



**INTRASTAT, VIES, ETS
INTEGRATED REPORTING
SOLUTION**



Revolux S.à R.L.

SKAT WS Test
for
Danmarks Statistik & SKAT
by
Revolux S.àR.L.

Testing the SKAT web services

Doc ID	RVL_064
Author	Revolux S.àR.L.
Distribution	Danmarks Statistik, Revolux
Version history	
12/03/2010	First draft

Category	Project Documentation
Document	SKATWSTest.docx

Summary

This document describes the application that was developed for testing the SKAT web services.

Table of Contents

1	Introduction	1
2	Summary	1
3	Contents, Requirements and Installation	3
3.1	Contents.....	3
3.2	Requirements	3
3.3	Installation.....	43
4	User Interface.....	4
5	Threads	7
6	Configuration	8
7	Test Input File	9
8	Application Log.....	9
9	Performance Log	9
10	Max Number of Cycles.....	10
11	Tips.....	11

Table of Figures

Figure 1	Installation result.....	4
Figure 2	User interface - idle	54
Figure 3	User interface - running.....	5
Figure 4	Confirm approval against production.....	76
Figure 5	Confirm start configuration	7
Figure 6	Threads stopped after max number of cycles.....	1110
Figure 7	Stop after max number of cycles.....	11

1 Introduction

The application allows testing the three SKAT web services:

- **VirkksomhedSoeg** AKA **GetSENumbers**
- **IndberetningRettighederHent** AKA **GetSERights** or **VIESAllowed**
- **ListeAngivelseFilOpret** AKA **SaveDeclaration**

The input to the application is a list of CVR numbers to test. The user interface allows setting various test scenarios that can simulate different (real world?) usages and loads of the SKAT web services.

The outcome of a test is a log file and a performance log. Obtaining the performance is log is the primary goal and purpose of the application.

The application uses the **same** code for accessing the web services and generating the declarations as IDEP.WEB does. Thus the tests carried out by the tool **are** realistic tests.

2 Summary

(See email 29/01/2010 – Peter de Vos - RE: SKAT web-service test (In reply to your phone-call))

The user provides a list of CVR numbers. These numbers will be tested, one after the other, all in one test **cycle**.

The user needs to provide an **interval** between two test-cycles and then user 'Starts' the tests.

A small status screen logs the progress of the program. This log is also written to a log-file. Each session is logged in its own log file with timestamps.

For each tested web-service the maximum response time is shown on the status screen as well, plus the time and date when this occurred.

The log file will be tab-separated, so it can be opened and processed with Excel. The format of this file is described in section [PERFORMANCE LOG](#).

A single test **cycle** is defined as follows:

```
For each valid CVR in the list
  Call service VirksomhedSøg
  For each returned SE number
    Call service IndberetningRettighederHent
    If ViesAllowed
      Call service ListeAngivelseFilOpret
      [ with some dummy data (up to N lines) ]

Take a random CVR-number (valid with respect to the Danish VAT algorithm)
Call service VirksomhedSøg
For each returned SE number
  Call service IndberetningRettighederHent
```

The random number assures we are getting through to their database and are not obtaining cached information. The random number is generated automatically based on the Danish VAT algorithm. **This might often generate CVR numbers that are not in use.**

The *one-after-the-other* approach does not simulate the situation where multiple declarations are sent simultaneously (which is a valid scenario and should also be tested). In order to accomplish this, the program could be run N times, in parallel, with a different interval-time and different CVR list. The results will go to separate log-files; one per running instance.

In order to simulate a realistic load (many simultaneous connections) each instance of the application can also run several **threads**. Each thread carries out the same test. The threads run in parallel (at the same time) which will effectively stress the web services – especially if several instances of the application are running with multiple threads. See section [THREADS](#).

Given that we should be able to handle 20.000 user, assuming they would log in 2 twice a month, and create 2 declarations per month, we would need to handle 40.000 CVR tests per month = 2000 per working day = 300 per hour = 5 per minute. Of course not evenly spread over the day; we should assume bursts.

This would more or less be covered by running three instances of the program:

- List of 4 CVRs; 2 of them returning 1 SE number; 2 returning 2 SE-numbers (this would generate 6 declarations).
Interval: 60 seconds
- List of one CVR; returning 1 SE number (generating one declaration)
Interval: 300 seconds
- List of 5 CVR; returning 2 SE number (generating 10 declaration)
Interval: 1000 seconds

None of these CVR numbers should be the same (but are allowed to be). The dummy declaration data is generated automatically with up to N (configurable) lines. Each line will contain a random Luxembourgish VAT number (valid with respect to the algorithm but not necessarily associated with a Luxembourgish company). This is necessary because it is not allowed to submit a declaration with the same VAT number several times:

Der er angivet flere rækker til samme varemottager for samme periode på fil

Each line can contain **random** or **fixed** amounts (Service value [if applicable], Invoice value, Triangular value).

In the reporting system of SKAT it should be easy to afterwards remove this data from the system.

If you would let these three instances of program run for a few days, it would give a proper indication of the reliability of the SKAT-web-services. (You might want to experiment with other intervals and perhaps more instances/threads in parallel. We leave that up to your imagination).

3 Contents, Requirements and Installation

3.1 Contents

The content of the distribution is a self-contained installation package (**Setup.exe** & **Setup.msi**) that allows installing it on any PC or server.

3.2 Requirements

The machine requires access to the SKAT web services, i.e. it must use an IP-address that is acceptable by SKAT and is allowed to call out through the firewall – presumably any PC in the DST network should be acceptable (provided it is allowed to call out).

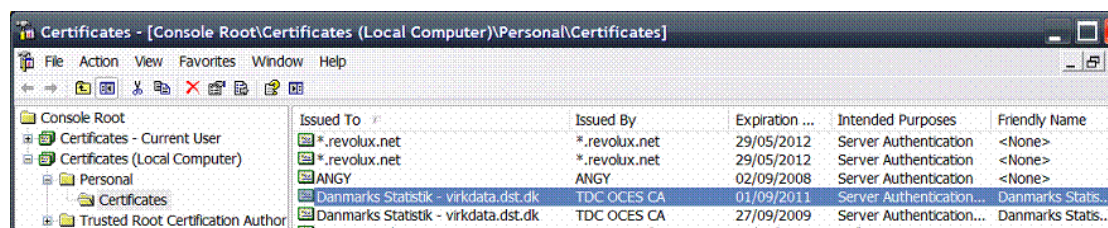
The machine must have Microsoft.NET [2.0.3.5](#) SP21 or higher installed.

The application can be installed on any of the IDEP.WEB servers, production or test-server, in- or outside the DMZ.

For ease of installation and operation we recommend running the application on one of the IDEP.WEB servers.

In the following two cases you should install DST's **signing** certificate (with thumbprint 1D086BE528709E93755F55CA7F2EFF8FC24CD6C5) in **Local Computer\Personal\Certificates**:

- If the tool is to be run outside of DST or Revolux, contact SKAT to open their firewall to another IP-address.
- If the tool is to be run in DST's LAN, ensure that it is allowed to call out to the SKAT web service end-points (in principle this should be the case).



Make sure the certificate is installed using the same account as is used for running the application. If the accounts are not the same use the WSE Certificate Tool to grant the rights.

3.3 Installation

Execute **setup.exe** and follow the instructions. A shortcut to the application will be placed on the installer's desktop. The tool allows easy access to its installation location through the user interface so the actual installation location is of little importance.

Once the application is installed the result is:

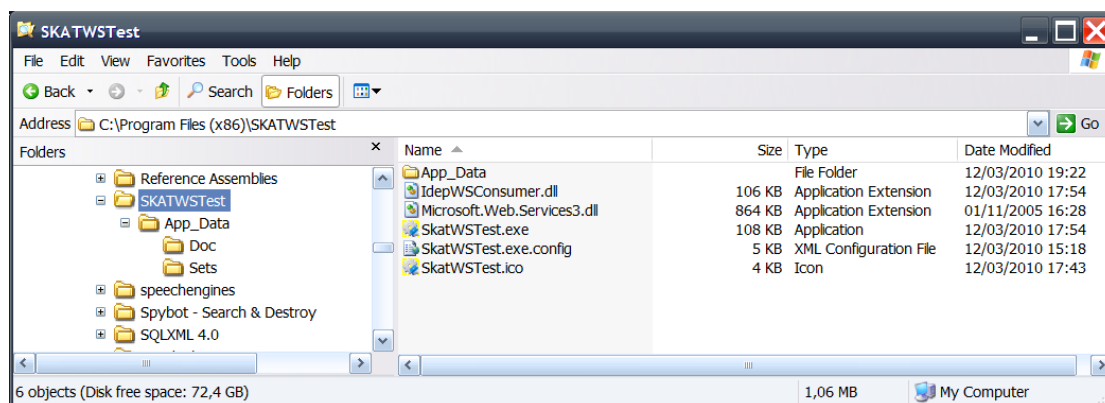


Figure 1 Installation result

The above illustrates the default installation (on a 64bit Windows XP).

The **App_Data** directory contains the **Doc** and **Sets** subdirectories which contain the documentation and sample CVR test data respectively.

The **SkatWSTest.exe.config** file is the application's configuration file. In principle there is not much that needs to be changed here. See section [CONFIGURATION](#).

4 User Interface

Using the shortcut (*SKAT Web Services Test*) on the desktop or starting **SkatWSTest.exe** directly the user interface is shown:

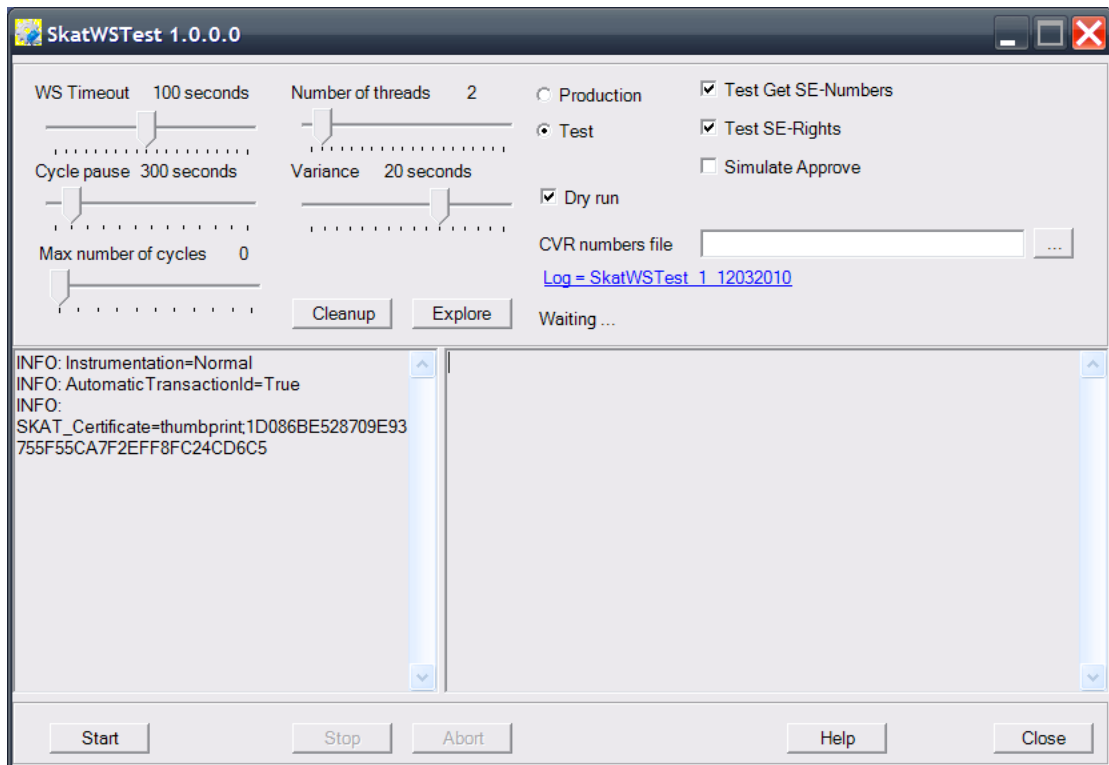


Figure 2 User interface - idle

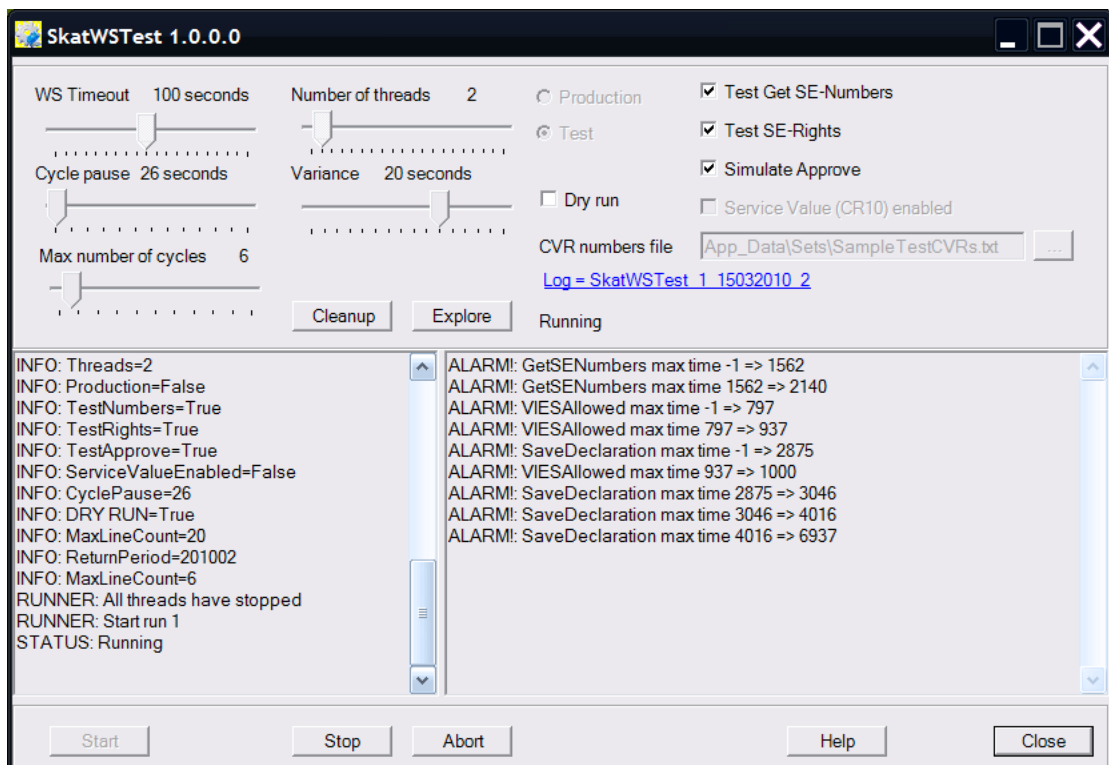


Figure 3 User interface - running

The various screen elements are explained below:

- **Left hand pane:** Displays the initial settings and shows information about the running state of the application (starting, stopping, stopped).

- **Right hand pane:** Displays the performance warnings. If a current maximum time for a web service has been exceeded, a message will be displayed in this pane.

The panes can be adjusted by moving the splitter between them. Everything that is shown in each of the panes can also be found back in the application's log files.

The control settings:

- **WS Timeout:** Web service timeout – the default setting is set in the application's configuration file. This is the maximum time the application will wait for a reply from the web service. IDEP.WEB also uses the 100 second timeout as a default. If this value is too low the web service calls may be aborted too soon if the load on SKAT's server is too high.
- **Cycle pause:** Pause between each test cycle.
- **Variance:** If this value is set a random number of seconds is added to each pause between cycles. The random value will be between 0 and the value of 'Variance'.
- **Max number of cycles:** Maximum number of test cycles that will be carried out. If this value is set to 0 the application will continue indefinitely. See also section [MAX NUMBER OF CYCLES](#).
- **Number of threads:** Stress factor – number of test cycles that are carried out simultaneously.
- **Production / Test:** Choose between testing against the production or test server. If you are testing against the production server with approval switched on you should be aware that you will be **submitting fake test reports to the real production server**. A warning will be issued if this is the case.
- **Dry run:** Runs the configured tests **without** calling the web services. This test does not generate a performance log.
- **Test Get SE-Numbers:** Switch on the calls to the **GetSENumbers (VirksomhedSoeg)** web service.
- **Test SE-Rights:** Switch on the calls to the **GetSERights/VIESAllowed (IndberetningRettighederHent)** web service.
- **Simulate Approve:** Switch on the calls to the **SaveDeclaration (ListeAngivelseFilOpret)** web service. **This will generate fake VIES reports**. To test this option Test SE-Rights must be switched on.
- **Service Value (CR10) enabled:** Switches on the 'Service Value' in the VIES declarations (only applicable if Approval) is switched on. **Currently this only works against the Test server**. So even in Production mode, the test server will be addressed instead. The end points for this service are configured in the application's configuration file.
- **CVR numbers files:** The name of the file containing the CVR numbers to test.
- **...** : Browse for such a file.
- **Cleanup:** Cleans up the **Temp** and **Log** directories of files older than one day. If you press shift while clicking this button, **all** files will be deleted.
- **Explore:** Opens the Windows explorer in the application's root directory.

- [Log=SkatWSTest_1_DDMMYYYY](#): Opens the current log file (**not** the performance log) in the default text editor (usually notepad).

The buttons:

- **Start**: Having adjusted all the settings pressing the 'start' button will display a confirmation box and start the tests.

If the approval against the production server is switched on you must confirm this:

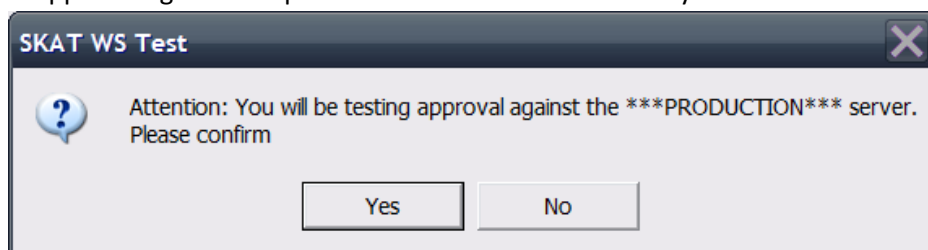


Figure 4 Confirm approval against production

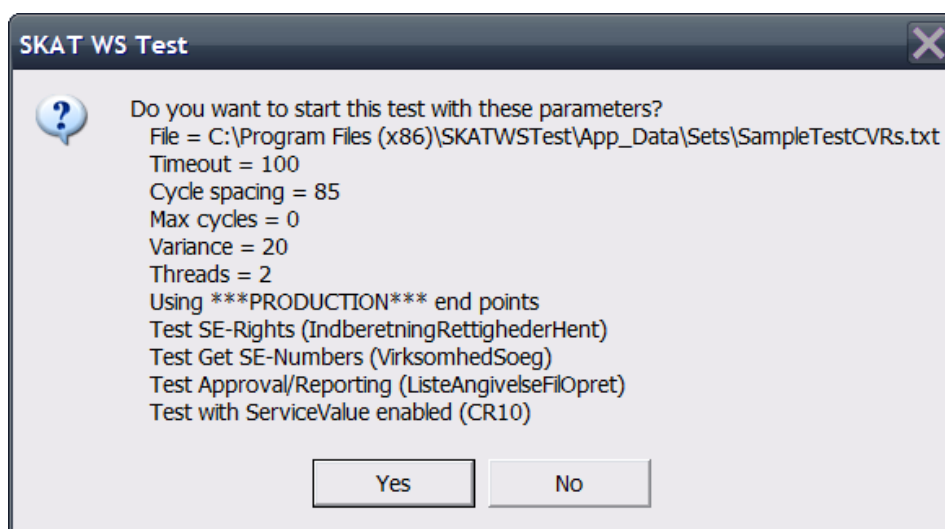


Figure 5 Confirm start configuration

- **Stop**: Stop the test as soon as possible. This may not be immediate if the worker threads are pausing between test cycles. A confirmation will be requested.
- **Abort**: Stop the test immediately.
- **Close**: Stops any test in progress and quits the application. If lengthy threads are operating you should first abort the tests and then close.

Some of the controls remain enabled while a test is active; this means you can interactively **change these settings while the test is running** (e.g. switch from dry run to real run).

5 Threads

A 'thread' can be seen as a separate process carrying out a complete test cycle.

The program supports multiple threads. On a multi-CPU machine these threads may run in parallel (i.e. **at the same time**).

This allows simulating a realistic load (many simultaneous connections) in instance of the application.

The number of threads can be seen as a 'stress multiplication factor'.

When multiple threads are used the cycle variance should also be used. This will add a random pause after each completed cycle which ensure that sometimes the web services are called at the same time by multiple threads and sometimes slightly delayed adding more 'realness' to the tests.

The multi-threaded nature of the application makes stopping it somewhat 'special' in a way that in principle the application must wait for all of the threads to finish. This becomes especially tricky if the threads work asynchronously (with random variance). If the 'cycle pause' is high, the application must wait a long time for each thread to finish (at least 'cycle pause' seconds). In such cases you might as well 'abort' the test which will kill the worker threads immediately. From the program's perspective this is not very 'elegant' (brutal) and might, for instance, interrupt a web service call in mid-stride. In any case 'abort' cannot do any harm and might only generate an exception message in the log.

6 Configuration

Some changes might be necessary in the file **SkatWSTest.exe.config**:

Once the end-points for the CR10 web services (test and production) are known.

```
<add key="WS_SaveVIESDeclarationCR10"
value="http://85.81.229.78:8080/ListeAngivelseFilOpret" />
<add key="WS_SaveVIESDeclarationTestCR10"
value="http://85.81.229.78:8080/ListeAngivelseFilOpret" />
```

If the declaration submission period for February 2010 has closed (i.e. the tool is used in another month than March 2010).

```
<add key="ReturnPeriod" value="201002"/>
```

It is possible to configure the Invoice, Triangular and Service values used in the declarations. They can be made random or fixed:

```
<add key="Values" value="1;0;R"/>
```

The above means the fixed value 1 is used as Invoice value, the fixed value 0 for Triangular value and **random** for service value. **This should possibly be agreed upon with SKAT.**

A part from the above changes nothing really needs to be changed in this file; other settings can also be set through the user interface.

```
<add key="Instrumentation" value="normal" />
```

The legal values of **Instrumentation** are **normal**, **high**, **extreme** and **off**. If the value is set to **extreme** many files will be generated in the **Temp** directory (each web service call and its result will be dumped as XML file).

It is also possible to switch on even more detailed logging of the web service calls. This will generate great amounts of log information! This should only be switched on in case of problems.

```
<microsoft.web.services3>
  <!--optional-->
  <diagnostics>
    <trace enabled="false" input="InputTrace.webinfo"
output="OutputTrace.webinfo" />
    <detailedErrors enabled="true" />
  </diagnostics>
</microsoft.web.services3>
```

In principle none of the other settings in this file need changing.

7 Test Input File

The input file should contain at most one valid CVR number per line. The format of the file is very flexible and may contain ordinary text too. Only recognised valid CVR numbers will be taken into consideration. Duplicate CVR numbers will be ignored.

Anything following comma, semi-colon, tab, hash will be ignored. The supplied file **SampleTestCVRs.txt** gives an example of what's possible.

8 Application Log

The application log will be generated in the **Log** directory (automatically created). This log will contain a very detailed trace of the operations (and errors encountered) of the test. It will also contain the settings that are active for the test.

The log also contains detailed information about any errors encountered in the test file.

An application log file will be called **SkatWSTest_1_DDMMYYYY[_NNN].txt**.

Each test run carried out in the **same session** will be appended to the **same** log file (this means that an overnight test will be stored in the same log as the previous day).

Each log file entry is time-stamped which will allow relating individual application log events to the performance log and vice-versa.

9 Performance Log

The performance log will be generated in parallel to the application log in the **Log** directory. Its name is **Per-formance_1_DDMMYYYY[_NNN].txt** (i.e. the same 'number' as the application log).

Each test run carried out in the same session will be appended to the same log file.

Each log file entry is time-stamped which will allow relating individual application log events to the performance log and vice-versa.

A performance log contains all relevant information for analysis of the performance and behaviour of the web services. It also contains relevant information for SKAT (the Transaction Id and timestamp).

The columns in this file are:

1. **Thread:** Thread number
2. **Cycle:** Test cycle number
3. **WS:** Web service name
4. **CVR:** CVR number used
5. **SE:** SE number use
6. **Time:** Time in milliseconds elapsed
7. **Alarm:** Contains **y** if the previous maximum time has been exceeded.
8. **Info:** Result returned by the web service.
9. **Linecount:** Number of declaration lines submitted (if applicable)
10. **StartTime:** Timestamp of the operation
11. **Tid:** **TransaktionIdentifikator** and **TransaktionTidused** in SKAT's **HovedOplysninger** element.

10 Max Number of Cycles

Setting 'Max number of cycles' to 0 (default) causes the application to carry out cycles indefinitely. If this value is set to non-zero value the testing threads will stop the activity once the max number of cycles is reached.

This setting can be set interactively during the tests. The activity will stop as soon as possible (e.g. if all threads are waiting to carry out the next cycle the activity might stop only after this wait time has elapsed).

In the example below, we have three threads and as max number of cycles 10. Thus the application will stop its activity **after** 3 cycles (3 threads * 3 cycles each + at least one more cycle >= 10)

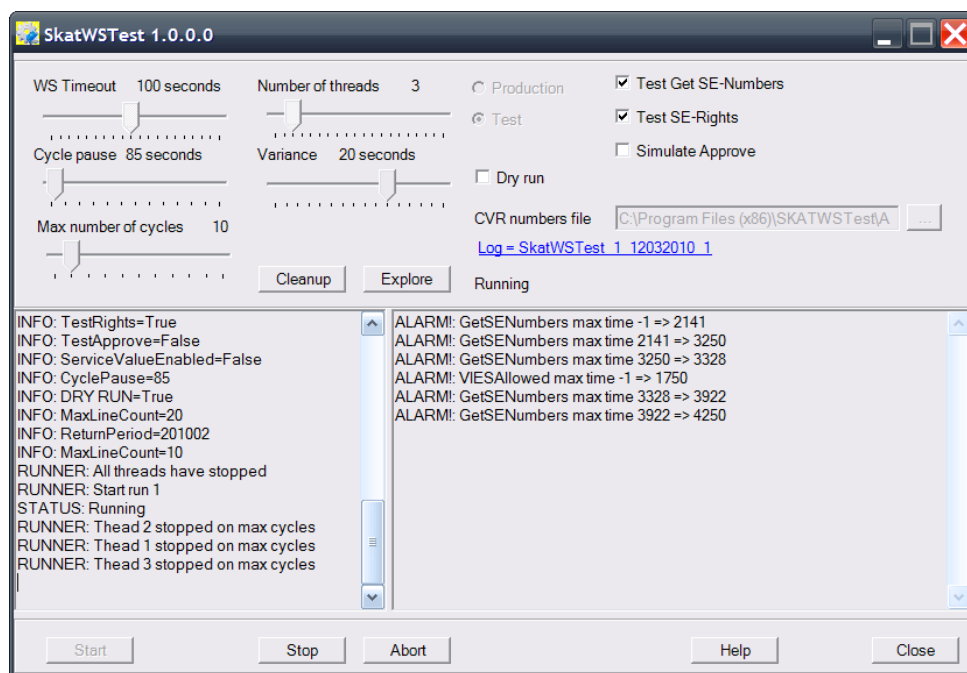


Figure 6 Threads stopped after max number of cycles

If 'Max number of cycles' is used the application is *not quite stopped* but just the activity threads have stopped their processing (they cannot signal the user interface that everything has stopped as the threads are not aware of each other).

In this case just 'Stop' the application using the 'Stop' button:

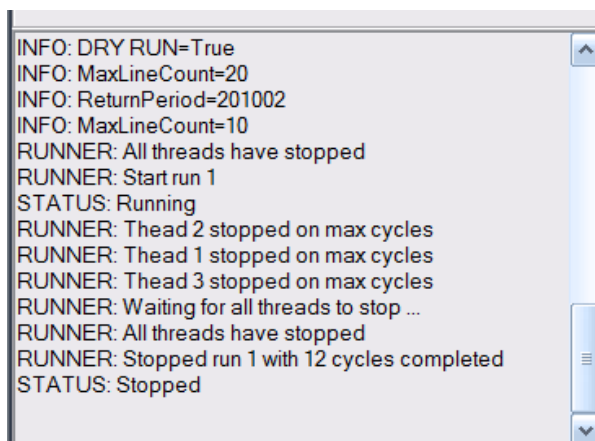


Figure 7 Stop after max number of cycles

In the above image we can observe that now the application has really stopped and it shows that 12 cycles have been carried out (3 threads each 3 full cycles and 2 threads 1 one cycle each).

11 Tips

Use 'Dry run' to get a feel of the application. This will **not** contact the SKAT web services. You can change some of the controls and observe the result in the report window and log file.

You can start several instances of the application to perform different tests in parallel.

Dragging the performance log file to an open Excel worksheet will import it right away.