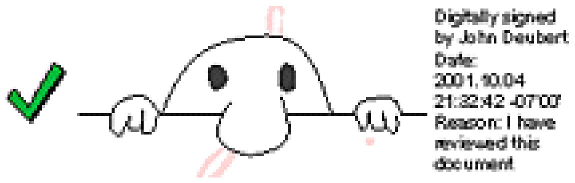
Table of Contents

Acumen Training

Acrobat Signatures, Part 2

Last month, we saw how to sign a PDF file using the Adobe Self-Sign Security mechanism that ships with Acrobat. As you recall, there were 4 steps to this process:



1. Create a User Profile that identifies you to the Acrobat signature mechanism. This is a one-time process that must preceed your signing any Acrobat Files. Among other things, you will select a password for yourself.

2. Log into the Self-Sign Security mechanism in Adobe Acrobat. You will need to supply your password.

3. Click on the Signature tool in Acrobat and then click and drag out a rectangular area on the document you want to sign.

4. Fill out the resulting dialog box, which includes confirming your password.

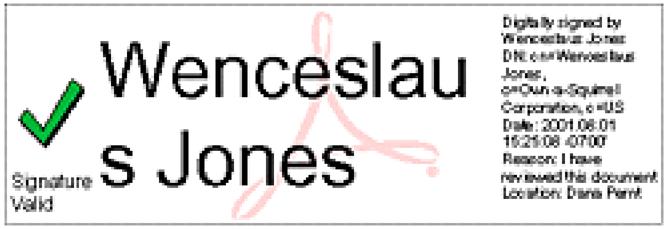
That's it; you have now signed the PDF file.

This month, we'll look at what to do with a signed PDF file. In particular, we shall see what is involved in sending a signed PDF document to another person and how that person can confirm the validity of your signature.

Next Page ->

Sending Signed Documents

You have electronically signed a PDF document. Now what?


Wenceslau
s Jones
Signature Valid
Digitally signed by Wenceslaus Jones
DN: cn=Wenceslaus Jones, o=Own-a-Squirrel Corporation, c=US
Date: 2001.08.01 16:21:08 -07'00'
Reason: I have reviewed this document
Location: Dana Point

Well, now you want to send that signed document to someone else, indicating your approval or agreement (or whatever).

The recipient, in turn, needs to confirm that the signature on the page is actually yours. To make this possible, you will need to send a *user certificate* to each recipient of your signed PDF documents.

User Certificates

Signature IDs

The password you choose when you create a user profile is one of two passwords associated with your profile. The second password is invisibly generated for you by Acrobat when you create your user profile.

The password you chose yourself lets you place signatures on the page. The other, internal password is what actually identifies you to the Self-Sign Signature mechanism. Think of it as your “signature ID.”

Next Page ->

Your signature ID is embedded in encrypted form in the PDF file as part of your Acrobat signature. To confirm that a signature is really yours, a recipient's copy of Acrobat must be able to compare the signature ID in the signature with a signature ID that is known to be yours.

CertExchange
WenceslausJones.fdf

This is why you send a user certificate to the document's recipient.

User Certificates

A user certificate is an FDF file ("Form Data Format") you export from within Acrobat that contains encrypted information about you as a document signer, including your signature ID.

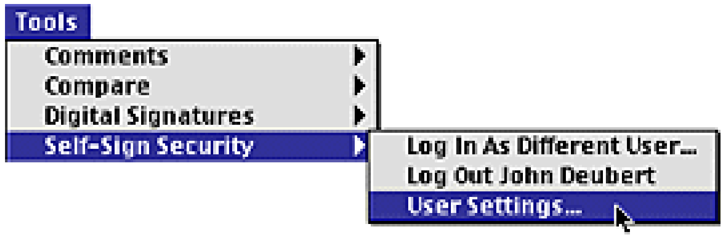
You must send this file to anyone to whom you will send signed PDF files. It allows their copy of Acrobat to verify that the PDF file was actually signed by you.

Next Page ->

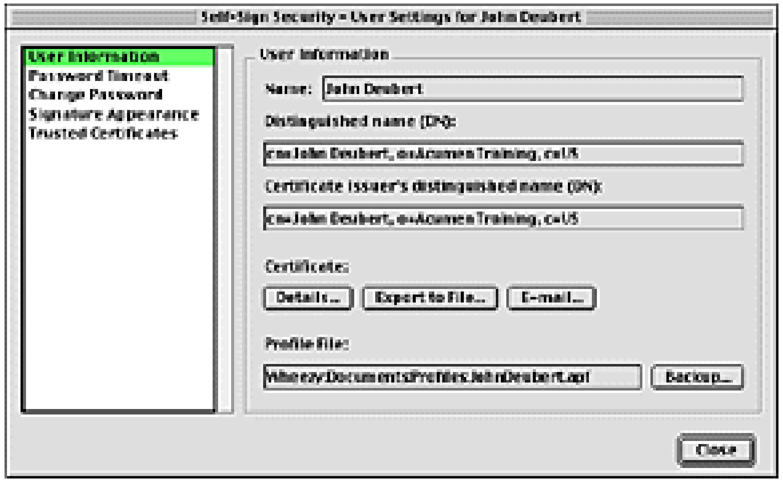
Creating a User Certificate

To create a user certificate, you must first log into the Self-Sign Security system. (See last month's article for detailed instructions on logging in.) Then:

1. Select *Self-Sign Security>User Settings* from the Acrobat *Tools* menu.
   You will now be looking at the *User Settings* dialog box (below, right).



2. Select *User Information* in the list on the left side of the dialog box.
   This panel presents signature information about you.



Name: John Deubert

Distinguished name (DN): cn=John Deubert, o=Acumen Training, c=US

Certificate issuer's distinguished name (DN): cn=John Deubert, o=Acumen Training, c=US

Profile File: Wheezy:Documents:Profiles:JohnDeubert.apf

3. Click on the *Export to file...* button.
   You will be asked to name the FDF file. Do so.

4. Note signature "fingerprint" information.
   Acrobat presents you a dialog with two long strings of text that constitute a "fingerprint" for this certificate. You might want to copy these; you can use them later to determine whether your certificate has become corrupted. I never used these, myself.

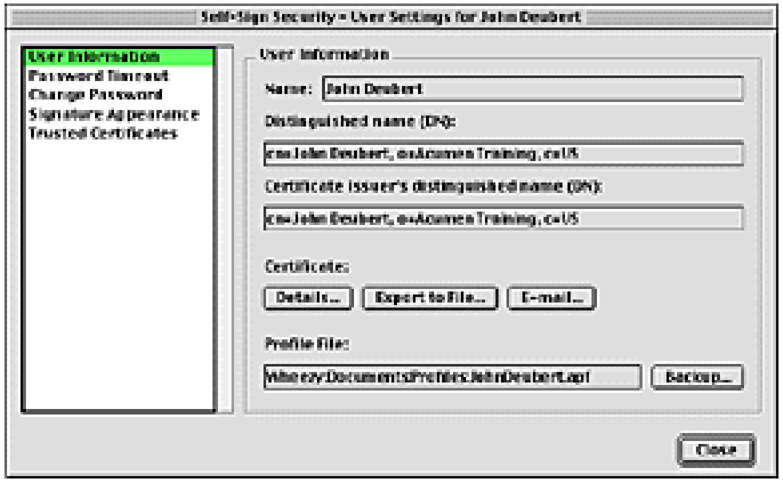
That's it. You have now created a User Certificate you may send to recipients of your signed PDF files.

Next Page ->

E-mailing Certificates

By the way, you can also create a use certificate and immediately e–mail it to someone directly from within Acrobat.

Simply click on the *E–mail* button in the User Settings dialog box and Acrobat will present you with another dialog box that lets you e-mail the certificate directly.


Self-Sign Security - User Settings for John Deubert

User Information

Name: John Deubert

Distinguished name (DN):

cn=John Deubert, o=Acumen Training, c=US

Certificate issuer's distinguished name (DN):

cn=John Deubert, o=Acumen Training, c=US

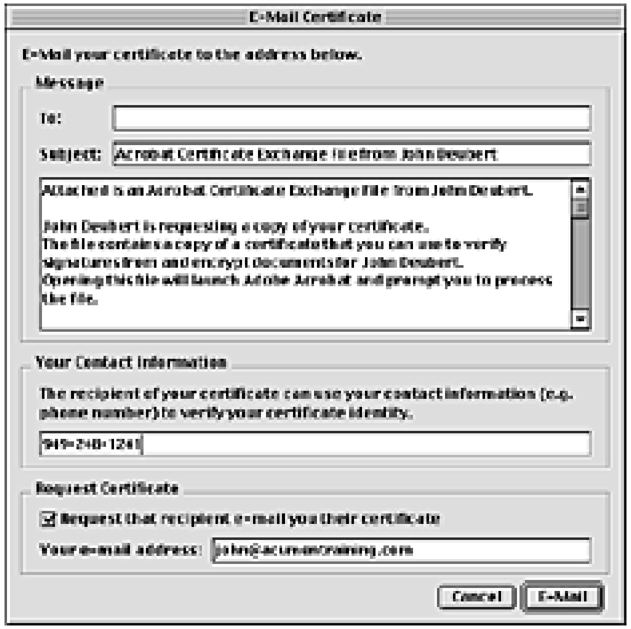
Certificate:

Profile File:

Wheezy:Documents:Profiles:JohnDeubert.apf

Most of this dialog box is reasonably straightforward and we'll not discuss it here.

Next Page ->


E-Mail Certificate

E-Mail your certificate to the address below.

Message

To:

Subject: Acrobat Certificate Exchange file from John Deubert

Attached is an Acrobat Certificate Exchange File from John Deubert.

John Deubert is requesting a copy of your certificate.
The file contains a copy of a certificate that you can use to verify signatures from and encrypt documents for John Deubert.
Opening this file will launch Adobe Acrobat and prompt you to process the file.

Your Contact Information

The recipient of your certificate can use your contact information (e.g. phone number) to verify your certificate identity.

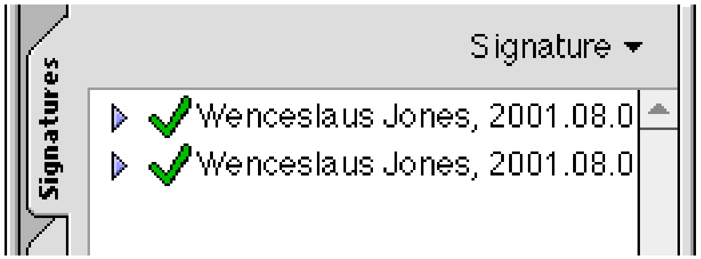
919-240-1241

Request Certificate

☑ Request that recipient e-mail you their certificate

Your e-mail address: john@acumentraining.com

Receiving a Signed PDF File



Wenceslaus Jones, 2001.08.0
Wenceslaus Jones, 2001.08.0

When you receive a signed PDF document from someone, you will need to verify that the signature was actually placed on the page by that person. In Acrobat, this is referred to as *validating* the signature.

The process is pretty easy: Acrobat can scan through a PDF file and declare valid (or not) all of the signatures in the document.

The question is: how can Acrobat know whether a signature is really yours?

The answer: the recipient must import a user certificate, sent by the signer, into his or her Acrobat. Acrobat can then compare the signature ID in the user certificate with that embedded in the signature.

What's a valid signature? A signature is valid if its signature ID matches that in your user certificate and the document has not been changed since it was signed.

Next Page ->

Trusted Certificates

A user certificate imported into a recipient’s Acrobat is referred to as a *trusted certificate.*

It is “trusted” in the sense that you are convinced the certificate actually came from the signer of the document.

Perhaps you have called that person to confirm he or she actually e-mailed you a certificate on such-and-such a date. Perhaps the signer personally gave you the certificate on a disk.

One way or another, you trust that this certificate truly came from the signer.

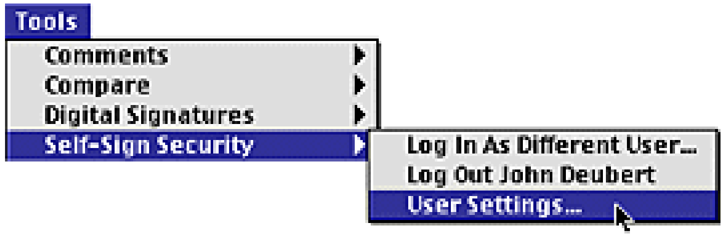
Next Page ->

Importing a Trusted Certificate

To import a trusted certificate:

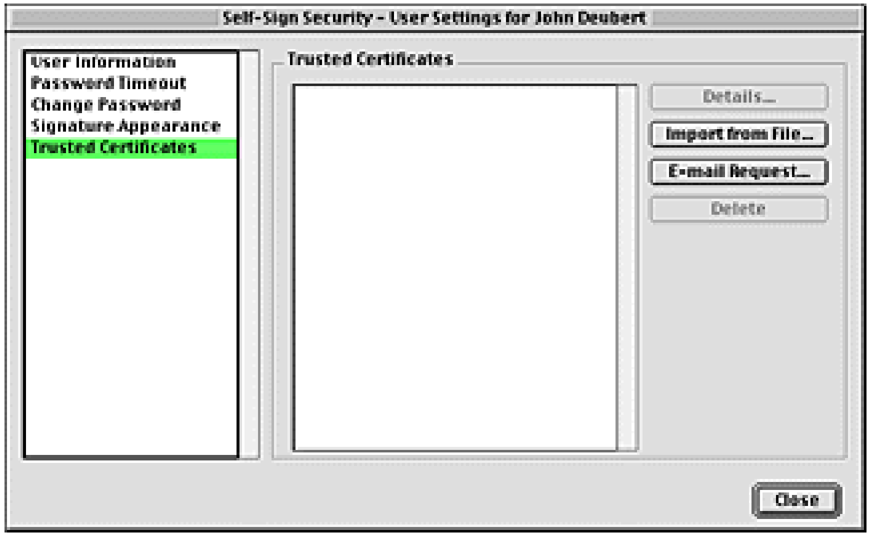
1. Log into the Self-Sign Signature system. (Select *Tools>Self-Sign Signatures>Log in…*

2. Select *Tools>Self-Sign Security>User Settings…*
   You will be looking at the User Settings dialog box.



3. Select *Trusted Certificates* from the list on the left.

   You are now looking at a list of all the trusted certificates you have imported in the past.



4. Click on the *Import from File* button.

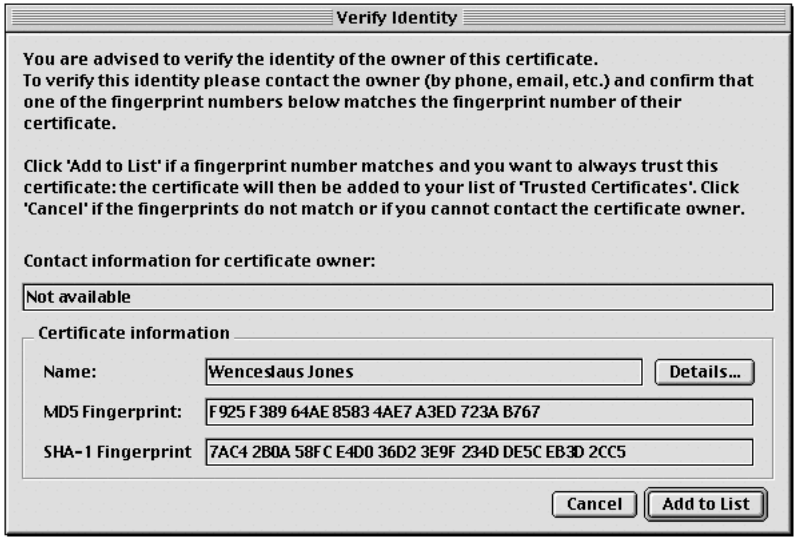
   Acrobat will ask you to select the FDF file that contains the signer's user certificate.

Next Page ->

5. Compare the signature "fingerprints," if you wish.

   Acrobat will display signature information associated with the user certificate. You may want to compare notes with the signer to see if these were the same signature information the signer was shown when the user certificate was created.

   I confess I never do this step.


Verify Identity

You are advised to verify the identity of the owner of this certificate.
To verify this identity please contact the owner (by phone, email, etc.) and confirm that one of the fingerprint numbers below matches the fingerprint number of their certificate.

Click 'Add to List' if a fingerprint number matches and you want to always trust this certificate: the certificate will then be added to your list of 'Trusted Certificates'. Click 'Cancel' if the fingerprints do not match or if you cannot contact the certificate owner.

Contact information for certificate owner:

Not available

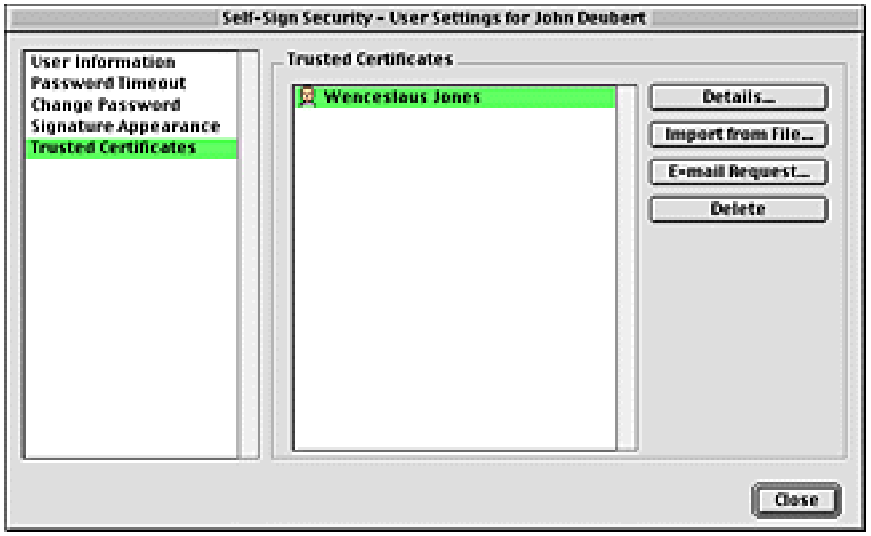
Certificate information

Name: Wenceslaus Jones
MD5 Fingerprint: F925 F389 64AE 8583 4AE7 A3ED 723A B767
SHA-1 Fingerprint 7AC4 2B0A 58FC E4D0 36D2 3E9F 234D DE5C EB3D 2CC5

We're done!

The name of the signer will now appear in the User Settings dialog box as a trusted certificate. Acrobat will recognize signatures by this person.

Next Page ->


Self-Sign Security - User Settings for John Deubert

Trusted Certificates: Wenceslaus Jones

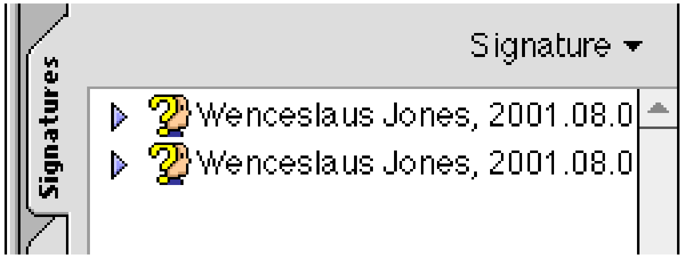
Validating Signatures

Having imported a trusted certificate, how do we determine whether the signatures in a PDF file are valid? I usually open the *Signatures* panel and work from there.

To validate a signature…

1. Log in, as usual.
2. Expose the Navigation pane, on the left side of the Acrobat window, and bring the *Signatures* tab to the front.

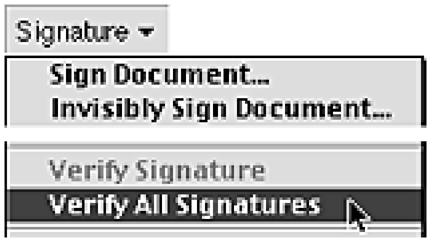
   The Signatures tab presents a list of the signatures in this document, each tagged with a question mark, indicating that it haven't been validated yet.

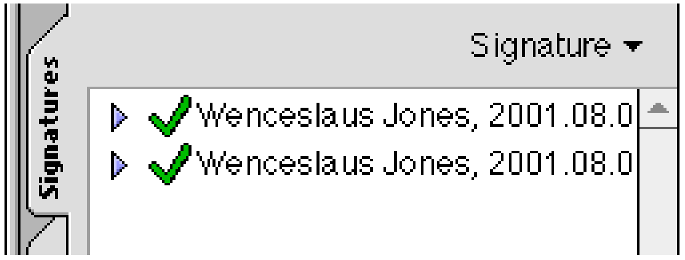


3. In the drop down menu attached to this pane, select *Verify all signatures.*

   Acrobat will scan through the PDF file comparing each signature in the file with the information in the trusted certificates you have imported. All valid signatures will have the question mark replaced by a check mark.

   You can be very confident that these are the signatures of the proper people.





Next Page ->

Summary

There seem to be a lot of steps associated with using Acrobat Self-Sign Security, but it's all pretty simple. In summary, here's the procedure we have described these past two issues. (You might want to print the next three pages as a cheat sheet.

Signing a document

Creating a Profile

1. In Acrobat, select *Tools>Self-Sign Security>Log in…*
2. In the resulting dialog box, click on the *New User Profile…* button.
3. In the *New User* dialog box, select a password for yourself and click *OK.*

Signing the document

1. In Acrobat, select *Tools>Self-Sign Security>Log in…*
2. In the *Log In* dialog box, select your name and type in your password; click *OK*.
3. Click on the *Signature* tool in the toolbar.
4. Click and drag out a rectangular region on the PDF page.
5. In the resulting dialog box, supply your password and fill in whatever other information you want (such as your reason for signing the document).

Next Page ->

Creating a user certificate	1. In Acrobat, log in.
	2. Select *Tools>Self-Sign Security>User Settings…*
	3. In left-hand side of the resulting dialog box, click on *User Information.*
	4. In the resulting controls, click on the *Export to File…* button
	5. Select a name for the certificate file.
	6. Click *OK* in the resulting dialog box that gives you the “fingerprint” information.
Sending a signed document	1. Create a user certificate, if necessary.
	2. Send the signed PDF file and the user certificate file to the recipient of the signed PDF file.
	(Note that you only have to send your user certificate once to a given recipient.)

Next Page ->

Receiving a signed PDF file

Importing a trusted certificate

1. In Acrobat, log in.
2. Select *Tools>Self-Sign Security>User Settings…*
3. In left-hand side of the resulting dialog box, click on *Trusted Certificates.*
4. Among the resulting controls, click on the *Import from file…* button.
5. Select the user certificate file sent to you by the document's signer.
6. Click *OK* in the resulting dialog box (it's those fingerprint strings again).

Validating a signature

1. In Acrobat, log in.
2. Expose the navigation pane (on the left side of the Acrobat window).
3. In the pop-up menu associated with the Signatures pane, select *Verify all signatures.*
4. Examine the result. Verified signatures will be tagged with a check mark.

Next Page ->

Final Remarks

Acrobat 4

All of the screen shots in our two-month discussion have been of Acrobat 5. The Self-Sign Security mechanism in Acrobat 4 is substantially the same. Dialog boxes look different, but otherwise it is very much the same mechanism.

Signature Plug-ins

What I've described here is the signature mechanism that ships with Adobe Acrobat. The signature mechanism, however, is built around a plug-in programmers' interface. Third party companies can and do create Acrobat plug-ins to implement other electronic signature methods specific to a particular company or need. Check out Planet PDF or PDF Zone for information on these.

Reverting to signed versions

If any of the signatures in a PDF file fail to validate because the document has changed since the signing, you can tell Acrobat to revert the document to the state it had when it was signed. We don't have space to talk about it here, but experiment by selecting *View Signed Version* in the pop-up menu of the *Signatures* pane.

Try it!

If you want to practice validating signatures, look among the Acrobat sample files on the Acumen Training web site. You will find a .sit/.zip file containing a signed PDF document and a user certificate file for the signature.

Return to Main Menu

Reencoding Fonts, Part 1

Here’s a common question from the newsgroups: how do you print accented and other non-ASCII characters in a PostScript font? An examination of the Adobe Standard Character Set (Appendix E in the PostScript Language Reference Manual) shows that most fonts contain a very large number of characters that are not immediately accessible; in particular, all the characters with accents and other diacritical marks simply cannot be conveniently accessed in a font as it ships from the foundry.

So how do we print these characters?

Actually, the process is very easy, though not obvious. We need to assign character codes to these characters within the font, a technique known as “re-encoding the font.” Once we have done this, *show* can print these characters within a string, as usual.

Doing this requires a small bit of code, but one you will probably need to apply to every font you use in industrial-strength PostScript code.

We’re going to look at this in two parts.

This month, we’ll look at a PostScript program that reencodes a font, inserting four characters into Helvetica. This program is easily understood, but its technique is not useful for driver output, that needs to assign codes to many characters within a font.

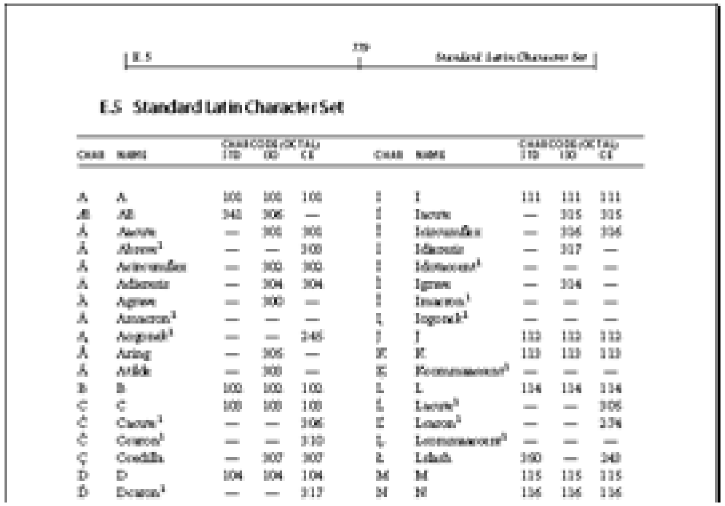
Next month, we’ll look at PostScript code more appropriate for generalized output, such as that generated by a PostScript driver.

Next Page ->

Background

Character Sets

Appendix E in the PostScript Language Reference Manual (PLRM) presents the sets of characters built into Adobe’s fonts; these sets have become standard for pretty much all font vendors. Of particular interest to our discussion is Table 5 in this appendix, the *Adobe Standard Latin Character Set.* (This is on page 779 of the PLRM.)



E.5 779 Standard Latin Character Set

E.5 Standard Latin Character Set

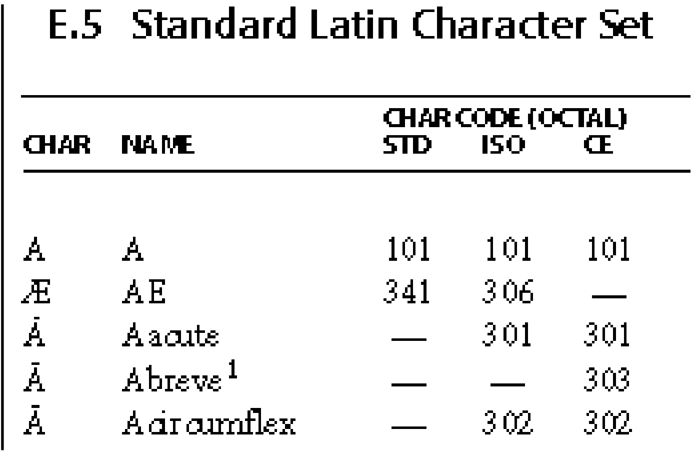
CHAR	NAME	CHAR CODE (OCTAL) STD	ISO	CE	CHAR	NAME	CHAR CODE (OCTAL) STD	ISO	CE
A	A	101	101	101	I	I	111	111	111
Æ	AE	341	306	—	Í	Iacute	—	315	315
Á	Aacute	—	301	301	Î	Icircumflex	—	316	316
Ă	Abreve[1]	—	—	303	Ï	Idieresis	—	317	—
Â	Acircumflex	—	302	302	İ	Idotaccent[1]	—	—	—
Ä	Adieresis	—	304	304	Ì	Igrave	—	314	—
À	Agrave	—	300	—	Ī	Imacron[1]	—	—	—
Ā	Amacron[1]	—	—	—	Į	Iogonek[1]	—	—	—
Ą	Aogonek[1]	—	—	245	J	J	112	112	112
Å	Aring	—	305	—	K	K	113	113	113
Ã	Atilde	—	303	—	Ķ	Kcommaaccent[1]	—	—	—
B	B	102	102	102	L	L	114	114	114
C	C	103	103	103	Ĺ	Lacute[1]	—	—	305
Ć	Cacute[1]	—	—	306	Ľ	Lcaron[1]	—	—	274
Č	Ccaron[1]	—	—	310	Ļ	Lcommaaccent[1]	—	—	—
Ç	Ccedilla	—	307	307	Ł	Lslash	350	—	243
D	D	104	104	104	M	M	115	115	115
Ď	Dcaron[1]	—	—	317	N	N	116	116	116

This table lists all of the characters in Adobe’s alphabetic fonts intended for the Western market. For each character, the table lists five columns of information:

Character shape - a picture of the character

Character name - the character’s name.

Character code - the character code assigned to the character in each of three character encoding schemes. We care about *STD,* Adobe’s Standard Encoding.



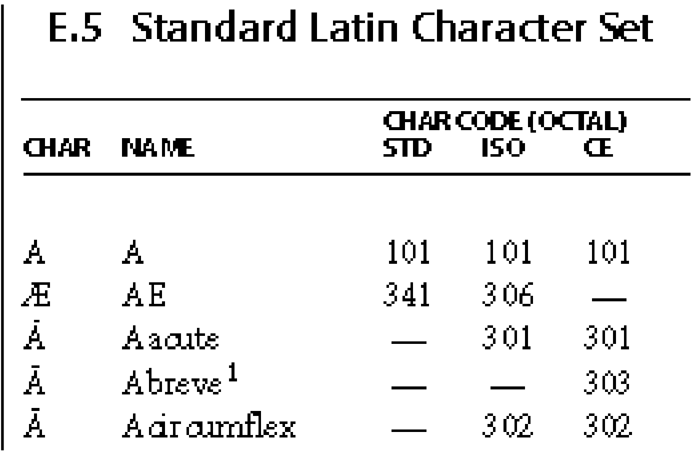
E.5 Standard Latin Character Set

CHAR	NAME	CHAR CODE (OCTAL) STD	ISO	CE
A	A	101	101	101
Æ	AE	341	306	—
Á	Aacute	—	301	301
Ă	Abreve[1]	—	—	303
Â	Acircumflex	—	302	302

Next Page ->

PostScript Encoding Arrays

As you recall from your PostScript classes, within a PostScript font, the correspondence between character codes and printed characters is specified by an array named *Encoding* inside the font dictionary. This is a 256-element array of character names.

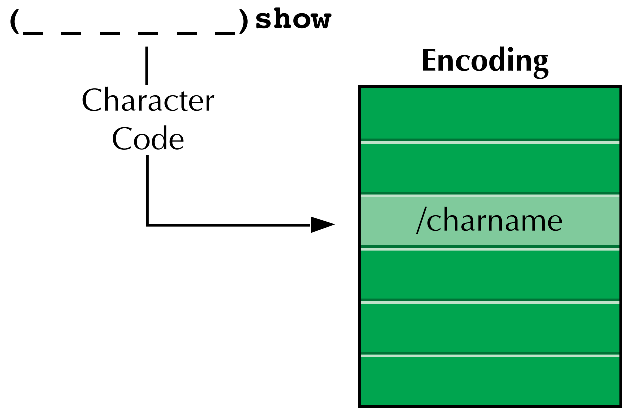


E.5 Standard Latin Character Set

CHAR	NAME	CHAR CODE (OCTAL) STD	ISO	CE
A	A	101	101	101
Æ	AE	341	306	—
Á	Aacute	—	301	301
Ă	Abreve[1]	—	—	303
Â	Acircumflex	—	302	302

Every character within a PostScript font has a name associated with it. The position of that name within the *Encoding* array determines the character code for that character.

The PostScript *show* operator takes each character code in its string argument and uses it as an index into *Encoding*, getting the name of the character it should print.

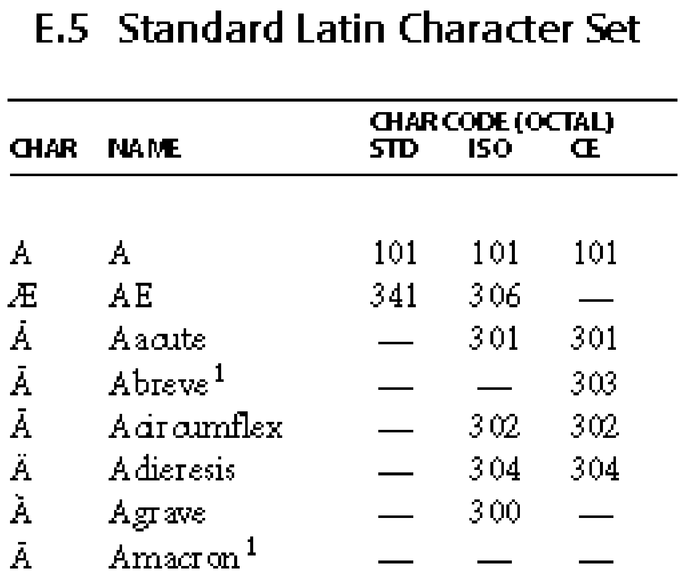


Next Page ->

Reencoding Fonts

Unencoded Characters

Examine the fragment of the Latin Character Set table reproduced at right. Several of the characters have no character codes listed in the STD encoding. These characters are *unencoded;* their shapes are defined in the font, but their names do not appear in the default Encoding array.



E.5 Standard Latin Character Set

CHAR	NAME	CHAR CODE (OCTAL) STD	ISO	CE
A	A	101	101	101
Æ	AE	341	306	—
Á	Aacute	—	301	301
Ă	Abreve[1]	—	—	303
Â	Acircumflex	—	302	302
Ä	Adieresis	—	304	304
À	Agrave	—	300	—
Ā	Amacron[1]	—	—	—

To gain access to the unencoded characters (and these include all of the accented characters), we must *reencode* the font. That is, we must insert the names of these characters into the current font's Encoding array.

Typically, we would insert the characters into positions consistent with the computer environment that is generating the PostScript code. That is, PostScript generated on the Macintosh would want to put the name *Aacute* into position *231* of the Encoding array, since that's the character code assigned to Á in that OS; in Windows, we'd place *Aacute* into position *193,* matching that environment's encoding.

Next Page ->

Example

¿Dónde está el camino a San José?

This month, we shall look at a program that creates a reencoded version of Helvetica, inserting four characters into that font's Encoding array. We'll then use the reencoded Helvetica to print a Spanish phrase.

If you have taken any of the Acumen Training PostScript classes, you will recognize this program.

Overview

PostScript fonts dictionaries are read-only; you can't change them. This being the case, whenever we *do* want to change a font, we have to make an entirely new font, identical to the original, save for the changes.

There are four steps to this process:

1. Make a new dictionary, the same size as the original font, plus any room needed for our changes.
2. Copy everything from our original font into our new dictionary. In PostScript LanguageLevel 1, we should *not* copy the *FID* entry in the font. LanguageLevels 2 and 3 don't care, so we'll not worry about this detail in our example.
3. Make the changes to our new dictionary.
4. Turn the new dictionary into a font dictionary by calling *definefont.*

Next Page ->

The Code

```
/Helvetica findfont dup          % Find the Helvetica font
length dict                      % Create a new dictionary the same size
copy begin                       % Copy Helv. into the new dict & move the
                                 %    new dict. to the dict stack.

/Encoding Encoding 256 array copy def % Make a writable copy of Encoding
Encoding 1 /oacute put           % Put new character names into Encoding
Encoding 2 /aacute put
Encoding 3 /eacute put
Encoding 4 /questiondown put

/Helvetica-Esp currentdict definefont pop   % Turn current dict into a
                                            %     a new font
end                              % Remove the dict from the dict stack

/Helvetica-Esp 18 selectfont

72 600 moveto
(\004D\001nde est\002 el camino a San Jos\003?) show

showpage
```

Next Page ->

The Code, Step-by-Step

Create a new dictionary

```
/Helvetica findfont dup       % =>  <<Helv>> <<Helv>>
```

Here we fetch the Helvetica font dictionary and leave it on the operand stack. We duplicate the resulting dictionary object, since we want to do two things with it: find its length and then copy its contents to a new dictionary.

```
length dict                   % =>  <<Helv>> <<dict>>
```

We hand one of our copies of Helvetica to the *length* operator, which returns the number of key-value pairs in that font dictionary. We then create a dictionary with room for that many key-value pairs. This leaves us with Helvetica and our new dictionary on the stack.

```
copy begin
```

Finally, we copy the contents of Helvetica into our new dictionary. The *copy* operator leaves our new dictionary on the operand stack; we move it to the top of the dictionary stack with *begin.*

At this point, our new dictionary is the current dictionary: the *def* operator will place its key-value pairs into this dictionary.

Next Page ->

Create a new Encoding array

```
/Encoding Encoding 256 array copy def
```

The Encoding array in our new dictionary was copied from Helvetica and is, therefore, read-only; we need to create a writable version of it. This line of PostScript creates a 256-element array, copies the contents of Helvetica's *Encoding* (taken from the current dictionary, *i.e.,* Helvetica) into it, and then *defs* it back into our new dictionary with the name *Encoding.*

```
Encoding 1 /oacute put        % Put new character names into Encoding
Encoding 2 /aacute put
Encoding 3 /eacute put
Encoding 4 /questiondown put
```

Now we use the *put* operator to place four character names into our new, writable Encoding array. Properly speaking, we don't have to have to insert *questiondown* into Encoding, since it already resides in Adobe's default Encoding. Still, in real life, you'd need to do this, since the default character code for this character doesn't match anyone else's encoding scheme.

In our font, character codes 1 through 4 will correspond to ó, á, é, and ¿.

Next Page ->

Turn the dictionary into a font dictionary

```
/Helvetica-Esp currentdict definefont pop
```

Finally, we turn the dictionary into a font dictionary with the *definefont* operator. This operator takes a name and a dictionary from the operand stack; it turns the dictionary into a font dictionary with the specified name and returns a copy of the new font dictionary on the operand stack.

In our case, we immediately discard the copy of the new font dictionary.

```
end
```

We tidy up by removing our new dictionary (now a font dictionary) from the dictionary stack.

Use the new font

```
/Helvetica-Esp 18 selectfont

72 600 moveto
(\004D\001nde est\002 el camino a San Jos\003?) show
```

¿Dónde está el camino a San José?

We can now use the reencoded font just like any other. We make it the current font with a call to *selectfont,* do a *moveto* and a *show,* printing a phrase in Spanish.

Note that within a PostScript string, a backslash followed by three octal digits inserts that character code into the string.

Next Page ->

Limitations to this example

This example inserts four characters, one at a time, into Helvetica’s Encoding array. This is fine if you want to make only a very minor change to the font’s encoding, but is not appropriate for the PostScript generated by a printer driver.

A driver needs to place a large number of characters into the font’s Encoding array, matching the environment’s character encoding. Doing this with a series of *put* operations is tedious, slow, and clunky. Particularly when you recall that a driver needs to reencode *every* font it uses. Every *findfont* must be followed by a reencoding.

Next month?

Next month, we’ll see how to do efficient reencoding in a driver. We shall define a *Reencode* procedure that may be called every time a driver calls *findfont.*

I’m hoping the anticipation will help get you through November.

Return to Main Menu

Schedule of Classes, Nov 2001 - Jan 2002

Following are the dates and locations of Acumen Training's PostScript and Acrobat classes. Clicking on a class name below will take you to the description of that class on the Acumen training website. (September has been completely consumed by on-site classes, so there are no Orange County classes this month.)

The PostScript classes are taught in Orange County, California.

PostScript Classes

PostScript Foundations	December 10 - 14
Advanced PostScript	January 21 – 25
PostScript for Support Engineers	January 14 – 18
Jaws Development	November 27 - 30

For more classes, go to www.acumentraining.com/schedule.html

PostScript Course Fees

PostScript classes cost $2,000 per student.
These classes may also be taught on your organization's site.

Registration →
Acrobat Classes →

Acrobat Class Schedule

On-Site Only	These classes are taught only on corporate sites. If you have an interest in any of these classes for your group, please see the Acumen Training website regarding arranging an on-site class.
Acrobat Essentials	This class teaches the student how to make perfect PDF files. It includes complete coverage of the meaning and proper settings of all of the Distiller Job Options.
Interactive Acrobat	Here we show you how to add bookmarks, links, buttons, sounds, movies, form fields, and other interactive features to an Acrobat file.
Creating Acrobat Forms	This class shows you how to make interactive forms in Adobe Acrobat. It steps you through creating the form, posting form contents to a server, and everything else you need to create a working PDF form.
Troubleshooting with Enfocus' PitStop	This class shows the student how to use all of the capabilities of this popular editing and preflight software.

Back to PostScript Classes

Return to First Page

Contacting John Deubert at Acumen Training

For more information	For class descriptions, on-site arrangements or any other information about Acumen's classes: **Web site:** http://www.acumentraining.com **email:** john@acumentraining.com **telephone:** 949-248-1241 **mail:** 25142 Danalaurel, Dana Point, CA 92629
Registering for Classes	To register for an Acumen Training class, contact us any of the following ways: **Register On-line:** http://www.acumentraining.com/registration.html **email:** registration@acumentraining.com **telephone:** 949-248-1241 **mail:** 25142 Danalaurel, Dana Point, CA 92629
Back issues	Back issues of the Acumen Journal are available at the Acumen Training website: www.acumenjournal.com/AcumenJournal.html

Return to First Page

What’s New at Acumen Training?

New Class: Creating Acrobat Forms

Acumen Training is now offering a class on Creating Acrobat Forms. This class for beginning to intermediate Acrobat users shows the student how to create interactive forms in Adobe Acrobat. Building real forms as we go, the class teaches the student:

- How to create buttons, text fields, pop-up menus, and all the other data-gathering fields available in Acrobat.
- How to embed interactive features in your forms, such as movies, sounds, and links to the World Wide Web.
- How to use the Acrobat Signature mechanism so that people who fill out your form may place an electronic signature on the PDF page.
- How to submit form data to an external server, there to be inserted in a database, etc.

Creating Acrobat Forms is a half-day class and, like all Acumen Training’s Acrobat classes, is taught on corporate sites only.

For more information, see the course description on the Acumen Training web site.

Return to First Page

Journal Feedback

If you have any comments regarding the *Acumen Journal,* please let me know. In particular, we are looking for three types of information:

Comments on usefulness. Does the Journal provide you with worthwhile information? Was it well written and understandable? Did you like it, hate it, or did it make you want to eat brussels sprouts? How could we make it better? Do you like the PDF format?

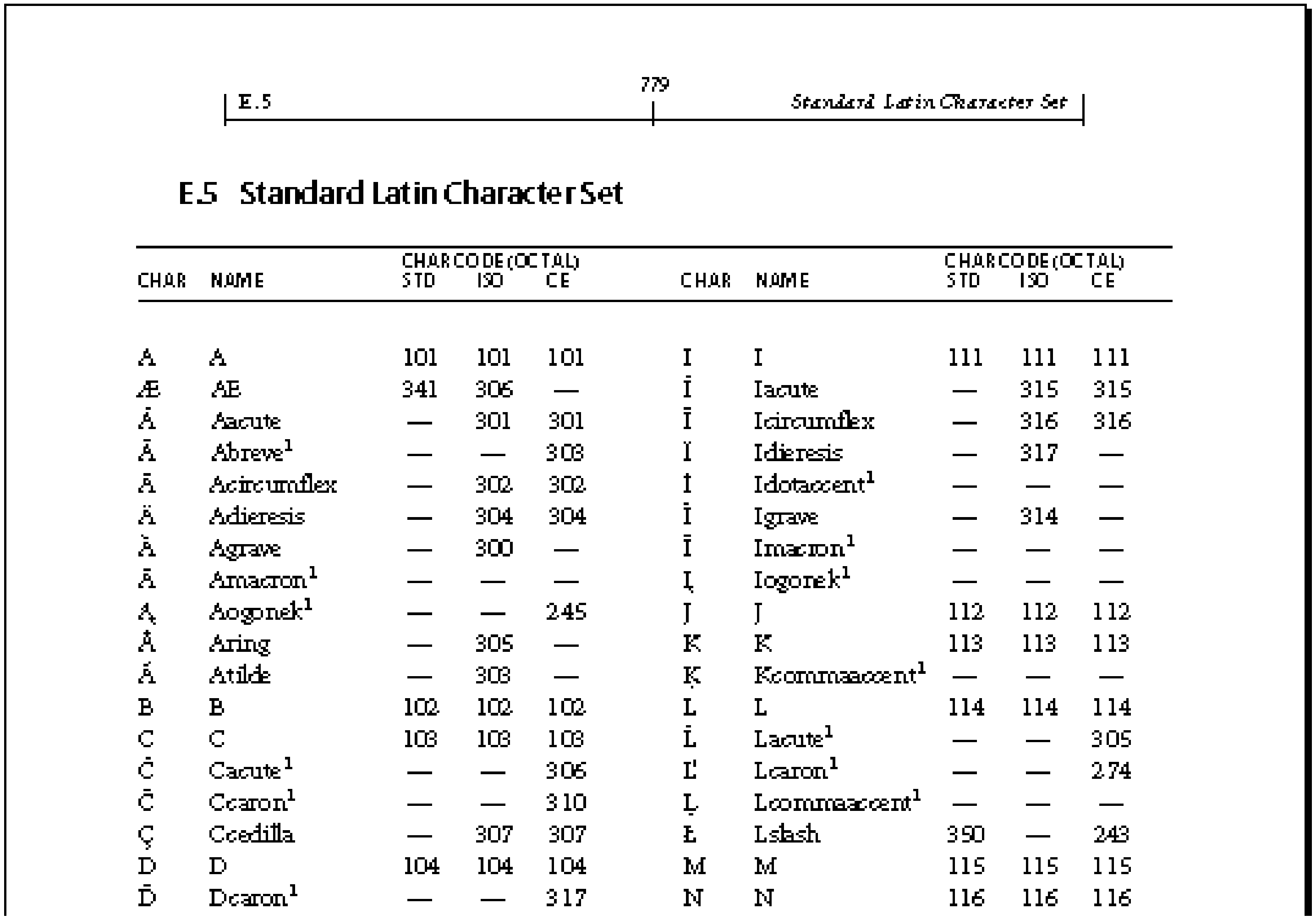
Suggestions for articles. Each Journal issue contains one article each on PostScript and Acrobat. What topics would you like us to address?

Questions and Answers. We are planning a Q&A section for future issues. Do you have any questions about Acrobat, PDF or PostScript?

Please send any comments, questions, or problems to:

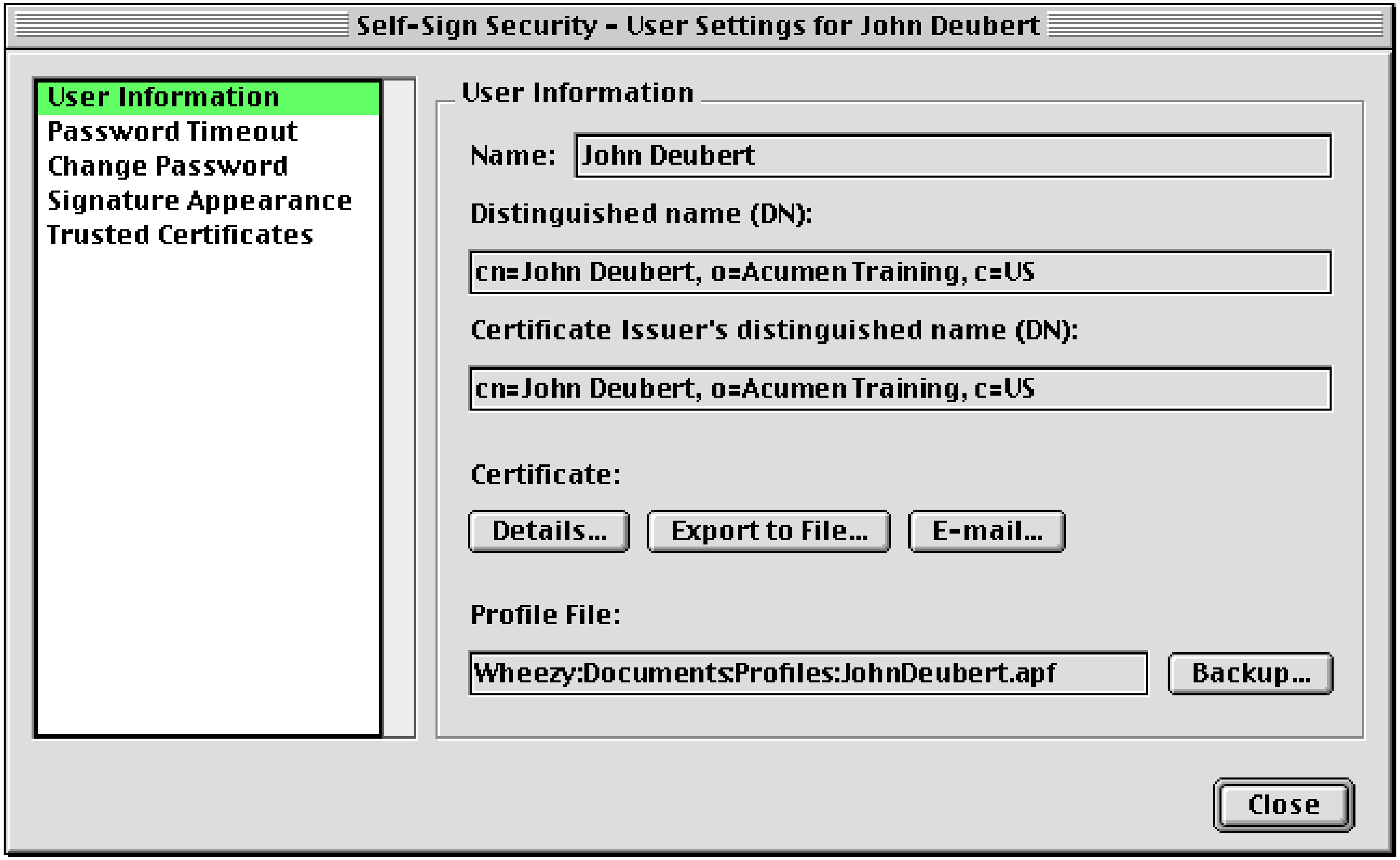
journal@acumentraining.com

Return to Menu



E.5 Standard Latin Character Set

CHAR	NAME	CHAR CODE (OCTAL) STD	ISO	CE	CHAR	NAME	CHAR CODE (OCTAL) STD	ISO	CE
A	A	101	101	101	I	I	111	111	111
Æ	AE	341	306	—	Í	Iacute	—	315	315
Á	Aacute	—	301	301	Î	Icircumflex	—	316	316
Ă	Abreve[1]	—	—	303	Ï	Idieresis	—	317	—
Â	Acircumflex	—	302	302	İ	Idotaccent[1]	—	—	—
Ä	Adieresis	—	304	304	Ì	Igrave	—	314	—
À	Agrave	—	300	—	Ī	Imacron[1]	—	—	—
Ā	Amacron[1]	—	—	—	Į	Iogonek[1]	—	—	—
Ą	Aogonek[1]	—	—	245	J	J	112	112	112
Å	Aring	—	305	—	K	K	113	113	113
Ã	Atilde	—	303	—	Ķ	Kcommaaccent[1]	—	—	—
B	B	102	102	102	L	L	114	114	114
C	C	103	103	103	Ĺ	Lacute[1]	—	—	305
Ć	Cacute[1]	—	—	306	Ľ	Lcaron[1]	—	—	274
Č	Ccaron[1]	—	—	310	Ļ	Lcommaaccent[1]	—	—	—
Ç	Ccedilla	—	307	307	Ł	Lslash	350	—	243
D	D	104	104	104	M	M	115	115	115
Ď	Dcaron[1]	—	—	317	N	N	116	116	116



User Information

Name: John Deubert

Distinguished name (DN):

cn=John Deubert, o=Acumen Training, c=US

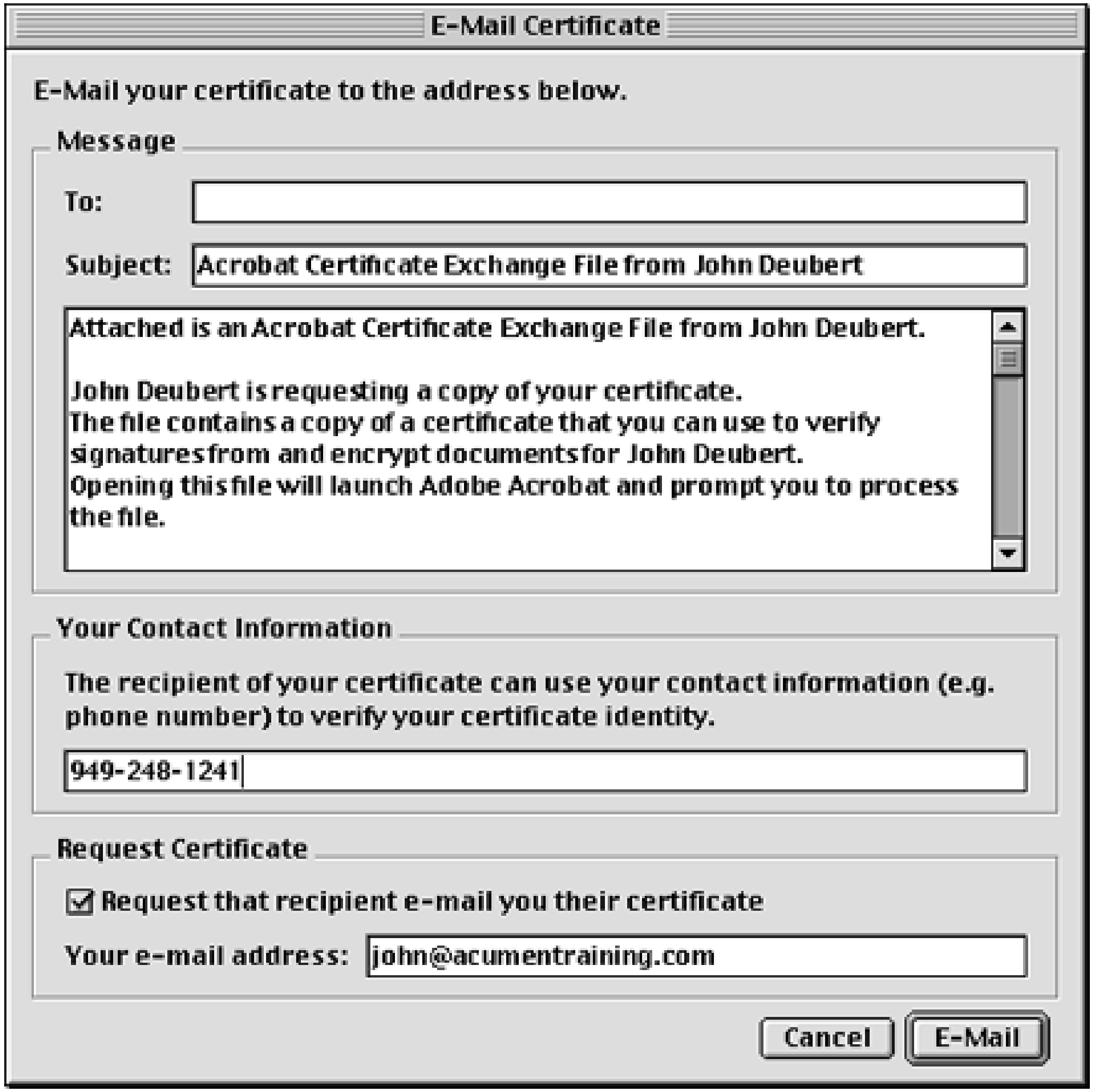
Certificate Issuer's distinguished name (DN):

cn=John Deubert, o=Acumen Training, c=US

Certificate:

Profile File:

Wheezy:Documents:Profiles:JohnDeubert.apf



E-Mail your certificate to the address below.

Message

To:	
Subject:	Acrobat Certificate Exchange File from John Deubert

Attached is an Acrobat Certificate Exchange File from John Deubert.

John Deubert is requesting a copy of your certificate.
The file contains a copy of a certificate that you can use to verify signatures from and encrypt documents for John Deubert.
Opening this file will launch Adobe Acrobat and prompt you to process the file.

Your Contact Information

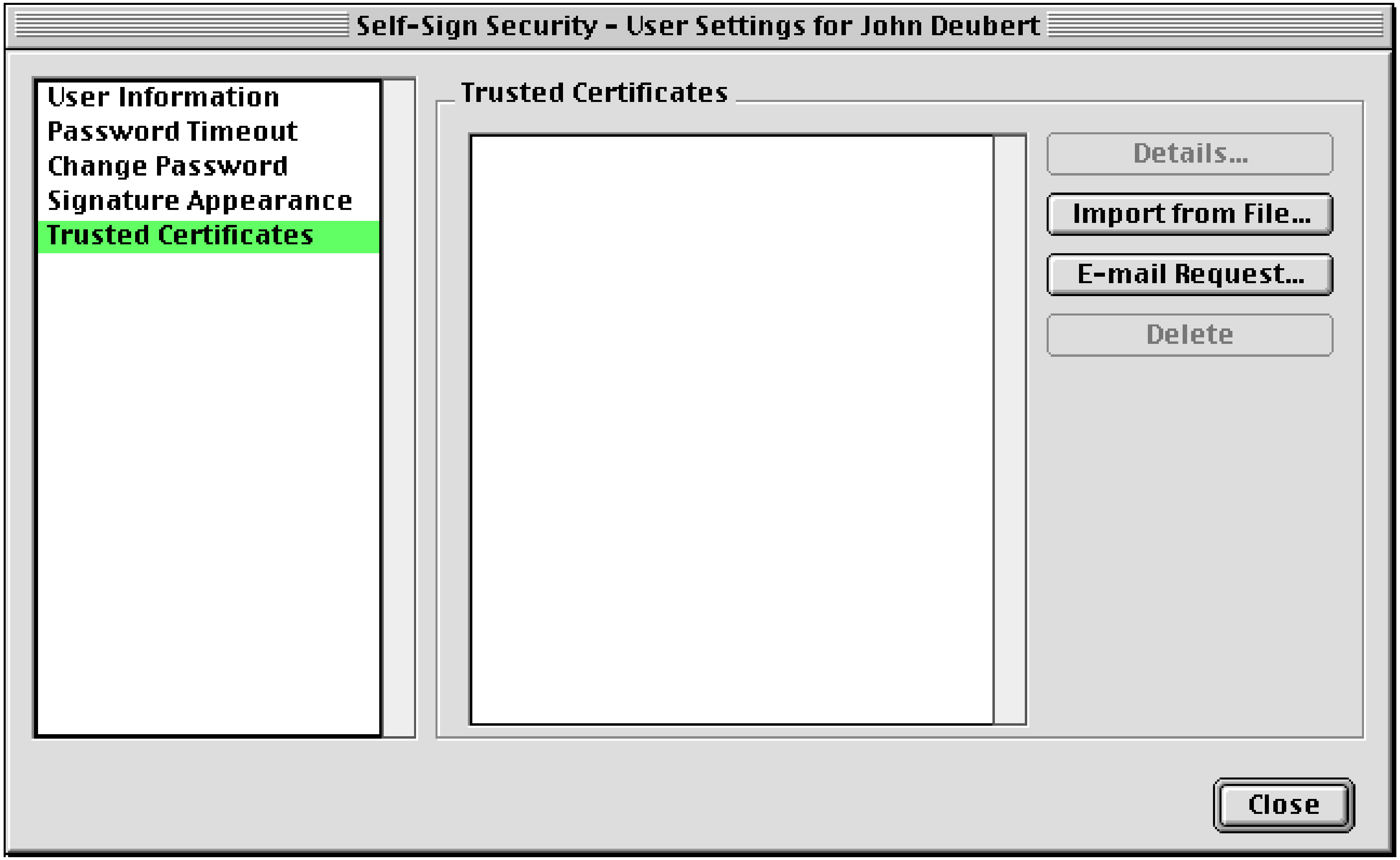
The recipient of your certificate can use your contact information (e.g. phone number) to verify your certificate identity.

949-248-1241

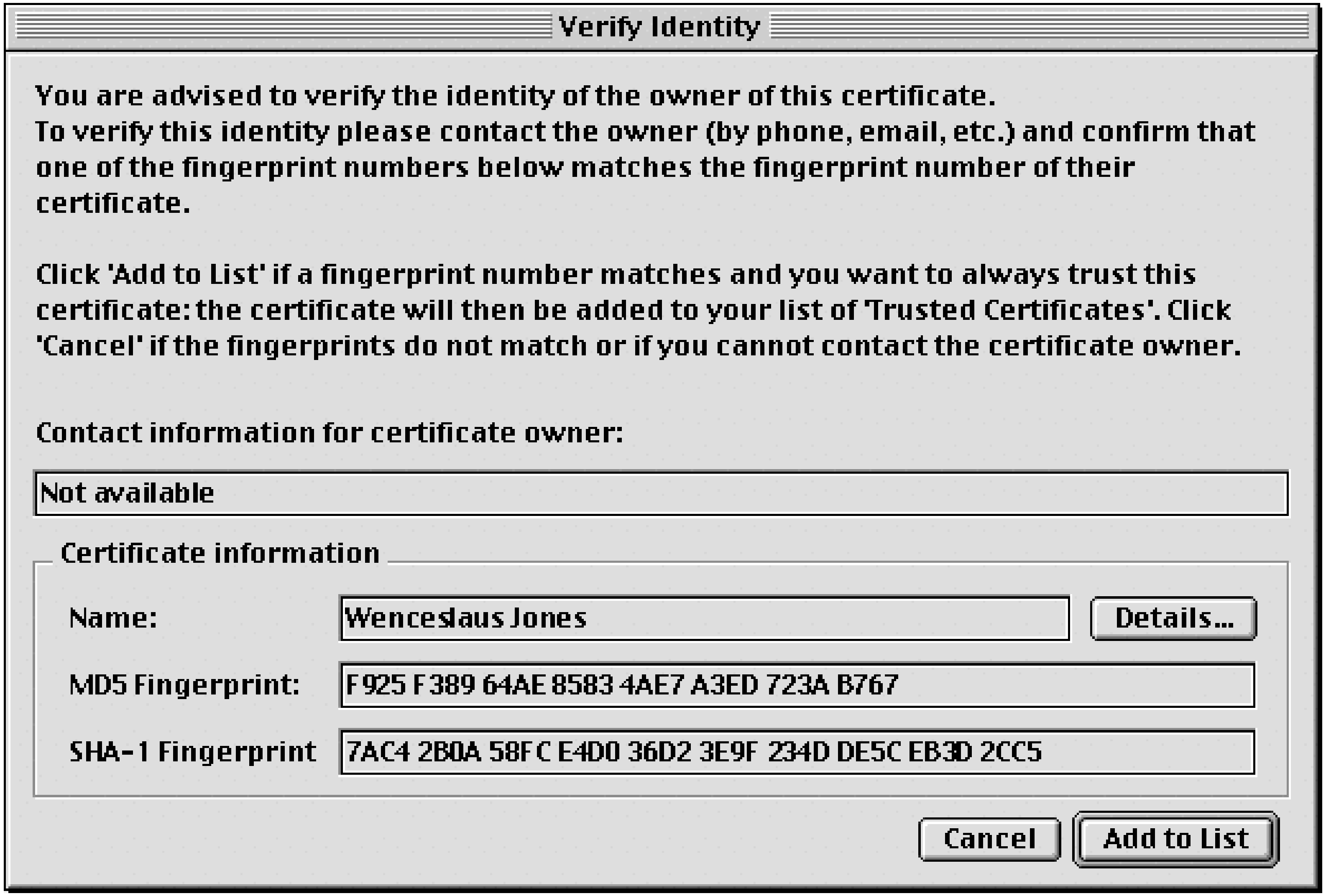
Request Certificate

[✓] Request that recipient e-mail you their certificate

Your e-mail address: john@acumentraining.com


Self-Sign Security - User Settings for John Deubert
User Information
Password Timeout
Change Password
Signature Appearance
Trusted Certificates
Trusted Certificates
Details...
Import from File...
E-mail Request...
Delete
Close

Verify Identity



You are advised to verify the identity of the owner of this certificate.
To verify this identity please contact the owner (by phone, email, etc.) and confirm that one of the fingerprint numbers below matches the fingerprint number of their certificate.

Click 'Add to List' if a fingerprint number matches and you want to always trust this certificate: the certificate will then be added to your list of 'Trusted Certificates'. Click 'Cancel' if the fingerprints do not match or if you cannot contact the certificate owner.

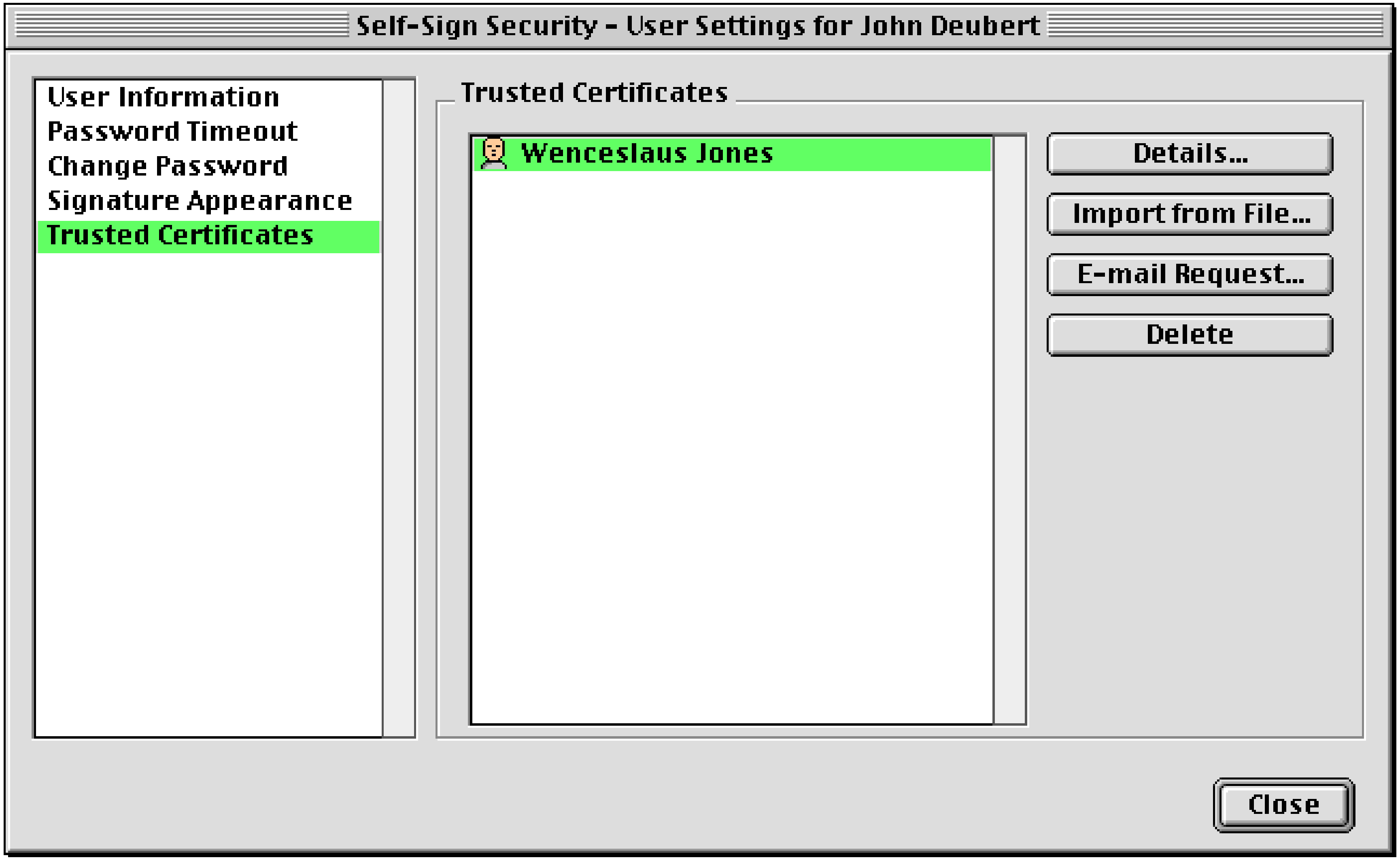
Contact information for certificate owner:

Not available

Certificate information

Name:	Wenceslaus Jones
MD5 Fingerprint:	F925 F389 64AE 8583 4AE7 A3ED 723A B767
SHA-1 Fingerprint	7AC4 2B0A 58FC E4D0 36D2 3E9F 234D DE5C EB3D 2CC5

Imported Certificate


Self-Sign Security - User Settings for John Deubert
User Information
Password Timeout
Change Password
Signature Appearance
Trusted Certificates
Trusted Certificates
Wenceslaus Jones
Details...
Import from File...
E-mail Request...
Delete
Close