# Talend User Component tRunTask

**Purpose**

This component runs job as task in the Talend Administration Center (TAC).
The advantages are:

- It is possible to create execution plans with nearly unlimited flexibility
- It enables the chaining of external job scheduler with the TAC
- It is not necessary to build “monster” jobs by embedding the whole process with tRunJob
- With chaining task with tRunTask it is always possible to monitor all single steps (one task)
- The used API does not need Talend specific libraries and is based on Open Source technology
- Since Talend release 5.6.1 the component returns the exit code from the job as return code.
- Has two modes:
  - Run a Task: Runs a task
  - Retrieve a task status: This mode reads the current status of a task and provide it as return values

**Talend-Integration**

This component can be found in the palette under Management (next to tRunJob)
This component provides several return values.

**Basic settings**

| Property | Content |
|---|---|
| Operational Mode | Run Task: Run a task<br>Check t |
| TAC URL | URL of the TAC (it is the same as used in Studio or in the browser) ***required*** |
| TAC login | User login (User need the Administrator role)<br>It is recommended to use an technical user |
| TAC password | Users password |
| Use task label | If false you need to know the ID of the task<br>(refer to the information view of a task in the TAC) |
| Task label is job name | In case of the task has as label the job name check this option. In this case the job name is used as task label. |
| Task ID | ID of the task ***required (if Use task label is false)*** |
| Task label | Label of the task ***required (if Use task label is true)*** |
| Job run by task | The job, which will be run by this task. This configuration expects to choose the job which is actually used in the already deployed task. It does NOT change the job for a task.<br>The purpose is to configure the context variables. |

Additional basic settings for mode: Run a task

| Property | Content |
|---|---|
| Context Parameters | The context parameters and its source. |
| Run task asynchronously | If true it starts the task and pools for its end. If false it waits for the end in the same http request (in long running task this could lead to broken pipes) |
| Wait until the end | Waits for the end of the job. Otherwise the job will be started and the component finish and the current job can continue. It is like fire and forget.<br>To prevent calling a task twice at the same time, the component checks at if the task is already running and waits until its end. |
| Check time cycle until job is running | If Run task asynchronously is true, we have to wait until the TAC has started the job the poll on its end.<br>Set the time in ms. |

| | |
|---|---|
| Check time cycle until job is running | If Run task asynchronously is true, we have to poll on its end. It depends on the experiences about the typical task run duration. It is not recommended to pool to often, because it could lead to a notable load for the TAC. |
| Wait until this task and none of the task in the list are running | If true, the component pools on the finish status of it own task and the listed tasks and start its own task after none of these task are currently running. |
| Die on error | Dies of the started task fails (means the job fails because of any problems within the job). If there are any errors in communication with the TAC, the tRunTask component will always fail. |

**Advanced settings**

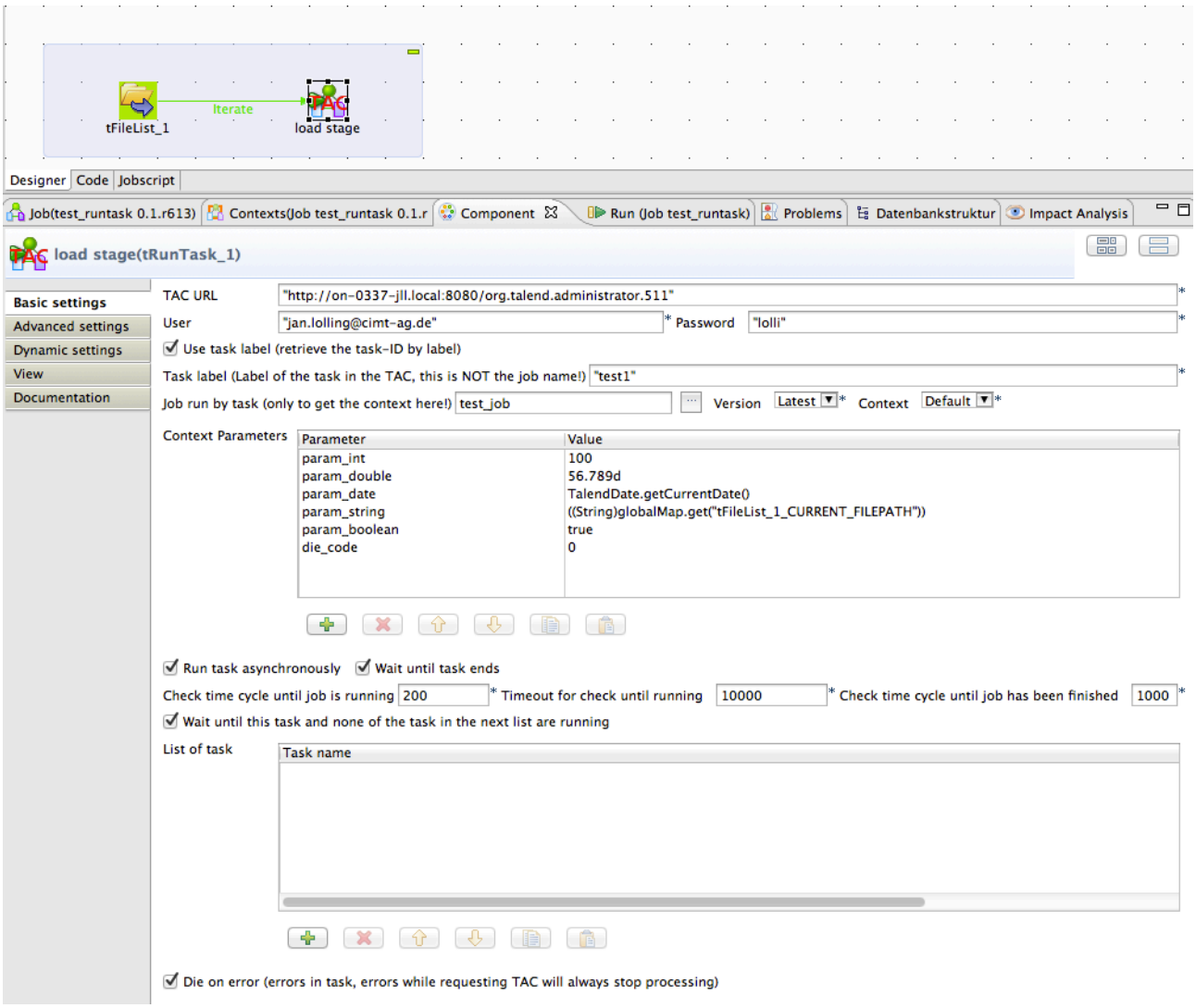
| Property | Content |
|---|---|
| Debug output requests and response | If true the component prints out the requests and the responses to standard output stream |
| Allow task generating or deploying | if false: the component fails if the task is not in the status: Read To Run.<br>if true: the component starts the job and checks the preparing status. This will also allow the task to be generated between two runs. |
| Maximum repetition in case of TAC errors | If the TAC returns errors the component repeat the request multiple times. If the maximum number of repetition is reached, the component fails. |
| Wait time between repetition | Wait time in milliseconds after a TAC error happened before the next attempt. |

**Return values**

| Return value | Content |
|---|---|
| ERROR_MESSAGE | Last error message. Unfortunately this is not the error message from the actually running job. This message is build from the tRunTask component. The current TAC web service does not provide this message. |
| TASK_ID | The task id retrieved from the TAC by the task label. All commands to the TAC related to task need the task ID. That's why the tRunTask component retrieves this ID at first. |
| RUN_DURATION | The time the task is running measured by the tRunTask component. It is not exactly the time because of the possible polling time delay. |
| RETURN_CODE | Before Talend release 5.6 the component is not able to get the real job exit code and returns 4 as exit code in case of the task run fails (real job failures).<br>Starting with Talend release 5.6 the component delivers the real exit code of the job. **This real exit code is only available in the "run" mode!**<br>In the status mode the component has no access to the real exit code and returns 4 in case of errors and 0 in case of OK. |
| HAS_ERRORS | The job has errors (the last run of the job (task) failed) |
| IS_READY_TO_RUN | The job is ready to run |
| IS_PREPARING | The job need preparing like generating or deploying |
| IS_RUNNING | The job is currently running |
| STATUS | The current status of the job as text |
| ERROR_STATUS | The current error status of the job |

**Scenario 1:**

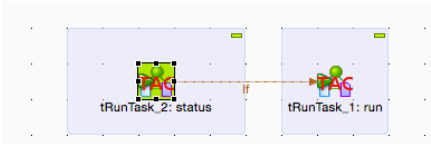
Simply running a task:




This scenario shows the way to implement a trigger, which starts a task in the TAC for every found file. There are a lot of other scenarios possible.
One of the most used scenarios is to trigger a task from another Job scheduler because of a company policy about scheduling. In large companies there are typically dedicated schedulers and with this component you can write a job with it self can started as simple script from such kind of schedulers.

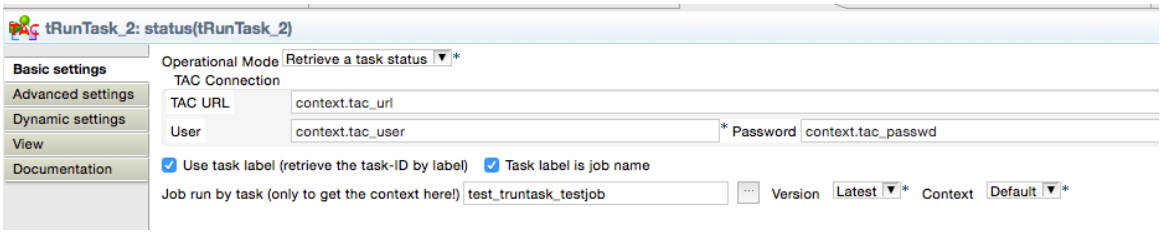
**Scenario 2: Watchdog job**

In this scenario a job check the status of a task and if the task has been failed the task will be run again.
In a real world scenario such job will be run also as task in the TAC and would be triggered several times within a hour.



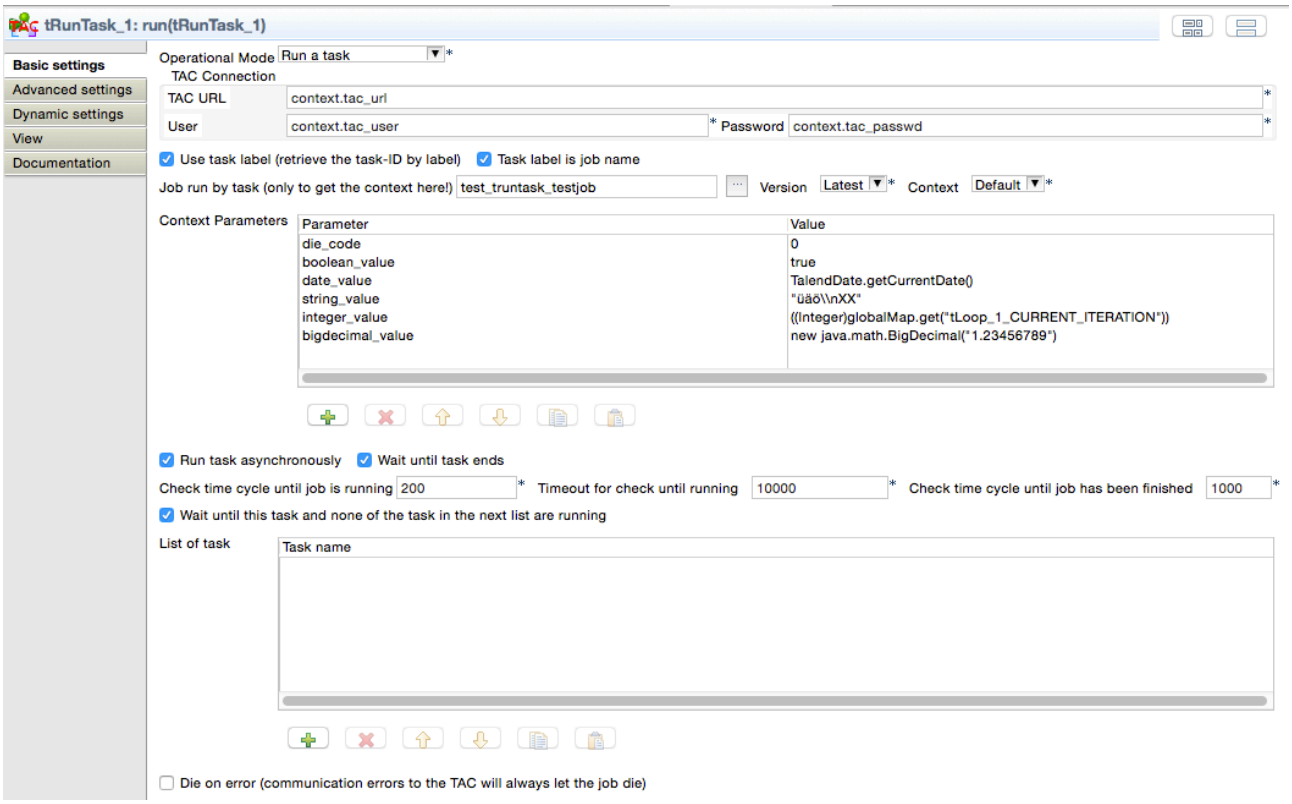

This is the simple job design.

Here the basic settings for the tRunTask_2. The enhanced label can be achieved by set as label in the View setting:
__UNIQUE_NAME__: __MODE__




The if condition is: ((Boolean)globalMap.get("tRunTask_2_HAS_ERRORS"))
This is a return value and can be put here by drag and drop.

If the task has errors, the task should be run again:
Here the basic settings of the tRunTask_1:



## Typical error messages

The correct run of the TAC is prerequisition before checking the following configuration problems.

**Error message:**
```
org.apache.http.conn.HttpHostConnectException: Connect to 111.11.11.11:8080
[/111.11.11.11] failed: Operation timed out
```

**What went wrong:**
The host cannot be reached. Check the IP address

**Error message:**
```
java.net.UnknownHostException: debian2.local: unknown error
```

What went wrong:
Server name is wrong or the server name cannot be resolved because missing domain.

**Error message:**
```
org.apache.http.client.HttpResponseException: Not Found
```

**What went wrong:**
This means the server and host are perhapse ok but the server does not know the path.
E.g. this is currently used:

http://debian1.local:8080/tac_611_tst

but is should be e.g.:

http://debian1.local:8080/tac_611_**prd**

**Error message:**
```
java.lang.Exception: Server error: No task with this label
\"test_truntask_testjob1\""
```

**What went wrong:**
The given task does not exist in the TAC. Please take care about the task label and keep in mind it is case sensitive.

**Error message:**
```
java.lang.Exception: Server error: Cannot find task with id=9999
```

**What went wrong:**
The given task id is wrong. Check the task id in the TAC in the task details.