COSIMA unterm Apfel

Installation von Ghostscript

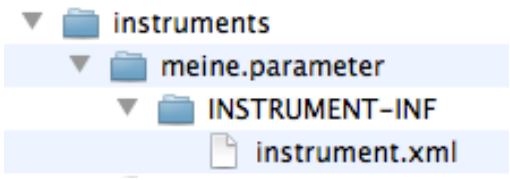
Mit Version 5.1 von Sign Live! CC kann über COSIMA sehr einfach aus jeder Anwendung heraus durch einfaches Drucken ein komplexer Konvertierungs- und Signaturworkflow ausgelöst werden. Obwohl diese Vereinfachung primär für die Windows-Plattform gedacht ist, funktioniert COSIMA auch unter Mac OS X. Dazu müssen aber zuvor einige Vorkehrungen getroffen werden.
So benötigt man zur Konvertierung des Postscript-Datenstroms, der aus der Druckschnittstelle kommt analog zu Windows einen PS-nach-PDF-Konverter. Diese Rolle übernimmt *ghostscript*. Um das aktuelle *ghostscript* auf Snow Leopard 10.6 zu installieren, verwendet man am besten *MacPorts* oder die graphische Alternative *Porticus*. Außerdem müssen auf dem Mac auch die Entwicklertools, also Xcode 3 installiert sein.
In einer Shell gibt man dazu folgenden Befehl ein:

```
[user@host]> sudo port install ghostscript +cups +no_x11 +universal
```

Die Parameter sorgen dafür, dass das aktuelle *ghostscript*-Package (derzeit Version 9.0.1) ohne X-Windows-Abhängigkeit installiert wird. Dadurch lässt sich *ghostscript* problemlos von der Kommandozeile aus verwenden ohne zuvor einen X11-Server gestartet zu haben. Testen kann man die erfolgreiche Installation durch den Aufruf von *ghostscript* in der *Terminal.app*:

```
[user@host]> gs
GPL Ghostscript  9.01 (2011-02-07)
Copyright (C) 2010 Artifex Software, Inc.  All rights reserved.
This software comes with NO WARRANTY: see the file PUBLIC for details.
GS> quit
```

Im nächsten Schritt müssen wir *Sign Live!* mitteilen, wo die *ghostscript*-Installation zu finden ist. Dazu legen wir im Profil-Verzeichnis der Anwendung `~/Library/Application Support/SignLiveCC_5.1/instruments` folgende Struktur an.



Die eigentlichen Anweisungen befinden sich dann in der XML-Datei *instrument.xml* und lauten wie folgt. Diese XML-Datei kann auch über [gs instrument] direkt heruntergeladen werden.

```
<?xml version="1.0" encoding="UTF-8"?>
<instrument id="com.mein.cosima.printer">
    <requires>
        <prerequisite
instrument="com.cabaret.pdf.exchange.ps.ExchangePSInstrument"
absent="skip"/>
    </requires>
    <extension point="com.cabaret.exchange.ps.ghostscript">
        <ghostscript>
            <library path="/opt/local/lib/libgs.dylib"/>
            <!-- this is autodetected normally -->
            <include path="/opt/local/share/ghostscript/9.01/lib"/>
            <include path="/opt/local/share/ghostscript/9.01/Resource"/>
            <include path="/opt/local/share/ghostscript/fonts"/>
            <arg value="-dNOPAUSE"/>
            <arg value="-dNOPAGEPROMPT"/>
            <arg value="-dNOPROMPT"/>
            <arg value="-dBATCH"/>
            <!-- ghostscript stdout is captured
                 <arg value="-q"/>
                 <arg value="-dQUIET"/>
                 <arg value="-sstdout=${${}
processor.output}.ghostscript.log"/>
                 -->
            <arg value="-I${${}processor.includepath}"/>
            <arg value="-sOutputFile=${${}processor.output}"/>
            <arg value="-sDEVICE=pdfwrite"/>
            <arg value="-r300"/>
            <arg value="-c"/>
            <arg value=".setpdfwrite"/>
            <arg value="-c"/>
            <arg value="save"/>
            <arg value="-c"/>
            <arg value="pop"/>
            <arg value="-f"/>
            <arg value="${${}processor.input}"/>
        </ghostscript>
    </extension>
</instrument>
```

Damit findet Sign Live! CC den *ghostscript*-Prozessor, wenn ein Druckvorgang auf den Sign Live! COSIMA Printer ausgelöst wird.

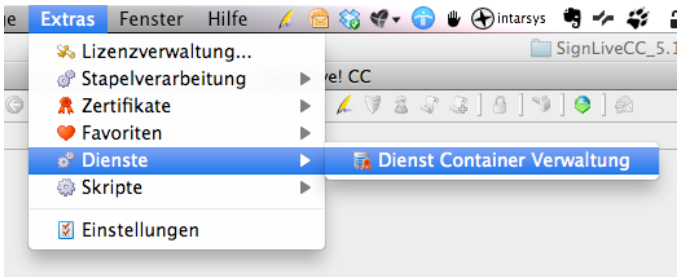
Exkurs: Warum der Umweg über Postscript?

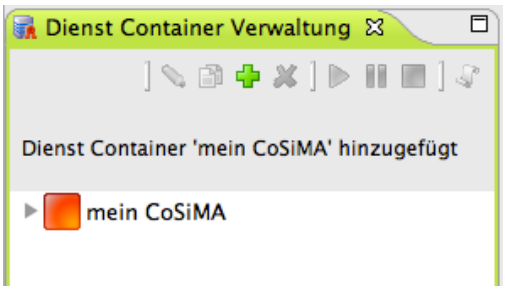
Vermutlich stellt sich die Frage, warum man auf einer Plattform wie Mac OS X, die von sich aus bereits alle Grafik und Druckausgaben in PDF erstellt, einen Umweg über einen Postscript-Konverter wie *ghostscript* machen muss. Das Problem besteht darin, dass der ursprüngliche PDF-Code, den eine Cocoa-Anwendung aus den Fensterinhalten erzeugt, nicht abgefangen werden kann und von Apple selbst über das CUPS-Drucksystem zunächst in ein Postscript-Format gewandelt wird, um dann anschließend z.B. an Postscript-Drucker weitergeleitet zu werden.
Natürlich kann man in Mac OS X im Druckdialog jederzeit angegeben, dass man die Druckausgabe direkt als PDF speichern möchte oder an die *Vorschau.app* übergeben will. Aber auch hier passiert nichts anderes, als dass im Mac-Betriebssystem ein Postscript-nach-PDF Konverter namens *pstopdf* aufgerufen wird, der analog zu *ghostscript* funktioniert. Bei *pstopdf* handelt es sich um einen relativ alte Version des Adobe Distillers für UNIX, den Apple vor einigen Jahren lizenziert und leicht angepasst hat. Im Gegensatz zu *ghostcript* versteht *pstopdf* wesentlich weniger Steuerargumente, da es in der Regel nicht vom Anwender direkt aufgerufen wird.
Wichtig ist, dass auch im Mac-Drucksystem immer eine Wandlung von Postscript zu PDF stattfindet und ein direktes Abgreifen des „Ur-PDF“ auf Shell-Ebene nicht möglich ist. Deshalb existiert in vielen Mac-Programmen häufig ein Menüpunkt „Exportieren als PDF …“.

COSIMA wird angelegt

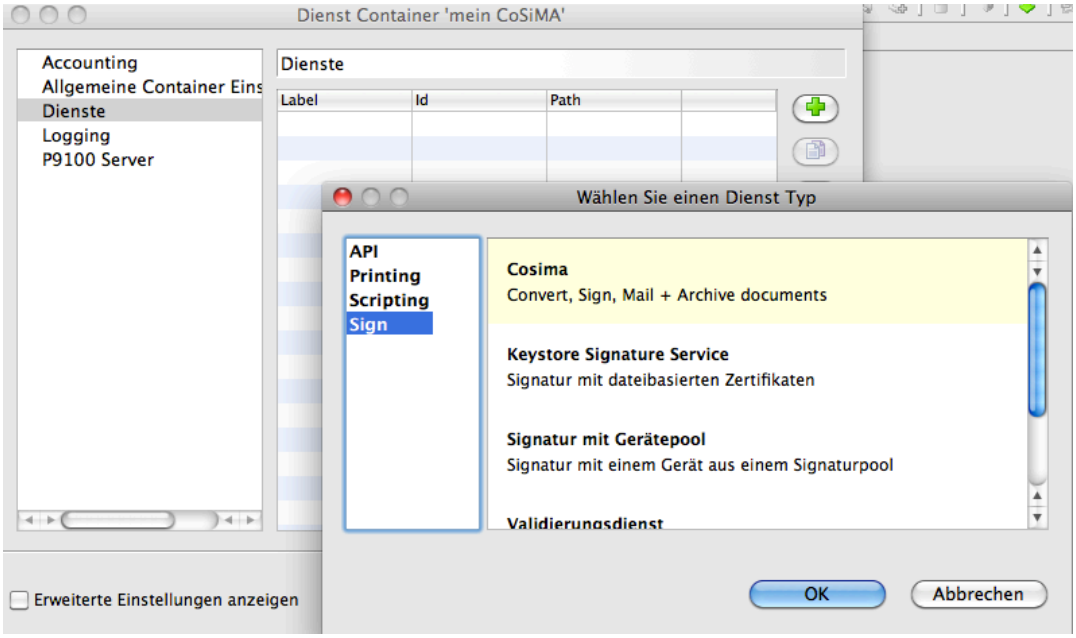
Nun wollen wir nun die Verbindung zwischen dem Druckdialog in Mac OS X und *Sign Live!* sowie die Aktivierung von COSIMA herstellen. Dazu richtet man über *Sign Live!* einen neuen Drucker ein. Nach dem Start von *Sign Live!* wird ein neuer Dienste-Container vom Typ *P9100* angelegt und dieser dann mit einem geeigneten Namen bezeichnet, z.B. „mein COSIMA“.

Ein Doppelklick auf „mein COSIMA“ öffnet die Konfigurationseinstellungen des Dienstecontainers. Dort muss nun ein neuer Dienst vom Typ „Sign->COSIMA“





hinzugeführt werden.

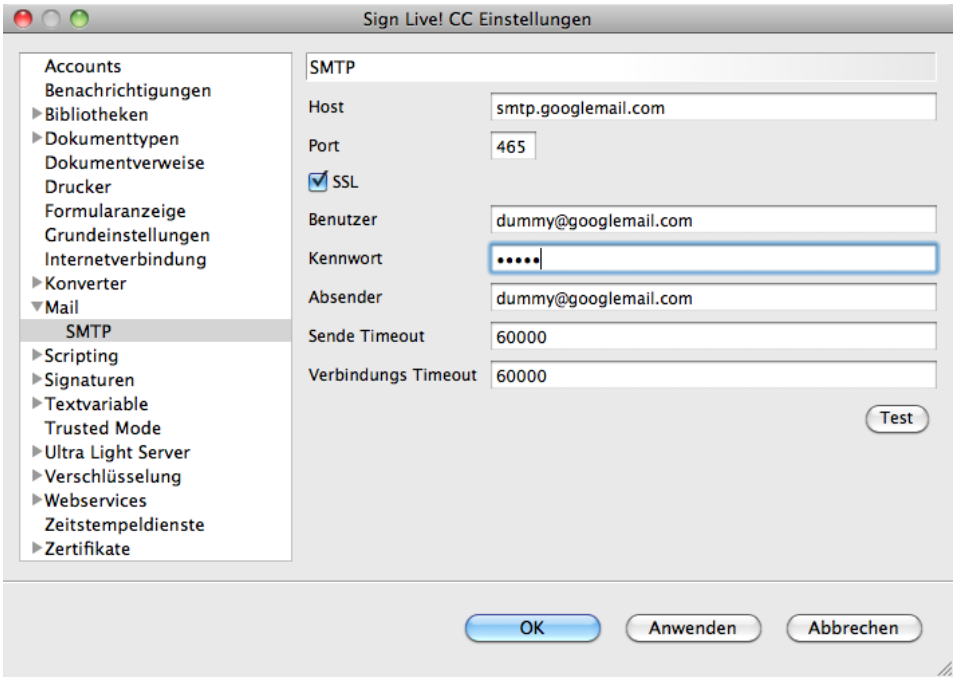


COSIMA versendet Mails

Bevor der neue Dienst gespeichert wird, ändern wir noch die Einstellungen des Prozessschritts „6 – Mail“. Dort ist standardmäßig angegeben, dass der Mail-Versand über „Simple MAPI (mit GUI)“ erfolgen soll. Da auf dem Mac diese Microsoft-Schnittstelle nicht existiert, ändern wir dies auf „Versand per SMTP“. Dabei wird später die signierte Datei direkt zu einem Mailserver via SMTP gesendet, ohne noch zuvor den Umweg über Mail.app oder andere Mail-Clients zu gehen.

Damit dies funktioniert, müssen natürlich die Zugangsdaten für den Mailserver erfasst werden. Da diese Angaben global für die gesamte Anwendung gelten, findet sich die

Eingabemaske unter „Extras->Einstellungen->Mail->SMTP“. Das notwendige Kennwort wird aus Sicherheitsgründen nicht im Klartext dargestellt.
In unserem Beispiel haben wir die Angaben zu einem Google Mail Konto erfasst. Natürlich sind auch beliebige andere SMTP-Server möglich. Um die Gültigkeit der Eingaben zu überprüfen, kann durch Klick auf den Knopf „Test“ eine entsprechende Testmail an einen beliebigen Adressaten von dem in der Maske erfassten SMTP-Konto aus gesendet werden. Ist dies erfolgreich, steht dem Mailversand in COSIMA nichts mehr im Wege.



Neben dem Schritt „Mail“ können noch zahlreiche weitere Prozessschritte eingerichtet und parametrisiert werden. Für den Anfang soll dies aber zunächst einmal genügen.

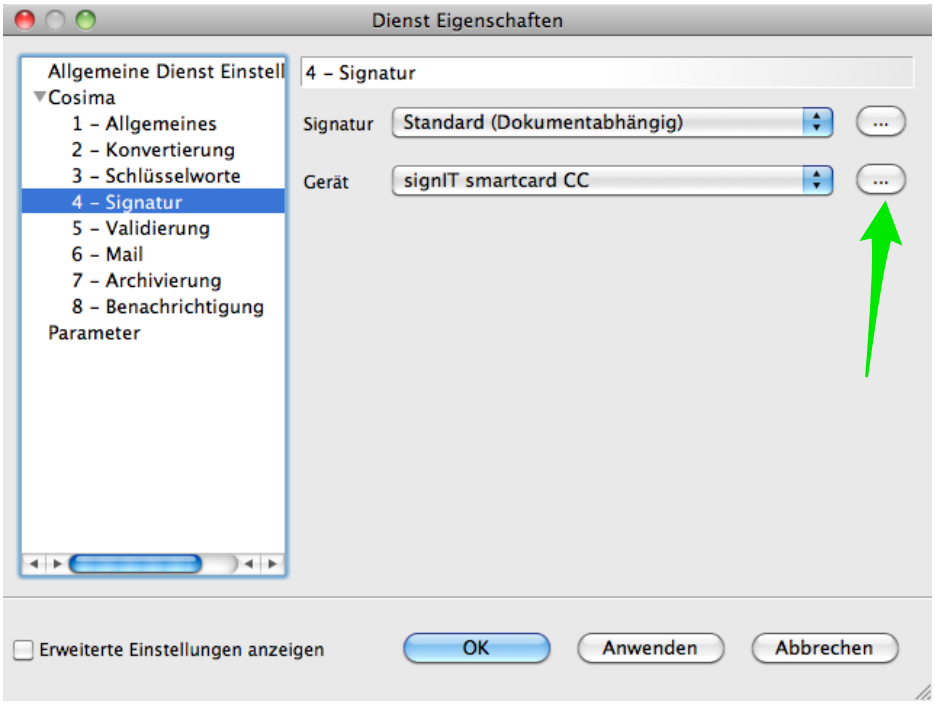
COSIMA als Drucker

Der COSIMA-Dienst kann nun gestartet werden, wodurch auf dem Mac ein neuer Netzwerkdrucker sichtbar wird. In den Systemeinstellungen richtet man einen neuen Drucker vom Typ *IP* ein. Als Protokoll wählen wir *HP Jetdirect - Socket*, als Adresse geben wir einfach localhost ein. Für die Warteliste kann „L1“ angegeben werden. Als Name bietet sich z.B. *Sign Live! COSIMA Printer* an. Als Druckertyp genügt *Allgemeiner Postscript-Drucker.* Mit „Hinzufügen“ schließen wir diese Installation ab.
Aktivieren wir nun in einem Programm wie OpenOffice oder Pages den Druck-Befehl und wählen als Zieldrucker den eben eingerichteten *Sign Live! COSIMA Printer*, so drucken wir das jeweilige Dokument quasi direkt in *Sign Live!*.

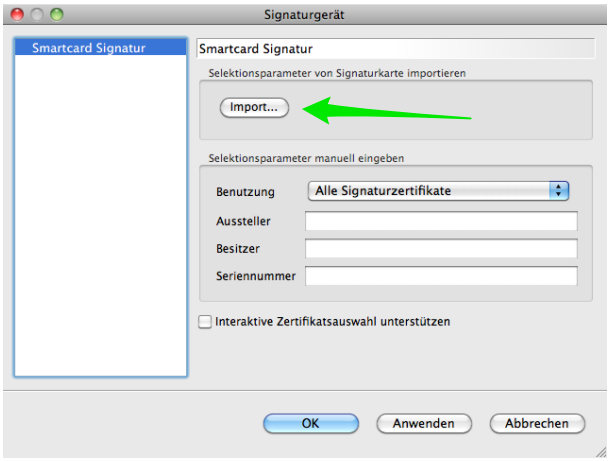
Und nun die Signatur

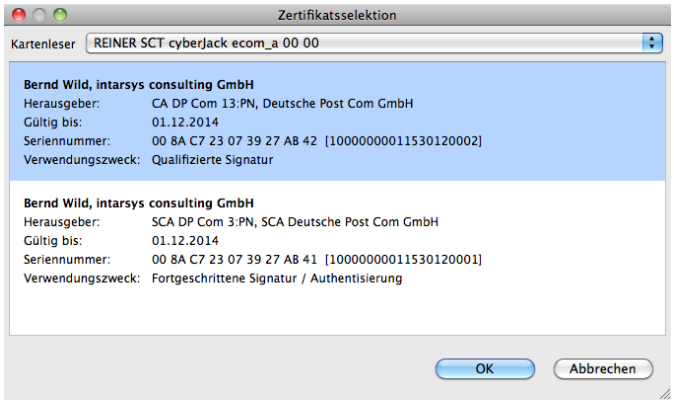
Welche Signaturform für das Signieren des Dokuments eingesetzt werden soll, legt man in den COSIMA-Einstellungen in Abschnitt „4 - Signatur“ fest. Standardmäßig stehen die Einstellungen auf *dokumentenabhängig* , d.h. PDF-Dokumente werden eingebettet sichtbar signiert und als Signatureinheit kommt *signIT easy* zum Einsatz, die Signatur mit Softwarezertifikaten. Soll ein anderes als das Demozertifikat aus dem Zertifikatsspeicher verwendet werden, so kann dies nun hier eingestellt werden.

Wir wollen mit der Smartcard signieren, also qualifizierte elektronische Signaturen erstellen, wie sie insbesondere für die Rechnungssignatur erforderlich sind. Dazu wählen wir als Signaturgerät „signIT smartcard CC“ aus.

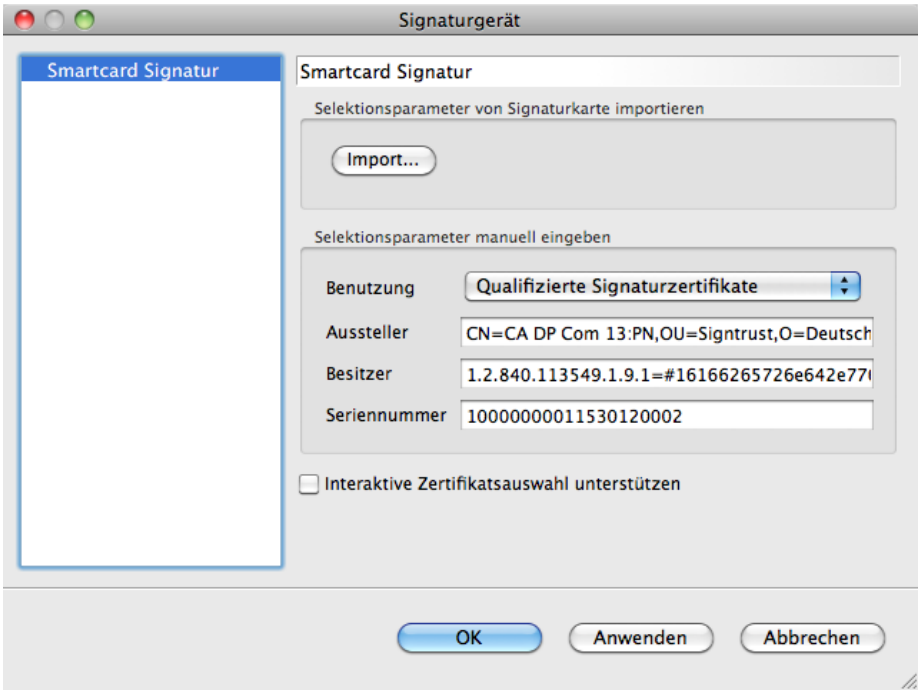


Danach muss von der im angeschlossenen Kartenleser befindlichen Smartcard das betreffende Zertifikat selektiert werden.



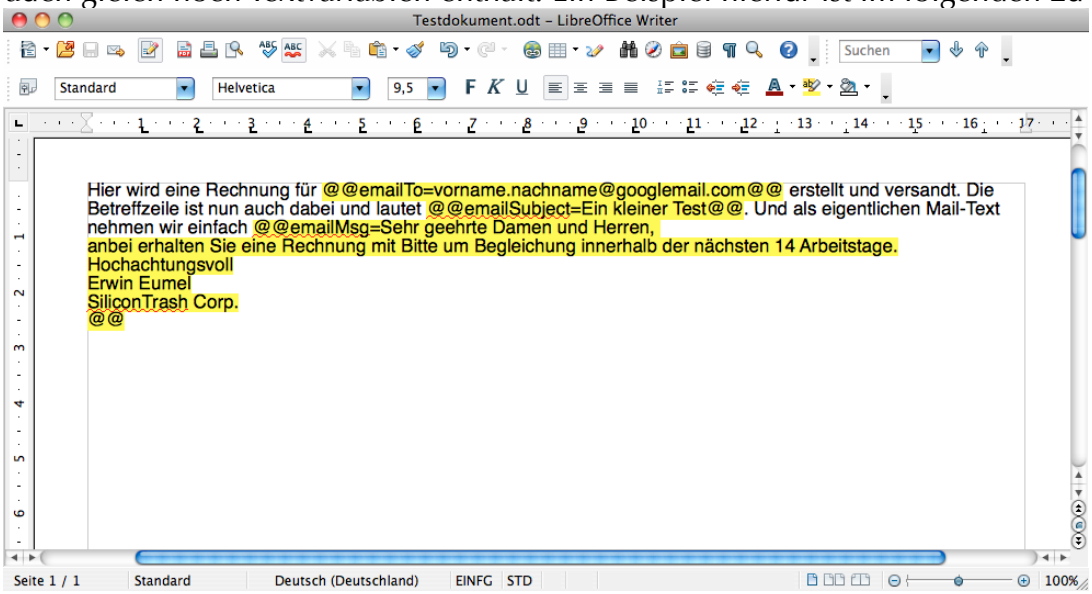


Ist das qualifizierte Signaturzertifikat ausgewählt, so werden die Kenndaten des Zertifikats als Selektionsparameter angezeigt. Ein Zertifikat wird i.d.R. über seine Seriennummer eindeutig identifiziert.



Textvariablen

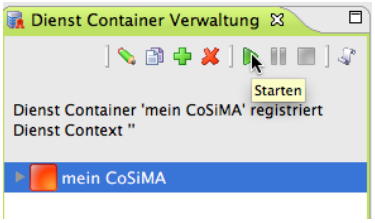
Im letzten Teil dieses Tutorials soll nun noch die Textmustererkennung besprochen werden. Damit ist es möglich, aus dem Dokumententext ganz bestimmte Textvariablen zu extrahieren, die für die Weiterverarbeitung bedeutsam sind. Dazu erstellen wir zunächst in einer geeigneten Textverarbeitung ein Dokument, das signiert und versandt werden soll und dabei auch gleich noch Textvariablen enthält. Ein Beispiel hierfür ist im folgenden zu sehen:

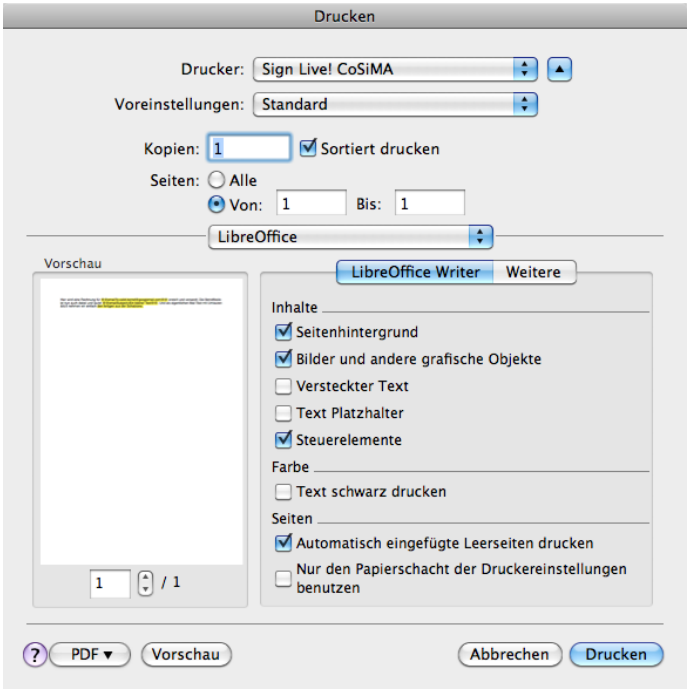


Die gelb hinterlegten Passagen enthalten die Textvariablen, wie sie auch im COSIMA-Handbuch dokumentiert sind. So besagt z.B. das Muster *@@emailTo=vorname.nachname@googlemail.com@@*, dass dieses Dokument nach der Konvertierung und Signatur an den Email-Empfänger mit der Adresse *vorname.nachname@googlemail.com* versendet werden soll. Natürlich kann diese Variablendefinition irgendwo im Dokument stehen. Wird sie mit der Textfarbe weiß ausgezeichnet, so taucht sie sichtbar nirgends auf, wird aber trotzdem von COSIMA erkannt und verarbeitet.
Eine Beschreibung aller vordefinierter Textvariablen findet sich in der Kurzanleitung zu COSIMA [SLCC Cosima Guide].
Achtung: Die Textmustererkennung funktioniert nicht, wenn als Textverarbeitung ein einfacher Editor wie z.B. TextEdit verwendet wird. Dies ist leider eine Eigenart der PDF-Generierung in Mac OS X, die auch dazu führt, dass PDFs aus derartigen Texteditoren zwar korrekt in Vorschau.app dargestellt werden aber nicht mehr durchsuchbar sind.

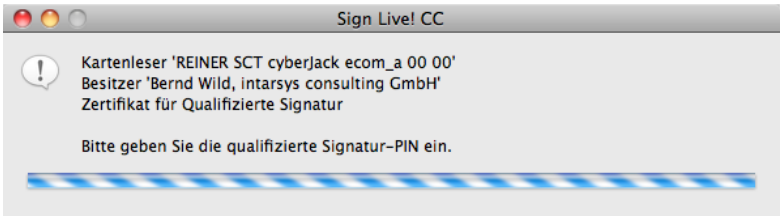
Und nun geht‘s los ...

Nachdem nun alle Einstellungen vorgenommen wurden, können wir den Dienst „mein CoSiMA“ starten. Danach wechseln wir zu unserer Textverarbeitung. Im vorliegenden Tutorial ist dies LibreOffice, natürlich kann man auch Pages, Word oder OpenOffice verwenden. Nun drucken wir das Dokument, das die Textvariablen enthält einfach auf den COSIMA-Drucker.





Man kann dann beobachten, dass die Druckeranwendung kurz im Dock erscheint. Nach einigen Sekunden drängt sich dann Sign Live! CC in den Vordergrund und fordert zur Eingabe der PIN auf dem angeschlossenen Kartenleser auf.



Wird die PIN auf dem Pinpad korrekt eingegeben, fährt der COSIMA-Prozess vollautomatisch mit der Signatur und dem Mailversand fort. Bei Bedarf kann man als letzten Schritt des COSIMA-Prozesses noch eine Benachrichtigung via Growl einbauen. Wie das funktioniert , zeigen wir beim nächsten Mal!

Viel Spass beim Signieren und Mailversenden mit Sign Live! CC COSIMA.

Referenzen und Links

[SLCC Cosima Guide]	Sign Live! CC Cosima Guide, Version 5.1, intarsys consulting GmbH, 2011, in *Sign Live Cosima Guide.pdf*
[gs instrument]	`http://www.intarsys.de/misc/downloads/mac/instrument.xml`