Tests

Table of Contents

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

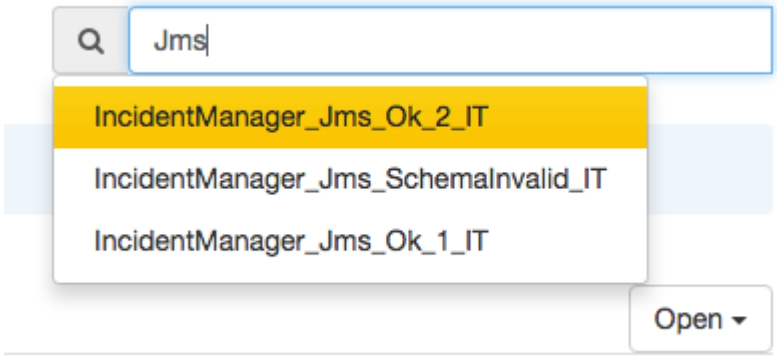
- Search tests
- List tests
- Open tests
- Execute tests
- Test designer
- Reporting

Search tests

Searching for test cases is very simple. Open the tests page and you will se 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.

List tests

Your project usually contains a lot of tests. All found tests are listed by their class and method. Also you can filter the displayed tests by the Java package. Each test can be opened in a editor for exploring further test details.

est cases

TestName

	Class	Method
	IncidentManager_Http_IT	testIncidentManager_Http_Ok_3
	IncidentManager_Http_IT	testIncidentManager_Http_Ok_4
lid_IT	IncidentManager_Http_IT	testIncidentManager_Http_Schema
ror_1_IT	IncidentManager_Http_IT	testIncidentManager_Http_FieldFo
ror_2_IT	IncidentManager_Http_IT	testIncidentManager_Http_FieldFo
	IncidentManager_Http_Ok_1_IT	testIncidentManager_Http_Ok_1
	IncidentManager_Http_Ok_2_IT	testIncidentManager_Http_Ok_1
	IncidentManager_Jms_IT	testIncidentManager_Jms_Ok_2
lid_IT	IncidentManager_Jms_IT	testIncidentManager_Jms_Schema
	IncidentManager_Jms_IT	testIncidentManager_Jms_Ok_1

est cases

TestName

	Class	Method
	IncidentManager_Http_IT	testIncidentManager_Http_Ok_3
	IncidentManager_Http_IT	testIncidentManager_Http_Ok_4
lid_IT	IncidentManager_Http_IT	testIncidentManager_Http_Schema
ror_1_IT	IncidentManager_Http_IT	testIncidentManager_Http_FieldFo
ror_2_IT	IncidentManager_Http_IT	testIncidentManager_Http_FieldFo
	IncidentManager_Http_Ok_1_IT	testIncidentManager_Http_Ok_1
	IncidentManager_Http_Ok_2_IT	testIncidentManager_Http_Ok_1
	IncidentManager_Jms_IT	testIncidentManager_Jms_Ok_2
lid_IT	IncidentManager_Jms_IT	testIncidentManager_Jms_Schema
	IncidentManager_Jms_IT	testIncidentManager_Jms_Ok_1

TestName

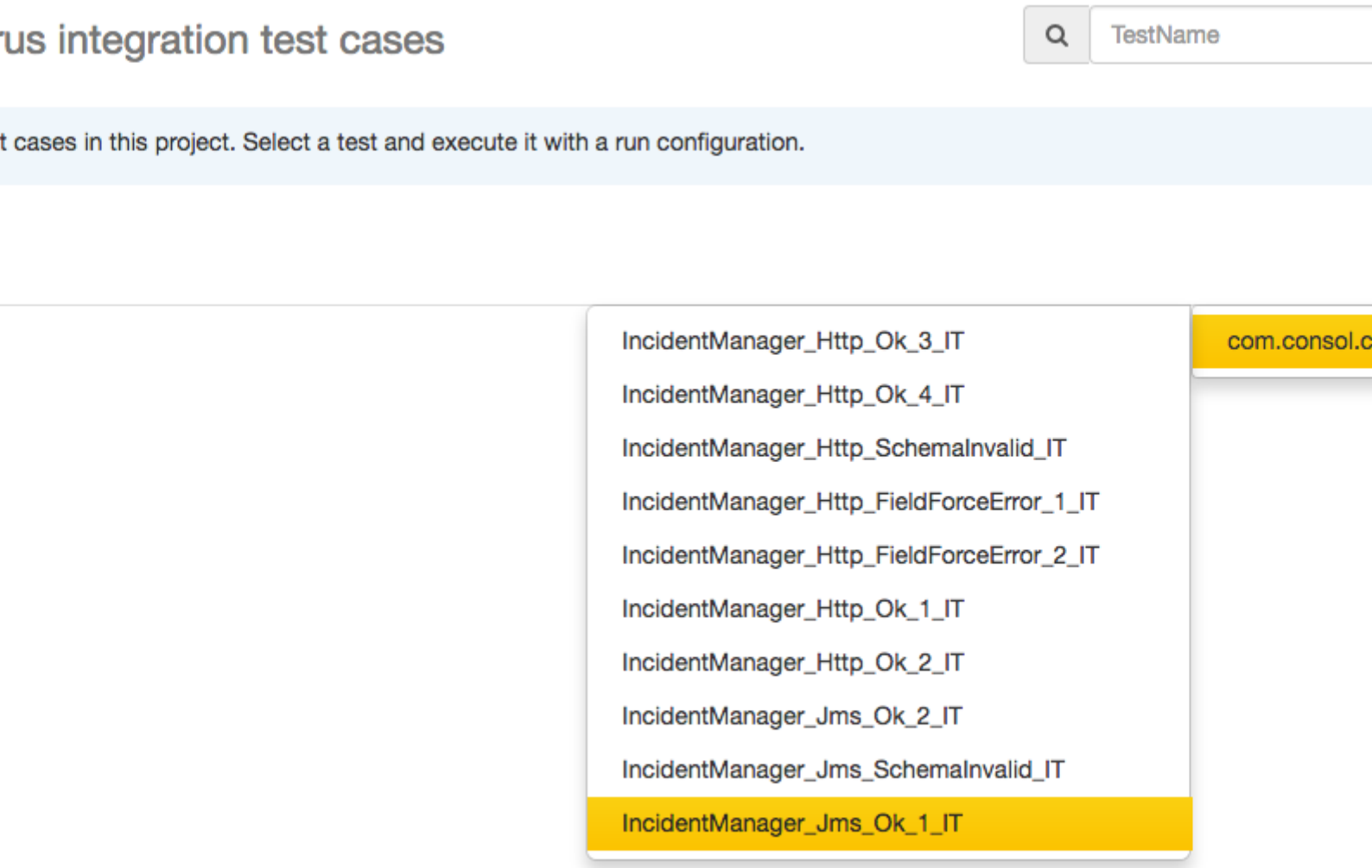
	Class	Method
	IncidentManager_Http_IT	testIncidentManager_Http_Ok_3
	IncidentManager_Http_IT	testIncidentManager_Http_Ok_4
	IncidentManager_Http_IT	testIncidentManager_Http_SchemaInvalid
	IncidentManager_Http_IT	testIncidentManager_Http_FieldForceError_1
	IncidentManager_Http_IT	testIncidentManager_Http_FieldForceError_2
	IncidentManager_Http_Ok_1_IT	testIncidentManager_Http_Ok_1
	IncidentManager_Http_Ok_2_IT	testIncidentManager_Http_Ok_1
	IncidentManager_Jms_IT	testIncidentManager_Jms_Ok_2
	IncidentManager_Jms_IT	testIncidentManager_Jms_SchemaInvalid
	IncidentManager_Jms_IT	testIncidentManager_Jms_Ok_1

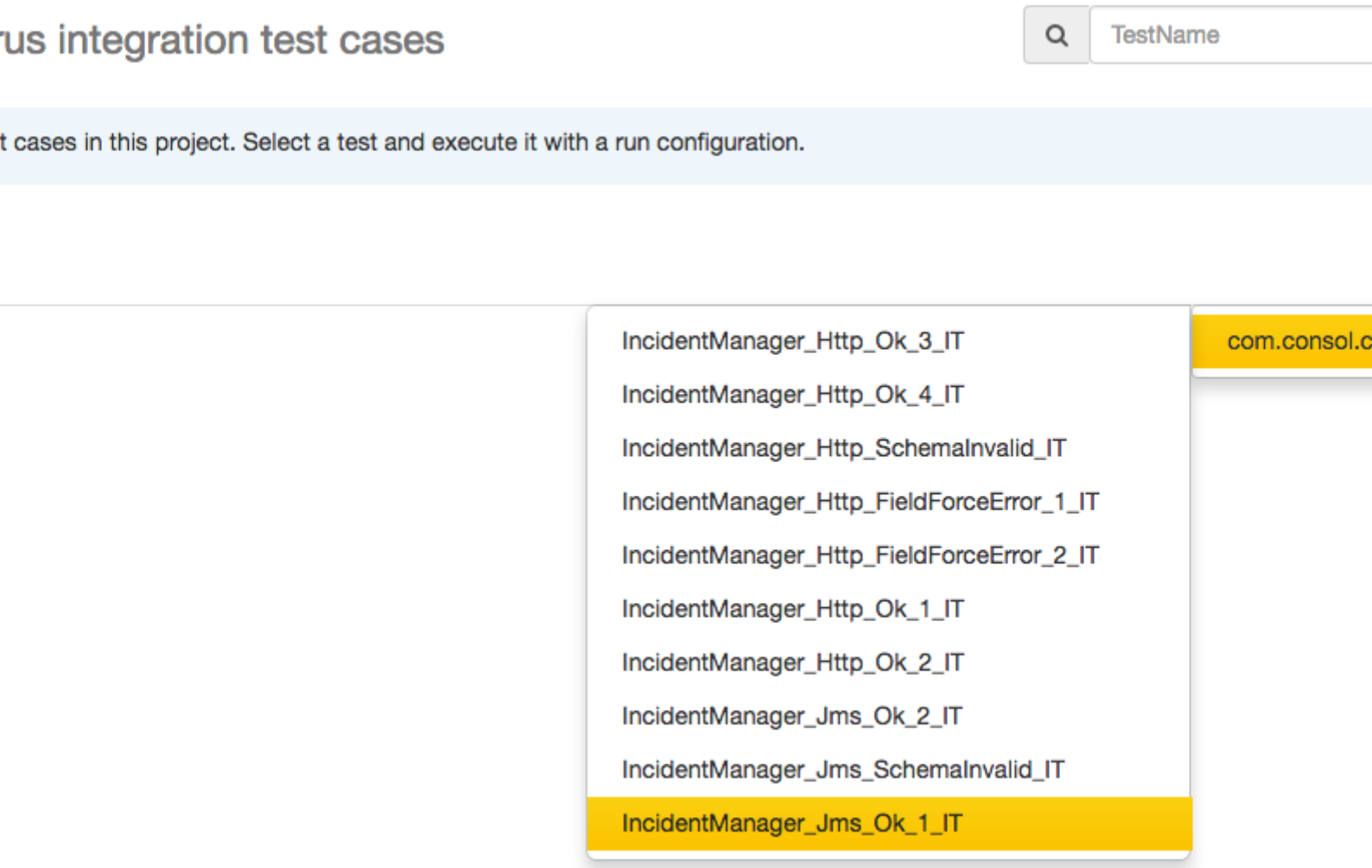
Editor

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

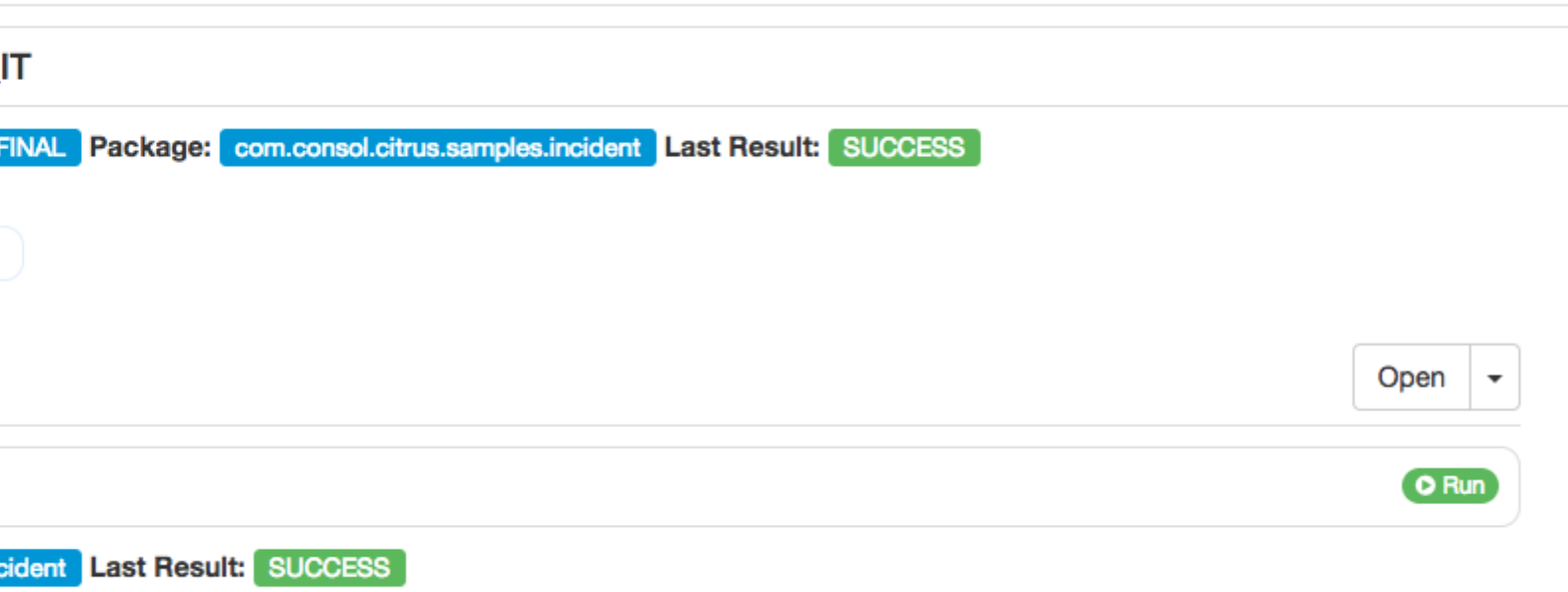
NOTE: Java DSL test cases might cause some problems when loading the test design view. this is because we do have to make the Java DSL code interpretation more stable.

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


rus integration test cases
TestName
t cases in this project. Select a test and execute it with a run configuration.
IncidentManager_Http_Ok_3_IT
IncidentManager_Http_Ok_4_IT
IncidentManager_Http_SchemaInvalid_IT
IncidentManager_Http_FieldForceError_1_IT
IncidentManager_Http_FieldForceError_2_IT
IncidentManager_Http_Ok_1_IT
IncidentManager_Http_Ok_2_IT
IncidentManager_Jms_Ok_2_IT
IncidentManager_Jms_SchemaInvalid_IT
IncidentManager_Jms_Ok_1_IT
com.consol.c



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



Next to to the test case info you can view the test source code.

```
r or authors.

sion 2.0 (the "License");
mpliance with the License.
t

ENSE-2.0

greed to in writing, software
ibuted on an "AS IS" BASIS,
Y KIND, either express or implied.
age governing permissions and


t;

usXmlTest;
estNGCitrusTest;




 extends AbstractTestNGCitrusTest {


r_Http_Ok_1_IT")
Ok_1() {
```

```
r or authors.

sion 2.0 (the "License");
mpliance with the License.
t

ENSE-2.0

greed to in writing, software
ibuted on an "AS IS" BASIS,
Y KIND, either express or implied.
age governing permissions and



t;

usXmlTest;
estNGCitrusTest;




 extends AbstractTestNGCitrusTest {


r_Http_Ok_1_IT")
Ok_1() {
```

```
r or authors.

sion 2.0 (the "License");
mpliance with the License.
t

ENSE-2.0

greed to in writing, software
ibuted on an "AS IS" BASIS,
Y KIND, either express or implied.
age governing permissions and



t;

usXmlTest;
estNGCitrusTest;




 extends AbstractTestNGCitrusTest {


r_Http_Ok_1_IT")
Ok_1() {
```

If your test uses the XML DSL to describe the test actions you can also view the XML sources.

```
ework.org/schema/testcase"
framework.org/schema/jms/testcase"
framework.org/schema/ws/testcase"
framework.org/schema/samples/IncidentManager/v1"
framework.org/schema/samples/NetworkService/v1"
ringframework.org/schema/beans" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
ww.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd
rk.org/schema/jms/testcase http://www.citrusframework.org/schema/jms/testcase/citrus-jms-testcase.xsd
rk.org/schema/ws/testcase http://www.citrusframework.org/schema/ws/testcase/citrus-ws-testcase.xsd
rk.org/schema/testcase http://www.citrusframework.org/schema/testcase/citrus-testcase.xsd">

1_IT">


te>

/last-updated-by>
</last-updated-on>


lication via Http message transport using SOAP request message. Opens a new incident and verifies
ce as well as final incident response.</description>


rus:randomUUID()"/>
itrus:randomNumber(6)"/>



 request message to IncidentManager via Http SOAP interface</message>


 fork="true">


p://www.citrusframework.org/schema/samples/IncidentManager/v1">

im:ticketId>
tDate('yyyy-MM-dd'T'00:00:00')</im:captured>

:component>
went wrong with the software!</im:description>


d>
m:firstname>
lastname>
r. 38, 80995 München</im:address>
```

```
ework.org/schema/testcase"
framework.org/schema/jms/testcase"
framework.org/schema/ws/testcase"
framework.org/schema/samples/IncidentManager/v1"
framework.org/schema/samples/NetworkService/v1"
ringframework.org/schema/beans" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
ww.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd
rk.org/schema/jms/testcase http://www.citrusframework.org/schema/jms/testcase/citrus-jms-testcase.xsd
rk.org/schema/ws/testcase http://www.citrusframework.org/schema/ws/testcase/citrus-ws-testcase.xsd
rk.org/schema/testcase http://www.citrusframework.org/schema/testcase/citrus-testcase.xsd">

1_IT">


te>

/last-updated-by>
</last-updated-on>


lication via Http message transport using SOAP request message. Opens a new incident and verifies
ce as well as final incident response.</description>


rus:randomUUID()"/>
itrus:randomNumber(6)"/>



 request message to IncidentManager via Http SOAP interface</message>


 fork="true">


p://www.citrusframework.org/schema/samples/IncidentManager/v1">

im:ticketId>
tDate('yyyy-MM-dd'T'00:00:00')</im:captured>

:component>
went wrong with the software!</im:description>


d>
m:firstname>
lastname>
r. 38, 80995 München</im:address>
```

```
ework.org/schema/testcase"
framework.org/schema/jms/testcase"
framework.org/schema/ws/testcase"
framework.org/schema/samples/IncidentManager/v1"
framework.org/schema/samples/NetworkService/v1"
ingframework.org/schema/beans" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
ww.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd
rk.org/schema/jms/testcase http://www.citrusframework.org/schema/jms/testcase/citrus-jms-testcase.xsd
rk.org/schema/ws/testcase http://www.citrusframework.org/schema/ws/testcase/citrus-ws-testcase.xsd
rk.org/schema/testcase http://www.citrusframework.org/schema/testcase/citrus-testcase.xsd">

1_IT">


te>

/last-updated-by>
</last-updated-on>


lication via Http message transport using SOAP request message. Opens a new incident and verifies
ce as well as final incident response.</description>


rus:randomUUID()"/>
itrus:randomNumber(6)"/>



request message to IncidentManager via Http SOAP interface</message>


fork="true">


p://www.citrusframework.org/schema/samples/IncidentManager/v1">

im:ticketId>
tDate('yyyy-MM-dd'T'00:00:00')</im:captured>

:component>
went wrong with the software!</im:description>


d>
m:firstname>
lastname>
r. 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 designer 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.

AL Package: com.consol.citrus.samples.incident Last Result: SUCCESS

100%

Http_Ok_1_IT: SUCCESS

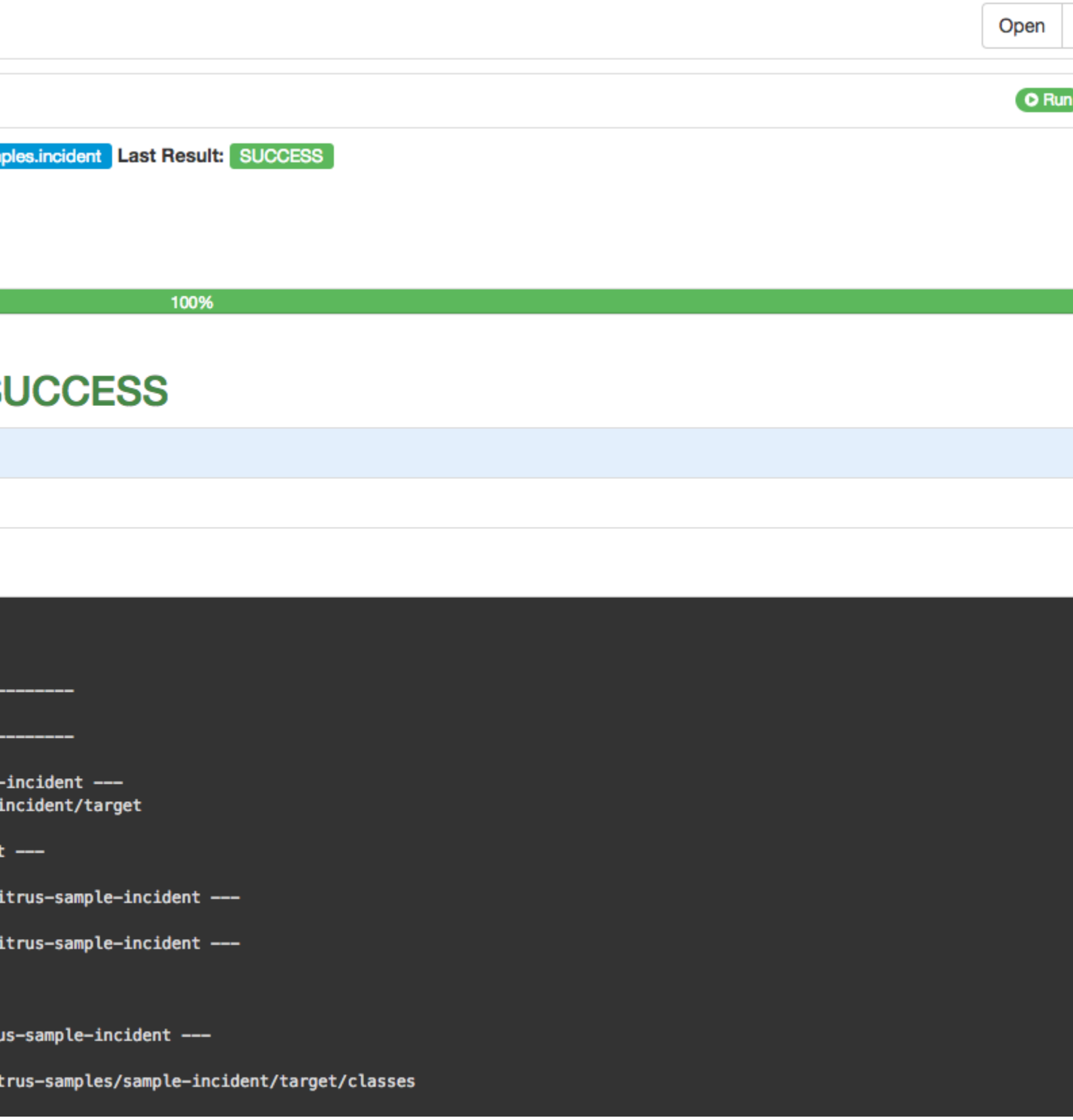
```
------------------------------------
2.7.1
------------------------------------

 (default-clean) @ citrus-sample-incident ---
te/Citrus/citrus-samples/sample-incident/target

default) @ citrus-sample-incident ---

l2java (apache-cxf-generate) @ citrus-sample-incident ---

esources (default-resources) @ citrus-sample-incident ---
iltered resources.


compile (default-compile) @ citrus-sample-incident ---
he module!
ers/christoph/Projekte/Citrus/citrus-samples/sample-incident/target/classes
```

AL Package: com.consol.citrus.samples.incident Last Result: SUCCESS

100%

Http_Ok_1_IT: SUCCESS

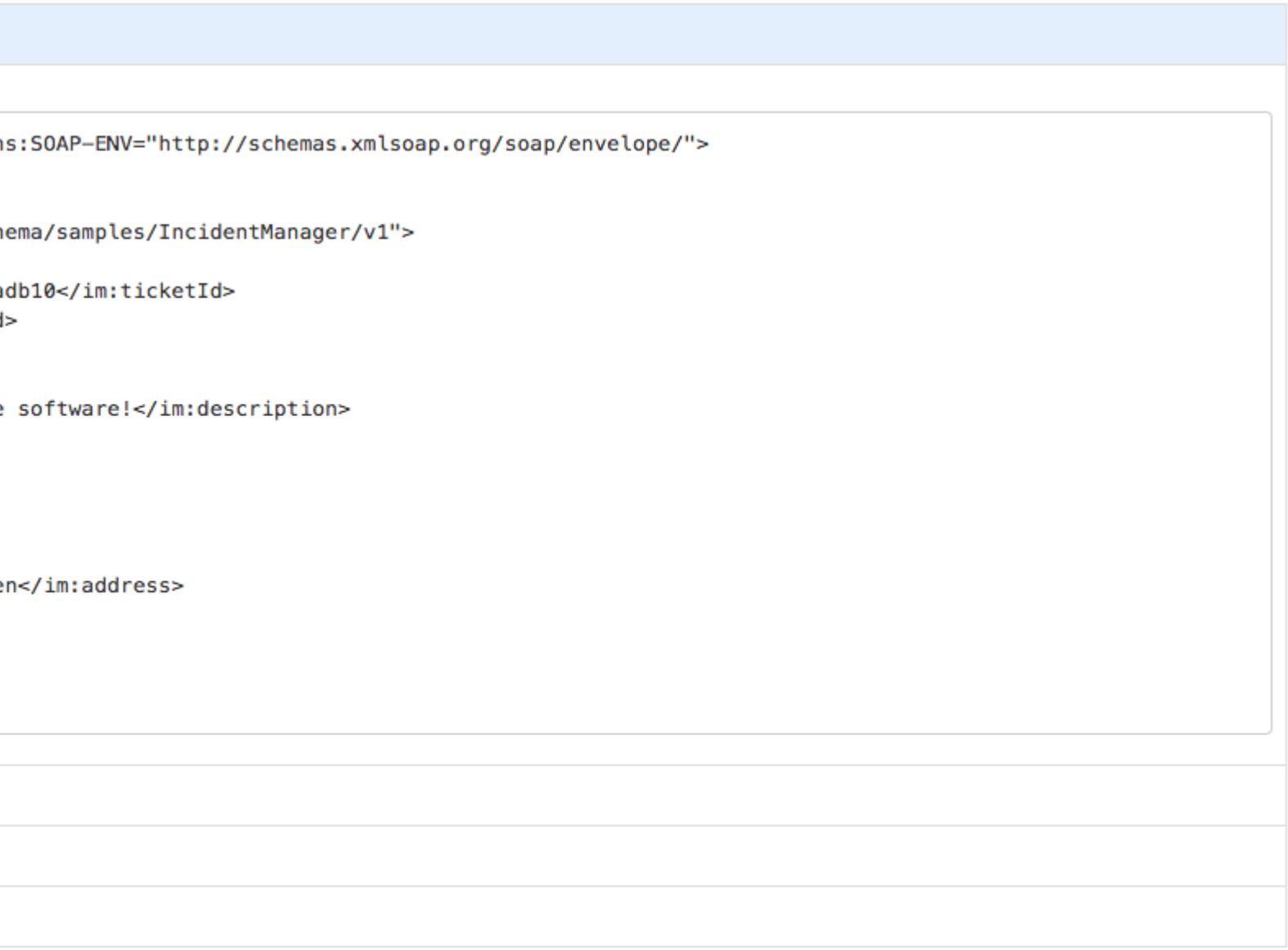
```
------------------------------------
2.7.1
------------------------------------

 (default-clean) @ citrus-sample-incident ---
te/Citrus/citrus-samples/sample-incident/target

default) @ citrus-sample-incident ---

l2java (apache-cxf-generate) @ citrus-sample-incident ---

esources (default-resources) @ citrus-sample-incident ---
iltered resources.


compile (default-compile) @ citrus-sample-incident ---
he module!
ers/christoph/Projekte/Citrus/citrus-samples/sample-incident/target/classes
```



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 table you are able to review all messages (inbound/outbound) that were part of the test run.

```
ng="UTF-8"?><SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">


"http://www.citrusframework.org/schema/samples/IncidentManager/v1">
>
Id>a9f13b9b-81ad-4308-975a-9b6a57fadb10</im:ticketId>
ed>2017-06-26T00:00:00</im:captured>
NEW</im:state>
ent>SOFTWARE</im:component>
ption>Something went wrong with the software!</im:description>
t>
>
128</im:id>
ame>Christoph</im:firstname>
me>Deppisch</im:lastname>
s>Franziskanerstr. 38, 80995 München</im:address>
r>
ent>
```

```
ng="UTF-8"?><SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">


"http://www.citrusframework.org/schema/samples/IncidentManager/v1">
>
Id>a9f13b9b-81ad-4308-975a-9b6a57fadb10</im:ticketId>
ed>2017-06-26T00:00:00</im:captured>
NEW</im:state>
ent>SOFTWARE</im:component>
ption>Something went wrong with the software!</im:description>
t>
>
128</im:id>
ame>Christoph</im:firstname>
me>Deppisch</im:lastname>
s>Franziskanerstr. 38, 80995 München</im:address>
r>
ent>
```



The message panel displays all inbound and outbound messages. Click on the message row to see the message content details. The mechanism for tracking inbound and outbound messages during a test run is done with either the admin connector library that you need to activate in your project. If for some reason you are not able to activate the connector library in your project the administration UI will try to read the messages from the normal Citrus logging output. This of course is only working if you have logging enabled in addition to using at least with logging level *DEBUG*.

If non of these approaches is working for you the admin UI will not display exchanged messages after the test run.

Test designer

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.

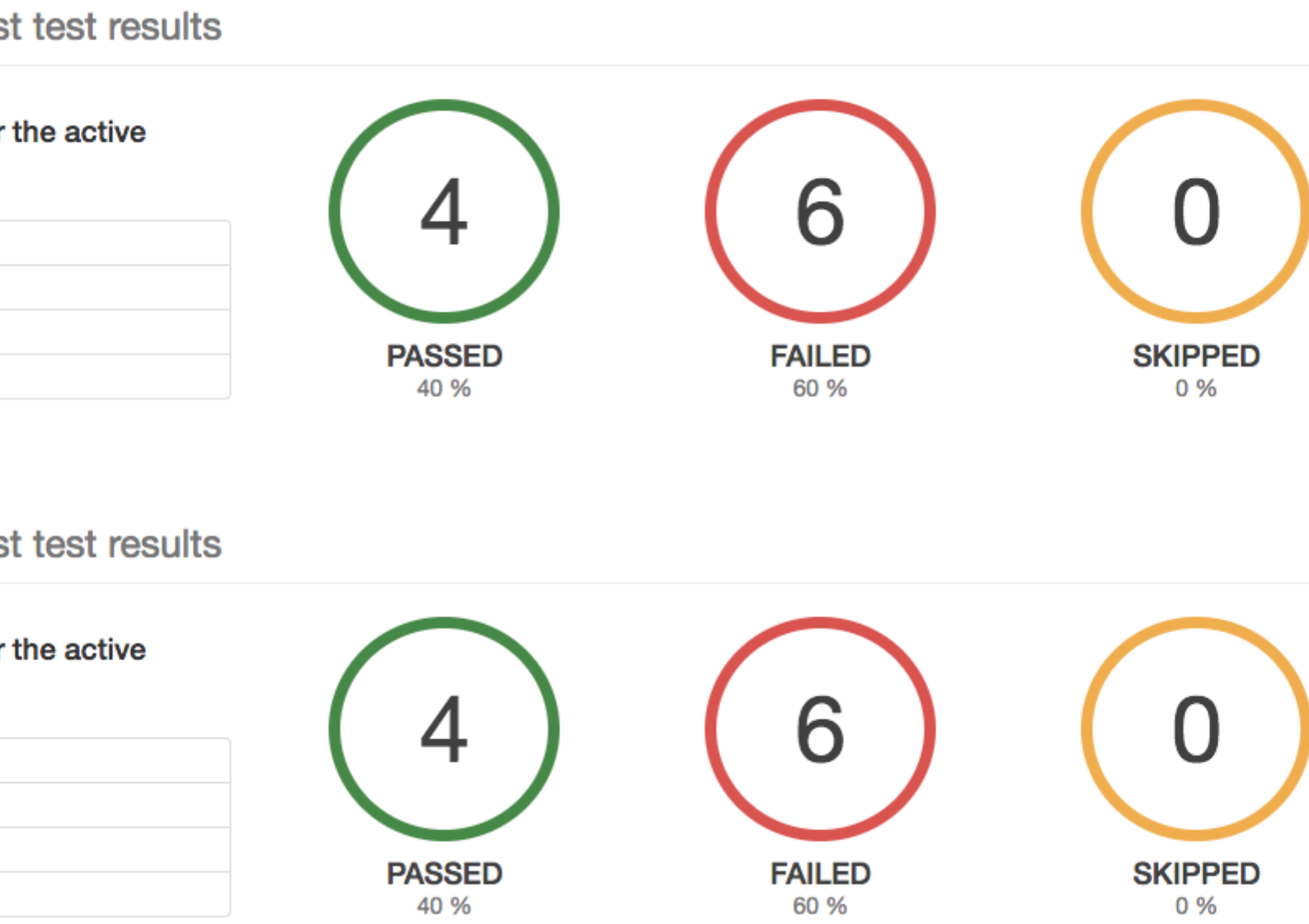

echo
send
echo
receive
echo
send
echo
receive


echo
send
echo
receive
echo
send
echo
receive

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

Test reporting

The administration UI is able to read and parse basic TestNG and JUnit reports. Usually these reports are written after each test run to the build output directory of your project. The admin UI will automatically find those reports and display the results to you.



When a test case is failing for some reason exception and failure information will be provided.