

Table of Contents

Introduction	0
Preconditions	1
Installation	2
Project	3
New project	3.1
Open project	3.2
Dashboard	3.3
Settings	3.4
Connector	3.5
Statistics	3.6
Configuration	4
Endpoints	4.1
Global Variables	4.2
Namespace Context	4.3
Data Dictionaries	4.4
Functions	4.5
Validation Matcher	4.6
Schema Repositories	4.7
Test management	5
Search tests	5.1
Open tests	5.2
Execute tests	5.3
Test design	5.4
Reporting	5.5
Further Reading	6

Citrus Admin UI

Welcome to the Citrus administration user interface

This is a web administration user interface for the integration test framework Citrus www.citrusframework.org written in Angular2. Major functionality objectives are project and configuration management as well as test execution and reporting.

NOTE: *This project is still in beta status and is still under construction!*

Stable features

Consider following features to be in a stable state:

- [Project](#)
 - [Open project](#)
 - [Dashboard](#)
 - [Settings](#)
- [Configuration](#)
 - [Endpoints](#)
 - [Global Variables](#)
 - [Namespace Context](#)
 - [Data Dictionaries](#)
 - [Functions](#)
 - [Validation Matcher](#)
 - [Schema Repositories](#)
- [Test management](#)
 - [Search tests](#)
 - [Open tests](#)

Unstable and heavily under construction

Consider following features to be under construction:

- [Project](#)
 - [New project](#)
 - [Statistics](#)
- [Test management](#)

citrus-admin

- [Execute tests](#)
- [Test design](#)
- [Reporting](#)

Please let us know if you are missing a feature and/or like to vote for features.

Preconditions

The administration UI is in early beta status and still under construction. Following from that we have to deal with some limitations and trade offs until the final version is released (hopefully end 2016, keep your fingers crossed!). However the administration UI is usable and some features are stable.

Please consider following limitations:

Java 8

The administration UI is implemented in Java 8. Following from that you need Java 8 to run it as web application on your machine.

Build tools

At the moment we are limited to supporting standard Citrus projects using Maven. Gradle or ANT projects are currently not supported. The build tool used is important because the administration UI is reading build information and uses Maven to execute test cases.

Citrus version

Your project should use Citrus version 2.0 or higher in order to be able to work with the administration UI. Earlier Citrus versions might work too but are not tested and will not get support with bugfixes.

Browsers

At this early state we do not support browsers other than Chrome. This does not mean that other browsers are not working with the administration UI but the features are not tested yet with other browsers. The Citrus development team is using Chrome so you can be sure that errors related to browser incompatibility will be fixed very soon for Chrome.

Citrus annotations

The administration UI is looking for all tests in your project. It is required that you use `@CitrusTest` and `@CitrusXmlTest` annotations on your test methods. Otherwise the test cases will not be found and displayed.

Citrus Java DSL

Citrus provides both XML and Java DSL for writing test cases. The administration UI is definitely able to read your XML test cases. The UI should also be able to read and manage your Java DSL test cases but this feature is not completely stable yet. There might be still some bugs and trade offs when reading Java DSL code. Please be curious and find out if the current working state is working with your project.

Installation

The Citrus administration UI is a web application that uses Spring boot and Angular2. First of all download the latest distribution which is a Java JAR file located at labs.consol.de/maven/repository:

Save the Java archive to a folder on your local machine and start the Spring boot web application. The downloaded JAR should be executable from command line like this:

```
java -jar citrus-admin-web-1.0.0-beta-2.jar
```

You will see the application starting up. Usually you will see some console log output. The web server should start within seconds. Once the application is up and running you can open your browser and point to '<http://localhost:8080>'.

That's it you are ready to use the Citrus administration UI. Next thing to do is to [open a project](#).

Project management

The basic objective of the administration UI is to manage your Citrus projects. The UI is able to open your project in order to view and edit your project configuration as well as your test cases.

You can do the following things regarding project management

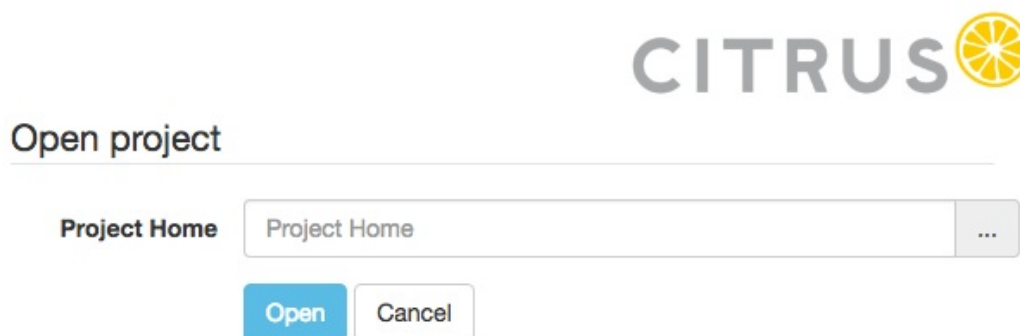
- [New project](#): Creates a new Citrus project from scratch.
- [Open project](#): Opens an existing Citrus project.
- [Dashboard](#): Shows all project related information such as test count, latest test run, configuration and so on.
- [Settings](#): Show and edit project settings such as folder layout, build types and versions.
- [Statistics](#): Shows common project statistics.

New project

Sorry, not ready yet!

Open project

You can open any Citrus project. The only prerequisite is that you have access to the project home on your local machine. When the administration UI is started you have to open a project first.



The project home selection form is displayed automatically when no project has been selected yet. You can preselect a project home when starting the administration UI by setting a system environment variable at startup:

```
java -Dproject.home=/Users/myaccount/path/tp/citrus/project/home -jar citrus-admin-web-1.0.0-beta-2.jar
```

When pre selecting a project home the project is opened automatically and the [project dashboard](#) is displayed. Now back to the project home selection if no project has been pre selected yet.

You need to specify the project home which is the root directory of your Citrus project. You can specify the complete path manually or pick the home directory over the file browser.

Select project home



Please select Citrus project home directory and confirm.

- Applications
- Desktop
- Documents
- Downloads
- Library
- Movies
- Music
- Pictures
- Projekte
- Public
- VirtualBox VMs

Select

Close

Once you have specified the project home you are ready to hit the **Open** button. Citrus will read the project information and open the [project dashboard](#). The administration UI is looking for several things in your project in order to gain information about the project. The files scanned are:

Path	Description
<code>\${project-home}/pom.xml</code>	Reads information from Maven POM
<code>\${project-home}/src/test/resources</code>	Reads XML test cases
<code>\${project-home}/src/test/resources/citrus-context.xml</code>	Reads Spring bean configuration
<code>\${project-home}/src/test/java</code>	Reads test cases

Customize project settings

It is possible that your project uses a different folder layout for test resources and test sources (e.g. `src/it/resources` and `src/it/java`). Then the project open operation will fail with errors. We can fix this by customizing the project settings manually in prior to opening the project.

There are two different approaches to customizing the project settings: First of all you can use system properties when starting the administration UI application:

citrus-admin

```
java -Dproject.home=/Users/myaccount/path/tp/citrus/project/home -  
Djava.source.directory=src/it/java -Dxml.source.directory=src/it/resources -jar citrus-admin-  
web-1.0.0-beta-2.jar
```

You can set the following system properties:

Property	Description
project.home	Preselect project on startup
root.directory	System root as base of all projects (default: user home directory)
java.source.directory	Java sources directory (default: <i>src/test/java</i>)
xml.source.directory	XML test sources directory (default: <i>src/test/resources</i>)
spring.application.context	Path to Spring application context file (default: <i>src/test/resources/citrus-context.xml</i>)

You can also use Spring boot properties, e.g. a custom server port:

```
java -Dserver.port=8181 -jar citrus-admin-web-1.0.0-beta-2.jar
```

A second approach would be to create a project settings file in your Citrus project root directory. The project settings are stored in a file called **citrus-project.json**. When you open a Citrus project for the first time the administration UI creates this project settings file automatically. But now we want to create this file manually in order to set custom directories and settings prior to opening the project. The setting file uses JSON data format and looks like this:

```
{
  "projectHome" : "~/Projects/Citrus/citrus-sample",
  "name" : "citrus-sample-project",
  "description" : "",
  "version" : "1.0.0",
  "testCount" : 0,
  "settings" : {
    "basePackage" : "com.consol.citrus",
    "citrusVersion" : "2.6",
    "springApplicationContext" : "src/it/resources/citrus-context.xml",
    "javaSrcDirectory" : "src/it/java/",
    "xmlSrcDirectory" : "src/it/resources/",
    "javaFilePattern" : "/*/*Test.java,/*/*IT.java",
    "xmlFilePattern" : "/*/*Test.xml,/*/*IT.xml",
    "build" : {
      "@class" : "com.consol.citrus.admin.model.build.maven.MavenBuildConfiguration",
      "type" : "maven",
      "properties" : [ ],
      "testPlugin" : "maven-failsafe-plugin",
      "profiles" : null
    }
  }
}
```

So you can force the administration UI to use these settings when opening the project. Just create the **citrus-project.json** file in the Citrus project home directory before opening the project.

Project dashboard

The dashboard gives you a quick overview of what your project looks like. Citrus reads information about your project such as name, package, description, test count, latest reports and so on.

The screenshot shows the Citrus Project Dashboard for a project named 'citrus-incident-test'. The dashboard has a blue header with navigation tabs: Citrus, Project (selected), Configuration, Tests, Statistics, and About. The main content area features a large title 'Project: citrus-incident-test' with a cube icon. Below the title, project details are listed: Projecthome, Name, Package, Version, and Description. An 'Edit settings' button is present. The dashboard is divided into three columns: Configuration, Tests, and Test results. The Configuration column lists various items like Endpoints, Schema Definitions, Functions, Validation Matcher, Data dictionaries, Namespaces, Message Validators, and Global Variables, with a 'Show configurations' button. The Tests column shows the project has 10 test cases and lists the latest tests, including IncidentManager_Http_Ok_2_IT, IncidentManager_Http_Ok_1_IT, IncidentManager_Http_IT, and several testIncidentManager_Http_Ok and testIncidentManager_Jms tests, with a 'Show all tests' button. The Test results column displays the date (05/20/2016), duration (14612), suite name (TestSuite), and a summary of test results: 0 Passed, 1 Failed, and 0 Skipped, with a 'Show test results' button.

Project: citrus-incident-test

Projecthome: /Users/christoph/Projekte/Citrus/citrus-samples/incident/citrus-test
Name: citrus-incident-test
Package: com.consol.citrus.samples
Version: 2.6-SNAPSHOT
Description: No project description available

[Edit settings](#)

Configuration

Citrus configuration items:

- » Endpoints
- » Schema Definitions
- » Functions
- » Validation Matcher
- » Data dictionaries
- » Namespaces
- » Message Validators
- » Global Variables

[Show configurations](#)

Tests

Your project has **10** test cases!

Latest tests:

- IncidentManager_Http_Ok_2_IT
- IncidentManager_Http_Ok_1_IT
- IncidentManager_Http_IT
 - testIncidentManager_Http_Ok_3
 - testIncidentManager_Http_Ok_4
 - testIncidentManager_Http_Schem...
 - testIncidentManager_Http_Field...
 - testIncidentManager_Http_Field...
- IncidentManager_Jms_IT
 - testIncidentManager_Jms_Ok_2
 - testIncidentManager_Jms_Schema...
 - testIncidentManager_Jms_Ok_1

[Show all tests](#)

Test results

Date: 05/20/2016
Duration: 14612
Suite Name: TestSuite

✓ Passed: 0
✗ Failed: 1
⏸ Skipped: 0


Total Tests: 1

[Show test results](#)

The project dashboard is a good starting point to discover your project with all [projects](#) [settings](#) and [test cases](#).

Project settings

Each Citrus project has properties and settings that influence the administration UI. These properties are project names, descriptions, versions and source folders. You can review and change these project related settings with an HTML form on the project settings page.


Project Settings & Information

Home

/Users/christoph/Projekte/Citrus/citrus-samples/incident/citrus-test

Name *

citrus-incident-test

Version *

1.0.0

Description

Settings:

Citrus *

2.6-SNAPSHOT

Java Sources *

src/test/java/

/**/*.Test.java,/**/*.IT.java

XML Sources *

src/test/resources/

/**/*.Test.xml,/**/*.IT.xml

Spring Application Context *

src/test/resources/citrus-context.xml

Package *

com.consol.citrus.samples

Build:

BuildType *

maven

TestPlugin

maven-failsafe-plugin

Profiles

Properties

Name

Value

Add

embedded = true

Save settings

Some project settings are read only at the moment, e.g. we do not support renaming of projects yet. If you want to rename a project or change the project version you need to do this manually in the Maven POM configuration.

If you save the project settings the administration UI will save the changes to the project settings file **citrus-project.json** which is located in your project home directory. This file uses the JSON syntax and looks like follows:

```

{
  "projectHome" : "~/Projects/Citrus/citrus-sample",
  "name" : "citrus-sample-project",
  "description" : "",
  "version" : "1.0.0",
  "settings" : {
    "basePackage" : "com.consol.citrus",
    "citrusVersion" : "2.6",
    "springApplicationContext" : "src/it/resources/citrus-context.xml",
    "javaSrcDirectory" : "src/test/java/",
    "xmlSrcDirectory" : "src/test/resources/",
    "javaFilePattern" : "/*/*Test.java,/*/*IT.java",
    "xmlFilePattern" : "/*/*Test.xml,/*/*IT.xml",
    "useConnector" : true,
    "build" : {
      "@class" : "com.consol.citrus.admin.model.build.maven.MavenBuildConfiguration",
      "type" : "maven",
      "properties" : [ ],
      "testPlugin" : "maven-failsafe-plugin",
      "profiles" : null
    }
  }
}

```

General settings

Each Citrus project works with Java classes and resources. These files are located in project folders inside the Maven project. Citrus admin is working with these defaults:

- **src/test/java/** folder for Java test classes
- **src/test/resources/** folder for test resources (e.g. configuration files)
- **/*/*Test.java,/*/*IT.java** file pattern for Java test classes
- **/*/*Test.xml,/*/*IT.xml** file pattern for XML test cases

You can customize these settings according to your project setup.

Build configuration

The administration web UI is able to execute tests. This test execution is done by calling the Maven build lifecycle for the opened project. You can adjust the build settings accordingly. By default Citrus admin uses the **maven-failsafe-plugin** to execute the Citrus tests. This causes Citrus to call

```
> mvn failsafe:integration-test
```

This executes all Citrus test cases. You can change this to **maven-surefire-plugin** so the Maven command looks like this:

```
> mvn test
```

In case you need to activate Maven profiles during the build you need to add those profiles to the build settings. Each profile name that you save to the build settings will result in some command line argument for the Maven build like this:

```
> mvn failsafe:integration-test -PmyProfile
```

Also when some system properties should be set during the Maven build you can add those properties to the build settings, too. This results in command line arguments for the Maven command:

```
> mvn failsafe:integration-test -DmyProperty=value
```

This is how to customize the Maven build that executes the Citrus tests in a project.

Admin connector library

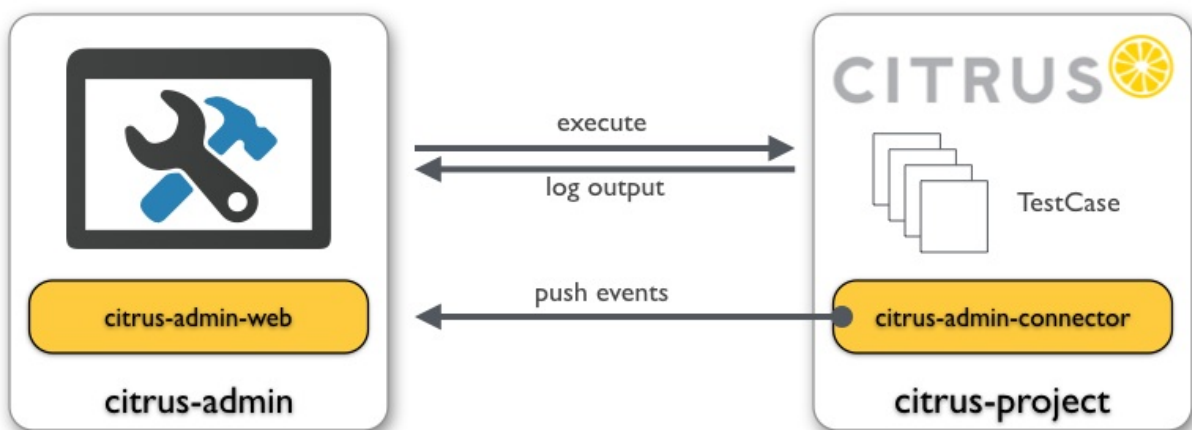
In the previous chapter we have seen how to customize the project and build settings for the active project. Now when the administration UI executes some Citrus tests we can make use of a special admin-connector library that provides detailed information on the test run and its outcome.

The admin UI tries to get test run information by parsing the logging output. We can extend this mechanism by adding a connector library as dependency to the target project.

In general this is a Maven dependency that is added to the target project:

```
<dependency>
  <groupId>com.consol.citrus</groupId>
  <artifactId>citrus-admin-connector</artifactId>
  <version>${citrus.admin.version}</version>
</dependency>
```

Basically this little helper library provides detailed information during the test run by pushing events to the admin UI.



The citrus-admin-connector is part of the Citrus project and is automatically loaded when tests are run. The connector library registers special test listeners and pushes information to the citrus-admin web application. This way the admin UI is able to display runtime information of the tests such as exchanged messages, test results and so on.

You can automatically activate/deactivate the connector library in the project settings. This will automatically add or remove the citrus-admin-connector Maven dependency for you.

Configuration

The Citrus components such as endpoints, variables, functions, schemas and dictionaries are configured in a Spring application context. The administration UI is able to read and change the Citrus components configuration. Each component category is represented in the configuration page.

Configuration Application context configuration

Endpoints	Schema Definitions	Global Variables	Functions	Validation Matcher	Data Dictionaries	Namespace Context
-----------	--------------------	------------------	-----------	--------------------	-------------------	-------------------

You can do show/edit the following configuration:

- [Endpoints](#)
- [Global Variables](#)
- [Namespace Context](#)
- [Data Dictionaries](#)
- [Functions](#)
- [Validation Matcher](#)
- [Schema Repositories](#)

Endpoint configuration

Endpoints are essential in a Citrus project. They define client and server components as well as producer and consumer for different message transports.

Endpoints

Schema Definitions

Global Variables

Functions

Validation Matcher

Data Dictionaries

Namespace Context

Endpoints


New ▾


Endpoint components create producer and consumers for sending and receiving messages to/from various endpoint destinations. Endpoint components can be of synchronous and asynchronous nature and get referenced in test cases when sending and receiving messages.


http-server	name: networkBackendHttpServer port: 18002 auto-start: true root-parent-context: true interceptors: serverInterceptors timeout: 10000	✕
jms	name: fieldForceOrderEndpoint destination-name: JMS.Citrus.v1.FieldForceOrder pub-sub-domain: false timeout: 5000	✕
jms	name: fieldForceNotificationEndpoint destination-name: JMS.Citrus.v1.FieldForceNotification pub-sub-domain: false timeout: 5000	✕
ws-client	name: incidentHttpClient request-url: http://localhost:18001/incident/IncidentManager/v1 interceptors: clientInterceptors fault-strategy: THROWS_EXCEPTION timeout: 5000	✕
ws-server	name: smsGatewayServer port: 18005 auto-start: true root-parent-context: true handle-mime-headers: true timeout: 10000	✕


First of all the list of all available endpoints in the project is displayed. Each endpoint represents a message transport such as SOAP, JMS, REST, Mail, FTP and so on. You can add new endpoints using the *New* context menu on the right. You need to chose the endpoint type first. Then a HTML form is displayed holding the endpoint settings.


New ▾


 camel


 channel


 ftp-client


 ftp-server


 http-client


 http-server


 jms


 jmx-client


 jmx-server


 mail-client


 mail-server


 rmi-client


 rmi-server


 ssh-client


 ssh-server

 vertx

 ws-client








 ws-server

 websocket-client

 websocket-server

New Endpoint

Type * http-client

Name *	<input type="text"/>
RequestUri	<input type="text"/>
RequestMethod	POST 
ErrorStrategy	propagateError 
PollingInterval	500
MessageCorrelator	<input type="text"/> 
MessageConverter	<input type="text"/> 
RequestFactory	<input type="text"/> 
RestTemplate	<input type="text"/> 
Charset	<input type="text"/>
ContentType	<input type="text"/>
Interceptors	<input type="text"/>
Timeout	5000
TestActor	<input type="text"/> 

Click save to add the new endpoint. Citrus is working with Spring XML configuration files. This means that the new endpoint component is saved as XML Spring bean to the basic Spring application context file. Usually this is a file located in *src/test/resources/citrus-context.xml* in your project. After you have saved the new component you will see that a new entry has been added to this file.








You can also edit endpoint components in the administration UI. Just click an an existing endpoint component and you will see the HTML form with all the settings to this endpoint.

Endpoints

[New ▾](#)

Endpoint components create producer and consumers for sending and receiving messages to/from various endpoint destinations. Endpoint components can be of synchronous and asynchronous nature and get referenced in test cases when sending and receiving messages.

Edit Endpoint

Type *	ws-client
Name *	incidentHttpClient
RequestUrl	http://localhost:18001/incident/IncidentManager/v1
WebServiceTemplate	<input type="text"/> 
MessageFactory	<input type="text"/> 
MessageSender	<input type="text"/>
MessageCorrelator	<input type="text"/> 
Interceptors	clientInterceptors
EndpointResolver	<input type="text"/> 
MessageConverter	<input type="text"/> 
FaultStrategy	<input type="text"/> 
PollingInterval	<input type="text"/>
Timeout	5000
TestActor	<input type="text"/> 

[Save](#) [Close](#)

If you save the changes Citrus will again change the Spring bean component in the XML configuration file. You can manually review the changes made. All manual changes in the Spring application context will also affect the administration UI. Just hit the reload button in your browser to reload the configuration.

Test management

Managing all your tests is a basic thing to do with the administration UI. You are able to search for tests, open tests and execute tests with the UI.

You can do the following things regarding test management

- [Search tests](#)
- [Open tests](#)
- [Execute tests](#)
- [Test design](#)
- [Reporting](#)

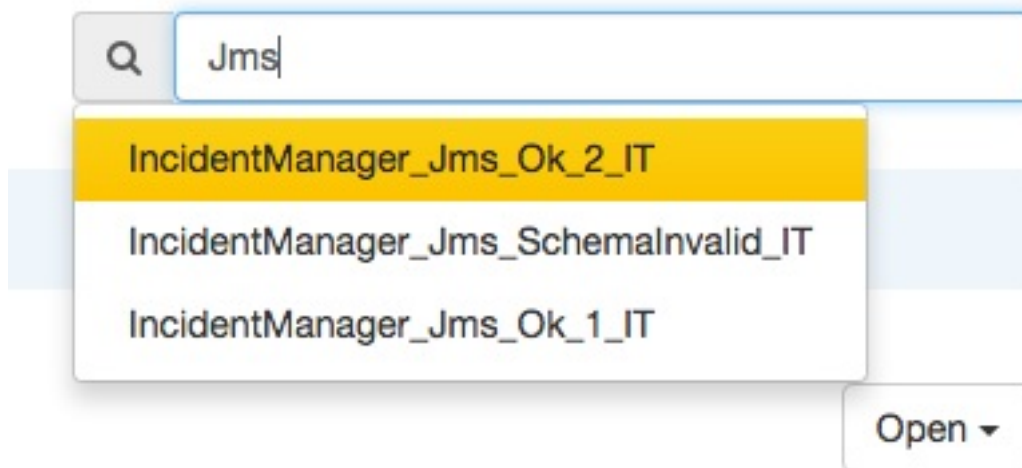
Search tests

Searching for test cases is very simple. Open the tests page and you will see a search form input field at the top right.



Citrus administration UI will search for tests in your project. These includes XML and Java DSL test cases. The search is a full text search on all test names available. So the result is always a matching list of tests.

You can open the test by selecting the test from the result list.



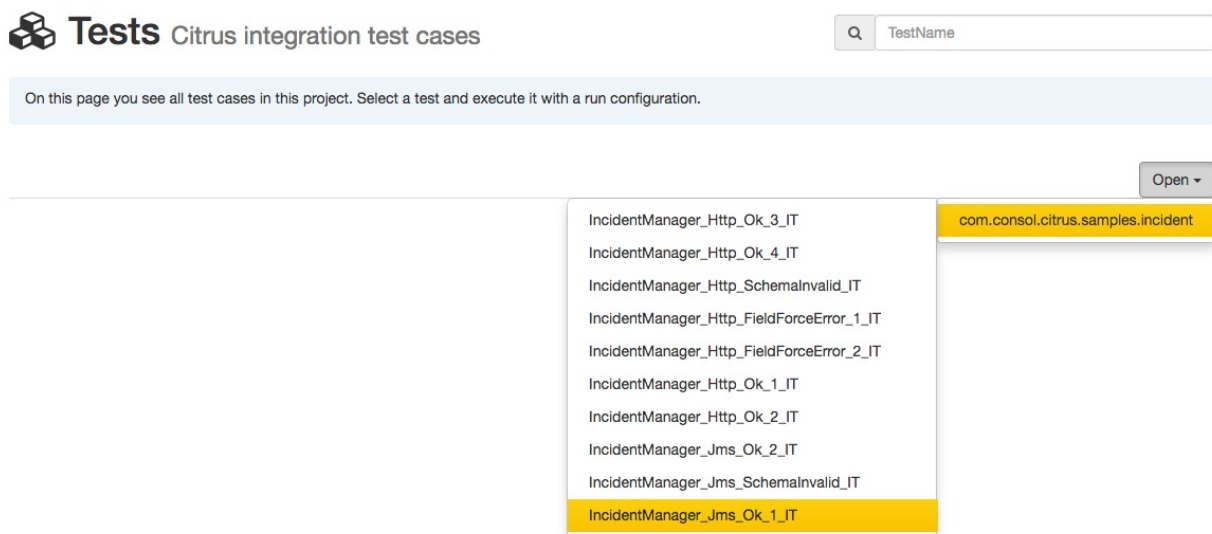
The test is opened and displayed in [test detail](#) panel.

Open tests

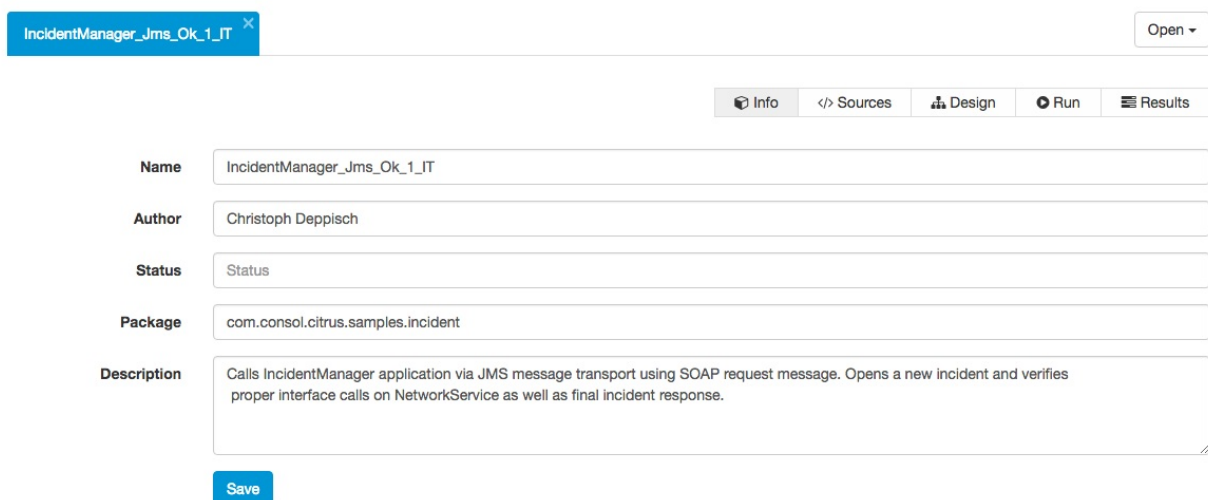
The administration UI is able to open your test cases. Both XML and Java DSL test cases are supported.

NOTE: Java DSL test cases might cause some problems when loading the test design view. this is because we do have to make the Java DSL code interpretation more stable.

You can open the tests using the *Open* context menu. All available test cases are grouped by their package.



Choose a test and open it in order to see the basic test case information such as name, author, status and description.



Next to the test case info you can view the test source code. This is either a XML test or the Java DSL code.

IncidentManager_Jms_Ok_1_IT
Open

Info
Sources
Design
Run
Results

Xml
Java

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <spring:beans xmlns="http://www.citrusframework.org/schema/testcase"
3     xmlns:jms="http://www.citrusframework.org/schema/jms/testcase"
4     xmlns:ws="http://www.citrusframework.org/schema/ws/testcase"
5     xmlns:im="http://www.citrusframework.org/schema/samples/IncidentManager/v1"
6     xmlns:net="http://www.citrusframework.org/schema/samples/NetworkService/v1"
7     xmlns:spring="http://www.springframework.org/schema/beans" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
8     xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd
9     http://www.citrusframework.org/schema/jms/testcase http://www.citrusframework.org/schema/jms/testcase/citrus-jms-testcase.xsd
10    http://www.citrusframework.org/schema/ws/testcase http://www.citrusframework.org/schema/ws/testcase/citrus-ws-testcase.xsd
11    http://www.citrusframework.org/schema/testcase http://www.citrusframework.org/schema/testcase/citrus-testcase.xsd">
12
13 <testcase name="IncidentManager_Jms_Ok_1_IT">
14 <meta-info>
15 <author>Christoph Deppisch</author>
16 <creationdate>2014-09-28</creationdate>
17 <status>FINAL</status>
18 <last-updated-by>Christoph Deppisch</last-updated-by>
19 <last-updated-on>2014-09-28T00:00:00</last-updated-on>
20 </meta-info>
21
22 <description>Calls IncidentManager application via JMS message transport using SOAP request message. Opens a new incident and verifies
23 proper interface calls on NetworkService as well as final incident response.</description>
24
25 <variables>
26 <variable name="ticketId" value="citrus:randomUUID()"/>
27 <variable name="customerId" value="citrus:randomNumber(6)"/>
28 </variables>
29
30 <actions>
31 <echo>
32 <message>Step 1: Send OpenIncident request message to IncidentManager via Http SOAP interface</message>
33 </echo>
34
35 <send endpoint="incidentJmsEndpoint" fork="true">
36 <message>
37 <payload>
38 <im:OpenIncident xmlns:im="http://www.citrusframework.org/schema/samples/IncidentManager/v1">
39 <im:incident>
40 <im:ticketId>${ticketId}</im:ticketId>
41 <im:captured>citrus:currentDate('yyyy-MM-dd'T'00:00:00')</im:captured>
42 <im:state>NEW</im:state>
43 <im:component>SOFTWARE</im:component>
44 <im:description>Something went wrong with the software!</im:description>
45 </im:incident>
46 <im:customer>
47 <im:id>${customerId}</im:id>
48 <im:firstname>Christoph</im:firstname>
49 <im:lastname>Deppisch</im:lastname>
50 <im:address>Franziskanerstr. 38, 80995 München</im:address>

```

At the moment these information is read only. Stay tuned for some code editing features that might come in the future. Another representation of the test sources is the [test design](#) view. The design view brings the test actions to a graphical representation.

Now a very powerful feature is to execute the test using the administration UI. Let's go and read about [test execution](#).

Execute tests

Test execution is a very powerful feature as it enables you to execute your tests within a browser environment almost everywhere. Just hit the *Run* button and Citrus will start a new background process that executes the test case immediately. At the moment we do only support Maven test execution. This means that a new Maven process is launched in background executing the test.

IncidentManager_Jms_Ok_1_IT Open ▾

Info </> Sources Design **Run** Results

▶ Run 100%

Console Messages

```

12:53:41,922 INFO citrus.Citrus| TEST SUCCESS IncidentManager_Jms_Ok_1 test case IncidentManager_Jms_Ok_1 (com.consolix.citrus.samples.incident)
12:53:41,923 INFO citrus.Citrus|
12:53:41,969 INFO citrus.Citrus|
12:53:41,969 INFO citrus.Citrus|
12:53:41,969 DEBUG citrus.Citrus| AFTER TEST SUITE
12:53:41,969 INFO citrus.Citrus|
12:53:41,969 INFO citrus.Citrus|
12:53:41,969 INFO citrus.Citrus| AFTER TEST SUITE: SUCCESS
12:53:41,971 INFO citrus.Citrus|
12:53:41,971 INFO citrus.Citrus|
12:53:41,972 INFO citrus.Citrus|
12:53:41,972 INFO citrus.Citrus| CITRUS TEST RESULTS
12:53:41,973 INFO citrus.Citrus| IncidentManager_Jms_IT.testIncidentManager_Jms_Ok_1 ..... SUCCESS
12:53:41,973 INFO citrus.Citrus|
12:53:41,973 INFO citrus.Citrus| TOTAL: 1
12:53:41,973 DEBUG citrus.Citrus| SKIPPED: 0 (0.0%)
12:53:41,973 INFO citrus.Citrus| FAILED: 0 (0.0%)
12:53:41,973 INFO citrus.Citrus| SUCCESS: 1 (100.0%)
12:53:41,974 INFO citrus.Citrus|
12:53:41,974 DEBUG report.HtmlReporter| Generating HTML test report
12:53:42,009 INFO report.HtmlReporter| Generated HTML test report
Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 4.636 sec - in TestSuite
12:53:42,105 INFO /| Destroying Spring FrameworkServlet 'smsGatewayServer-servlet'
12:53:42,109 INFO /| Destroying Spring FrameworkServlet 'networkBackendHttpServer-servlet'

Results :

Tests run: 1, Failures: 0, Errors: 0, Skipped: 0

[WARNING] File encoding has not been set, using platform encoding UTF-8, i.e. build is platform dependent! The file encoding for reports output files should
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 19.966 s
[INFO] Finished at: 2016-05-13T12:53:42+02:00
[INFO] Final Memory: 17M/131M
[INFO] -----
process completed successfully

```

As you can see the test log output is forwarded to your browser. Also the test progress and result (success or failure) is tracked by the administration UI. In the messages tab you are able to review all messages (inbound/outbound) that have been exchanged during the test run.

The screenshot shows the Citrus Admin interface. At the top, there's a tab labeled 'IncidentManager_Jms_Ok_1_IT' with a close button. Below it, a toolbar contains buttons for 'Info', 'Sources', 'Design', 'Run', and 'Results'. A 'Run' button is also present on the left. A green progress bar indicates 100% completion. Below the progress bar, there are tabs for 'Console' and 'Messages'. The main area is divided into 'Outbound' and 'Inbound' sections. The 'Inbound' section is active, showing a message details panel. The message content is an XML payload for an 'OpenIncident' with details like ticketId, state, component, and customer information. Below the message content, there's a button with an envelope icon and a double arrow. The bottom section shows the HTTP headers and the start of the XML body for an 'AnalyseIncident' message.

The message panel displays all inbound and outbound messages. Click on the envelope to see the message content details. The mechanism for tracking inbound and outbound messages in the administration UI is working with a special message listener that you have to add to your Citrus project:

```
<bean class="com.consol.citrus.admin.connector.WebSocketPushMessageListener">
  <property name="host" value="localhost"/>
  <property name="port" value="8080"/>
</bean>
```

As you can see the connector is pushing message data to the administration UI using a WebSocket API on the administration UI server. The *host* and *port* properties are customizable, default values are *localhost* and *8080*. When a test is executed the message listener will automatically connect and push messages exchanged to the administration UI.

Of course the administration UI server has to be accessible during the test run. The message listener will automatically test the server connectivity at the beginning of the test run. In case the administration UI is not accessible the message push feature is

simply disabled. So you can continue to work with your Citrus project even if the administration UI is not started.

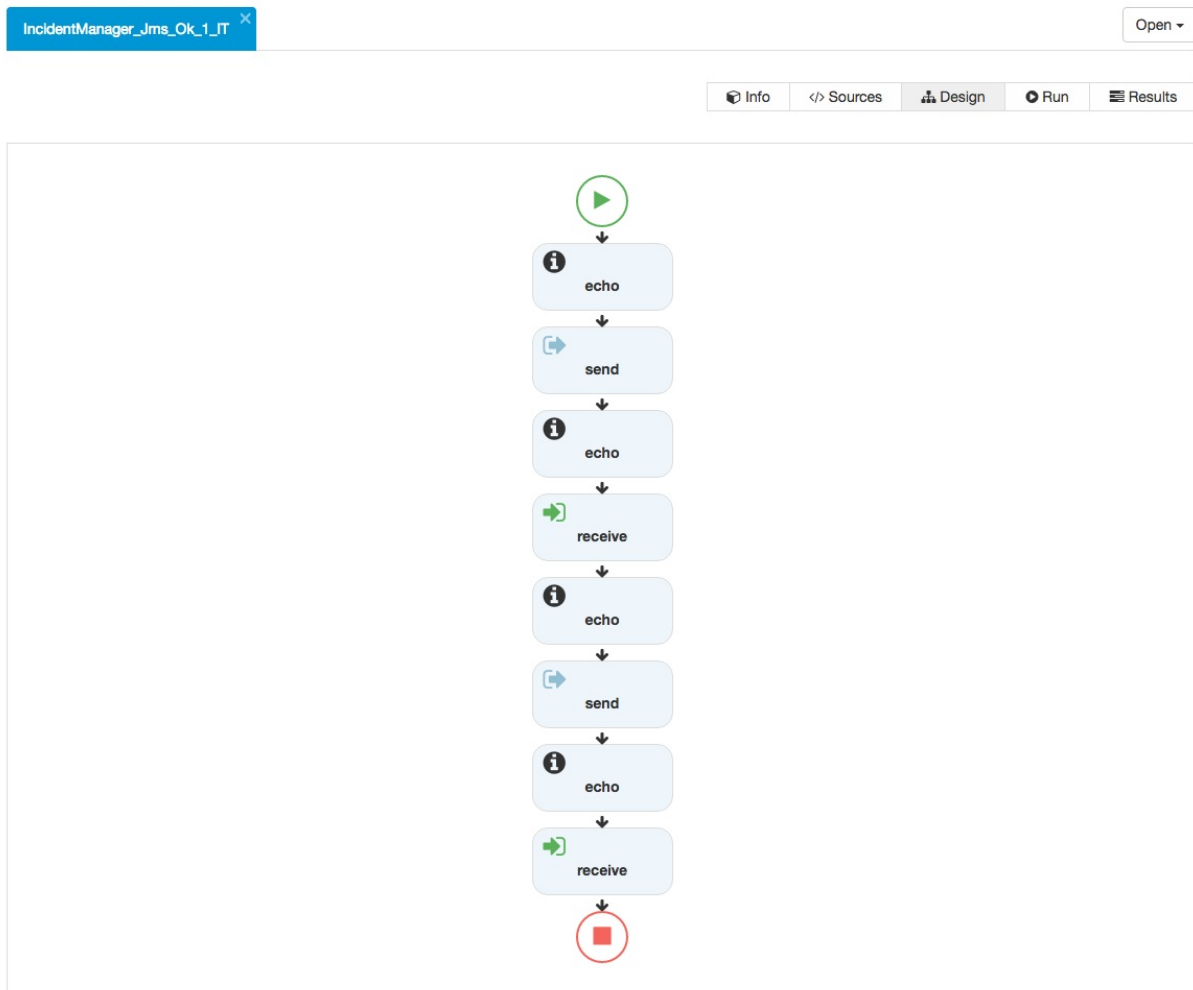
Do not forget to add the **citrus-admin-connector** as dependency to the Citrus project Maven POM:

```
<dependency>
  <groupId>com.consol.citrus</groupId>
  <artifactId>citrus-admin-connector</artifactId>
  <version>1.0.0-beta-2</version>
</dependency>
```

This is how the administration UI is able to track messages exchanged during a test run. Stay tuned for more features related to the test execution and message exchange.

Test design

The next feature is quite experimental. In the test design view Citrus tries to give you a graphical representation of the test actions in your test. Each test action is displayed as a graphical node. If you enter the action with the mouse or if you click on a test action node some more details are displayed.



NOTE: The test design view is not complete for all test actions. Some actions may not be displayed or may have limited display.

Further reading

- The [Citrus reference manual](#) gives you a detailed description of all Citrus features.
- Examples:
 - [Sample projects](#)
 - [Devoxx '15 sample project](#) with Microservices and Docker
 - [JavaLand '16 sample project](#) with Arquillian
- [ChangeLog](#) shows the release history.
- [Contributing](#) explains how you can contribute to this project. Pull requests are highly appreciated!