Oracle® Communications Convergent Charging Controller

Voucher Print Shop Operations Guide

Release 6.0

May 2016

ORACLE®

Copyright

Contents

About This Document

Scope

This document describes how Printshop:

- Generates and distributes the security key to the operator
- Decrypts the operator-provided voucher batch files

It also explains the format of the voucher batch file.

It does not include detailed design of the service.

Audience

This guide is intended for use by personnel of the print shop who will be responsible for the end-to-end voucher printing process.

Related Documents

The following documents are related to this document:

- *Charging Control Services Technical Guide*
- *Voucher Manager User's Guide*

Document Conventions

Typographical Conventions

The following terms and typographical conventions are used in the Oracle Communications Convergent Charging Controller documentation.

Formatting Convention	Type of Information
Special Bold	Items you must select, such as names of tabs. Names of database tables and fields.
Italics	Name of a document, chapter, topic or other publication. Emphasis within text.
Button	The name of a button to click or a key to press. **Example:** To close the window, either click **Close**, or press **Esc**.
Key+Key	Key combinations for which the user must press and hold down one key and then press another. Example: **Ctrl+P** or **Alt+F4**.
`Monospace`	Examples of code or standard output.
`Monospace Bold`	Text that you must enter.
variable	Used to indicate variables or text that should be replaced with an actual value.
menu option > menu option >	Used to indicate the cascading menu option to be selected. Example: **Operator Functions > Report Functions**
hypertext link	Used to indicate a hypertext link.

Specialized terms and acronyms are defined in the glossary at the end of this guide.

Terminology

This topic explains any terminology specific to this manual.

Operator

An operator is the telecommunications service provider which generates the vouchers or calling cards which need printing.

Chapter 1

System Overview

Overview

Introduction

This chapter provides an overview of the software and formats used in preparing a voucher batch file for printing.

In this chapter

This chapter contains the following topics.

Introduction

Charging Control Services files and encryption

Charging Control Services (CCS) produces encrypted voucher and account batch files for printing. The encryption is used to provide security for the vouchers or subscriber accounts the files hold. Before the files are printed, the encrypted files must be decrypted using the same public private key pair that was used for the encryption.

For more information about how CCS generates vouchers and accounts, see *Charging Control Services User's Guide* and *Charging Control Services Technical Guide*.

Public and private key encryption

Public and private key encryption (also known as asymmetric encryption) involves a pair of keys:

1 a public key which is used encrypt the file, and
2 a private key which is used to decrypt the file.

Both keys are generated by the holder of the private key. The public key is made available to others who want to send encrypted files to the private key holder. In this case, the print shop will generate the public and private keys and provide the public key to the operator.

For more information about:

- generating keys, see *Managing Public/Private Key Pairs* (on page 2).
- decrypting files, see *Decrypting Files* (on page 9).

More information about public and private key encryption is widely available in publications and on the Internet.

Recommended software

Oracle uses GnuPG to encrypt batch files. These files can be decrypted using any software which supports gnupg public private keys. This guide covers the GnuPG command line tool, and the GPG4Win WindowsXP-compatible software.

Note: Other software such as PGP can also be used successfully for generating and exporting keys and decrypting files. Please use the software which is most suitable for your platform.

For more information about GnuPG (including downloadable software), see http://www.gnupg.org.

For more information about GPG4Win (including downloadable software), see http://www.gpg4win.org.

For more information about PGP (including purchasable software), see http://www.pgp.com.

Managing Public/Private Key Pairs

Generating GPG keys

A public and private GPG key can be generated from a pass-phrase. The private key is held only by the print shop and used only to decode the encrypted batch file. The public key is used to encrypt the file and must therefore be supplied to the operator who will be responsible for generating the voucher batch file.

For more information about using GPG keys with exported files, see *Print Shop Operations Guide*.

Generating keys using gpg

Follow these steps to generate a key using GnuPG.

Important: Additional documentation is available at http://www.gnupg.org. Always consult the recent documentation for your version of GnuPG if you are unsure of any steps in the procedure.

Step	Action
1	Log into the machine which has the GnuPG tool installed.
2	Run the gpg binary. **Example command:** `./gpg --gen-key` **Result:** Text similar to the following will appear: `gpg (GnuPG) 1.4.6; Copyright (C) 2006 Free Software Foundation, Inc.` `This program comes with ABSOLUTELY NO WARRANTY.` `This is free software, and you are welcome to redistribute it` `under certain conditions. See the file COPYING for details.` `gpg: directory `/home/users/cmorris/.gnupg' created` `gpg: can't open `/gnupg/options.skel': No such file or directory` `gpg: keyring `/home/users/cmorris/.gnupg/secring.gpg' created` `gpg: keyring `/home/users/cmorris/.gnupg/pubring.gpg' created` `Please select what kind of key you want:` `(1) DSA and Elgamal (default)` `(2) DSA (sign only)` `(5) RSA (sign only)` `Your selection?`

3	Enter the kind of algorithm you have agreed with the operator you are printing for.
	Result: Text similar to the following will appear:
	`DSA keypair will have 2048 bits.`
	`ELG-E keys may be between 2048 and 4096 bits long.`
	`What keysize do you want? (2048)`

Step	Action

4	Enter the keysize you have agreed with the operator.
	Result: Text similar to the following will appear.
	`Requested keysize is 2048 bits`
	`Please specify how long the key should be valid.`
	`0 = key does not expire`
	`<n> = key expires in n days`
	`<n>w = key expires in n weeks`
	`<n>m = key expires in n months`
	`<n>y = key expires in n years`
	`Key is valid for? (0)`

5	Enter the expiry period you have agreed with the operator.
	Result: Text similar to the following will appear.
	`Key does not expire at all`
	`Is this correct? (y/N) y`

6	If all the details entered so far are correct, type y and press **Enter**.
	Result: Text similar to the following will appear:
	`You need a user ID to identify your key; the software constructs the user ID from the Real Name, Comment and Email Address in this form:`
	`"Heinrich Heine (Der Dichter) <heinrichh@duesseldorf.de>"`
	
	`Real name:`

7	Type your real name and press **Enter**. **Result:** Text similar to the following will appear. `Email address:`
8	Type your email addresss and press **Enter**. **Result:** Text similar to the following will appear. `Comment:`

9	Type a comment and press Enter. The comment should identify who the printshop is, and may also identify the operator.
	Result: Text similar to the following will appear.
	`You selected this USER-ID:`
	`"ExampleUser (TelcoEurope-Printshop) <example.user@xample.com>"`
	
	`Change (N)ame, (C)omment, (E)mail or (O)kay/(Q)uit?`

10	If the details which have been displayed are correct, type O and press **Enter**.
	Result: Text similar to the following will appear.
	`You need a Passphrase to protect your secret key.`
	
	`Enter a passphrase:`

11	Type a passphrase and press **Enter**. **Important:** • This passphrase must be entered when the files are decrypted. If the passphrase is not available, the files will not be able to be decrypted and a new pair of keys and batch file will have to be generated. • The passphrase is an important contributor to the overall security of the encryption. Ensure you follow any guidelines set by the operator, and that you pick a secure password. **Result:** Text similar to the following will appear: `Confirm passphrase:`
12	Type the passphrase from step 11 again and press **Enter**. **Result:** Text similar to the following will appear as gpg generates the keys. `We need to generate a lot of random bytes. It is a good idea to perform some other action (type on the keyboard, move the mouse, utilize the disks) during the prime generation; this gives the random number`

Step	Action

```
generator a better chance to gain enough entropy.
++++++++++++++++++.++++++++++++++++++..+++++.++++++++++.++++++++++.++++++
++++.++++++++++.+++++++++++++++.+++++.++++++++++++++++++++++++>+++++.......a...s.
.l.......fg......a.w.....f.h......a..wi...u.r......w...i.r..u...........i.e.+u+f++
h+
waicuywf7t9urytg87rthgre76hg679t84yg8o7ytgsehuty9prgyt9e8yg6iueyg98ser759fs7e5iuyf
h8awstyfniurg9o8yfhit87siru5ye9f865w3i75uy45giluyhtg75fjwiu45ynfiuyhs987towefny559
87s9r8tgw5oitgusd9p8tg7seoigt98rg75iou3syh98th4i7948gh7ioWe need to generate a lot
of random bytes. It is a good idea to perform
some other action (type on the keyboard, move the mouse, utilize the
disks) during the prime generation; this gives the random number
generator a better chance to gain enough entropy.
++++++++++++++++++++++++++++++..8.++++++++++++++++++++++++++++++..++++++++++++++
+++++++++..++++5++++++..+++++++++++++++.++++++++++++++++++++++++++..u.++++++++++>++
++++++++>.+++++.8.......89..73.9..h.........................................
.....................................<..+++++>+++++.<+++++>..+++++.....<+++++>+
++++>+++++rfe.<+++++<+++++>++++++++er++^ui^h^hie
gpg: /home/users/exuser/.gnupg/trustdb.gpg: trustdb created
gpg: key 2E72F865 marked as ultimately trusted
public and secret key created and signed.

gpg: checking the trustdb
gpg: 3 marginal(s) needed, 1 complete(s) needed, PGP trust model
gpg: depth: 0  valid:   1  signed:   0  trust: 0-, 0q, 0n, 0m, 0f, 1u
pub   2048D/2E72F865 2009-01-16
      Key fingerprint = DB01 04E7 79E3 ECDD DEAD  B829 4ECB 48D6 2E72 F865
uid                  ExampleUser (TelcoEurope-Printshop)
<example.user@example.com>
sub   2048g/69372FCB 2009-01-16
```

Exporting keys using gpg

Follow these steps to export keys which have been generated by gpg.

Step	Action
1	On the machine where the gpg generated the keys, instruct gpg to export the keys. **Example command:** `gpg --armour --export <email_addr_from_key_creation> > <file>.key` **Result:** gpg will export the public key to the specified file. If you cat the file, it will look similar to the example key shown below:

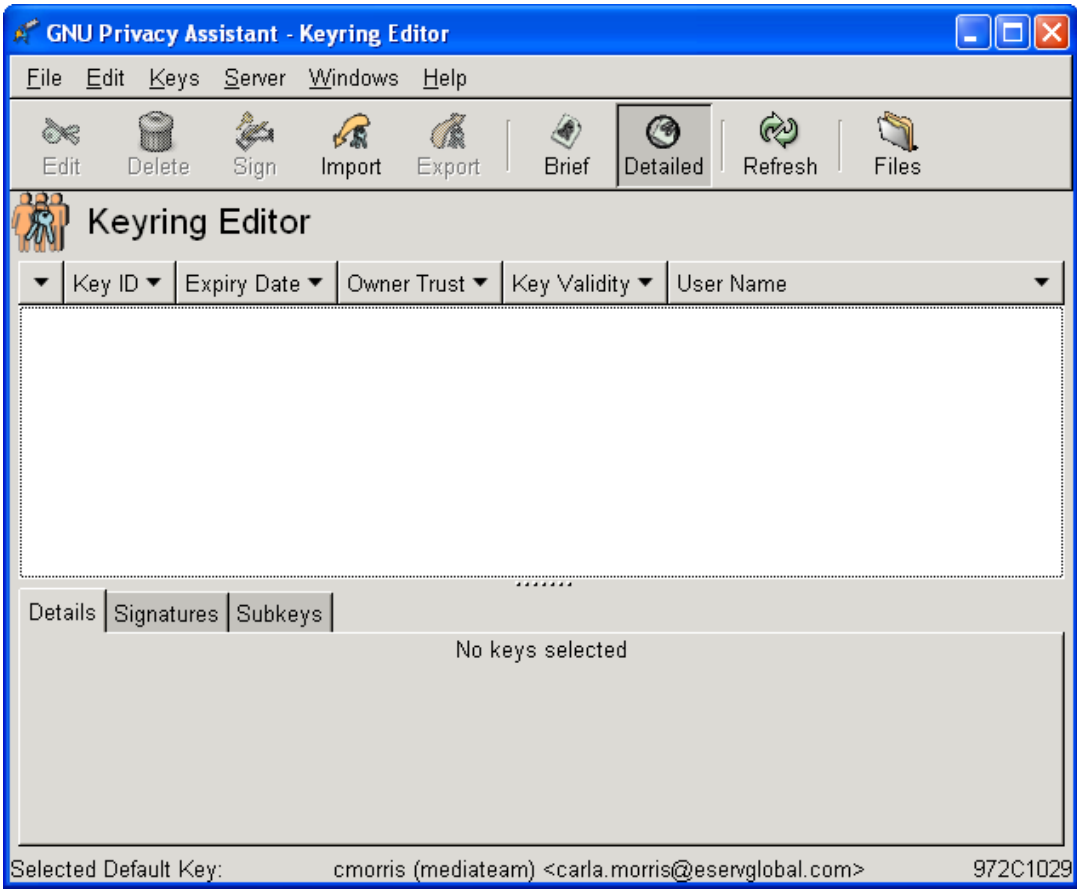
```
-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: GnuPG v1.4.7 (GNU/Linux)

mQGiBEfsGcYRBADQn11NUtpDz4PMF8XrCZBRKMBzlk1scAIxhqI2TKK8Ec46tMn6cbVXyZaE0yKbLUjKxi
bYBDeWBQQT8Sd+5wADBQf8CkTb6UCy9rI0mG5MQIphIGtiQLeJWCwGoZJW3q5kr6/4zhfPl7DNiUMk0Mv5
aGZMSzMRdlS+HXp4flZJ3yazyHkmfk7KQOZ84g12bHlj+g7jyJwHYn8SrgMDTYe/oYyZkMeLf6qr9jLB90
4dDua8eo2zi5kidRGegt4XEpfjFA1RfVxhCjhlUQi/5IumuFTHBQpIe3ebFJ1JzpIMFOnouMuFsLmNvbT6
IYAQTEQIUIGyh4g85SUaxYCjyBfYM3zMpe7kc0Rg+IAnn9Q36nt0e7Ic0oFL8wiQiTBACMKF7Ry71xtiNO
YLCQgHAwIEFQIIAwgcTnikm6NJcmtaZj8btI7E5rikVQYxMiXnlNrDKX9b01JSHedlbKFpwgOUSm1tIzp4
qQj3ltUEvqkuR/JdAIVSrgDwwdMVXHvwbitwXKdYyaPHz4zUr+wT/1pqk6eodUuxUyzGSd/9LpZlI3eiH1
jVmvP3kaDrQ3BecTxmCoX4BVfqqu4GIqXx2QWAgMBAh4BAheAAAoJEFR/mQrBD3ZEVDExm/mGUl90IqHIY
pXfmx7zQeBBoYhka4gXm4G4wuqlocQEW2T3bkRRJa5mg87ACkBkPANcIh5YbKYviVX0lJUeh3HzQVITl02
Hwk2iL9V/Lp8PCkYz3e2Ei3QPLmOQLbZJd1j6PKOZYpzfAWKtH5Vnge0u0RuL4F8AKXVcSYIQpZ/5PT17R
qjCOQmFCNmGkiDxABlGA6aY45Lh2ZpZWxkQGVzZXJ2Z2AIG17RbaIodvISBTisHn/0tA6zjlUWQMAYB2mH
JO9IHlXDRMAJwKWoBiLJ0OgcnX2Cvnw7MmKawbx7kCDQRH7BnGEAgAmCEwNyyXyabQKoQtZIt5fzTgokn1
BAUCR+wZxgIbAwA9rQ1cGF1bHNjaG9maWVsZCAoQmxhaCkgPFBhdWBDc0R374AnjwuU2NobCdOfdwmqyUo
DD0LlZvbUyNP0wCgkYqA02kjimc1yS5Mlq4a1z1oQA0EALhKCcDpQWdPNClvWbIwgQnrQpdf271YS9IyEs
HXFB3mnnUe12i4QjQRdmyJOxVTa2ISF75sr6Lvv5tjInCPky709cjsPkTYjOYPiWUNRxPHazlYpW8tmP9w
7cUZRiw/1X91PQhph9gw3QO+zh0/2zge/ZZtxr/CiDp0JqtyAJ6APmguzTQ7g/4rguu5oM0SqTrPo3sS6f
kEzWN+OWCj8wUDGvK+V4RRQzFDoC7hfLUNyi1yGpZRII1hSvHrthe6oWoSwPXyl292YEsiGHU7TCfizadz
aVLPppdCc89pwz3kCqkp1oQ6xEEdNVjcghoBt2HNgg/cLMEIQP+yccNiOzaHc+RCY7DqfjfzBFSuRUpjAq
FBFeBLx7IhJBBgRAgAJBQJH7BnGAhsMAAoJEFR/mQrOHUMnzxvYmh5TnerAMMtjQoJXjhY5SAJ4gWB2Pc9
Ujw8rGFoO5Zg==
=Qu4J
-----END PGP PUBLIC KEY BLOCK-----
```

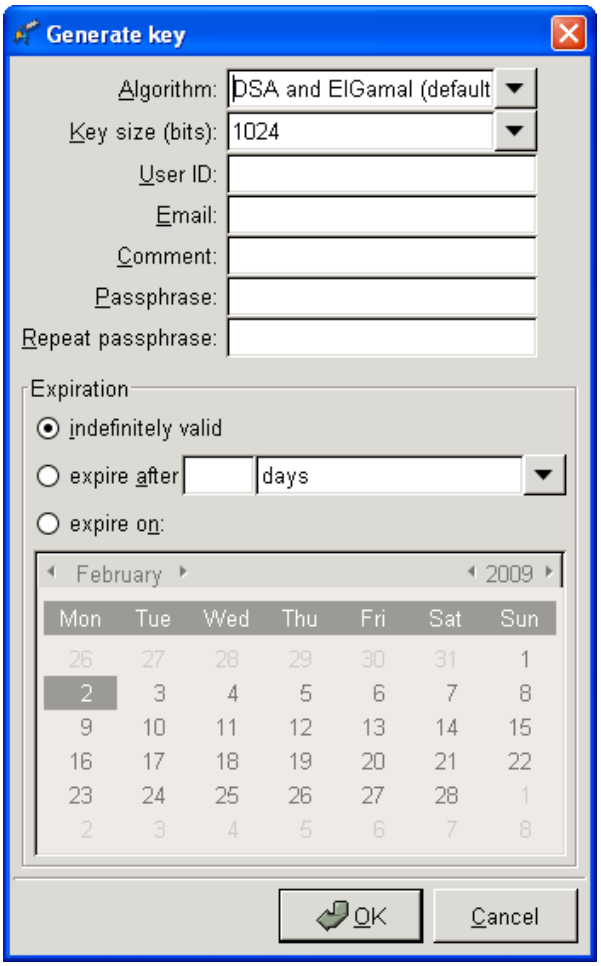
Step	Action
2	Send the public key to the operator. The operator will import it into Charging Control Services and will use it when generating voucher and subscriber account (calling card) batches.

Generating keys using Gpg4Win

Follow these steps to generate a new key using Gpg4Win.

Step	Action
1	Start GNU Privacy Assistant - Keyring Editor. **Result:** The GNU Privacy Assistant - Keyring Editor screen will open. 
2	Select Keys, New Key. **Result:** The Generate key screen opens.

Step	Action



3 Select values from the **Algorithm:** and **Key size (bits):** fields. The values you pick should have been agreed with the operator you are printing for.

4 In the **User ID:** field, enter your email adress.

5 In the **Email:** field, enter your email address.

6 In the **Comment:** field enter a description of this key. The comment should identify who the printshop is, and may also identify the operator.

7 In the **Passphrase:** and **Repeat passphrase:** fields, type the passphrase to use with this key.

Important:

- This passphrase must be entered when the files are decrypted. If the passphrase is not available, the files will not be able to be decrypted and a new pair of keys and batch file will have to be generated.
- The passphrase is an important contributor to the overall security of the encryption. Ensure you follow any guidelines set by the operator, and that you pick a secure password.

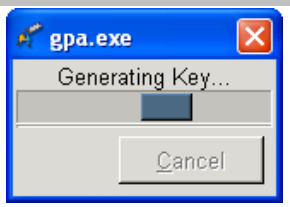
8 In the **Expiration** area, select an expiry date as agreed with the operator.

9 Click **OK**.

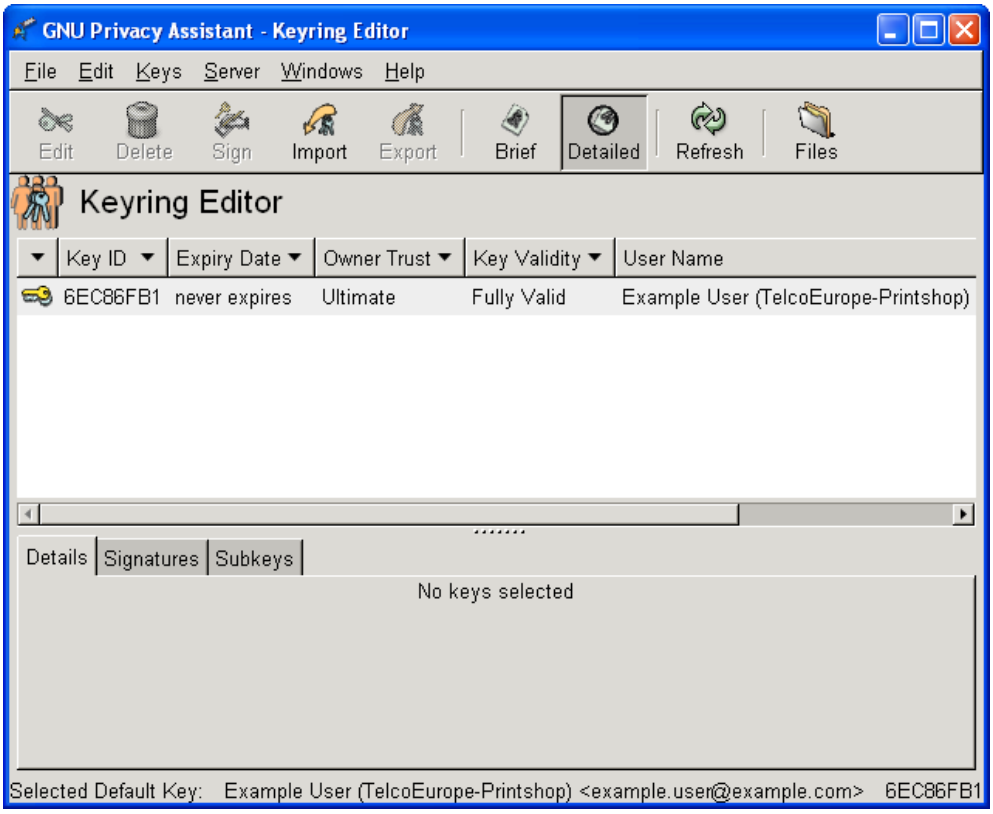
Results:

- The gpa.exe screen displays with a moving bar that indicates it is generating the new key.

Step	Action



- When the new key has been generated, it appears in the GNU Privacy Assistant - Keyring Editor screen.

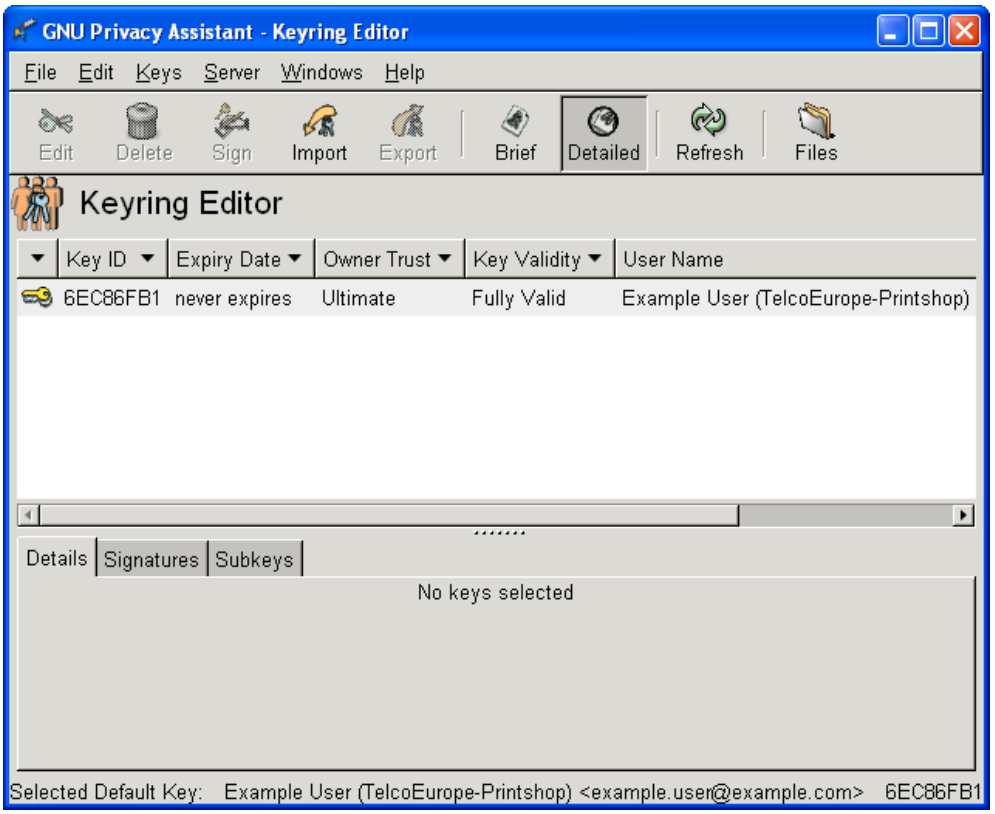


Exporting keys using Gpg4Win

Follow these steps to export keys which have been generated by gpg.

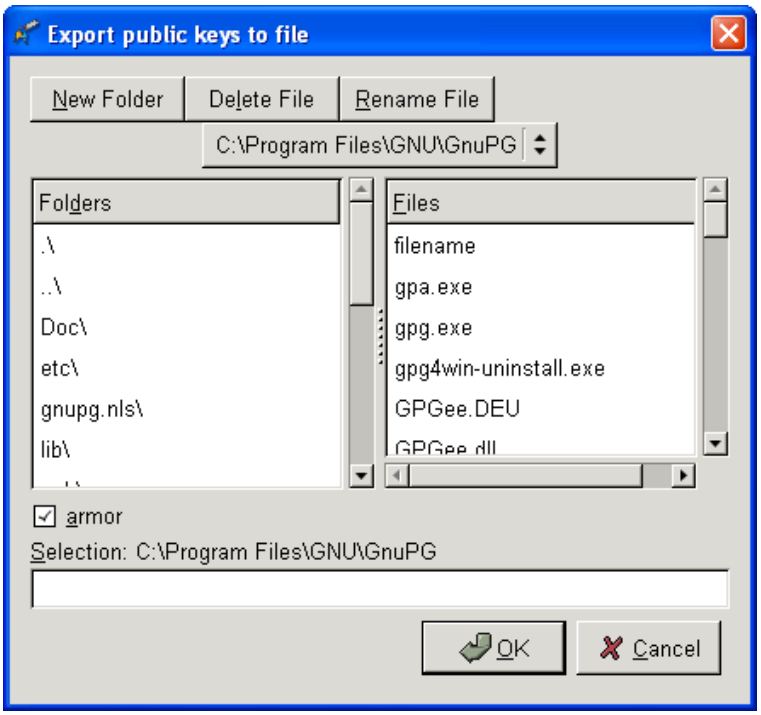
Step	Action
1	On the machine where Gpg4Win generated the keys, start GNU Privacy Assistant - Keyring Editor. **Result:** The GNU Privacy Assistant - Keyring Editor screen will open.

Step	Action



2 Select the key you want to export and click **Export**.

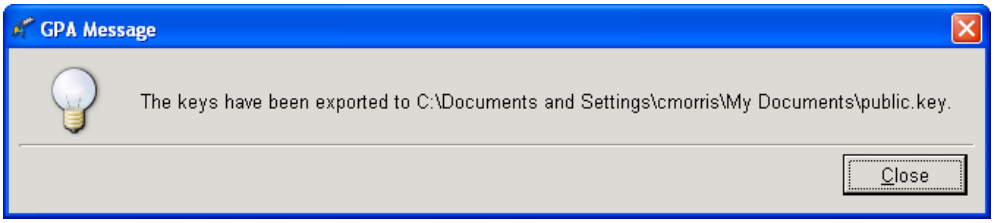
Result: The Export public keys to file screen appears.



3 Browse to the directory you want the exported keys to be stored in.

4 Type the name of the file you want the exported keys to be stored in and click **OK**.

Step	Action
	Result: Gpg4Win exports the public key to the specified file, and then displays the GPA Message prompt.



5	Click **Close**.
6	Send the public key to the operator. The operator will import it into Charging Control Services and will use it when generating voucher and subscriber account (calling card) batches.

Decrypting Files

Introduction

The batch file provided by the operator for printing will have been encrypted using the public key provided by the printshop. This file will need to be decrypted using the matching private key. There are two methods for decrypting the files.

The voucher batch file should be placed on the Printshop target PC. The PGP software on the PC should be used to decrypt the voucher batch file.

Decrypting files using gpg

Follow these steps to decrypt a file using gpg.

Step	Action
1	Copy the batch file to the machine where the key was exported from.

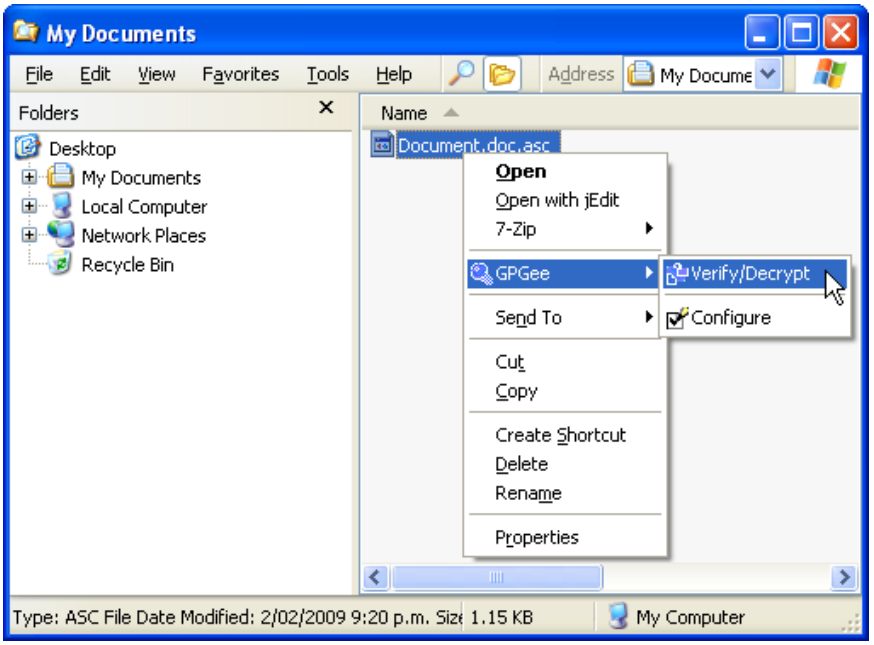
2	Use gpg to decrypt the batch file.
	Example command: `gpg -o batchFile.txt --decrypt batchFile.gpg`
	Result: Text similar to the following will appear.
	`You need a passphrase to unlock the secret key for`
	`user: "ExampleUser (TelcoEurope-Printshop) <example.user@example.com>"`
	`2048-bit ELG-E key, ID 69372FCB, created 2009-01-16 (main key ID 2E72F865)`
	
	`Enter passphrase:`

3	Type the passphrase used when the key was generated and press **Enter**.
	Result: gpg will use the passphrase to decrypt the file. Text similar to the following will appear.
	`user: "ExampleUser (TelcoEurope-Printshop) <example.user@example.com>"`
	`2048-bit ELG-E key, ID 69372FCB, created 2009-01-16 (main key ID 2E72F865)`
	The decryption should now be complete.

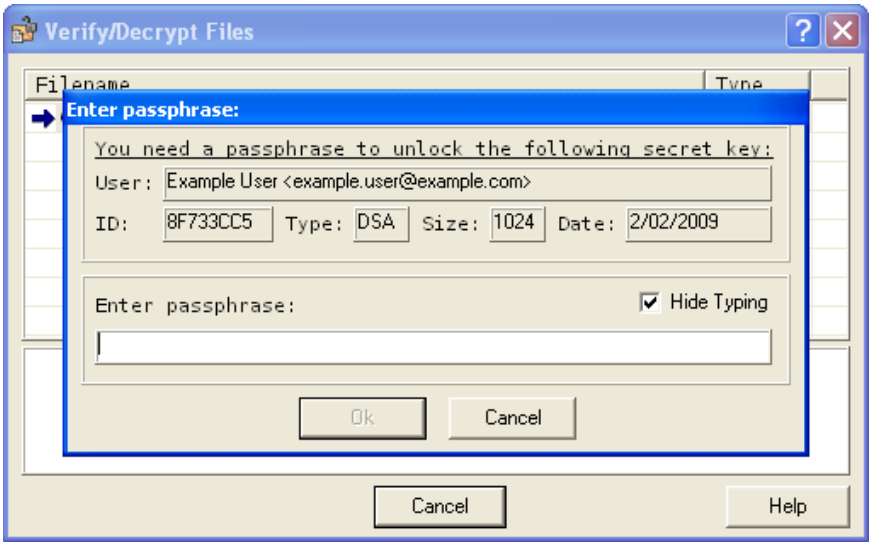
Decrypting files using Gpg4Win

Follow these steps to decrypt an encrypted file using Gpg4Win.

Step	Action
1	Using Windows Explorer, browse to the file you want to decrypt.
2	Select the file and right-mouse-click on the document. **Result:** The right-mouse-click menu appears.

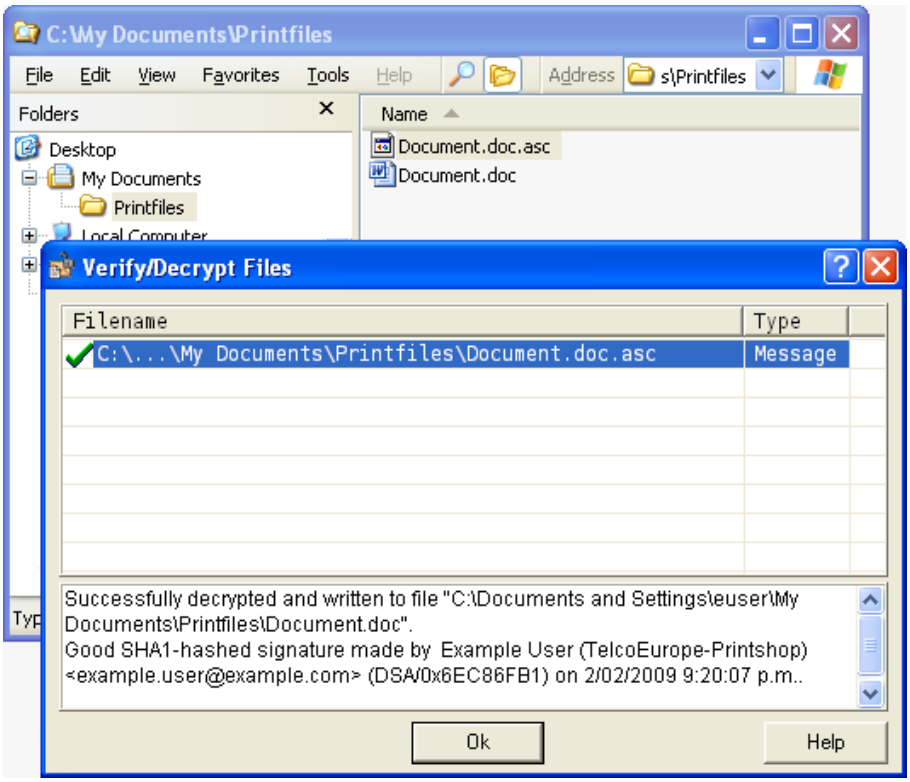


3	Select GPGee, Verify/Decrypt. **Result:** The Verify/Decrypt Files and Enter passphrase screens open.



4	Type the passphrase for the specified key in the **Enter passphrase:** field.
5	Click **Ok**. **Result:** The results of the file decryption will be displayed in the Verify/Decrypt Files screen and the decrypted file will be saved to the same directory as the souce file.

Step	Action



Exported voucher batch files

Voucher batch file format is controlled by the security library, and the voucher writer plugin used to generate the batch. Which libraries and plugins are used is defined by the Authentication Module (PAM) and the Authentication Rule specified in the New Voucher Batch screen.

Header fields are in the format "<Key field name>=<value>". Key field names always start with an alphabetic character. This makes it easy to distinguish them from voucher records (which always start with a number).

The following header fields are used in the voucher batch file header, (although downstream processors should detect any "<Key field name>=<value>" lines).

Header field	Description
BilingEngineName=<str>	The name of the Voucher and Wallet Server where the voucher resides.
VoucherTypeName=<str>	The name of the voucher type as created on the Convergent Charging Controller platform. The voucher type contains the following information: • Pre-use expiry period (number of days and hours that this voucher is valid in a pre-use state) • Wallet expiry period (change the current wallet expiry date by this many days and hours) • Voucher number length • Voucher PIN length • A list of all the balance types, associated values and balance expiry date modifications which will be changed/updated

Header field	Description
	when this voucher is redeemed **Note:** It will be up to the operator to provide the details of the voucher type described here to the print shop so that any specific voucher details can be printed on the final vouchers.
AuthRuleName=<str>	The name of the authentication rule which was used for creating the voucher number and PIN.
AuthModName=<str>	The name of the pluggable authentication module (PAM) (Convergent Charging Controller specific) used for creating the voucher PIN.
VoucherBatchBatch=<str>	A two character identifier (non unique) for this voucher batch.
VoucherBatchID=<int>	The system generated ID for this voucher batch.
OriginalCount=<int>	The number of vouchers created in this batch.
StartOfRange=<int>	Beginning of the range of voucher numbers.
EndOfRange=<int>	End of the range of voucher numbers.

A line consisting of a single equal sign (=) terminates the header lines. All subsequent lines are voucher detail records.

CCS3 DES voucher batch example

This text shows an example export voucher batch file generated by ccsVoucher_CCS3 using the DES encryption library (and a bespoke voucher file writer plugin to format the non-header details), but no GnuPG key.

```
#
# Voucher file for batch 83
# Generated by ccsVoucher at Tue Nov 11 12:55:27 2008
# (key=value or
voucherserialnumber,vouchernumber,vouchersecret,vouchercontext,voucherprivate_secret
)
#
BillingEngineName=PCDEV
VoucherTypeName=DES
AuthRuleName= DES (VL=10 VP=4)
AuthModName=DES
VoucherBatchBatch=
VoucherBatchID=83
OriginalCount=2
StartOfRange=1000000001
EndOfRange=1000000002
=
#
# Voucher records start
#
1000000001,8986
1000000002,4887
#
# End of voucher records
#
```

CCS3 CB10 voucher batch example

This text shows an example export voucher batch file generated by ccsVoucher_CCS3 using the 'CB10 HRN' encryption library using the 'HRNGEN' encryption algorithm, but no GnuPG key.

```
#
# Voucher file for batch 85
```

```
# Generated by ccsVoucher at Tue Nov 11 12:55:27 2008
# (key=value or voucherbatch,preuseexpiry,hrn,serialnumber)
#
BillingEngineName=PCDEV
VoucherTypeName=CB10
AuthRuleName=CB10 (S=14 R1=2 R2=2 R3=0)
AuthModName=CB10 HRN
VoucherBatchBatch=
VoucherBatchID=85
OriginalCount=2
StartOfRange=00000000000001
EndOfRange=00000000000002
=
#
# Voucher records start
#
85,20090101000000,631599527570333589,1000000138
85,20090101000000,855619036698319621,1000000139
#
# End of voucher records
#
```

CCS3 CB10 GPG voucher batch example

This text shows an example export voucher batch file generated by ccsVoucher_CCS3 using the 'CB10 HRN' encryption library using the 'HRNGEN' encryption algorithm, and GnuPG encryption.

Note: This file has been decrypted using the gpg key.

```
#
# Voucher file for batch 86
# Generated by ccsVoucher at Tue Nov 11 12:55:27 2008
# (key=value or voucherserialnumber,hrnserialnumberseed,hrn,nrnlength,hrnc)
#
BillingEngineName=PCDEV
VoucherTypeName=CB10 HRN
AuthRuleName= CB10 (S=14 R1=2 R2=2 R3=0)
AuthModuleName=CB10 HRN
VoucherBatchBatch=
VoucherBatchID=86
OriginalCount=2
StartOfRange=00000000000003
EndOfRange=00000000000004
=
#
# Voucher records start
#
86,20090101000000,057195727842702414,1000000138
86,20090101000000,363323157948027866,1000000139
#
# End of voucher records
#
```

Exported card/account batch files

Subscriber account/calling card batch file format is controlled by the account writer plug-in used to generate the batch. Which libraries are used is defined by the authentication name specified in the New Subscriber Batch screen.

Header fields are in the format "*Key_field_name*=*value*". Key field names always start with an alphabetic character. This makes it easy to distinguish them from voucher records (which always start with a number).

The following header fields are used in the voucher batch file header, (although downstream processors should detect any "*Key_field_name*=*value*" lines).

Header field	Description
AccountBatchID=*int*	The ID of the subscriber account batch.
ServiceProviderID=*int*	The ID number of the service provider the subscriber batch belongs to. When ccsAccount is started by the screens the value of this field is populated by the id of the service provider which is selected in the **Service Provider** field of the Subscriber Management screen when the **New** button is clicked.
AccountTypeID=*int*	The product type the subscriber batch has. When ccsAccount is started by the screens the value of this field is populated by the **Product Type** field on the New Subscriber Batch screen.
maxConcurrent=*int*	The maximum number of concurrent connections wallets generated with this subscriber batch can have. When ccsAccount is started by the screens the value of this field is populated by the **Maximum Concurrent Accesses** field on the New Subscriber Batch screen.
BatchSize=*int*	The number of subscriber accounts in this batch. When ccsAccount is started by the screens the value of this field is populated by the **Batch Size** field on the New Subscriber Batch screen.
RangeStart=*int*	Beginning of the range of subscriber account numbers. When ccsAccount is started by the screens the value of this field is populated by the **Card Number Start Range** field on the New Subscriber Batch screen.
RangeEnd=*int*	End of the range of subscriber account numbers. When ccsAccount is started by the screens the value of this field is populated by the **Card Number End Range** field on the New Subscriber Batch screen.
AuthenticationModuleID=*int*	The ID of the authentication module used for: • Encryption and/or random generation of PINs for this batch • (optionally) sends the output file for encryption by gpg. When ccsAccount is started by the screens the value of this field is populated by the **PAM Name** field on the New Subscriber Batch screen.
BillingEngineID=*int*	The ID number of the Voucher and Wallet Servers .
CurrencyID=*int*	The ID of the currency the wallets generated with this subscriber batch will use. When ccsAccount is started by the screens the value of this field is populated by the **Wallet Currency** field on the New Subscriber Batch screen.
LimitType=*str*	The type of limit the wallets generated with this subscriber batch will use.
BalanceType=*int*	The balance type ID of the balance type this wallet will have any initial value stored in.

A line consisting of a single equal sign (=) terminates the header lines. All subsequent lines are voucher detail records.

Card/account output file

This text shows an example export subscriber account/calling card output file.

```
# Account Batch Output File
# Generated Wed Dec 31 01:24:29 2008
#
AccountBatchID=59
ServiceProviderID=1
AccountTypeID=7
maxConcurrent=1
BatchSize=10
RangeStart=8815000000
RangeEnd=8819990000
AuthenticationModuleID=4
BillingEngineID=2
CurrencyID=2
LimitType=DEBT
BalanceType=1
=
Dec 31 01:24:29.861203 ccsAccount(15179) NOTICE: Beginning account generation.
16309877,3415992,7,G8.H3zCjoKzbY,8800127
19052821,0363266,7,G8fRbQy015unk,8800128
18627603,5447142,7,G82efn9Gh2gSY,8800129
16635167,9003194,7,G8nkF67MOzS9g,8800130
19498256,8441931,7,G8tfZtbQvbOIg,8800131
18758105,8744644,7,G8CSYLULMZtww,8800132
17349265,3517347,7,G8GH/BMl4HHzs,8800133
16223817,0064708,7,G8MbgIe4gPO.U,8800134
16089674,7771756,7,G8lXd7ySSzsVw,8800135
16405822,1207166,7,G8JugOSguxjqg,8800136
Dec 31 01:24:35.514685 ccsAccount(15179) NOTICE: Progress 10/10 (100.0%) Complete
Dec 31 01:24:35.515578 ccsAccount(15179) NOTICE: Account generation complete.
```

Glossary of Terms

CCS

1) Charging Control Services (or Prepaid Charging) component.

2) Common Channel Signalling. A signalling system used in telephone networks that separates signalling information from user data.

Convergent

Also “convergent billing”. Describes the scenario where post-paid and pre-paid calls are handed by the same service platform and the same billing system. Under strict converged billing, post-paid subscribers are essentially treated as “limited credit pre-paid”.

HRN

Hidden Reload Number

PC

Point Code. The Point Code is the address of a switching point.

PIN

Personal Identification Number

Service Provider

See Telco.

Telco

Telecommunications Provider. This is the company that provides the telephone service to customers.

Telecommunications Provider

See Telco.

Index