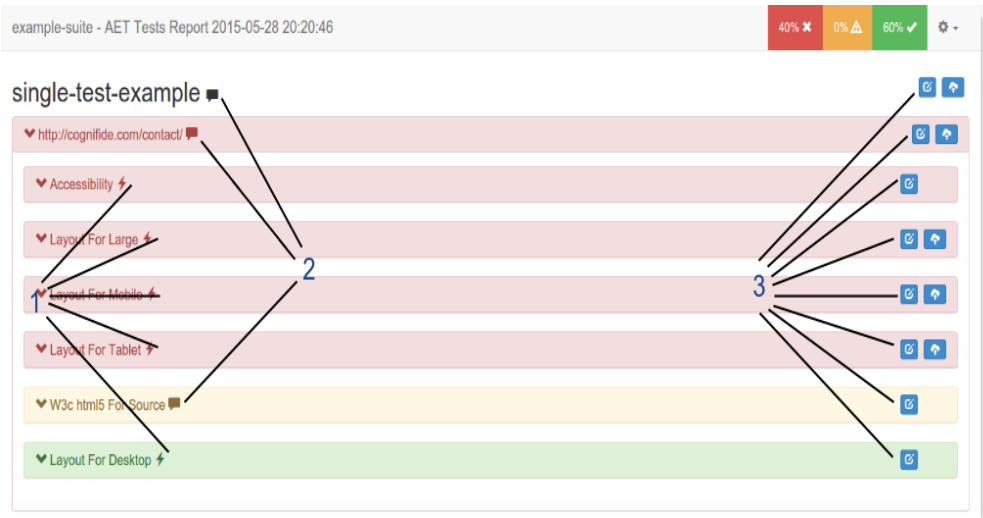


Comments module provides ability to create comment on 3 levels - for:

1. Test,
2. Url,
3. Test result.

Each comment level provides same features.

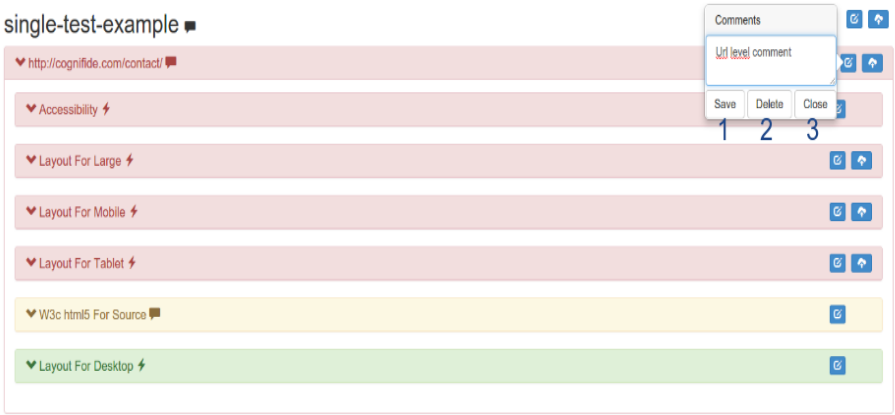
Navigation



1. Tooltip icon is missing when corresponding comment does not exist,
2. Tooltip icon is available when corresponding comment exists,
3. Comment icon - shows comment's form for the corresponding item and closes other comments' forms.

Form

Comment's form enables user to save and delete comment. Any one form is available at the same time.



1. Saves or overwrites (if already exists) comment. Successful action shows message:

   Comment saved succeesfully!

In case when comment was not saved, message:

2. Delete comment if exists. Successful action shows message:

In case when comment was not deleted, message:

3. Closes form.

7. Known bugs and workarounds - 1.3

This section contains known bugs and issues. Some of them have proposed workarounds presented below.

Issue

1 Some urls/links were not tested and only information about failure is present.

2 After hiding some element on the page layout with Hide Modifier, this element is still visible.

3 Sleep Modifier doesn't work for Screen Collector when changing resolution.

4 Adobe Flash plugin is not installed

Full report in Jenkins workspace has been downloaded but build status is marked as failed. In console following log ca

5
```
[xUnit] [INFO] - [JUnit] - No test report file(s) were found with the pattern
'target/xunit-report.xml' relative to 'c\:...' for the testing framework 'JUnit'.
Did you enter a pattern relative to the correct directory? Did you generate the result report(s)
[xUnit] [ERROR] - No test reports found for the metric 'JUnit' with the resolved pattern 'target/
Configuration error?.
```

6 Tests on AEM author instances are failing.

7 Test suite with more than 1000 urls fails.

8 There are some artifacts on my layout screen after collecting screenshot with changed resolution of browser.

While running test using Jenkins, job build fails and following exception is presented in console output:

9.
```
[ERROR] Failed to execute goal com.cognifide.aet:aet-maven-plugin:1.0.1:run (default-cli)
on project aet-stress-tests: Failed to save report: java.security.cert.CertificateException:
No subject alternative names present
```

10. Why trying to use rebase action from html report stored in Jenkins workspace, nothing happens and js console shows s

```
Mixed Content: The page at
'https://jenkins ... s/target/report-full.html'
was loaded over HTTPS, but requested an insecure XMLHttpRequest endpoint 'http://{REST_API_URL}..
This request has been blocked; the content must be served over HTTPS.
```

11. Many comparators with the same consumed resource type and module name, but different parameters. Results on rep every comparator.

```
<test name="QATest" useProxy="rest">
 <collect>
  <open/>
  <source/>
 </collect>
 <compare xmlns="http://www.cognifide.com/aet/compare/">
  <source comparator="w3c-html5" />
  <source comparator="w3c-html5" errors-only="false"/>
  <source comparator="w3c-html5" errors-only="true"/>
 </compare>
 <urls>
  <url href="http://www.onet.pl/" description="onet"/>
 </urls>
</test>
```

12. My page is not fully rendered although sleep duration is set on long duration.

/**/ .jira-issue { background: none repeat scroll 0 0 #F5F5F5; border: 1px solid #CCCCCC; border-radius: 3px; display: 0 0 0 2px; font-size: 8pt; line-height: 14px; } .jira-issue .jira-issue-key { padding-left: 2px; } .jira-issue .icon { height: 12p jira-macro-single-issue-export-pdf { line-height: 12px; padding-bottom: 1px; } .jira-status { padding-left:0; padding-botto .jira-issue .summary { padding:2px; } /**/ AETS-142 - Problem with Accordion component Open

/**/ .jira-issue { background: none repeat scroll 0 0 #F5F5F5; border: 1px solid #CCCCCC; border-radius: 3px; display: 0 0 0 2px; font-size: 8pt; line-height: 14px; } .jira-issue .jira-issue-key { padding-left: 2px; } .jira-issue .icon { height: 12p jira-macro-single-issue-export-pdf { line-height: 12px; padding-bottom: 1px; } .jira-status { padding-left:0; padding-botto .jira-issue .summary { padding:2px; } /**/ AETS-143 - Problem with changing resolution within the same test New