Automatisches Testen der GUI für OpenOffice.org 2.0 für Mac OS X (X11-Version)

Version 1.3 vom 14.8.2006

Erstellt mit **OOo 2.0.3**

Platform / Os : **Mac OS X**

Viel Spaß beim Testen von OpenOffice.org 2!

Dieses Dokument wird ihnen vom deutschen OpenOffice.org-Dokumetations-Projekt und dem Mac-Porting-Team zur Verfügung gestellt: http://de.openoffice.org

1 Überblick

Das Dokument dient dazu, Einsteigern in das automatische Testen der GUI (graphic user interface = Benutzeroberfläche) von OpenOffice.org – die sog. „Smoketests“[1] - auf dem Mac das eine oder andere zu erleichtern.

Um diese Tests durchzuführen, gibt es eine eigene Software, die OOo „fernsteuert“, Eingaben simuliert und prüft, ob die Ergebnisse den Erwartungen entsprechen: Das TestTool.

Zu diesem gibt es eine Reihe von Scripten, die die einzelnen Tests durchführen. Diese werden an jede neue Version von OOo speziell angepasst und müssen daher immer in der passenden Version benutzt werden.

Einige grundlegende englische Texte findet man hier:

http://qa.openoffice.org/qatesttool/index.html

Danke für Jörg „Jogi“ Sievers, Tino Rachnui und Thorsten Bosbach für die vielen nützlichen Tipps, Ergänzungen und Korrekturen.

1.1 Wichtige Änderungen zu vorherigen Verisonen

1.1.a Allgemein

Aufgrund schlechter Erfahrungen muss generell vom Einsatz von Clientsoftware für CVS abgeraten werden – es werden gerne nicht alle Dateien aus dem CVS geholt, was zu Fehlern im Testlauf führt.

Der checkout per CVS-Kommandozeile oder tk-cvs hat sich als zuverlässigste Variante erwiesen[2].

1.1.b Änderungen in der Software ab OOo 2.0.3RC4

Das testtool kann jetzt auch mit dem Blank im Namen des Ooo-Pfades umgehen – man muss den Namen des Anwendungsordners nicht mehr ändern (Kap. 3.3.a)

Das Script soffice muss nicht mehr als „testtool“ kopiert werden – tettool.bin starte jetzt auch direkt. Gilt bisher aber nur für PPC – auf Intel-Macs muss das Script weiterhin kopiert und zum Aufruf des testtools benutzt werden. (Kap. 3.3.b)

1 Sie heißen angeblich deswegen „smoketests“, weil sie automatisch durchlaufen und man derweil in früheren Zeiten eine Zigarette rauchen ging, da der Rechner dadurch blockiert war.
2 Sun-intern wird wohl auch jEdit mit grundsput eingesetzt - grundsput läuft aber wohl nicht auf Macs

2 Download der benötigten tools

Das TestTool wird mit der Ooo-Installation mitgeliefert. Es befindet sich im Ordner …/program. Die einzelnen Testscripte werden für jede neue OOo-Version angepasst. Daher muss man sie als check-out aus dem CVS beziehen.

2.1 Check out der Scripte aus dem CVS-Server

Aufgrund der inzwischen gemachten negativer Erfahrungen mit MacCVSClient (und vom Hörensagen auch bei anderen Clients) wird unbedingt empfohlen, die Dateien aus dem CVS-repository per cvs-Kommandozeilenbefehl auszuchecken oder Grundsput zu verwenden.

Das geht auch sehr einfach – siehe Kap. 2.1.d

2.1.a Installation eines CVS-Clients

Wer es nicht lassen kann oder einfach selber mal ausprobieren möchte: Es gibt mehrere grafische CVS-Clients für den Mac, z.B.

- Concurrent Versions Librarian (ca. 3MB)
- LinCVS (ca. 7 MB, deutsch, ziemlich fett, sieh gut aus, leicht zu lernen)
- MacCVSClient (ca. 600 kB, eher schlicht) http://www.heilancoo.net/MacCVSClient/

Die sind alle freeware und können z. B. von http://www.versiontracker.com geladen werden.

- Und für Profis: jEdit mit Grundsput-Plugin. Damit kann man CVS und Editieren von Tests gleichzeitig machen. www.jedit.org (Danke an Jörg „Jogi“ Sievers für den Tipp)

Um einen CVS-Client oder die Kommandozeile auf dem Mac nutzen zu können, müssen als erstes die developer-tools nachinstalliert werden:
Installations-DVD einlegen – im Ordner XcodeTools das XcodeTools.mpkg öffnen und installieren. Dabei brauchen nur die eigentlichen Developer Tools installiert werden, den Rest kann man weglassen, was ca. 2 GB auf der Platte spart.

2.1.b Konfiguration eines GUI-Clients am Beispiel LinCVS

Zum ersten Download des ca. 109 MB großen repository auf den „checkout“-Button drücken und folgendes eingeben (die ersten Werte: Benutzer, CVS-Server usw. kann man auch vorher als Profil anlegen).Am Ende kommt eine Abfrage, ob man das Verzeichnis im Workspace anzeigen will – sollte man tun. Danach genügt dann Markieren des Verzeichnisses ein klick auf „update“, um die neuesten Versionen zu laden.

Einstellungen:

Benutzer:	anoncvs
cvs-server:	anoncvs.services.openoffice.org
CVS-Repository:	cvs
CVS-Modul:	qa/qatesttool
checkout nach:	/Users/Shared/OOoQA
CVS-Zugang:	Passwort

2.1.c Konfiguration eines GUI-Clients am Beispiel MacCVSClient

Nach dem runter laden, entpacken und starten des Programms folgende Werte bei den Preferences (tab „Login Profiles“) eingeben:

user:	anoncvs
host:	anoncvs.services.openoffice.org
root:	cvs
CVS-Modul:	qa/qatesttool
checkout nach:	/Users/Shared/OOoQA
CVS-Zugang:	Pserver
Port:	Default lassen
	No Password

Ein Klick auf Test zeigt, ob eine Verbindung aufgenommen werden kann.

Dann in den Dialog Repository > Check out… folgendes eingeben:

login:	Das in den Preferences angegeben login profile
module:	qa/qatesttool
Folder:	Users:Shared:OOoQA
	Normal Checkout Process folders recursively: an Prune empty folders: an Pretend: aus

„OK“ startet den checkout. In dem dann irgendwie aufgehenden Konsolenfenster sieht man den Fortschritt – der erste checkout kann wie gesagt was dauern, es sind ja ca. 100 MB.

Später genügt dann die Auswahl dieses Ordners unter File > Open Sandbox… und dann Sandbox > update, um das zu aktualisieren. Besser ist es allerdings, das alte qatesttool-Verzeichnis zu löschen und einen neuen checkout zu machen - derzeit gibt es mit einigen wenigen Dateien da immer mal ein „there is a file in the way“-Problem.

Wenn das hier kommt, dann antwortet der Server nicht – passiert schon mal im Internet.

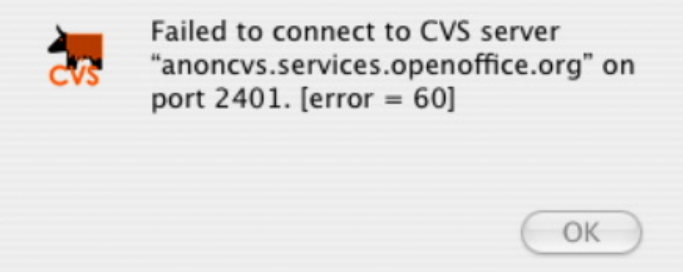

Abbildung 1: MacCVS - Client Verbindungsfehler

2.1.d And the winner is: Es geht auch einfach an der Kommandozeile

Wenn man mal über seine Mac-user-Schatten springt, tippt man das bisschen am besten direkt an der Kommandozeile ein. Da geht sowohl im Programm „Terminal“ wie auch im X11-Terminal.

Bei mir würde das für einen ersten checkout so aussehen:

Weil cvs immer in das aktuelle Verzeichnis auspackt, wechseln wir erst mal da hin.

```
cd /Users/Shared/OOoQA/
```

In diesem Verzeichnis wird dann später beim eigentlichen checkout das Verzeichnis qatesttool angelegt, in das alle Scripte usw. geschrieben werden.

Als nächstes muss man sich einmal auf dem Server einloggen:

```
cvs -d :pserver:anoncvs@anoncvs.services.openoffice.org/cvs login
```

Bei der Passwortabfrage einfach return drücken, ohne etwas einzugeben. Das login merkt sich cvs, man muss es später nicht mehr machen.

Und dann der eigentliche checkout:

```
cvs -d :pserver:anoncvs@anoncvs.services.openoffice.org/cvs -z3 -R co qa/qatesttool
```

und los geht’s…

Jeder weitere checkout bzw. update kann dann mit

```
cd /Users/Shared/OOoQA/
cvs -d :pserver:anoncvs@anoncvs.services.openoffice.org/cvs -z3 -R co qa/qatesttool
```

gemacht werden.

Da kommen dann auch „Es ist eine Datei im Weg“-Meldungen - aber ich denke, die kann man ignorieren.

2.2 Referenzen

Eine detaillierte Anleitung zum TestTool kann aus dem CVS herunter geladen werden. Dort finden sich letztlich auch das TestTool und die jeweiligen Scripten. Hat man die Scripte aus dem CVS geholt, liegt schon einiges im Unterordner „documentation“

http://qa.openoffice.org/qatesttool/index.html

Auf das Ooo-CVS kann auch online lesend zugegriffen werden:

http://qa.openoffice.org/source/browse/qa/qatesttool/

dort finden sich das aktuelle TestTool, die Scripte usw. auch einzeln zum download.

2.3 Andere Tests

Neben den GUI-Tests gibt es noch die TCM-Tests (Test Case Management). Diese sind umgangssprachliche Beschreibungen von normalen Vorgängen im Rahmen der Programmnutzung. Die der Tester selbst manuell nachvollziehen soll. Ich zitiere mal von http://wiki.services.openoffice.org/wiki/De:2.0.2_TCM_Test:

„Die Tests sollen nicht der Freigabe einer Version dienen, sondern helfen, sich einen Überblick über allgemeine Qualität und Bedienbarkeit zu verschaffen. Hauptaugenmerk ist dabei die Lokalisierung! Das beinhaltet die Übersetzung der Oberfläche und Hilfe, aber auch Umgang mit Sonderzeichen, Sprachabhängigen Formatierungen ... Die im TCM beschriebenen Testfälle sollen als "Leitfaden" verstanden werden, welche Bereiche der Software zu testen sind. Neben dem Erfüllen des konkreten Testfalles soll dabei auch erfasst werden, ob OpenOffice.org im gegebenen Bereich verständlich und logisch bedienbar ist oder dem Anwender unnötige Hürden in den Weg legt.“

Wer sich eher für diese Art von Tests interessiert, soll bitte Kontakt mit dem Autor oder den auf der Seite genannten Verantwortlichen aufnehmen.

Man findet auch hier nähere Infos dazu: http://l10n.openoffice.org/localization/About_TCM.html

3 Installation der Testumgebung

3.1 Neuen Benutzer anlegen

Es ist eine gute Idee, die Testumgebung und das zu testende Programm in einen neuen Benutzer anzulegen, damit vorhandene Konfigurationen nicht beschädigt werden (z. B. Benutzereinstellungen). Man kommt dann auch öfters in den Genuss der tollen „Benutzer-Umschalt-Animation“ von Apple :-)

Am einfachsten geht das, wenn man in den Systemeinstellungen – Benutzer einen neuen User anlegt – hier in den Beispielen den Benutzer „test“.

Für jede neue Version, die man testen will, sollte im test-user nicht nur das OpenOffice.org-Programm, sondern auch das Verzeichnis „OpenOffice.org 2.0“ im Library-Ordner des test-users (~/Library/Application Support/OpenOffice.org 2.0) gelöscht werden.

Man kann auch den Benutzer „test“ ganz neu anlegen – in den Systemeinstellungen > Benutzer den alten Testuser löschen und neu anlegen. Dann ist allerdings auch die testtool-Konfiguraton weg.

Das meiste, was Apple einem neuen Benutzer automatisch mit auf den Weg gibt, braucht dieser Nutzer nicht, man kann die Ordner „Bilder“, „Filme“usw. also der Übersichtlichkeit wegen erst mal löschen.
Die Ordner „Öffentlich“ und „Library“ sollte man beibehalten – der erste stellt eine einfache Austauschmöglichkeit mit anderen Benutzern auf dem Rechner dar, der zweite ist sowieso für das System notwendig.

3.2 Vorbereitung des Testverzeichnisses

Es empfiehlt sich, im Verzeichnis „Für alle Benutzer“ einen neuen Ordner „OOoQA“ anzulegen. (Bei den weiteren Pfadangaben in diesem Dokument wird von diesem Installationsort ausgegangen)

In diesen Ordner sollte ein Verzeichnis „logs“ angelegt werden. Ebenfalls sollten hierhin der Ordner qatesttool mit den Testscripten (die „sandbox“ des CVS-client) abgelegt werden.

Will man mehrere Versionen testen, dann empfiehlt es sich, hinter “OOoQA“ noch die Version zu schreiben, um mehrere Script-Versionen parallel halten zu können.

Jetzt ist eine gute Gelegenheit, den CVS-checkout wie oben beschrieben zu machen.

3.3 Vorbereiten von OOo

Jetzt braucht man noch eine zu testende Version von OpenOffice.org. Es wird dringend empfohlen, nur auf eine parallel zur Arbeitsversion installierte *Kopie* der Installation zu testen. Bei einem crash des TestTools bleibt die benutze OOo-Version möglicherweise beeinträchtigt zurück, z. B. weil Konfigurationsinformationen geändert wurden.

Um dieses Risiko auszuschließen sollte die zu testende Version neben der „normalen“ in einen eigenen, am besten neu angelegten Benutzer installiert werden. Ich installiere z. B: die zu testende OOo-Version immer ins home-Verzeichnis des test-users.

3.3.a Name der zu testenden Ooo-Installation ändern (nur bis 2.03RC3 notwendig)

Dabei muss vor allem das Leerzeichen aus dem Namen des Pakets entfernt werden. Sinnvoll ist es, die zu testende Version als OpenOffice2.0 (ohne Leerzeichen vor der 2) in das Home-Verzeichnis des test-users zu kopieren (hiervon wird im weiteren Beispielen ebenfalls ausgegangen). Man kann sie auch Ooo2.0Test oder wie auch immer nennen – wichtig ist: Kein Leerzeichen im Namen, damit kommt das TestTool derzeit nicht klar.

Ab Version 2.0.3RC4 kann der Name bleiben, wie er ist.

3.3.b Das script „soffice“ kopieren

Zuerst ins „program“-Verzeichnis wechseln[1]

```
cd /Users/test/OpenOffice.org\ 2.0.app/Contents/MacOS/program/
```

dann

```
cp soffice testtool
```

Das Script soffice setzt einige Umgebungsvariablen und startet dann ein Program, das wie es selbst heißt, aber noch ein „.bin“ angehängt hat. Diese Umgebungsvariablen benötigt auch das testtool.bin – daher sollte es ebenfalls über das script aufgerufen werden. Da das script aber sich selbst & .bin startet, muss es für das testtool halt kopiert und umbenannt werden.

Angeblich kann das ab Version 2.0.3 unterbleiben – testtool.bin lässt sich direkt ohne Fehlermeldung starten. Beim RC4 funktioniert es nur auf PPC, auf Intel noch nicht - ausprobieren.

3.3.c X11 anpassen

Als letztes sollten noch die X11-Einstellungen an OOo angepasst werden: Tastatur-Kurzbefehle aus und Farben auf 16,7 Mio.

Man kann sich dabei auch gleich einen neuen Menüeintrag zum starten des Testtools anlegen: In Programme > Menü anpassen einfach den Aufruf als weiteren X11-Menübefehl hinterlegen. Dazu „Hinzufügen“ klicken, einen sinnigen Namen (z. B. TestTool) vergeben und den vollständigen Pfad zu dem gerade kopierten script als Befehl einkopieren.

1 Der Backslash „\“ vor dem Blank muss an der Kommandozeile geschrieben werden, damit das Blank als Teil des Namens erkannt wird. Man kann stattdessen auch den Name in Anführungszeichen setzen:

```
cd /Users/test/“OpenOffice.org 2.0.app“/Contents/MacOS/program/
```

4 Starten und Konfigurieren des Testtools

4.1 Der erste Start

Das TestTool wird aus dem X11-Terminalfenster heraus mit dem Befehl

```
~/OpenOffice.org\ 2.0.app/Contents/MacOS/program/testtool
```

gestartet.

Sinnigerweise würde man sich ein Alias davon auf den Schreibtisch oder ins Dock legen – das funktioniert aber leider nicht. Aber mit „Tab“ und „Pfeil nach oben“ in der Shell kann man es sich einfacher machen (oder man legt wie oben beschrieben einen Befehl in X11 an).

Für nicht-Linuxer:

Der **Tab** ergänzt einen begonnene Namen automatisch. Gehen Sie in das Terminalfenster, Tippen Sie „/A“ und drücken die Tab-Taste, wird auf „/Applications“ ergänzt. Funktioniert aber nur bei eindeutigen Namen.

Hat man einen Befehl einmal eingegeben, kann man ihn mit „**Cursor nach oben**“ - Taste wieder aus dem „Gedächtnis“ des Terminalfensters holen; das bleibt selbst bei einem Neustart erhalten. Je nach Einstellung kann man dabei fast beliebig viele Befehle zurück gehen.

Bevor man Lostesten kann, muss man aber noch einige Konfigurationseinstellungen vornehmen.

4.2 Einrichten des TestTools

Als erstes muss das TestTool gestartet werden.

Dann im Menü „Extras“ die Einstellungen aufrufen.

4.2.a Reiter „Generisch“

Hier müssen zwei Parameter und die zugehörigen Werte eingetragen werden: „GUI Platform“ mit „12“ und „OooProgramDir“ mit dem Verzeichnis des zu testenden OpenOffice.org-Version – entsprechend den obigen Vorgehen also[1]:

```
~/OpenOffice.org 2.0.app/Contents/MacOS/
```

Wie werden diese nun eingetragen?

In das oberen Eingabefeld („Bereich“) wird „GUI Platform“ geschrieben. Die Taste „Neu“ rechts daneben klicken.

Der Cursor steht jetzt im unteren Eingabefeld („Einstellung“). Hier einfach „12“ 'reinschreiben und wieder „Neu“ (diesmal das untere) klicken. Fertig.

Jetzt oben den Text „GUI Platform“ einfach mit „OOoProgramDir“ überschreiben – wieder „Neu“ klicken und unten den Pfad eintragen – nochmal unten „Neu“ klicken - fertig.

4.2.b Reiter „Profile“

Hier kann man sich ein Profil „Standard“ anlegen – einfach wieder bei „Profile“ eintragen und „Neu“ klicken. Dann müssen noch die drei Pfade eingetragen werden.

Ist das Tool wie oben beschrieben installiert, ergeben sich hier folgende Einträge:

- Log Basisverzeichnis: /Users/Shared/OOoQA/log
- Basisverzeichnis: /Users/Shared/OOoQA/qatesttool
- HID Verzeichnis: /Users/Shared/OOoQA/qatesttool/global/hid

1 Die Tilde ~ (Option-n und dann Leerzeichen, da es ein deadkey ist) steht in Unix-Systemen als Abkürzung für das Home-Verzeichnis. Das ist auf dem Mac das Verzeichnis /Users/<loginname>/
Es darf aber auch die Tilde in die Befehlszeile geschrieben werden.

„Automatisch neu laden“ sollte angehakt ein, die beiden anderen Optionen können ausgeschaltet bleiben.

4.2.c Reiter „Misc“

Hier sollte als Host „localhost“ (ohne Anführungszeichen) stehen, damit auf der lokalen Maschine getestet wird.

4.2.d Reiter „Schrift“

Bestimmt Schriftart und -größe in der log-Datei.

4.2.e Eintrag in die Konfigurationsdatei

Als letztes muss noch die Konfigurationsdatei .testtoolrc manuell ergänzt werden. Dazu erst mal das TestTool beenden und dann an der Kommandozeile den Befehl

```
pico ./Library/.testtoolrc
```

eingeben. Es öffnet sich ein kleiner Texteditor. Mit den Kursortasten ganz ans Ende fahren und folgenden Eintrag machen:

```
[Crashreporter]
UseProxy=false
ProxyServer=
ProxyPort=
AllowContact=false
ReturnAdress=uwe@office.org
```

Im Ergebnis sollte die Datei in etwa so aussehen wie im Anhang 1 wiedergegeben

4.2.f ...oder einfacher:

Den im Anhang 1 angegebene Text als nur-Text-Datei unter dem Namen .testtoolrc im richtigen Verzeichnis abspeichern.
Danach das TestTool starten und die Einträge nochmal kontrollieren – fertig.

5 Und jetzt geht's los

5.1 Der erste Test

Eine ggf. laufende Instanz des Test-OOo muss beendet werden – das Testscript startet OOo neu. Prinzipiell sollte kein anderer soffice – Prozess mehr laufen . Das kann man mit Dienstprogramme > Aktivitäts-Anzeige überprüfen. (Oben recht „alle Prozesse" auswählen und im Filter „soffice" eingeben)

- Es bietet sich an, das Script „first.bas" zu benutzen ;-) - es liegt im Verzeichnis ~/OOoQA/qa-testtool/framework/first
  - Datei > Öffnen
  - Script auswählen und öffnen. Der Text des Scripts wird angezeigt.
- Programm > Start. Das Script läuft los – unten rechts ist eine Statuszeile.
- Nach dem die Scriptdateien geladen und alles parat ist, wird Ooo automatisch gestartet und der Test beginnt.
- Das Ergebnis wird in ein zweites Dokument geschrieben, das sich automatisch öffnet – das ist die log-Datei. Diese sollte nach Ende des Tests in das log-Verzeichnis gespeichert werden.

5.2 Scripts, die man immer laufen lassen sollte

IM Ordner „framework" finden sich

- **First.bas** Die Abschnitte zu StarOffice dürfen Fehler melden – wir testen ja OpenOffice.org.
- **TopTen.bas** sollte auch immer laufen.

Beide Tests benötigen je ca. 16 Minuten.

5.3 Das script „ooo_releasetest.sh"

Wenn man das qatesttool augecheckt hat, findet man dort im Unterordner „script" und dort nach Betriebssystemen unterteilt zwei Scripts:

- **OooTestRun_unix.sh** – das lässt alle überhaupt verfügbaren Testst laufen. Sollte man nur anwerfen, bevor man in Urlaub fährt.
- **ooo_releasetests.sh** – das sind die aus Sicht der QA wichtigen Tests für ein neues Release, da sie zu großen Teilen die Ressourcen testen, also das, was bei Fehlern das OpenOffice.org unweigerlich zum Crash bringt.

Die Unix-Variante der scripts läuft eigentlich auch auf dem Mac – man muss dazu lediglich einige kleine Anpassungen vornehmen. Dazu kann man das Script einfach mit TextEdit oder jEdit öffen.

Für das Script ooo_releasetests.sh:

Erstens müssen zu Beginn des scripts zwei Pfade angepasst werden:

```
# set location of testscripts
# (the directory, where directory 'qatesttool' exists)
sLocation=/Users/Shared/OOoQA/


# set location of TestTool
# (full path including executable 'testtool')
sTestTool="/Users/ooo-test/OpenOffice.org 2.0.app/Contents/MacOS/program/testtool"
```

(Die Pfade beziehen sich wieder auf die hier im Text verwendete Beispielinstallation.)

Zweitens muss noch der Codeteil

```
# kill office, if exists
```

```
#killall -9 soffice.bin
pkill -9 soffice.bin
ps -fe | grep $USER | grep "soffice.bin" | grep -v "grep"
```

in das hier geändert werden:

```
# kill office, if exists
 /usr/bin/killall -9 soffice.bin
 #pkill -9 soffice.bin
 ps -Aj | grep $USER | grep "soffice.bin" | grep -v "grep"
```

d.h., statt dem pkill muss der /usr/bin/killall – Befehl mit der kompletten Pfadangabe verwendet werden und beim ps – Befehl müssen die Parameter von „-fe“ in „-Aj“ geändert werden

und zuletzt muss noch ein Stück weiter unten

```
if ps -p $testtoolpid > /dev/null ; then
```

bzw,

```
while ps -p $testtoolpid > /dev/null ;
```

geändert werden in

```
if ps -p $testtoolpid | grep $testtoolpid > /dev/null ; then
```

und

```
while ps -p $testtoolpid | grep $testtoolpid > /dev/null ;
```

(also jeweils ein „| grep $testtoolpid “ einfügen.)

Als letztes muss das Script noch lauffähig gemacht werden, d. h. es muss als ausführbare Datei gekennzeichnet werden. Dazu bitte in einer beliebigen Shell in das Verzeichnis wechseln, in dem das Script liegt.

```
cd /Users/Shared/OOoQA/qatesttool/script/unix/
```

Dann den Befehl

```
chmod a+x ooo_releasetests.sh
```

eingeben.

Zum Starten des Scripts muss man den vollständigen Pfad und den Dateinamen des scripts an der X11-Eingabeaufforderung eingeben. Achtung: Das script benötigt ca. einen Tag zum durchlaufen.

Leider wirft es aus unerfindlichen Gründen gelegentlcih Fehler ab, die bei manuellem Starten des Einzeltests nicht kommen - daher bei Fehlern lieber nochmal den betreffenden Test direkt aufrufen.

5.4 Interpretation der Ergebnisse

Das Ergebnis des Tests wird in ein zweites Dokument geschrieben, das sich automatisch öffnet - das ist die log-Datei oder resultfile. Diese sollte nach Ende des Tests in das "log"-Verzeichnis gespeichert werden (passiert automatisch).

Tritt ein Fehler auf, so sollte in den '///-gekennzeichneten Kommentarzeilen stehen, was passiert – wenn nicht, dann darf man gerne allein schon deswegen dem Testautor mal ein Issue schreiben (der Autor steht in jeder einzelnen *.inc- und auch *.bas- Datei unter „owner“ – nicht den Autor der o.g. „superscripte“)

Zunächst einmal sollte man aber *diesen einzelnen* testcase alleine nochmal laufen lassen, um evtl. den Fehler live zu sehen. Kann man es reproduzieren, ist man auf jeden Fall dichter am Problem. Das einzelne Resultfile sollte man dann auch unter 6. an das Issue hängen und nicht etwa den ganzen testlauf.

Hierzu kann man im testscript auch Brechpunkte (**breakpoints**) setzen – die halten das Script an der entsprechenden Stelle an und warten einfach auf eine „OK“ zum weitermachen. Script (oder include-file) öffnen, mit dem Cursor an die gewünschte Stelle fahren. Im Menü den Befehl Programm > Brechpunkt setzen/löschen geben; der Brechpunkt wird durch einen roten Punkt neben dem Script angezeigt.

Bei der Eingrenzung der Bedingungen, unter denen ein Fehler auftritt (falls er das nicht immer tut – und viele Fehler tun und diesen gefallen ja nicht) ist viel von der Phantasie des Benutzers abhängig. Mein Lieblingsvorgehen: Herausfinden, was genau im testscript nicht funktioniert hat und dann versuchen, das manuell im Programm nachzuvollziehen.

Wenn ich den Fehler habe, mit verschiedenen Variationen im Umfeld ausprobieren, um möglichst genau die Bedingungen benennen zu können, unter denen er auftritt. Gerade bei portierungsspezifischen Fehlern kann das sehr hilfreich sein.

6 Melden von Fehlern

Fehler werden grundsätzlich in Issuetracker (früher Issuezilla) abgewickelt.

Bei Unsicherheiten kann man sich vorher auf der Mailingliste des mac-porting-teams oder im chat auf irc.freenode.net, channel openoffice.org-de (deutesche gruppe) oder openoffice.org (internationale Entwickler) schlau machen.

Es ist meistens durchaus sinnvoll, das log des testtools mit dem Fehler drin an den issue anzuhängen (das Resultat des einzelnen Tests – nicht einen ganzen Testlauf, falls man eines der größeren Scripten aus 5. hat laufen lassen)

6.1 RTFM!

Grundlegend:

http://qa.openoffice.org/issue_handling/pre_submission.html

oder auf Deutsch:

http://de.openoffice.org/dev/pre_submission_de.html

etwas ausführlicher

http://qa.openoffice.org/issue_handling/project_issues.html

Neue Issues:

http://qa.openoffice.org/issue_handling/bug_writing_guidelines.htm

6.2 Wohin damit?

Fehler in OOo, die unter Mac OSX auftreten und die Mac-port-spezifisch sind, sollten in Issuezilla component „porting“, subcomponent „MacOSX“ reported werden.

Wenn es sich dabei offensichtlich nicht um Mac OS-port-spezifische Fehler handelt, sollte Sie den entsprechenden Komponenten zugeordnet werden.

7 FAQ

7.1 Issue tracker macht Pickel

Beipackzettel lesen:

http://de.openoffice.org/dev/pre_submission_de.html

7.2 Crashreporter wird angemeckert

Es erscheint folgende Meldung auf der X11-Konsole:

```
UwesPB:~ OOoTest$ /Applications/OpenOffice.org 2.0.app/
Contents/openoffice.org/program/testtool.bin
sh: line 1: crash_report: command not found

Fatal exception: Signal 6
Please turn on Enable Crash Reporting and Automatic Display of Crashlogs in the Console
application
Abort trap
```

Unter Mac OS X ist der Crashreporter nutzlos (da der port nicht von Sun gemacht wird), daher ist er standardmäßig deaktivert. Es gibt aber testscripte, die das Vorhandensein des Crahsreporter testen und dann mit dieser Fehlermeldung reagieren. Kann ignoriert werden.

7.3 Was passiert noch alles auf meinem Rechner

Das TestTool legt eine unsichtbare Datei Namnes .testtoolrc an, die die Konfigurationsdaten des TestTools enthält. Bis Version 2.0.1 liegt die im Home-Verzeichnis des Benutzers, danach in ~/Library/Application Support/ Das kann man sich mit

```
nano <Pfad zur Datei>/.testtoolrc
```

anschauen (ctrl + x drücken, um nano, den kleinen Texteditor, zu beenden)

8 Anhang 1: Die Datei .testtoolrc

Die Datei liegt für ältere Versionen im Home-Verzeichnis des users.

Ab Version 2.0.2 liegt sie in ~/Library/Application Support/ (also in der Library des jeweiligen users)

Das hier wiedergegeben Beispiel funktioniert für:

- Benutzer „ooo-test“,
- für den im Home-Verzeichnis die OpenOffice.org-Installation mit entferntem Blank ist.
- In /User/Shared/ sind die Ordner „OOoQA“ und in diesem die Unterordner
  - „qatesttool“ und
  - „log“

  angelegt (in qatesttool befinden sich die Testscripte – letztlich der checkout aus dem CVS)

Die Datei würde dann so aussehen:

```
[WinGeom]
WinParams=0,0,876,875;53;0,44,1280,976;

[Misc]
ServerTimeout=4500
StopOnSyntaxError=0
AutoReload=1
AutoSave=0
ScriptFontName=Courier
ScriptFontStyle=normal
ScriptFontSize=12

[Path]
BaseDir=/Users/Shared/OOoQA/qatesttool
LogBaseDir=/Users/Shared/OOoQA/log
HIDDir=/Users/Shared/OOoQA/qatesttool/global/hid
*.bas=/Users/Shared/OOoQA/qatesttool/framework/first
LastFilterName=Quelltexte (*.BAS)

[GUI Platform]
Current=12
All=12

[Communication]
TTPort=12479
Host=localhost
UnoPort=12480

[LRU]
LRU1=/Users/Shared/OOoQA/qatesttool/framework/first/first.bas
MaxLRU=4

[OOoProgramDir]
Current=/Users/ooo-test/OpenOffice.org 2.0.app/Contents/MacOS
All=/Users/ooo-test/OpenOffice.org 2.0.app/Contents/MacOS

[Crashreporter]
UseProxy=false
ProxyServer=
ProxyPort=
AllowContact=true
ReturnAddress=uwealtmann@openoffice.org
```

9 Credits

Autor: **Uwe Altman** uwealtmann@openoffice.org

Kontakt: **Deutsches Dokumentationsprojekt OpenOffice.org –** http://de.OpenOffice.org

Versionshistorie:

Version	*Date*	*Kommentar*
1.0	22.05.06	Erste Version
1.1	23.5.06	Einarbeitung der Kommentare von Jörg Sievers, Christian Lohmeier und Cedric Bosdonnat
1.2	24.5.06	Anpassungen des scripts 000_releasetests.sh vervollständgt.
1.3	14.8.06	Einige kleine Änderungen, insbesondere ab Version 2.0.3, Nachgetragen.

10 Lizenz

Appendix

Public Documentation License Notice

The contents of this Documentation are subject to the Public Documentation License Version 1.0 (the "License"); you may only use this Documentation if you comply with the terms of this License. A copy of the License is available at *http://www.openoffice.org/licenses/PDL.html*.

The Original Documentation is Smoketests_auf_dem_Mac-1.0de.odt.The Initial Writer of the Original Documentation is Uwe Altmann Copyright © 2006. All Rights Reserved. (Initial Writer contact(s): uwealtmann@openoffice.org).

Contributor(s):

NOTE: The text of this Appendix may differ slightly from the text of the notices in the files of the Original Documentation. You should use the text of this Appendix rather than the text found in the Original Documentation for Your Modifications.