PGP® Command Line

User’s Guide
**Version Information**


**Copyright Information**

Copyright © 1991–2007 by PGP Corporation. All Rights Reserved. No part of this document can be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without the express written permission of PGP Corporation.

**Trademark Information**

PGF, Pretty Good Privacy, and the PGP logo are registered trademarks and Rest Secured is a trademark of PGP Corporation in the US and other countries. IDEA is a trademark of Ascend Tech AG. Windows and ActiveX are registered trademarks of Microsoft Corporation. AOL is a registered trademark, and AOL Instant Messenger is a trademark, of America Online, Inc. Red Hat and Red Hat Linux are trademarks or registered trademarks of Red Hat, Inc. Linux is a registered trademark of Linus Torvalds. Solaris is a trademark or registered trademark of Sun Microsystems, Inc. AIX is a trademark or registered trademark of International Business Machines Corporation. HP-UX is a trademark or registered trademark of Hewlett-Packard Company. SSH and Secure Shell are trademarks of SSH Communications Security, Inc. Rendezvous and Mac OS X are trademarks or registered trademarks of Apple Computer, Inc. Symantec, the Symantec logo, and LiveUpdate are registered trademarks of Symantec Corporation. All other registered and unregistered trademarks in this document are the property of their respective owners.

**Licensing and Patent Information**

The IDEA cryptographic cipher described in U.S. patent number 5,214,703 is licensed from Ascend Tech AG. The CAST encryption algorithm is licensed from Northern Telecom, Ltd. PGP Corporation has secured a license to the patent rights contained in the patent application Serial Number 10/655,563 by The Regents of the University of California, entitled Block Cipher Mode of Operation for Constructing a Wide-blocksize block Cipher from a Conventional Block Cipher. Some third-party software included in PGP Universal Server is licensed under the GNU General Public License (GPL). PGP Universal Server as a whole is not licensed under the GPL. If you would like a copy of the source code for the GPL software included in PGP Universal Server, contact PGP Support (http://www.pgpp.com/support). PGP Corporation may have patents and/or pending patent applications covering subject matter in this software or its documentation; the furnishing of this software or documentation does not give you any license to these patents.

**Acknowledgments**

This product includes or may include:

- The Zip and ZLib compression code, created by Mark Adler and Jean-Loup Gailly, is used with permission from the free Info-ZIP implementation, developed by zlib (http://www.zlib.net). • libxml2, the XML C parser and toolkit developed for the Gnome project and distributed and copyrighted under the MIT License found at http://www.opensource.org/licenses/mit-license.html. Copyright © 2007 by the Open Source Initiative. • bzip2 1.0, a freely available high-quality data compressor, is copyrighted by Julian Seward, © 1996-2005. • Application server (http://www.jakarta.apache.org/), web server (http://www.apache.org/), Jakarta Commons (http://jakarta.apache.org/commons/license.html) and log4j, a Java-based library used to parse HTML, developed by the Apache Software Foundation. The license is at www.apache.org/licenses/LICENSE-2.0.txt. • Castor, an open-source, data-binding framework for moving data from XML to Java programming language objects and from Java to databases, is released by the ExoLab Group under an Apache 2.0-style license, available at http://www.castor.org/license.html. • Xalan, an open-source software library from the Apache Software Foundation that implements the XSLT XML transformation language and the XPath XML query language, is released under the Apache Software License, version 1.1, available at http://xml.apache.org/xalan-j/GPLicense1.1.txt, an open-source implementation of the Java Management Extensions (JMX), is released under an Apache-style license, available at http://www.apache.org/licenses/license. • PGP, Pretty Good Privacy, and the PGP logo are registered trademarks of PGP Corporation. All Rights Reserved. No part of this document can be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without the express written permission of PGP Corporation. • The Zip and ZLib compression code, created by Mark Adler and Jean-Loup Gailly, is used with permission from the free Info-ZIP implementation, developed by zlib (http://www.zlib.net). • jpeglib version 6b is based in part on the work of the Independent JPEG Group. (http://www.jpeg.org) • libxml2 is the XML C parser and toolkit developed for the Gnome project and distributed and copyrighted under the MIT License found at http://www.opensource.org/licenses/mit-license.html. • PC/SC Lite is a free implementation of PC/SC, a specification for Smart Cards. • Postfix, an open source mail transfer agent (MTA), is released under the IBM Public License 1.0, available at http://www.opensource.org/licenses/ibmpl.php. • PostgreSQL, a free software object-relational database management system, is released under a BSD-style license, available at http://www.opensource.org/licenses/ibmpl.php. • PostgreSQL is a free software object-relational database management system, is released under a BSD-style license, available at http://www.postgresql.org/about/license. • Vixie-cron is the Vixie version of cron, a standard UNIX daemon that runs specified programs at scheduled times. Copyright © 1993, 1994 by Paul Vixie; used by permission.

**Export Information**

Export of this software and documentation may be subject to compliance with the rules and regulations promulgated from time to time by the Bureau of Export Administration, United States Department of Commerce, which restricts the export and re-export of certain products and technical data.

**Limitations**

The software provided with this documentation is licensed to you for your individual use under the terms of the End User License Agreement provided with the software. The information in this document is subject to change without notice. PGP Corporation does not warrant that the information meets your requirements or that the information is free of errors. The information may include technical inaccuracies or typographical errors. Changes may be made to the information and incorporated in new editions of this document, if and when made available by PGP Corporation.
## Contents

1  **PGP Command Line Basics** ................................................. 7  
   Important Concepts .......................................................... 7  
   Getting Started ............................................................... 8  

2  **Installation** ................................................................. 9  
   Overview ................................................................. 9  
   System Requirements ...................................................... 10  
   Installing on AIX ............................................................ 13  
   Installing on HP-UX ......................................................... 15  
   Installing on Mac OS X ...................................................... 16  
   Installing on Red Hat Enterprise Linux or Fedora Core ............... 18  
   Installing on Solaris .......................................................... 20  
   Installing on Windows ....................................................... 22  

3  **Licensing** .................................................................. 25  
   Overview ................................................................. 25  
   License Recovery ............................................................ 26  
   Using a License Number .................................................... 27  
   Using a License Authorization ............................................ 28  
   Re-Licensing ................................................................. 29  
   Through a Proxy Server .................................................... 30  

4  **The Command-Line Interface** ........................................ 31  
   Overview ................................................................. 31  
   Flags and Arguments ....................................................... 33  
   Configuration File .......................................................... 36  
   Environment Variables ..................................................... 41  
   Standard Input, Output, and Error ...................................... 42  
   Specifying a Key ............................................................. 43  
   ‘Secure’ Options ............................................................. 44  

5  **First Steps** ................................................................. 45  
   Overview ................................................................. 45  
   Creating Your Keypair ..................................................... 46  
   Protecting Your Private Key .............................................. 47  
   Distributing Your Public Key .............................................. 48  
   Getting the Public Keys of Others ...................................... 50  
   Verifying Keys ............................................................. 51
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>Cryptographic Operations</td>
<td>53</td>
</tr>
<tr>
<td></td>
<td>Overview</td>
<td>53</td>
</tr>
<tr>
<td></td>
<td>Commands</td>
<td>54</td>
</tr>
<tr>
<td>7</td>
<td>Key Listings</td>
<td>71</td>
</tr>
<tr>
<td></td>
<td>Overview</td>
<td>71</td>
</tr>
<tr>
<td></td>
<td>Commands</td>
<td>72</td>
</tr>
<tr>
<td>8</td>
<td>Working with Keyservers</td>
<td>81</td>
</tr>
<tr>
<td></td>
<td>Overview</td>
<td>81</td>
</tr>
<tr>
<td></td>
<td>Commands</td>
<td>81</td>
</tr>
<tr>
<td>9</td>
<td>Managing Keys</td>
<td>87</td>
</tr>
<tr>
<td></td>
<td>Overview</td>
<td>89</td>
</tr>
<tr>
<td></td>
<td>Commands</td>
<td>89</td>
</tr>
<tr>
<td>10</td>
<td>Miscellaneous Commands</td>
<td>133</td>
</tr>
<tr>
<td></td>
<td>Overview</td>
<td>133</td>
</tr>
<tr>
<td>11</td>
<td>Options</td>
<td>139</td>
</tr>
<tr>
<td></td>
<td>Boolean Options</td>
<td>140</td>
</tr>
<tr>
<td></td>
<td>Integer Options</td>
<td>150</td>
</tr>
<tr>
<td></td>
<td>Enumeration Options</td>
<td>160</td>
</tr>
<tr>
<td></td>
<td>String Options</td>
<td>170</td>
</tr>
<tr>
<td></td>
<td>List Options</td>
<td>180</td>
</tr>
<tr>
<td></td>
<td>List Options</td>
<td>180</td>
</tr>
<tr>
<td></td>
<td>File Descriptors</td>
<td>184</td>
</tr>
<tr>
<td>A</td>
<td>Lists</td>
<td>187</td>
</tr>
<tr>
<td></td>
<td>Basic Key List</td>
<td>187</td>
</tr>
<tr>
<td></td>
<td>Detailed Key List</td>
<td>193</td>
</tr>
<tr>
<td></td>
<td>Key List in XML Format</td>
<td>206</td>
</tr>
<tr>
<td></td>
<td>Detailed Signature List</td>
<td>213</td>
</tr>
<tr>
<td>B</td>
<td>Usage Scenarios</td>
<td>219</td>
</tr>
<tr>
<td></td>
<td>Secure Off-Site Backup</td>
<td>219</td>
</tr>
<tr>
<td></td>
<td>PGP Command Line and PGP Desktop</td>
<td>220</td>
</tr>
<tr>
<td></td>
<td>Compression Saves Money</td>
<td>221</td>
</tr>
<tr>
<td></td>
<td>Surpasses Legal Requirements</td>
<td>222</td>
</tr>
<tr>
<td>C</td>
<td>Quick Reference</td>
<td>223</td>
</tr>
<tr>
<td></td>
<td>Commands</td>
<td>223</td>
</tr>
<tr>
<td></td>
<td>Options</td>
<td>225</td>
</tr>
<tr>
<td></td>
<td>Environment Variables</td>
<td>229</td>
</tr>
<tr>
<td>Section</td>
<td>Page</td>
<td></td>
</tr>
<tr>
<td>------------------------------------------</td>
<td>------</td>
<td></td>
</tr>
<tr>
<td>Configuration File Variables</td>
<td>230</td>
<td></td>
</tr>
<tr>
<td>D  Command Comparison</td>
<td>233</td>
<td></td>
</tr>
<tr>
<td>E  Codes and Messages</td>
<td>235</td>
<td></td>
</tr>
<tr>
<td>Messages Without Codes</td>
<td>236</td>
<td></td>
</tr>
<tr>
<td>Messages With Codes</td>
<td>236</td>
<td></td>
</tr>
<tr>
<td>Exit Codes</td>
<td>256</td>
<td></td>
</tr>
<tr>
<td>F  Frequently Asked Questions</td>
<td>257</td>
<td></td>
</tr>
<tr>
<td>G  Glossary</td>
<td>261</td>
<td></td>
</tr>
<tr>
<td>Index</td>
<td>271</td>
<td></td>
</tr>
</tbody>
</table>
This chapter describes some important PGP Command Line concepts and gives you a high-level overview of the things you need to do to set up and use PGP Command Line.

Important Concepts

The following concepts are important for you to understand:

- **PGP Command Line**: A software product from PGP Corporation that automates the processes of encrypting/signing, decrypting/verifying, and file wiping; it provides a command-line interface to PGP technology.

- **command-line interface**: An interface where you type commands at a command prompt. PGP Command Line uses a command-line interface.

- **keyboard input**: PGP Command Line was designed so that all relevant information can be entered at the command line, thus requiring no further input from the keyboard to implement the commands.

- **scripting**: PGP Command Line commands can be easily inserted into scripts to be used for automating tasks. For example, if your company regularly copies a large database to an off-site backup and then stores it there, PGP Command Line commands can be added to the script that does this so that the database is encrypted before it is transmitted to the off-site location and then decrypted when it arrives. PGP Command Line commands are easily added to shell scripts or scripts written with scripting languages (such as Perl or Python, for example).

- **environment variables**: Environment variables control various aspects of PGP Command Line behavior; for example, the location of the PGP Command Line home directory. Environment variables are established on the computer running PGP Command Line.

- **configuration file variables**: When PGP Command Line starts, it reads the configuration file, which includes special configuration variables and values for each variable. These settings affect how PGP Command Line operates. Configuration file variables can be changed permanently by editing the configuration file or overridden on a temporary basis by specifying a value for a configuration file variable on the command line.

- **Self-Decrypting Archives (SDAs)**: PGP Command Line lets you create SDAs, compressed and conventionally encrypted archives that require a passphrase to decrypt. SDAs contain an executable for the target platform, which means the recipient of an SDA does not need to have any PGP software installed to open the archive. You can thus securely transfer data to recipients with no PGP software installed. You will have to communicate the passphrase of the SDA to the recipient, however.
- **Additional Decryption Key (ADK):** PGP Command Line supports the use of an ADK, which is an additional key to which files or messages are encrypted, thus allowing the keeper of the ADK to retrieve data or messages as well as the intended recipient. Use of an ADK ensures that your corporation has access to all its proprietary information even if employee keys are lost or become unavailable.

- **PGP Zip archives:** The PGP Zip feature lets you encrypt/sign groups of files or entire directories into a single compressed archive file. The archive format is tar and the supported compression formats are Zip, BZip2, and Zlib.

---

**Getting Started**

Now that you know a little bit about PGP Command Line, let’s go deeper into what you need to do to get started using it:

1. **Install PGP Command Line.** Specific instructions for installing PGP Command Line on the supported platforms are in Chapter 2, Installation.

2. **License the software.** PGP Command Line functionality is extremely limited until you license the software. Refer to Chapter 3, Licensing for more information.

3. **Create your default key pair.** Most PGP Command Line operations require a key pair (a private key and a public key). Refer to “Creating Your Keypair” on page 46 for more information.

4. **Protect your private key.** Because your private key can decrypt your protected data, it is important that you protect it. Do not write down or tell someone the passphrase. It is a good idea to keep your private key on a machine that only you can access, and in a directory that is not accessible from the network. Also, you should make a backup of the private key and store it in a secure location.

   Refer to “Protecting Your Private Key” on page 47 for more information.

5. **Exchange public keys with others.** In order to encrypt data to someone you need their public key; and they need yours to encrypt data to you.

   Refer to “Getting the Public Keys of Others” on page 50 for more information about how to obtain public keys.

6. **Verify the public keys you get from the keyserver.** Once you have a copy of someone’s public key, you add it to your public keyring. When you get someone’s public key, you should make sure that it has not been tampered with and that it really belongs to the purported owner. You do this by comparing the unique fingerprint on your copy of someone’s public key to the fingerprint on that person’s original key.

   For more information about validity and trust, refer to *An Introduction to Cryptography* (it was put onto your computer during installation). For instructions how to verify someone’s public key, see “--fingerprint” on page 72.

7. **Start securing your data.** After you have generated your key pair and have obtained public keys, you can begin encrypting, signing, decrypting, and verifying your data.
Installation
Instructions for All Platforms

This chapter lists the system requirements for, and tells you how to install PGP Command Line onto, the six supported platforms: AIX, HP-UX, Mac OS X, Linux, Solaris, and Windows. It also includes uninstall instructions.

Overview

PGP Command Line can be installed on these platforms:

- Windows Vista, Windows Server 2003 (SP 1), Windows XP (SP 2), Windows XP 64-bit, Windows 2000 (SP 4)
- HP-UX 11i and above (PA-RISC only)
- IBM AIX 5.2 and above
- RedHat Enterprise Linux 3.0 and above (x86 only)
- Fedora Core 3 and above (x86_64)
- Sun Solaris 9 (SPARC only)
- Apple Mac OS X 10.4 and above (Universal binary)

PGP Command Line uses a specific directory for the application data such as the configuration file, and a specific directory (called the home directory) for the files it creates, such as keyring files.

On any UNIX system, the application data and the home directory are identical and they are configured through the $HOME environment variable. For more information, refer to the installation instructions for the specific UNIX platform.

On Windows, the application data directory is used to store data such as the configuration file PGPrefs.xml. The home directory is called “My Documents” and is used to store keys. These two directories can be named differently, depending on the specific version on Windows. For more information, refer to “To Install on Windows” on page 22.

You can also use the --home-dir option on the command line to specify a different home directory. Using this option affects only the command it is used in and does not change the PGP_HOME_DIR environment variable.

Using --home-dir on the command line overrides the current setting of the PGP_HOME_DIR environment variable.
System Requirements

In general, system requirements for PGP Command Line are the same as the system requirements for the host operating system.

In addition to the hard drive space required by the base operating system, PGP Command Line requires additional space for both the data on which cryptographic operations (such as encryption, decryption, signing, and verifying) will be applied and temporary files created in the process of performing those operations.

For a given file being encrypted or decrypted, PGP Command Line can require several times the size of the original file in free hard drive space (depending on how much the file was compressed), enough to hold the original file or files and the final file resulting from the encryption or decryption operation.

In cases where PGP Zip functionality is used on a file, PGP Command Line may also require several times the size of the original file or files in free hard drive space, enough to hold the original file, a temporary file created when handling the archive, and the final file resulting from the encryption or decryption operation. Make sure you have adequate free hard drive space on your system before using PGP Command Line.

**Windows Vista (32-bit)**

<table>
<thead>
<tr>
<th>Component</th>
<th>Requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Computer and processor</td>
<td>PC with 1 GHz 32-bit (x86) processor</td>
</tr>
<tr>
<td>Memory</td>
<td>1 gigabyte (GB) of RAM or higher recommended (64 MB minimum supported; may limit performance and some features)</td>
</tr>
<tr>
<td>Hard disk</td>
<td>15 GB of available space</td>
</tr>
<tr>
<td>Drive</td>
<td>DVD-ROM drive</td>
</tr>
<tr>
<td>Display</td>
<td>Support for DirectX 9 graphics with WDDM driver, 128 MB of graphics memory (minimum), Pixel Shader 2.0 in hardware, 32 bits per pixel</td>
</tr>
</tbody>
</table>

**Windows Server 2003 Standard Edition**

<table>
<thead>
<tr>
<th>Component</th>
<th>Requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Computer and processor</td>
<td>PC with a 133-MHz processor required; 550-MHz or faster processor recommended (Windows Server 2003 Standard Edition supports up to four processors on one server)</td>
</tr>
<tr>
<td>Memory</td>
<td>128 MB of RAM required; 256 MB or more recommended; 4 GB maximum</td>
</tr>
<tr>
<td>Hard disk</td>
<td>1.25 to 2 GB of available hard-disk space</td>
</tr>
<tr>
<td>Drive</td>
<td>CD-ROM or DVD-ROM drive</td>
</tr>
<tr>
<td>Display</td>
<td>VGA or hardware that supports console redirection required; Super VGA supporting 800 x 600 or higher-resolution monitor recommended</td>
</tr>
</tbody>
</table>
## Datacenter Edition

<table>
<thead>
<tr>
<th>Component</th>
<th>Requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Computer and processor</td>
<td>Minimum: 400 MHz processor for x86-based computers; recommended: 733 MHz processor</td>
</tr>
<tr>
<td>Memory</td>
<td>Minimum: 512 MB of RAM; recommended: 1 GB of RAM</td>
</tr>
<tr>
<td>Hard disk</td>
<td>1.5 GB hard-disk space for x86-based computers</td>
</tr>
<tr>
<td>Other</td>
<td>Minimum: 8-way capable multiprocessor machine required; maximum: 64-way capable multiprocessor machine supported</td>
</tr>
</tbody>
</table>

## Enterprise Edition

These system requirements apply only to the 32-bit version of Windows Server 2003 Enterprise Edition; no support is provided for 64-bit versions of Windows Server 2003 Enterprise Edition.

<table>
<thead>
<tr>
<th>Component</th>
<th>Requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Computer and processor</td>
<td>133-MHz or faster processor for x86-based PCs; up to eight processors supported on either the 32-bit</td>
</tr>
<tr>
<td>Memory</td>
<td>128 MB of RAM minimum required; maximum: 32 GB for x86-based PCs with the 32-bit version</td>
</tr>
<tr>
<td>Hard disk</td>
<td>1.5 GB of available hard-disk space for x86-based PCs; additional space is required if installing over a network</td>
</tr>
<tr>
<td>Drive</td>
<td>CD-ROM or DVD-ROM drive</td>
</tr>
<tr>
<td>Display</td>
<td>VGA or hardware that supports console redirection required</td>
</tr>
</tbody>
</table>

## Web Edition

<table>
<thead>
<tr>
<th>Component</th>
<th>Requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Computer and processor</td>
<td>133-MHz processor (550 MHz recommended)</td>
</tr>
<tr>
<td>Memory</td>
<td>128 MB of RAM (256 MB recommended; 2 GB maximum)</td>
</tr>
<tr>
<td>Hard disk</td>
<td>1.5 GB of available hard-disk space</td>
</tr>
</tbody>
</table>

## Windows XP (32-bit)

<table>
<thead>
<tr>
<th>Component</th>
<th>Requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Computer and processor</td>
<td>PC with 300 megahertz (MHz) or higher processor clock speed recommended; 233-MHz minimum required; Intel Pentium/Celeron family, AMD K6/Athlon/Duron family, or compatible processor recommended</td>
</tr>
<tr>
<td>Memory</td>
<td>128 megabytes (MB) of RAM or higher recommended (64 MB minimum supported; may limit performance and some features)</td>
</tr>
<tr>
<td>Hard disk</td>
<td>1.5 gigabyte (GB) of available hard disk space</td>
</tr>
</tbody>
</table>
### Windows XP (64-bit)

<table>
<thead>
<tr>
<th>Component</th>
<th>Requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Computer and processor</td>
<td>PC with AMD Athlon 64, AMD Opteron, Intel Xeon with Intel EM64T support, Intel Pentium 4 with Intel EM64T support</td>
</tr>
<tr>
<td>Memory</td>
<td>256 megabytes (MB) of RAM or higher recommended</td>
</tr>
<tr>
<td>Hard disk</td>
<td>1.5 gigabyte (GB) of available hard disk space</td>
</tr>
<tr>
<td>Drive</td>
<td>CD-ROM or DVD-ROM drive</td>
</tr>
<tr>
<td>Display</td>
<td>Super VGA (800 × 600) or higher resolution video adapter and monitor supporting 800 × 600 or higher-resolution monitor recommended</td>
</tr>
</tbody>
</table>

### Windows 2000

<table>
<thead>
<tr>
<th>Component</th>
<th>Requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Computer and processor</td>
<td>133 MHz or higher Pentium-compatible CPU</td>
</tr>
<tr>
<td>Memory</td>
<td>At least 64 megabytes (MB) of RAM; more memory generally improves responsiveness</td>
</tr>
<tr>
<td>Hard disk</td>
<td>2 GB with 650 MB free space</td>
</tr>
<tr>
<td>Drive</td>
<td>CD-ROM or DVD-ROM drive</td>
</tr>
<tr>
<td>Display</td>
<td>VGA or higher resolution monitor</td>
</tr>
</tbody>
</table>

### IBM AIX 5.2 and 5.3

PGP Command Line runs on the range of IBM eServer p5, IBM eServer pSeries, IBM eServer i5 and IBM RS/6000, as supported by IBM AIX 5.2 and 5.3.

### HP-UX 11i

PGP Command Line runs on the list of PA-RISC workstation and servers supported by HP-UX 11i, as specified at [http://docs.hp.com/en/5187-2239/ch03s01.html](http://docs.hp.com/en/5187-2239/ch03s01.html).

### Solaris 9

<table>
<thead>
<tr>
<th>Component</th>
<th>Requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Computer and processor</td>
<td>SPARC (32- and 64-bit) platforms</td>
</tr>
<tr>
<td>Memory</td>
<td>64 MB minimum (128 MB recommended)</td>
</tr>
<tr>
<td>Hard disk</td>
<td>600 MB for desktops; one GB for servers</td>
</tr>
</tbody>
</table>
Red Hat Enterprise Linux and Fedora Core

<table>
<thead>
<tr>
<th>Component</th>
<th>Requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Computer and processor</td>
<td>x86 for Red Hat Enterprise Linux, x86_64 for Fedora Core; see Red Hat or</td>
</tr>
<tr>
<td></td>
<td>Fedora websites for hardware compatibility</td>
</tr>
<tr>
<td>Memory</td>
<td>256 MB minimum</td>
</tr>
<tr>
<td>Hard disk</td>
<td>800 MB minimum</td>
</tr>
</tbody>
</table>

Mac OS X

<table>
<thead>
<tr>
<th>Component</th>
<th>Requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Computer and processor</td>
<td>Macintosh computer with PowerPC G3, G4, or G5 processor</td>
</tr>
<tr>
<td>Memory</td>
<td>128 MB of physical RAM</td>
</tr>
</tbody>
</table>

Installing on AIX

This section tells you how to install, change the home directory, and uninstall on AIX.

To Install on AIX

You need to have root or administrator privileges on the machine on which you are installing PGP Command Line.

To install PGP Command Line onto an AIX machine:

1. If you have an existing version of PGP Command Line installed on the computer, uninstall it.

2. Download the installer application called PGPCommandLine960AIX.tar to a known location on your system.

3. Untar the package first. You will get the following file:

   PGPCommandLine960AIX.rpm

4. Type: `rpm -ivh PGPCommandLine960AIX.rpm`

5. Press Enter.

By default, the PGP Command Line application, pgp, is installed into the directory `/opt/pgp/bin`. You need to add this directory to your PATH environment variable in order for the application to be found.

For sh-based shells, use this syntax:

   PATH=$PATH:/opt/pgp/bin

For csh-based shells, use this syntax:
set path = ($path /opt/pgp/bin)

Also, in order to access the PGP Command Line man page, you need to set the MANPATH environment variable appropriately.

For sh-based shells, use this syntax:

    MANPATH=$MANPATH:/opt/pgp/man; export MANPATH

For csh-based shells, use this syntax:

    setenv MANPATH "/opt/pgp/man"

By adding the option --prefix to the rpm command, you can install PGP Command Line in a location other than the default:

1. If you have an existing version of PGP Command Line installed on the computer, uninstall it.
2. Download the installer application called PGPCommandLine960AIX.tar to a known location on your system.
3. Untar the package first. You will get the following file: PGPCommandLine960AIX.rpm
4. Type: `rpm --prefix=/usr/pgp -ivh PGPCommandLine960AIX.rpm`
5. Press Enter.

This command installs the application binary in the directory /usr/pgp/bin/pgp, libraries in /usr/pgp/lib, and so on.

You will need to edit the environmental variable LIBPATH to include the new library path (/usr/pgp/lib) so that PGP Command Line can function in a location other than the default.

### Changing the Home Directory on AIX

The home directory is where PGP Command Line stores the files that it creates and uses; for example, keyring files.

By default, the PGP Command Line installer for AIX creates the PGP Command Line home directory at $HOME/.pgp. If this directory does not exist, it will be created. For example, if the value of $HOME for user “alice” is /usr/home/alice, PGP Command Line will attempt to create /usr/home/alice/.pgp.

The PGP Command Line installer will not try to create any other part of the directory listed in the $HOME variable, only .pgp.

If you want the home directory changed on a permanent basis, you will need to create the $PGP_HOME_DIR environment variable and specify the path of the desired home directory.

### Uninstalling on AIX

Uninstalling PGP Command Line on AIX requires root privileges, either through su or sudo.
To uninstall PGP Command Line on AIX:

1. Type the following command and press Enter:
   
   ```bash
   rpm -e pgpcmdln
   ```

2. PGP Command Line is uninstalled.

---

**Installing on HP-UX**

This section tells you how to install, change the home directory, and uninstall on HP-UX.

**To Install on HP-UX**

You need to have root or administrator privileges on the machine on which you are installing PGP Command Line.

To install PGP Command Line onto an HP-UX machine:

1. If you have an existing version of PGP Command Line installed on the computer, uninstall it.

2. Download the installer file called `PGPCommandLine960HPUX.tar` to a known location on your system.

3. Untar the package first. You will get the following file:
   
   `PGPCommandLine960HPUX.depot`

4. Type: `swinstall -s /absolute/path/to/PGPCommandLine960HPUX.depot`

5. Press Enter.

The PGP Command Line application, `pgp`, is installed into the directory `/opt/pgp/bin`. You need to add this directory to your PATH environment variable in order for the application to be found.

For sh-based shells, use this syntax:

```bash
PATH=$PATH:/opt/pgp/bin
```

For csh-based shells, use this syntax:

```csh
set path = ($path /opt/pgp/bin)
```

Also, in order to access the PGP Command Line man page, you need to set the `MANPATH` environment variable appropriately.

For sh-based shells, use this syntax:

```bash
MANPATH=$MANPATH:/opt/pgp/man; export MANPATH
```

For csh-based shells, use this syntax:

```csh
setenv MANPATH "/opt/pgp/man"
```
Changing the Home Directory on HP-UX

The home directory is where PGP Command Line stores the files that it creates and uses; for example, keyring files.

By default, the PGP Command Line installer for HP-UX creates the PGP Command Line home directory in $HOME/.pgp. If this directory does not exist, it will be created. For example, if the value of $HOME for user “alice” is /usr/home/alice, PGP Command Line will attempt to create /usr/home/alice/.pgp.

The PGP Command Line installer will not try to create any other part of the directory listed in the $HOME variable, only .pgp.

If you want the PGP Command Line home directory changed on a permanent basis, you can define the $PGP_HOME_DIR environment variable and specify the path of the desired home directory.

Uninstalling on HP-UX

Uninstalling PGP Command Line on HP-UX requires root privileges, either su or sudo.

To uninstall PGP Command Line on HP-UX:

1. Type the following command and press Enter:

   swremove pgpcmdln

2. PGP Command Line is uninstalled.

Installing on Mac OS X

To Install on Mac OS X

To install PGP Command Line onto a Mac OS X computer:

1. Close all applications.

2. Download the installer application, PGPCommandLine960MacOSX.tgz, to your desktop.

3. Double-click on the file PGPCommandLine960MacOSX.tgz.

4. If you have Stuffit Expander, it will automatically first uncompress this file into PGPCommandLine960MacOSX.tar, and then untar it into PGPCommandLine960MacOSX.pkg.

Caution

You may encounter an issue generating 2048- or 4096-bit keys on HP-UX systems running PGP Command Line if you have altered the maximum number of shared memory segments that can be attached to one process, as configured by the shmseg system parameter. If you encounter this issue, reset the shmseg system parameter to its default value of 120. Consult your HP-UX documentation for information on how to alter system parameters.
5 Double-click on the file PGPCommandLine960MacOSX.pkg.

6 Follow the on-screen instructions.

The Mac OS X PGP Command Line application, pgp, is installed into /usr/bin/.

After you run PGP Command Line for the first time, its home directory will be created automatically in the directory $HOME/Documents/PGP. This directory may already exist if PGP Desktop for Mac OS X is already installed on the system.

**Changing the Home Directory on Mac OS X**

The home directory is where PGP Command Line stores the files that it creates and uses; for example, keyring files.

By default, the PGP Command Line installer for Mac OS X creates the PGP Command Line home directory at $HOME/Documents/PGP. If this directory does not exist, it will be created.

The PGP Command Line installer will not try to create any other part of the directory listed in the $HOME variable, only PGP.

If you want the home directory changed permanently, you need to create the $PGP_HOME_DIR environment variable and specify the path of the desired home directory.

**Uninstalling on Mac OS X**

Uninstalling PGP Command Line on Mac OS X requires administrative privileges.

To uninstall PGP Command Line on Mac OS X:

1 Using the Terminal application, enter the following commands:
   
   ```
   rm -rf /usr/bin/pgp
   rm -rf /Library/Frameworks/PGP*
   rm -rf /Library/Receipts/PGP*
   ```

2 PGP Command Line is uninstalled.

Preferences and keyrings are not removed when PGP Command Line is uninstalled.
Installing on Red Hat Enterprise Linux or Fedora Core

To Install on Red Hat Enterprise Linux or Fedora Core

You need to have root or administrator privileges on the machine on which you are installing PGP Command Line.

If you want to use the XML key list functionality in PGP Command Line, you need to upgrade libxml2 to Version 2.6.8; the default is Version 2.5.10. If you attempt to use the XML key list functionality without upgrading, you will receive an error.

To install PGP Command Line onto a Linux machine:

1. If you have an existing version of PGP Command Line installed on the computer, uninstall it.

2. Download the installer file called `PGPCommandLine960Linux.tar` to a known location on your system.

3. Untar the package first. You will get the following file: `PGPCommandLine960Linux.rpm`

4. Type: `rpm -ivh PGPCommandLine960Linux.rpm`

5. Press Enter.

The PGP Command Line application, pgp, is installed by default into `/usr/bin/`.

By adding the option `--prefix` to the `rpm` command, you can install PGP Command Line in a location other than the default. Perform the following steps:

1. If you have an existing version of PGP Command Line installed on the computer, uninstall it.

2. Download the installer file called `PGPCommandLine960Linux.tar` to a known location on your system.

3. Untar the package first. You will get the following file: `PGPCommandLine960Linux.rpm`

4. Type: `rpm --prefix=/opt -ivh PGPCommandLine960Linux.rpm`

5. Press Enter.

This command will install the application binary in the directory `/opt/bin/pgp`, libraries in `/opt/lib`, etc. You will need to edit the environment variable `LD_LIBRARY_PATH` to include the new library path for the software to function in any location other than the default.
Changing the Home Directory on Linux

The home directory is where PGP Command Line stores the files that it creates and uses; for example, keyring files.

By default, the PGP Command Line installer for Linux creates the PGP Command Line home directory at $HOME/.pgp. If this directory does not exist, it will be created. For example, if the value of $HOME for user “alice” is /usr/home/alice, PGP Command Line will attempt to create /usr/home/alice/.pgp.

The PGP Command Line installer will not try to create any other part of the directory listed in the $HOME variable, only .pgp.

If you want the home directory changed on a permanent basis, you need to create the $PGP_HOME_DIR environment variable and specify the path of the desired home directory.

Uninstalling on Linux

Uninstalling PGP Command Line on Linux requires root privileges, either su or sudo.

To uninstall PGP Command Line on Linux:

1. Type the following command and press Enter:

   ```
   rpm -e pgpcmdln
   ```

2. PGP Command Line is uninstalled.
Installing on Solaris

This section tells you how to install, change the home directory, and uninstall on Solaris.

To Install on Solaris

You need to have root or administrator privileges on the machine on which you are installing PGP Command Line.

To install PGP Command Line onto a Solaris machine in the default directory:

1. If you have an existing version of PGP Command Line installed on the computer, uninstall it.
2. Download the installer file called PGPCommandLine960Solaris.tar to a known location on your system.
3. Untar the package first. You will get the following file: PGPCommandLine960Solaris.pkg
4. Type `pkgadd -d PGPCommandLine960Solaris.pkg` and press Enter.
5. At the first prompt, enter “1” or “all” to install the package.

If the directories `/usr/bin` and `/usr/lib` are not owned by root:bin, the install application `pkgadd` will ask if you want to change the ownership/group on these directories. It is not necessary to change them, but as an admin you may do so if you wish.

By default, the PGP Command Line application, `pgp`, is installed into the directory `/opt/pgp/bin`. You need to add this directory to your PATH environment variable in order for the application to be found.

For sh-based shells, use this syntax:

```
PATH=$PATH:/opt/pgp/bin
```

For csh-based shells, use this syntax:

```
set path = ($path /opt/pgp/bin)
```

Also, in order to access the PGP Command Line man page, you need to set the MANPATH environment variable appropriately.

For sh-based shells, use this syntax:

```
MANPATH=$MANPATH:/opt/pgp/man; export MANPATH
```

For csh-based shells, use this syntax:

```
setenv MANPATH "/opt/pgp/man"
```
To install PGP Command Line on Solaris into a directory other than the default location:

1. If you have an existing version of PGP Command Line installed, uninstall it.

2. Download the installer application PGPCommandLine960Solaris.tar to a known location on your system.

3. Untar the package first. You will get the following file: PGPCommandLine960Solaris.pkg

4. Type: `pkgadd -a none -d PGPCommandLine960Solaris.pkg`
   (This will force an interactive installation).

5. Press Enter.

6. At the first prompt, enter “1” or “all” to install the package.

7. You will be asked to enter the path to the package’s base directory.
   
   If you enter /usr/pgp, the binary will be installed to /usr/pgp/bin/pgp, libraries will be installed to /usr/pgp/lib, and so on.
   
   You need to edit the environment variable LD_LIBRARY_PATH to include the new library path (/usr/pgp/lib) so that PGP Command Line can function in this location.

### Changing the Home Directory on Solaris

The home directory is where PGP Command Line stores the files that it creates and uses; for example, keyring files.

By default, the PGP Command Line installer for Solaris creates the PGP Command Line home directory in $HOME/.pgp. If this directory does not exist, it will be created. For example, if the value of $HOME for user “alice” is /usr/home/alice, PGP Command Line will attempt to create /usr/home/alice/.pgp.

The PGP Command Line installer will not try to create any other part of the directory listed in the $HOME variable, only .pgp.

If you want the PGP Command Line home directory changed on a permanent basis, you can define the $PGP_HOME_DIR environment variable and specify the path of the desired home directory.

### Uninstalling on Solaris

Uninstalling PGP Command Line on Solaris requires root privileges, either su or sudo.

To uninstall PGP Command Line on Solaris:

1. Type the following command and press Enter:
   
   `pkg rm PGPcmdln`
   
   To uninstall with no confirmation, use: `pkg rm -n PGPcmdln`

2. PGP Command Line is uninstalled.
Installing on Windows

This section tells you how to install, change the home directory, and uninstall on Windows.

Because PGP Command Line and PGP Desktop use different versions of the PGP SDK, if you want to install both applications on one system, install the 32-bit version of PGP Command Line (you can do this on the 64-bit version of Windows XP). The 64-bit version of PGP Command Line cannot be installed on a system that already has PGP Desktop installed on it; the installer will halt when it sees that PGP Desktop is already installed.

To Install on Windows

To install PGP Command Line onto a supported Windows system:

1. Close all Windows applications.
2. Download the installer application, PGPCommandLine960Win32.zip, to a known location on your system.
3. Unzip the file PGPCommandLine960Win32.zip. You will get the following file: PGPCommandLine96Win32.msi.
4. Double click on PGPCommandLine960Win32.msi.
5. Follow the on-screen instructions.
6. If prompted, restart your machine. A restart is needed only if other PGP products are also installed on the same machine.

The Windows PGP Command Line application, pgp.exe, is installed into:

C:\Program Files\PGP Corporation\PGP Command Line\

After you run PGP Command Line for the first time, its home directory will be created automatically in the user’s home directory:

C:\Documents and Settings\<user>\My Documents\PGP\

Application data is stored in the directory:

C:\Documents and Settings\<user>\Application Data\PGP Corporation\PGP

Locations may be different for the different Windows versions.

Changing the Home Directory on Windows

The home directory is where PGP Command Line stores its keyring files. If a different PGP product has already created this directory, PGP Command Line will also use it (thus, PGP Command Line can automatically use existing PGP keys).

PGP Command Line data files, such as keys, are stored in the home directory:

C:\Documents and Settings\<user>\My Documents\PGP\
PGP Command Line application files, such as the configuration file PGPprefs.xml, are stored in:

C:\Documents and Settings\<user>\Application Data\PGP Corporation\PGP\

If you want the home directory changed on a permanent basis, you need to create the PGP_HOME_DIR environment variable and specify the path of the desired home directory.

To create the PGP_HOME_DIR environment variable on a supported Windows system:

1. Click **Start**, select **Settings**, select **Control Panel**, and then select **System**.
   
   The System Properties dialog appears.

2. Select the **Advanced** tab, then click **Environment Variables**.
   
   The Environment Variables screen appears.

3. In the User Variables section, click **New**.
   
   The New User Variable dialog appears.

4. In the **Variable name** field, enter **PGP_HOME_DIR**. In the **Variable value** field, enter the path of the home directory you want to use. For example:

   C:\PGP\PGPhomedir\

5. Click **OK**.

   The Environment Variables screen reappears. PGP_HOME_DIR appears in the list of user variables.

**Uninstalling on Windows**

To uninstall PGP Command Line on a supported Windows system:

1. Navigate to the **Add or Remove Programs** Control Panel.

2. Select PGP Command Line from the list of installed programs.

3. Click **Remove**, then follow the on-screen instructions.

   PGP Command Line is uninstalled.
Licensing
Instructions for Licensing PGP Command Line

PGP Command Line requires a valid license to operate. This chapter describes how to license your copy of PGP Command Line.

Overview

PGP Command Line requires a valid license to support full functionality. If you use PGP Command Line without entering a license or after your license has expired, only basic functionality will be available; you will only be able to get help and version information; perform a speed test; list keys, user IDs, fingerprints, and signatures; export public keys and keypairs; and license PGP Command Line.

As PGP Command Line will not operate normally until licensed, you should license it immediately after installation.

When your license gets within 60 days of expiration, PGP Command Line begins issuing warnings that license expiration is nearing. There is no grace period once the license expiration date has been reached.

PGP Command Line supports the following licensing scenarios:

- **Using a license number**: This is the normal method to license PGP Command Line. You must have your license number and a working connection to the Internet.

- **Using a license authorization file**: This licensing method uses licensing information in a file that was obtained from PGP Corporation. This method does not require a working connection to the Internet.

- **Re-licensing**: If you have already licensed PGP Command Line on a system but want to re-license it with a new license number (to support additional functionality, for example), use this method. You must have your new license number and a working connection to the Internet.

- **Through a proxy server**: If you connect to the Internet through a proxy server, use this method to license PGP Command Line. You must have your license number and the appropriate proxy server information.

All of these scenarios are described in detail below.
License Recovery

When you first enter your PGP Command Line license, one option is `--license-email`, which takes a valid email address.

You are not required to use `--license-email` to license your copy of PGP Command Line, but it is required if you want to take advantage of the license recovery feature.

The license recovery feature provides an automated mechanism for retrieving your original licensing information for those occasions when you need to enter it again.

Here is how the license recovery feature works: When you first license your copy of PGP Command Line, you enter a License Name, License Organization, your License Number, and a License Email. The license authorizes, and you begin using PGP Command Line.

Several months pass. The hardware hosting PGP Command Line fails and it is no longer usable. You need to reinstall PGP Command Line on a new system. You still have your PGP Command Line license number, but you enter your company name differently in License Organization; you didn’t remember exactly how you entered it several months ago, and this time you picked a slightly different form (or maybe you even mis-typed it by mistake).

Not a big deal, you think; what difference could it make? But when you attempt to authorize the license, it doesn’t work.

What happened is that when you re-license PGP Command Line, you must enter the same information exactly as you did the first time or it will not license correctly.

At this point the license recovery feature kicks in. When you attempt to re-license PGP Command Line, and you enter a valid license, but the License Name or License Organization you enter is different, the license recovery feature sends an email message to the License Email you entered the first time you licensed PGP Command Line.

The email message includes the License Name and License Organization you used when you first licensed PGP Command Line. You can now license PGP Command Line on the new system using the information in the message.

The key to the license recovery feature is entering a valid email address when you first license PGP Command Line. The license recovery feature will only use the email address you enter when you first license a specific PGP Command Line license. You can’t add or change the email address at a later time; if you don’t enter it the first time you license, the license recovery feature won’t work for that particular PGP Command Line license.

If the license recovery feature isn’t available for a PGP Command Line license, but you need your original License Name or License Organization, you need to contact PGP Support. Refer to pgpsupport.com for more information.
Using a License Number

If you have a license number and a working Internet connection you can license your copy of PGP Command Line.

Use `--license-authorize` to license PGP Command Line.

The following options are required:

- `--license-name <Name>`
  Where `<Name>` is your name or a descriptive name.

- `--license-organization <Org>`
  Where `<Org>` is the name of your company.

- `--license-number <Number>`
  Where `<Number>` is a valid license number.

The following option is not required but is recommended:

- `--license-email <EmailAddress>`
  Where `<EmailAddress>` is a valid email address, generally the email address of the PGP Command Line administrator.

Before deciding not to enter a license email, be sure to refer to "License Recovery" on page 26. Not entering a license email when you first license your copy of PGP Command Line negates the license recovery feature for your PGP Command Line license. If you decide not to enter a license email, you will see a warning message but your license will authorize.

For example:

```plaintext
pgp --license-authorize --license-name "Alice Cameron"
     --license-organization "Example Corporation"
     --license-number "aaaaa-bbbbb-cccc-cdddd-eeeee-fff"
     --license-email "acameron@example.com"
```

(When entering this text, it all goes on a single line.)
Using a License Authorization

If you have both a license number and a license authorization (a text file) from PGP Corporation instead of just a license number, you need to list the name of the license authorization file in the command.

You may need a license authorization if you are having problems authorizing your license number or if the system hosting PGP Command Line is not connected to the Internet.

Use `--license-authorize` to license PGP Command Line using a license authorization.

The following options are required:

- `--license-name <Name>`
  Where `<Name>` is your name or a descriptive name.

- `--license-organization <Org>`
  Where `<Org>` is the name of your company.

- `--license-number <Number>`
  Where `<Number>` is a valid license number.

The following option is not required but is recommended:

- `--license-email <EmailAddress>`
  Where `<EmailAddress>` is a valid email address, generally the email address of the PGP Command Line administrator.

Before deciding not to enter a license email, be sure to refer to “License Recovery” on page 26. Not entering a license email when you first license your copy of PGP Command Line negates the license recovery feature for your PGP Command Line license. If you decide not to enter a license email, you will see a warning message but your license will authorize.

For example:

```
pgp --license-authorize --license-name "Alice Cameron"
   --license-organization "Example Corporation"
   --license-number "aaaaa-bbbbb-cccccc-ddddd-eeeee-fff"
   license-auth.txt --license-email "acameron@example.com"
```

(When entering this text, it all goes on a single line.)

In this example, the text file “license-auth.txt” is shown after the license number.
Re-Licensing

If you have already licensed your copy of PGP Command Line on a system, but you need to re-license it on the same system (if you have purchased a new license with additional capabilities, for example), you must use the `<force>` option to override the existing license.

You can use a license number or a license authorization when you are re-licensing.

Use `--license-authorize` to re-license PGP Command Line.

The following options are required:

- `--license-name <Name>`
  Where `<Name>` is your name or a descriptive name.

- `--license-organization <Org>`
  Where `<Org>` is the name of your company.

- `--license-number <Number>`
  Where `<Number>` is a valid license number.

- `--force`

The following option is not required but is recommended:

- `--license-email <EmailAddress>`
  Where `<EmailAddress>` is a valid email address, generally the email address of the PGP Command Line administrator.

The following option is optional:

- `<LicenseAuthFilename>`
  Where `<LicenseAuthFilename>` is the name of the text file from PGP Corporation that includes license authorization information.

Before deciding not to enter a license email, be sure to refer to "License Recovery" on page 26. Not entering a license email when you first license your copy of PGP Command Line negates the license recovery feature for your PGP Command Line license. If you decide not to enter a license email, you will see a warning message but your license will authorize.

For example:

```
pgp --license-authorize --license-name "Alice Cameron"
--license-organization "Example Corporation"
--license-number "aaaaa-bbbbb-cccccc-dddd-ddeee-fff"
--license-email "acameron@example.com" --force
```

(When entering this text, it all goes on a single line.)
Through a Proxy Server

If the Internet access of the system hosting PGP Command Line is via an HTTP proxy connection, you can still license your copy of PGP Command Line directly; you simply need to add the necessary proxy information.

Use `--license-authorize` to license PGP Command Line via a proxy server.

The following options are required:

- `--license-name <Name>`
  Where `<Name>` is your name or a descriptive name.

- `--license-organization <Org>`
  Where `<Org>` is the name of your company.

- `--license-number <Number>`
  Where `<Number>` is a valid PGP Command Line license number.

- `--proxy-server <Server>`
  Where `<Server>` is the IP address or fully qualified domain name of the proxy server PGP Command Line must go through to reach the Internet.

The following options are not required; they are only needed when the proxy server requires authentication:

- `--proxy-username <Username>`
  Where `<Username>` is a valid username on the proxy server.

- `--proxy-passphrase <Passphrase>`
  Where `<Passphrase>` is the passphrase for the username you entered.

The following option is not required but is recommended:

- `--license-email <EmailAddress>`
  Where `<EmailAddress>` is a valid email address, generally the email address of the PGP Command Line administrator.

Before deciding not to enter a license email, refer to “License Recovery” on page 26. Not entering a license email when you first license your copy of PGP Command Line negates the license recovery feature for your PGP Command Line license. If you decide not to enter a license email, you will see a warning message but your license will authorize.

For example:

```
pgp --license-authorize --license-name "Alice Cameron"
   --license-organization "Example Corporation"
   --license-number "aaaaa-bbbbb-ccccc-ddddd-eeeee-fff"
   --proxy-server "proxyserver.example.com"
   --proxy-username "acameron"
   --proxy-passphrase "a_cameron1492sailedblue"
   --license-email "acameron@example.com"
```
(When entering this text, it all goes on a single line.)
The Command-Line Interface

How to Enter Commands

This chapter describes the command-line interface of the PGP Command Line product: what it is, how to use it, how to get help, flags and arguments, the configuration file, and environment variables.

Overview

PGP Command Line uses a command-line interface. You enter a valid command and press Enter. PGP Command Line responds appropriately based on what you entered (if you entered a valid command) or with an error message (if you entered an invalid or incorrectly structured command).

All PGP Command Line commands have a long form: the text “pgp,” a space, two hyphens “--”, and then the command name. Some of the more common commands have a short form: one hyphen and then a single letter that substitutes for the command name.

The --version command, for example, tells you what version of PGP Command Line you are using. It does not have a short form:

%pgp --version [Enter]

From here on, the command prompt (% in this example) and [Enter] will not be shown.

The response is:

PGP Command Line 9.6
Copyright (C) 2007 PGP Corporation
All rights reserved.

The --help command tells you about the commands available in PGP Command Line. The long form is:

pgp --help

The short form is:

pgp -h

The response to either version of the --help command is:

PGP Command Line 9.6
Copyright (C) 2007 PGP Corporation
All rights reserved.
Commands:
Generic:
-h --help this help message
and so on.
Some more examples of the command line:

1. `pgp --encrypt report.doc --recipient Alice`
   
   `report.doc:encrypt (0:output file report.doc.pgp)`
   
   Encrypts a file (the output filename will be `report.doc.pgp`) to the recipient “Alice.”

2. `pgp -e report.doc -r Alice`

   `report.doc:encrypt (0:output file report.doc.pgp)`

   Does the same as above, but using the short forms of the encrypt and the recipient flags.

3. `pgp -er Alice report.doc`

   `report.doc:encrypt (0:output file report.doc.pgp)`

   Combines multiple command short forms. “Alice” must come after the “r” because it is a required argument to `--recipient`.

4. `pgp -er Alice report.doc --output NewReport.pgp`

   `report.doc:encrypt (0:output file NewReport.pgp)`

   Changes the name of the file that is produced.
Flags and Arguments

PGP Command Line uses flags, commands, options, and arguments:

- **Flags** come in two different types, commands and options. Commands are flags that control what PGP Command Line does in its current invocation; they have no effect on subsequent invocations of PGP Command Line. Options change the behavior of the current command. Some options require an argument, described below, while others do not. The order in which flags are listed on the command line has no effect on their behavior.

- **Arguments** are required as the next parameter when an option flag is used. Arguments must immediately follow their flags. Where the flag/argument pair are on the command line does not change what the flag/argument pair does. Except when setting lists, in which case the command is read left to right; so when searching keyservers, for example, the listed keyservers are searched in the order in which they are provided on the command line.

Flags and arguments must be separated by a space on the command line. Extra spaces are ignored. If a space between parts of an argument is required, the entire argument must be between quotes.

In some cases, there can be multiple names for a single flag.

For example:

```
--textmode and --text (same flag with two names)
```

It is also possible to provide an option that has no effect on the current operation. Flags that have no bearing on the current operation are ignored, unless they cause an error, in which case the command returns an error.

For example:

```
--list-keys Alice with the option --encrypt-to-self
(the option --encrypt-to-self will be ignored)
```

Flags

As noted above, flags have both long and short forms. To combine multiple long forms, you simply write them out separated by a space. For example, to encrypt a file and armor the output:

```
pgp --encrypt ... --armor
```

You can, however, combine multiple short forms into a single flag. For example, to encrypt and sign at the same time:

```
pgp -es ...
```

When combining short forms, if at any time an option is used in the list that requires an argument, the list must be terminated and followed by the argument. For example: `-ear recipient`
Arguments

An argument is required as the next parameter when some option flags are used. There are several kinds of arguments, differentiated by how they are structured or what kind of information is provided.

The kinds of arguments are:

- Booleans
- Integers
- Enumerations
- Strings
- Lists
- File descriptor
- No parent

Each of these kinds of arguments is described below.

**Booleans**

Booleans are a special kind of argument. They never take a direct argument themselves. Instead, the behavior changes by how the flag is specified. To disable a Boolean, specify it with the prefix `--no-` instead of the normal `--`.

When the short form is used for a Boolean flag, there is no way to specify the disabled version of the flag.

For example:

- `--reverse-sort` (activates reverse sorting)
- `--no-compress` (deactivates compression, the reverse of `--compress`)
- `-t` (activates text mode; to deactivate text mode, the long form must be used, `--no-text`)

**Integers**

Integers are arguments that take a numeric value.

For example:

- `--wipe-passes 8` (sets the number of wipe passes to eight)
Enumerations

Enumerations are arguments that take a string, which is then converted to the correct value by PGP Command Line. This string will be one of several possible for each flag.

For example:

- **--sort-order userid** (sort by user ID)
- **--overwrite remove** (sets the file overwrite behavior to remove files if they exist)

Strings

String arguments take a string. If the string you want to use contains any spaces, the entire string must be in quotes (this indicates that all of the pieces belong to the same argument). In some cases, an empty string ("") can be passed as an argument.

On Windows systems, strings are read in as double-byte character strings and converted to UTF-8 for use by the PGP SDK or for output. On all other platforms, UTF-8 is used.

For example:

- **--default-key 0x8885BE88** (sets the key with this key ID as the default key)
- **--output "New File.txt.pgp"** (sets the output filename to a filename with a space in it)
- **--passphrase ""** (specifies a blank passphrase)
- **--expiration-date 2007-12-27** (specifies an expiration date of Dec. 27, 2007)

Lists

List arguments are the same as string arguments except you can supply more than one string.

For example:

- **--recipient bob --recipient bill** (sets both Bob and Bill as recipients)
- **-r bob -r bill** (same command using the short form of the flag)

File descriptors

File descriptor arguments behave like integer arguments, but instead of storing the value of the descriptor, PGP Command Line reads a string value from the descriptor. These string values always have a string type counterpart.

If you need to specify the data in UTF-8 format on a Windows system, use the “8” versions of the file descriptor options.

For example:

- **--passphrase-fd 4** (read passphrase from fd 4 and use it as if **--passphrase** had been supplied)
- **--passphrase-fd8 7** (read a UTF-8 passphrase from fd 7)
No parent
The final kind of arguments are those that have no parent flag. These arguments behave like lists and follow the same rules. They are used in different ways, depending on the operation being performed, but they can occur anywhere in the command line except after a flag that has a required argument.

These arguments can represent users or represent files.

For example

```
--list-keys Alice Bob Bill (list all keys that match any one of these users)
--encrypt file1.txt file2.txt file3.txt (encrypt multiple files with the same command)
```

Configuration File
Generally, the configuration file PGPprefs.xml cannot be changed by PGP Command Line itself: any changes need to be edited manually (on Mac OS X, the configuration file is `com.pgp.desktop.plist`, located in `/user’s home directory/Library/Preferences/`).

Starting with the PGP Command Line version 9.0, there is one operation that will change the configuration file: when you authorize a license, this information is saved in the file PGPprefs.xml for future use.

The configuration file PGPprefs.xml is located in the following locations:

- $HOME directory on any Unix platform
- The exact location depends on the version of Windows, but it is always the directory that holds the application data.

By changing some of the settings in the PGPprefs.xml file, you will change how PGP Command Line works as long as this file is not replaced.

Note that those configuration file settings that do not begin with “CL” are shared among all PGP applications on the system.

Like arguments, the configuration file settings come in different types: Boolean, Integer, Enumeration, List, and String.

Boolean configuration file settings you can use with PGP Command Line are:

- **ADK warning level** (adkWarning). Enables warning messages for ADK actions such as adding an ADK, skipping an ADK, or when an ADK is not found. Refer to “--warn-adk” on page 148 for more information.
- **Encrypt to self** (encryptToSelf). When on, all files or messages you encrypt to someone else are also encrypted to your key, which means you can decrypt those encrypted files/messages at a later time, if you wish. The default is off. See “--encrypt-to-self” on page 142 for more information.
- **Fast keygen** (fastKeyGen). Establishes the setting for fast key generation, on or off. The default is on. See “--fast-key-gen” on page 143 for more information.
- **Halt on error** (`CLhaltOnError`). When on, causes PGP Command Line to halt operations when an error occurs. Does not apply to all operations. The default is **off**. See “—halt-on-error” on page 144 for more information.

- **Keyring cache** (`CLkeyringCache`). When on, stores keyrings in memory for each access. The default is **off**. See “—keyring-cache” on page 144 for more information.

- **Large Keyrings** (`CLlargeKeyrings`). Checks keyring signatures only when necessary. See “—large-keyrings” on page 144 for more information.

- **Marginal is invalid** (`marginalIsInvalid`). Establishes whether marginally trusted keys are considered valid. The default is **true**, which means that marginally valid keys are not valid. See “—marginal-as-valid” on page 145 for more information.

- **Passphrase cache** (`CLpassphraseCache`). When on, automatically saves your passphrase in memory until you log off or purge the passphrase cache. The default is **off**. See “—passphrase-cache” on page 146 for more information.

Integer configuration file settings you can use with PGP Command Line are:

- **Keyring cache timeout** (`CLkeyringCacheTimeout`). Establishes the number of seconds a keyring stays cached in memory. The default is **120 seconds**. See “—keyring-cache-timeout” on page 153 for more information.

- **Keyservice timeout** (`CLkeyserviceTimeout`). Establishes the number of seconds to wait before a keyservice operation times out. The default is **120 seconds**. See “—keyservice-timeout” on page 153 for more information.

- **Number of wipe input passes** (`CLfileWipeInputPasses`). Establishes the number of wipe passes for input files. The default is **3 passes**. See “—wipe-input-passes” on page 158 for more information.

- **Number of wipe passes** (`fileWipePasses`). Establishes the number of passes used by the **—wipe** command. The default is **3 passes**. See “—wipe” on page 136 for more information.

- **Number of wipe temp passes** (`CLfileWipeTempPasses`). Establishes the number of wipe passes for temporary files. The default is **3 passes**. See “—wipe-temp-passes” on page 159 for more information.

- **Number of wipe overwrite passes** (`CLfileWipeOverwritePasses`). Establishes the number of wipe passes when overwriting an existing output file. The default is **3 passes**. See “—wipe-overwrite-passes” on page 159 for more information.

- **Passphrase cache timeout** (`CLpassphraseCacheTimeout`). Establishes the number of seconds a passphrase stays cached in memory. The default is **120 seconds**. See “—passphrase-cache-timeout” on page 154 for more information.

Enumeration configuration file settings you can use with PGP Command Line are:

- **Automatic import of keys** (`CLautoImportKeys`). Establishes behavior when keys are found during non-import operations. The default is **all**. See “—auto-import-keys” on page 160 for more information.
- **Compression Level** (`CLcompressionLevel`). Sets the compression level for the current operation. The default is **default**. See “`--compression-level`” on page 161 for more information.

- **Enforce ADK** (`CLenforceADK`). Establishes the ADK enforcement policy. The default is **attempt**. See “`--enforce-adk`” on page 162 for more information.

- **Input cleanup** (`CLinputCleanup`). Establishes what to do with input files after they have been used. The default is **off**. See “`--input-cleanup`” on page 165 for more information.

- **Manual import of keys** (`CLmanualImportKeys`). Establishes behavior when keys are found during an import. The default is **all**. See “`--manual-import-keys`” on page 167 for more information.

- **Manual import of key pairs** (`CLmanualImportKeyPairs`). Establishes behavior when key pairs are found during import. The default is **pair**. Refer to “`--manual-import-key-pairs`” on page 166 for more information.

- **Sort order** (`CLsortOrder`). Changes the sort order for writing key lists. The default is **any**. See “`--sort-order, --sort`” on page 167 for more information.

- **Overwrite** (`CLoverwrite`). Establishes what to do when an operation tries to create an output file but it already exists. The default is **off**. See “`--overwrite`” on page 167 for more information.

List configuration file settings you can use with PGP Command Line are:

- **Always encrypt to keys** (`alwaysEncryptToKeys`). Specifies additional recipients for encryption. Use the 32- or 64-bit key ID to specify the key(s) to use. Refer to “`--additional-recipient`” on page 180 for more information.

- **Default keyserver names and associated values** (`keyservers`). Specifies default keyservers. The default is ldap://keyserver.pgp.com:389/. If you supply a keyserver on the command line, those keyservers listed in the configuration file are ignored.

String configuration file settings you can use with PGP Command Line are:

- **Comment** (`commentString`). Specifies a comment string to be used in armored output blocks. The default is **not set**. Refer to “`--comment`” on page 170 for more information.

- **Default signing key** (`CLdefaultKey`). Specifies a key to be used by default for signing. The default is **not set**. See “`--default-key`” on page 171 for more information.

- **License Authorization** (`CLlicenseAuthorization`). Specifies the license authorization. The default is **not set**. See “`--license-name, --license-number, --license-organization, --license-email`” on page 173 for more information.

⚠️ Because licensing information is stored somewhat differently, PGP Corporation recommends that you do not directly edit the license-related configuration file settings; instead, use the license authorization commands described in Chapter 3, Licensing.
- **License Name (CLlicenseName).** Specifies the name of the licensee. The default is **not set.** See “--license-name, --license-number, --license-organization, --license-email” on page 173 for more information.

- **License Number (CLlicenseNumber).** Specifies the license number. The default is **not set.** See “--license-name, --license-number, --license-organization, --license-email” on page 173 for more information.

- **License Organization (CLlicenseOrganization).** Specifies the organization of the licensee. The default is **not set.** See “--license-name, --license-number, --license-organization, --license-email” on page 173 for more information.

- **Output File (CLoutputFile).** Specifies the output file (default is not set in the configuration file; defaults to stdout). The output file is used for output messages. See “--output-file” on page 174 for more information.

- **Private keyring file (privateKeyringFile).** The filename or path and filename to the private keyring file. The default is *secring.skr*, located in the default PGP Command Line home directory. See “--private-keyring” on page 175 for more information.

- **Public keyring file (publicKeyringFile).** The filename or path and filename to the public keyring file. The default is *pubring.pkr*, located in the default PGP Command Line home directory. See “--public-keyring” on page 176 for more information.

- **Random seed filename (rngSeedFile).** Sets the location of the random seed file. By default, the random seed file is located in the PGP Command Line data directory. See “--random-seed” on page 177 for more information.

- **Status File (CLstatusFile).** Specifies the status file. The default is not set in the configuration file; defaults to stderr. The status file is used for status messages, using a file name (with or without the path information). See “--status-file” on page 178 for more information.
Keyserver Configuration File Settings

Here is the keyserver section of the PGPprefs.xml file, with brief explanations of specific settings:

```
<key>keyservers</key>
<array>
  <dict>
    <key>title</key>
    <string>keyserver.example.com</string>
    (name of the keyserver)
    <key>domain</key>
    <string></string>
    <key>hostname</key>
    <string>keyserver.example.com</string>
    (hostname of the keyserver)
    <key>port</key>
    <integer>389</integer>
    (keyserver port)
    <key>protocol</key>
    <integer>1</integer>
    (keyserver protocol: 1 = LDAP, 2 = HTTP, 3 = LDAPS and 4 = HTTPS (currently not supported))
    <key>type</key>
    <integer>1</integer>
    (keyserver type: 1 = HTTP, 2 = HTTPS (currently not supported))
    <key>keyserverType</key>
    <integer>100</integer>
    (keyserver type: 100 = PGPLDAP, 101 = PGPLDAPS, 102 = PGPVKD, 103 = X509LDAP, 104 = X509LDAPS, 105 = PGPHTTP)
    <key>baseDN</key>
    <string></string>
    <key>authKeyId</key>
    <string></string>
    (not used)
    <key>authAlgorithm</key>
    <integer>0</integer>
    (not used)
    <key>flags</key>
    <integer>0</integer>
    (not used)
```
Environment Variables

PGP Command Line behavior can be changed using environment variables. For information about defining environment variables, refer to the section that describes the platform you are using in Chapter 2, Installation.

Environment variables have the lowest priority compared to the command line and the configuration file. Settings for either will override environment variables. However, if a value for an item is not specified in either, the environment variable will be used. Environment variables cannot be disabled; if they are present, they are implemented. To disable an environment variable, remove it. Setting a Boolean environment variable will activate it, regardless of the value to which it is set.

Environment variables that can be implemented for PGP Command Line are:

- **PGP_LOCAL_MODE**: This is a Boolean environment variable that forces PGP Command Line to run in local mode. The default is unset. See “--local-mode” on page 145 for more information.
  
  Usage: PGP_LOCAL_MODE=1

- **PGP_NO_BANNER**: This is a Boolean environment variable that turns off the banner when a command is run. The default is unset. See “--banner” on page 141 for more information.
  
  Usage: PGP_NO_BANNER=1

- **PGP_HOME_DIR**: This is a string environment variable that overrides the default home directory, pointing it to the path supplied in the variable. The default is unset. See “--home-dir” on page 172 for more information.
  
  Usage: PGP_HOME_DIR=/usr/bin/alice

- **PGP_PASSPHRASE**: This is a string environment variable that lets you set your passphrase. The default is unset. For more information, See “--passphrase” on page 174 for more information.
  
  Usage: PGP_PASSPHRASE="Now is the time for all good men"

- **PGP_NEW_PASSPHRASE**: This is a string environment variable that lets you set a new passphrase. The default is unset. See “--new-passphrase” on page 173 for more information.
  
  Usage: PGP_NEW_PASSPHRASE="to come to the aid of their country."

- **PGP_SYMMETRIC_PASSPHRASE**: This is a string environment variable that lets you set a passphrase for symmetric encryption. The default is unset. See “--symmetric-passphrase” on page 178 for more information.
  
  Usage: PGP_SYMMETRIC_PASSPHRASE="Now is the time"

- **PGP_EXPORT_PASSPHRASE**: This is a string environment variable that lets you set the export passphrase. The default is unset. See “--export-passphrase” on page 171 for more information.
  
  Usage: PGP_EXPORT_PASSPHRASE="For All Good Men"
Standard Input, Output, and Error

PGP Command Line writes different data to several different places by default. Any user output generated by PGP Command Line is written to standard output (stdout), including version information, key list data, and so on. Any status information generated by command line is sent to standard error (stderr).

When encrypting and decrypting, PGP Command Line reads and writes files by default. These files can be overridden with the special argument "-" to either --input or --output. This behavior is set so that PGP Command Line doesn’t have to wait for input if you forget something: it will generate an error that you can detect.

The behavior of PGP Command Line changes depending on the operating system you are using, while the syntax changes depending on the shell.

When you work with PGP Command Line, you can use standard input (stdin) in two ways: by redirecting an existing file, or by typing (pasting in) data.

Redirecting an Existing File

You can use your shell to redirect input to PGP Command Line from an existing file.

The command looks like:

```
pgp -er user -i - -o file.pgp <file.txt
```

Example:

```
pgp -er "bob@example.com" -i - -o newnote.pgp <newnote.txt
stdin:encrypt (0:output file newnote.pgp)
```

In this case, the file newnote.txt was encrypted with Bob’s key and saved as newnote.pgp.

Entering Data

Instead of redirecting an existing file, you can also type (or paste in) the data that needs to be encrypted. The command looks like:

```
pgp -er user -i - -o file.pgp
(type/paste in the data to be encrypted)
```

Example:

```
pgp -er "bob@example.com" -i - -o newnote.pgp
(stdin:encrypt (0:output file newnote.pgp)
```

(This text is the file newnote.txt, which will be signed by Bob.)

```
^Z
```

In addition to specifying the end of file, you also need to specify an output file name (such as “newnote.pgp”), since the input file name was not specified.

```
pgp --decrypt newnote.pgp --passphrase sm1t4
```
If you now decrypt newnote.pgp, the decrypted file newnote will not have an extension since the input was not in a file format.

On platforms where buffered standard input/output (I/O) is disabled by default, you cannot type or paste into stdin. Instead, you need to enable standard I/O using \texttt{--buffered-stdin} (see \texttt{--buffered-stdin} for details).

\textbf{End-of-File}

Depending on the shell you use, the end of file will be announced in different ways:

- On Windows, enter \texttt{^Z} (\texttt{ctrl-z}) on a separate line.
- On UNIX, enter \texttt{^D} (\texttt{ctrl-d}) anywhere in the text. The end of file character is shell-dependent and will vary on different systems.

\section*{Specifying a Key}

When you need to specify a key or keys as input for a PGP Command Line operation, there are two methods you can use:

- **Match by user ID:** To match by user ID, supply some of the text in the user ID(s) you want to match. A case insensitive search of the user IDs of the keys on the local keyring is made. All keys that match the supplied text will be returned; for example, searching on ‘ex’ would return all keys on the local keyring from the domain ‘example.com’, as well as a key whose user ID was ‘dexter@pgp.com’. \textit{This is a convenience feature that makes it easy for you to match multiple keys on the local keyring.}

  Searching by user ID can return no keys, one key, or multiple keys, depending on the supplied text and the user IDs of the keys on the local keyring. Matching by user ID is best for operations where you want your search to return multiple keys; for example, the list operations (\texttt{--list-keys}, \texttt{--fingerprint}, and so on). Match by user ID can be used for operations that work only on a single key, but as it may return multiple keys, match by user ID may not be the best choice for these operations.

- **Match by key ID:** To match by key ID, supply the key ID of the specific key you want used for the operation (0xABCD1234, for example). The key IDs of the keys on the local keyring will be searched. If the key with the specified key ID is found on the local keyring, it will be used for the operation; if not, the operation will terminate.

  Searching by key ID will return either no keys or one key. Matching by key ID is best for those cases where the search must exactly match one key (\texttt{--default-key}, for example) or where only a single key can be used for the operation; for example, most of the key edit operations (\texttt{--split-key}, \texttt{--revoke}, and so on).
‘Secure’ Options

The descriptions of some options in PGP Command Line mention that they are “secure,” as in “This option is not secure” or “--auth-passphrase is secure.”

In this context, “secure” means that the option’s argument is saved in non-pageable memory (when that option is available to applications). Options that are not “secure” are saved in normal system memory.
First Steps
An Overview of What To Do First

This chapter describes the first steps you need to take to get up and running with PGP Command Line.

Overview

The first steps for getting up and running with PGP Command Line are:

1. **Install PGP Command Line.**
   - Installation for all supported platforms is fully described in Chapter 2, Installation.

2. **License your copy of PGP Command Line.**
   - Licensing is required for normal operation of PGP Command Line. Refer to Chapter 3, Licensing and “--license-authorize” on page 134 for more information about licensing PGP Command Line.

3. **Create your key pair.**
   - Most of the things you do with PGP Command Line require a key pair (a private key and a public key). How to create your key pair is described later in this chapter in “Creating Your Keypair” on page 46.

4. **Protect your private key.**
   - No one but you should know the password or have access to your private key. How to protect your private key is described later in this chapter in “Protecting Your Private Key” on page 47.

5. **Distribute your public key.**
   - In order for others to verify your signature or encrypt data so that only you can decrypt it, they will need your public key.
   - One way to distribute your public key is to post it to a keyserver so that others can obtain it. The best way to do this is to post your public key to the PGP Global Directory (keyserver.pgp.com), a free, public keyserver hosted by PGP Corporation. It provides quick and easy access to the universe of PGP keys.
   - You can also export your public key to a file, which you can then distribute in any number of ways. For information about how to post your public key to a keyserver and extract your public key to a file, refer to “Distributing Your Public Key” on page 48.

6. **Obtain the public keys of others.**
You need someone’s public key to be able to encrypt data so that only they can decrypt it. You can get public keys from a keyserver (as long as the key is posted, of course). And if you receive someone’s public key in a file, you can import it. For more information about how to get a public key from a keyserver and how to import a key, refer to “Getting the Public Keys of Others” on page 50.

7 Verifying the public keys you get.

It is important to make sure the public keys you get actually belong to the person or organization they appear to be from. For instructions on how to verify a public key, refer to “Verifying Keys” on page 51.

8 Start securing your data.

Creating Your Keypair

The first thing you need to do after installing PGP Command Line is to make sure you have a usable PGP key pair, as most PGP Command Line operations require a key pair.

A key pair consists of two keys:

- Private key (stored in `secring.skr`) that only you have.
- Public key (stored in `pubring.pkr`) that you can distribute freely to the people you correspond with.

Keys are stored on keyrings. There’s one keyring for private keys (`secring.skr`), and one keyring for public keys (`pubring.pkr`).

If you are using a Windows or Mac OS X system, you may already have a key pair generated by PGP Desktop. If you do have an existing key pair you want to use with PGP Command Line and you distributed your public key to the people who will be encrypting data to you, you need to make sure the environment variable (`PGP_HOME_DIR`) is defined and points to the directory where your existing key pair is located.

If you have PGP Desktop installed on the same Windows or Mac OS X computer as PGP Command Line, and you installed PGP Desktop into the default directory, then PGP Command Line will automatically locate and use your existing keyrings.

If you do not have a PGP key pair, you will need to create one for use with PGP Command Line.

Use the `--gen-key` command to create a new key pair.

To create a key pair:

1 On the command line, enter:

```
pgp --gen-key <user> --key-type <type> --encryption-bits <bits> --passphrase <pass> [--signing-bits <bits>] [options]
```

where:
<user> is a user ID that people can use to locate your public key. A common user ID is your name and email address in the format: “Alice Cameron <alice@example.com>”. If your user ID contains spaces, you must enclose it in quotation marks.

<type> means you are creating either an RSA or a DH key. <bits> is the number of bits of the key (usually 1024 - 4096).

<passphrase> is a passphrase of your choice. If your passphrase includes spaces, enclose it in quotation marks.

For more information, refer to “--gen-key” on page 100.

2 Press Enter when the command is complete.

PGP Command Line responds by generating your key pair.

The --gen-key command automatically creates your key pair and a public and a private keyring in the home directory, then puts your new private and public keys onto their respective keyrings. You can create empty keyring files without generating a key pair at the same time using the --create-keyrings command.

Protecting Your Private Key

If someone gets your private key and manages to guess your passphrase or finds it written on a Post-it®, they can impersonate you. They can open messages encrypted to you and they can sign messages, making them appear to be from you.

It is very important to protect your private key! Don’t let anyone get a copy of it and don’t ever give anyone the passphrase.

By default, all generated keys (private and public) are stored in the directory to which the environment variable points (which is PGP_HOME_DIR, if set).

Otherwise:

- UNIX: $HOME/.pgp
- Windows: C:\Documents and Settings\<current user>\My Documents\PGP
- Mac OS X: $HOME/Documents/PGP

You can locate your keyrings using the --version -v command.

Once the keys are generated, you can store them in any location you choose (provided you don’t forget to adjust the environment variable to point to the new location). Moving your keys to a different location is one way to protect them from someone who might get access to your system.
It is also a good practice to make a backup copy of your keys. Make sure to be especially careful with your private key, storing it on a machine only you can access and in a directory that cannot be accessed via a network. You may also choose to implement additional security precautions.

### Distributing Your Public Key

People need your public key to encrypt information that only you can decrypt and to verify your signature.

There are three main methods available to distribute your public key:

- **Post your public key to the PGP Global Directory.** The PGP Global Directory is a free, publicly available keyserver hosted by PGP Corporation that provides quick and easy access to the universe of PGP keys. *If you aren’t in an email domain protected by a PGP Universal Server, the PGP Global Directory is your source for trusted keys.*

- **Post your public key to another keyserver.** Once posted, people can get a copy of your public key and use it to encrypt data that only your private key can decrypt.

  How to use PGP Command Line to post your public key to a keyserver is described below.

- **Export your public key to a text file.** Once exported to a text file, you can distribute your public key however you like: attached to an email message, pasted into the body of an email message, or copied to a CD.

  How to use PGP Command Line to extract your public key to a text file is described in “Exporting Your Public Key to a Text File” on page 49.

### Posting Your Public Key to a Keyserver

You can post your public key to a private keyserver or a public keyserver; the procedure is the same in both cases.

Use the **--keyserver-send** command to post your public key to a keyserver.

To post a public key to a keyserver:

1. On the command line, enter:

```
pgp --keyserver-send <input> --keyserver <ks>
```

where:

- `<input>` is the user ID, portion of the user ID, or key ID of the public key you are posting.
- `<ks>` is the name of the keyserver to which you are posting.

For example:
pgp --keyserver-send alice@example.com --keyserver ldap://keyserver.example.com

If there are multiple keys with user IDs that match the input, all of them will be posted. To make sure only a specific key is posted, use the key ID as the input.

pgp --keyserver-send 0x12345678 --keyserver ldap://keyserver.pgp.com

Only the specified key will be posted to ldap://keyserver.pgp.com, a public keyserver.

2 Press Enter when the command is complete.

PGP Command Line responds by posting the public key(s) to the specified keyserver.

Once you have posted your public key to a keyserver, you should search the keyserver for your public key to make sure it was correctly posted.

How to search for a key on a keyserver is described in “Finding a Public Key on a Keyserver” on page 50.

Exporting Your Public Key to a Text File

Once you have extracted your public key to a text file, it is easy to distribute. You can attach it to an email message, paste it into the body of an email, or copy it to a CD.

Use the --export command to export your public key.

To export a public key:

1 On the command line, enter:

    pgp --export <input>

where:

- <input> is the user ID, portion of the user ID, or the key ID of the key you want to export.

    By default, keys are exported as ASCII armor (.asc) files into the directory currently active on the command line.

For example:

    pgp --export example

All keys with the string “example” anywhere in them would be exported into separate .asc files.

    pgp --export "Alice C <acameron@example.com>"

Only keys that exactly match this user ID would be exported. The filename would be Alice C.asc.

2 Press Enter when the command is complete.

PGP Command Line creates the .asc file(s) in the appropriate directory.
Getting the Public Keys of Others

To encrypt data to a specific person, you need to encrypt it with their public key. Naturally, you have to get their public key onto your keyring first.

To get a public key onto your keyring, you must first find the public key on a keyserver and then import it from the keyserver onto your keyring.

Finding a Public Key on a Keyserver

In order to get a public key onto your keyring, you have to find the right key. In many cases, you can get the key you need from a keyserver. You use the same procedure for a public keyserver and a private keyserver.

Use the `--keyserver-search` command to search a keyserver for a key.

To search a keyserver for a key:

1. On the command line, enter:

   ```
   pgp --keyserver-search <input> --keyserver <ks>
   ```

   where:

   - `<input>` is the user ID, portion of the user ID, or the key ID of the key for which you are searching.

     If you are searching by key ID, only an exact match will be found (you can find the key ID of your key using the `--list-keys` command). If you are searching by user ID, any key whose user ID contains the user ID or portion of the user ID you enter will be found. So a search by user ID could return many matches, where a search by key ID will return only one key.

   - `<ks>` is the name of the keyserver you want to search.

     You can enter more than one keyserver, separated by a space. Only results from the first keyserver where there is a match will be returned.

   For example:

   ```
   pgp --keyserver-search example.com --keyserver ldap://keyserver.pgp.com
   ```

   This search would return keys that have “example.com” in the user ID and are on keyserver.pgp.com, a public keyserver.

2. Press **Enter** when the command is complete.

   PGP Command Line responds by listing the key or keys that match the search criteria you specified in the following format:

   ```
   Alg Type Size/Type Flags Key ID User ID
   --- ---- --------- ----- --------- -------
   DSS pub  2048/1024 [-----] 0x1234ABCD Alice C <ac@example.com>
   ```
Importing a Public Key from a Keyserver

Once you have found the key you want on the keyserver, you need to get the key from the keyserver onto your keyring.

Use the `--keyserver-recv` command to locate a key on a keyserver and import it onto your keyring.

To import a key from a keyserver:

1. On the command line, enter:

   ```
   pgp --keyserver-recv <input> --keyserver <ks>
   ```

   where:

   `<input>` is the user ID, portion of the user ID, or key ID of the key you want to get onto your keyring.

   To get a specific key, use the key ID. To get one or more keys, use the user ID or portion of the user ID.

   `<ks>` is the name of the keyserver you want to search.

   You can enter more than one keyserver to search, separated by a space. Only results from the first keyserver where there is a match will be returned.

   For example:

   ```
   pgp --keyserver-recv 0xABCD1234 --keyserver ldap://keyserver.pgp.com
   ```

   The key with the key ID shown would be imported if it were on the specified keyserver.

2. Press Enter when the command is complete.

   PGP Command Line responds by listing the key(s) it found on the specified keyserver that matched the criteria you specified and that the key(s) was imported:

   ```
   pgp:keyserver receive (2504:successful search on ldap://keyserver.pgp.com)
   0xABCD1234:keyserver receive (0:key imported as Alice C <ac@example.com>).
   ```

   If you want to make sure the key was imported onto your keyring, use the `--list-keys` command (the short form is `-l`) to see what keys are currently on your keyring.

Verifying Keys

If you have information you want to send to someone privately, and you are going to the trouble to encrypt it so that it stays private, then it is probably also important that you make sure the public key you have obtained and are going to use to encrypt your important information is actually from the person or organization that you believe it to be from.
One way to do this is to compare the fingerprint of the public key you have with the fingerprint of the real key. You could, for example, call the person on the phone and ask them to read the fingerprint of their key.

Some people also put the fingerprint of their PGP key on their Web site or on their business card, making it easy to compare the fingerprint of the real key with the fingerprint of the public key you have.

Use the **--fingerprint** command to see the fingerprint of any of the keys currently on your keyring; refer to “--fingerprint” on page 72 for more information.

To view the fingerprint of a key:

1. On the command line, enter:

   ```
   pgp --fingerprint <input>
   ```
   
   where:

   `<input>` is the user ID, portion of the user ID, or key ID of the key whose fingerprint you want to see.

   If you don’t enter any input, PGP Command Line will display the fingerprints of all keys on your keyrings.

   For example:

   ```
   pgp --fingerprint 0xABCD1234
   ```

   The user ID and the fingerprint of the key with the key ID shown would display if it were on either keyring.

   ```
   pgp --fingerprint
   ```

   The user IDs and the fingerprints of all keys on both keyrings would display.

2. Press **Enter** when the command is complete.

   PGP Command Line responds by listing the user ID of the key(s) it found that matched the criteria you specified and the fingerprint of that key using the following format:

   ```
   Alice Cameron <alice@example.com>
   896A 4A96 9C3A 3BEC  C87C EA8B 2CDB B87B 2CEB 53CC
   ```
Cryptographic Operations

Descriptions and Examples of Cryptographic Commands

This chapter describes the commands used in PGP Command Line that relate to cryptographic operations. These commands are:

- **--armor**, which converts a file to ASCII armor format (page 54).
- **--clearsign**, which creates a clear signature (page 56).
- **--decrypt**, which decrypts encrypted data (page 57).
- **--detached**, which creates a detached signature (page 59).
- **--dump-packets**, which dumps the packets in a PGP message (page 60).
- **--encrypt**, which encrypts your data (page 61).
- **--export-session-key**, which exports the session key that was used to encrypt data to a separate file (page 64).
- **--list-sda**, which lists the contents of an SDA (page 65).
- **--list-archive**, which lists the contents of a PGP Zip archive (page 65).
- **--sign**, which signs your data (page 66).
- **--symmetric**, which encrypts data using a symmetric cipher (page 68).
- **--verify**, which lets you verify data without creating any output (page 69).

Overview

This chapter covers four of PGP Command Line’s most significant cryptographic operations: encrypting, signing, decrypting, and verifying:

- **Encrypt**: A method of scrambling information to render it unreadable to anyone except the intended recipient, who must decrypt it to read it. You use PGP Command Line to encrypt your important information so that if it is stolen from a hard drive or intercepted while in transit, it is of no value to the person who has taken it because they cannot decrypt it.

- **Sign**: When you sign a message or file, PGP Command Line uses your private key to create a digital code that is unique to both the contents of the message/file and your private key. Only your public key can be used to verify your signature.

- **Decrypt**: When you receive decrypted data, it’s of no value until you decrypt it. To do this, you need to use the private key of the key pair that includes the public key that was used to encrypt the data.

- **Verify**: In addition to decrypting your data so that you can use it, you should also verify the files you use with PGP Command Line, including data, signature, and key files, to make sure they have not been tampered with.
For more information about these cryptographic operations, refer to *An Introduction to Cryptography*, which was installed with PGP Command Line.

## Commands

The commands that relate to encrypting and signing are described in the following sections.

### --armor (-a)

Armors data, produces a PGP armored file, and changes the default file extension from `.pgp` or `.sig` to `.asc`. The resulting ASCII armored data format is used with email systems that only allow ASCII printable characters. It converts the plaintext by expanding groups of three binary 8-bit bytes into four (4) printable ASCII characters, and the resulting file expands in size by approximately 33%.

The usage format is:

```
pgp --armor <input> [<input2> ...] [options]
```

Where:

- `<input>` is the file to be armored. It is either in the current directory, or its location has to be defined using a relative or absolute path. Multiple files can be armored.

- `[options]` let you modify the command:
  - `--comment`. Saves a comment at the beginning of the file with the header tag 'Comment'.
  - `--compress`. Compresses the output file.
  - `--compression-algorithm`. Sets the compression algorithm. The default for this option is `zip`.
  - `--eyes-only`. Text inputs that are processed using this option can only be decrypted to the screen.
  - `--input-cleanup`. This option will clean up the input file, depending on the arguments you specify: off (default), remove, or wipe.
  - `--output`. Lets you specify a different name for the armored file.
  - `--overwrite`. Sets the overwrite behavior when PGP Command Line tries to create an output file with the same name that already exists in the directory. This option accepts the following arguments: off (default), remove, rename, or wipe.
  - `--temp-cleanup`. Cleans up the temporary file(s), depending on the arguments you specify: off, remove, or wipe (default). For large encryption jobs, this option should be set to remove to speed up the process.
  - `--text`. Forces the input to canonical text mode. Do not use with binary files. Automatic detection of file types is not supported.
-v|--verbose. Gives a verbose (detailed) report about the operation.

The option --compression-algorithm is allowed when --armor is the primary operation (armor only). When --armor is combined with --sign or --encrypt operations, check these operations for details about setting the compression algorithm.

Examples:

1. `pgp --armor report.txt --overwrite remove`
   The ASCII armored output file “report.txt.asc” replaced the existing file with the same name, which was removed by overwriting.

2. `pgp -a report.txt --compression-algorithm zlib`
   The ASCII armored file “report.txt.asc” is compressed using the ZLIB compression algorithm.

Using --armor as an option with other commands to armor a file:

The usage format is:

```bash
gpg command1 input command2 user [--passphrase] pass --armor
```

Examples:

1. `pgp --sign report.txt --signer <alice@example.com> --passphrase "cam3r0n" --armor`
   The output file is an armored file “report.txt.asc,” which contains Alice’s signature.

2. `pgp -er "Bill Brown" report.txt --armor --comment "Urgent"
   Creates the ASCII armored file “report.txt.asc,” which is encrypted for Bill and has the plaintext comment “Urgent” displayed on top of the encrypted file:

   -----BEGIN PGP MESSAGE-----
   Version: PGP Command Line v9.6.0 (OSX)
   Comment: Urgent

   qANQR1DBwEwDRB9gEpFtI3MBB7OL7GQa1xrOLCp54FKg/
   FN4KZN1r+DrD31G10P
   e5xyNUQcYnQ2YqYZO2kDuFkOJ11E1HyixLs4m4ETYxhT3Eh/
   VA+yIjqqBH0w16k
   MXzGNfNFcp8SoQZGVL0m6bLWOtRY/5W2E90B0iB+f3Pv/VHsN5gDO/
   FmvzREJke
   ..
--clearsign

Causes the document to be wrapped in an ASCII-armored signature but otherwise doesn’t modify the document. The signed message can be verified to ensure that the original document has not been changed. To verify the signed message, use --verify.

The usage format is:

```
pgp --clearsign <input> [ <input2> ... ] --signer <user>
   --passphrase <pass> [options]
```

Where:

- `<input>` is the name of the file to be clear-signed. It is required. You can clear-sign multiple files by listing them, separated by a space.

- `<user>` is the user ID, portion of the user ID, or the key ID of the clear-signer. The private key of the clear-signer must be on the keyring. If `<user>` is not specified, the default key is used.

- `<pass>` is the passphrase of the private key of the clear-signer. It is required.

- `[options]` let you modify the command. Options are:
  - `--comment` saves a comment at the beginning of the file with the header tag 'Comment'.
  - `--input-cleanup` cleans up the input file, depending on the arguments you specify: off (default), remove, or wipe.
  - `--overwrite` sets the overwrite behavior when PGP Command Line tries to create an output file with the same name that already exists in the directory. This option accepts the following arguments: off (default), remove, rename, or wipe.
  - `--temp-cleanup` cleans up the temporary file(s) depending on the arguments you specify: off, remove, or wipe (default). For large encryption jobs, this option should be set to remove to speed up the process.
  - `--text` forces the input to canonical text mode. Do not use with binary files (automatic detection of file types is not supported).
  - `-v|--verbose` gives a verbose (detailed) report about the operation.

Example:

```
pgp --clearsign newnote.txt --signer bob@example.com
   --passphrase sm1t4
newnote.txt:sign (0:output file newnote.txt.asc)
```

The resulting file "newnote.txt.asc" will have the unchanged text, "wrapped" between the header and the footer such as this:

```
-----BEGIN PGP SIGNED MESSAGE-----
Hash: PGP SIGNED MESSAGE-----
... SHA256
```

(the unchanged text in the file "new.note.asc")
--decrypt

Decrypts encrypted data.

If data being decrypted is also signed, the signature is automatically verified during the decryption process.

The usage format is:

```
pgp --decrypt <input> [<<input2>  ...] [<<inputd>...] [options]
```

Where:

- `--archive`. When you decrypt archives, note the following:
  - if you specify `--archive`, the contents of the archive are extracted
  - if you don’t specify `--archive`, only the .tar file is extracted

- `<inputd>` Additional detached signature target files are allowed. Note that PGP Command does not write output when decrypting detached signature files.

- `--eyes-only` Text inputs that are processed using this option can only be decrypted to the screen: the recipient must view the output on screen when decrypting a message. The default is off.

When decrypting data that is marked for your eyes only, PGP Command Line generates an error if the option `--eyes-only` is not specified.

- `--input-cleanup` cleans up the input file, depending on the arguments you specify: off (default), remove, or wipe.

- `--output` lets you specify a different name for the decrypted file.

- `--overwrite` sets the overwrite behavior when PGP Command Line tries to create an output file and it already exists. It accepts the following arguments: off (default), remove, rename, or wipe.
--passphrase is used for [asymmetrically] encrypted files

--sda. When decrypting SDAs, the option --sda must be specified or PGP Command Line will not be able to find PGP data.

To decrypt an SDA, you need either --symmetric-passphrase or --passphrase. Note that the symmetric passphrase cannot have an empty string (""), while the asymmetric passphrase can have an empty string because such passphrase references a private key.

When decrypting SDAs or archives, files will be automatically overwritten. The option -o (output) can be used to specify the output directory; this directory will be created if it does not exist.

--symmetric-passphrase is used for symmetrically encrypted files.

--temp-cleanup cleans up the temporary file(s), depending on the arguments you specify: off, remove, or wipe (default). For large encryption jobs, this option should be set to remove to speed up the process.

-v|--verbose gives a verbose (detailed) report about the operation.

Examples:

1. `pgp --decrypt note.txt.pgp --symmetric-passphrase "cam3r0n" --overwrite remove`
   Decrypts the file to "note.txt" and removes the existing file with the same name by overwriting it.

2. `pgp --decrypt keyshares.exe --sda --symmetric-passphrase "sm1t4"`
   keyshares.exe:decrypt (0:directory created successfully)
   keyshares.exe:decrypt (0:output file keyshares\Alice Cameron-1-Bob Smith.shf)
   keyshares.exe:decrypt (0:output file keyshares\Alice Cameron-2-John Jones.shf)
   keyshares.exe:decrypt (0:output file keyshares\Alice Cameron-3-Bill Brown.shf)
   keyshares.exe:decrypt (0:output file keyshares\pgp)
   keyshares.exe:decrypt (0:SDA decoded successfully)
   Decrypts a SDA.

3. `pgp --decrypt keyshares.exe --symmetric-passphrase "sm1t4"
   keyshares.exe:decrypt (3031:input does not contain PGP data)
   If you don’t enter the option --sda. PGP Command Line will not recognize the SDA you want to decrypt and uncompress.

4. `pgp --decrypt note.txt.sig --passphrase sm1t4`
   note.txt:decrypt (1082:detached signature target file)
   note.txt.sig:decrypt (3038:signing key 0x6245273E Bob Smith <bob@example.com>)
note.txt.sig:decrypt (3040:signature created
2005-10-28T12:44:38-07:00)

note.txt.sig:decrypt (3035:good signature)

Decrypts the detached signature file "note.txt.sig". When decrypting detached
signature files, you will get only a status message as output.

5  pgp --decrypt bobsarchive.pgp --passphrase "sm1t4"
bobsarchive.pgp:decrypt (0:output file bobsarchive.tar)

Decrypts the archive file into a tar file.

6  pgp --decrypt bobsarchive.pgp --passphrase sm1t4 --archive
bobsarchive.pgp:decrypt (0:output file .\note.txt)
bobsarchive.pgp:decrypt (0:output file .\report.doc)

Decrypts the archive file into the actual archived files "note.txt" and report.doc,
with their path information included.

--detached (-b)

Signs data and creates a detached signature. If you use this command to sign a
document, both the document and detached signature are needed to verify the signature.
To verify the signed message, use --verify.

The usage format is:

    pgp --detached <input> [<input2> ...] --signer <user>
    --passphrase <pass> [options]

Where:

<input> is the name of the file for which the detached signature is being created. It
is required. You can create a detached signature for multiple files by listing them,
separated by a space.

<user> is the user ID, portion of the user ID, or the key ID of the signer. It is
required. The private key of the signer must be on the keyring.

<pass> is the passphrase of the private key of the signer. It is required.

[options] let you modify the command. Options are:

--armor armors the data and changes the file extension from .sig to .asc.

--comment saves a comment at the beginning of the file with the header tag
"Comment". It works only if --armor is specified as well.

--input-cleanup cleans up the input file, depending on the arguments you
specify: off (default), remove, or wipe.

--output lets you specify a different name for the created file.

--overwrite sets the overwrite behavior when PGP Command Line tries to create
an output file that already exists. This option accepts the following arguments: off
(default), remove, rename, or wipe.
--temp-cleanup cleans up the temporary file(s), depending on the arguments you specify: off, remove, or wipe (default). For large encryption jobs, this option should be set to remove to speed up the process.

--text forces the input to canonical text mode. Do not use this option with binary files (automatic detection of file types is not supported).

--v|--verbose gives a verbose (detailed) report about the operation.

Examples:

1. `pgp -b note.txt --passphrase sm4 --signer "Bob Smith"
   note.txt:sign (0:output file note.txt.sig)
   Output is the file note.txt.sig, which contains Bob's detached signature.

2. `pgp --verify note.txt.sig`
   note.txt:verify (1082:detached signature target file)
   note.txt.sig:verify (3038:signing key 0x6245273E Bob Smith <bob@example.com>)
   note.txt.sig:verify (3040:signature created 2005-10-28T12:44:38-07:00)
   note.txt.sig:verify (3035:good signature)
   note.txt.sig:verify (0:verify complete)
   The detached signature is verified:

--dump-packets, --list-packets

Dumps the packet information in a PGP message. Input is a list of files or standard input; output is always a standard output.

This command uses the normal output format for data blocks and displays hexadecimal values in the format "NN".

The usage format is:

```
pgp --dump-packets <input> [<input2> ...] [options]
```

Where:

<input> is a list of files or standard input.

<input2> are additional files.

[options] let you modify the command. Options are:

--buffered-stdio enables buffered stdio for stdin and stdout.

Example:

```
pgp --dump-packets TrainingDetails.msg
Processing file TrainingDetails.msg
New: unknown(tag 16)(4049 bytes)
Old: Trust Packet(tag 12)(46 bytes)
```
--encrypt (-e)

Encrypts a document to specified recipients. Input is either the standard input or a list of files. Output is either the standard output, a list of files, or an archive. If you use standard input, note that it cannot be combined with other inputs.

```
Old: Reserved(tag 0)(2 bytes)
```

When encrypting, the preferred cipher and compression algorithms of the recipient is used. If there is more than one recipient, the most compatible algorithm is used. Note that you cannot specify a one-time cipher or compression algorithm with --encrypt.

The usage format is:

```
pgp --encrypt <input> [<input2> ...] --recipient <user>
[-r <user2> ...] [options]
```

Where:

- `<input>` is the name of the file to be encrypted. It is required. You can encrypt multiple files by listing them, separated by a space. The default output filename for an encrypted file is `<input_filename>.pgp`. Note that stdin can be used only by itself and cannot be combined with other inputs.

- `<user>` is the user ID, portion of the user ID, or the key ID of the recipient. It is required. The public key of the recipient must be on the keyring. You must specify a recipient; you cannot encrypt to your own key by not specifying a recipient. You can encrypt the file to multiple recipients by listing them, separated by a space.

- `[options]` let you modify the command. Options are:
  - `--adk` can be used only together with the option --sda. Note that if any of the keys used with the option --adk have ADKs, they will also be used.
  - `--archive` saves the output as an archive. It cannot be used with the options --text-mode or --sda. When using --archive, directories can be in the input file: without this option, the directories are skipped.
  - `--cipher` If the option --cipher is used, the existing cipher will be forcefully overridden and the key preferences and algorithm lists in the SDK will be ignored. This can create messages that don’t comply with the OpenPGP standard. This option must be used together with the option --force.

Caution

The --encrypt command is not used for symmetric encryption; instead, use the --symmetric command, described in “--symmetric (-c)” on page 68.
--comment saves a comment at the beginning of the file with the header tag 'Comment'. It works only if --armor is specified as well.

--compress toggles compression. If enabled, the preferred compression algorithm of the recipient is used.

--compression-algorithm If the option --compression-algorithm is used, the existing compression algorithm will be forcefully overridden and the key preferences and algorithm lists in the SDK will be ignored. This can create messages that do not comply with the OpenPGP standard. This option must be used together with the option --force.

--encrypt-to-self lets you encrypt to the default key in addition to any other specified keys. The default is off.

--eyes-only Text inputs that are processed using this option can only be decrypted to the screen.

--force required to use --compression-algorithm and --cipher.

--input-cleanup cleans up the input file, depending on the arguments you specify: off (default), remove, or wipe.

--output lets you specify a different name for the encrypted file.

--overwrite sets the overwrite behavior when PGP Command Line tries to create an output file that already exists. This option accepts the following arguments: off (default), remove, rename, or wipe.

--root-path can only be used with either --sda or --archive.

--sda cannot be used together with the command --sign (such as -es). For more information, refer to the option --sda.

--sign lets you sign the encrypted file.

--temp-cleanup cleans up the temporary file(s) depending on the arguments you specify: off, remove, or wipe (default). For large encryption jobs, this option should be set to remove to speed up the process.

--text forces the input to canonical text mode. Do not use with binary files (automatic detection of file types is not supported).

-v | --verbose gives a verbose (detailed) report about the operation.

Refer to the descriptions of these options or to the man page for information about how to use these options.

Examples:

1. pgp --encrypt report.txt README.rtf -r "Bill Brown" -r "Mary Smith" -r "Bob Smith"

The files 'report.txt' and 'README.rtf' are encrypted to multiple recipients.
2  `pgp -er "Bob Smith" report.txt --eyes-only`
   The output file "readme.txt.pgp" is encrypted for Bob's "eyes only", which means that he can read the file only on the screen.

3  `pgp -e report.doc -r "Bob Smith" --output newreport.pgp -v`
   The output file is "newreport.pgp", and the on-screen message contains the following detailed information about the performed operation:

   `pgp:encrypt (3157:current local time 2005-11-05T12:13:09-08:00)`
   `/Users/bobsmith/.pgp/pubring.pkr:open keyrings (1006:public keyring)`
   `/Users/bobsmith/.pgp/secring.skr:open keyrings (1007:private keyring)`
   `0x4A8C54B8:encrypt (1030:key added to recipient list)`
   `report.doc:encrypt (3048:data encrypted with cipher AES-128)`
   `report.doc:encrypt (0:output file newreport.pgp)`

4  `pgp -er "Bob Smith" report.doc --output /Users`
   `myreport.doc:encrypt (0:output file myreport.doc.pgp)`
   `report.doc:encrypt (0:output file /Users/report.doc.pgp)`
   You have encrypted the file `report.doc` to the specified directory.

5  `pgp -er "Bob Smith" *.doc`
   `myreport.doc:encrypt (0:output file myreport.doc.pgp)`
   `report.doc:encrypt (0:output file report.doc.pgp)`
   Both files with the extension `.doc` were encrypted for the user Bob.

6  `pgp -er "Bob Smith" *.doc -output /Users`
   `myreport.doc:encrypt (0:output file /Users/myreport.doc.pgp)`
   `report.doc:encrypt (0:output file /Users/report.doc.pgp)`
   You have encrypted all files with the extension `.doc` to another directory.

7  `pgp -er "Bob Smith" *.doc --output archive.pgp`
   `pgp:encrypt (3028:multiple inputs cannot be sent to a single output file)`
   Nothing happened since the archive mode was not enabled.

8  `pgp -er "Bob Smith" *.doc --output archive.pgp --archive`
   `pgp00000.tmp:encrypt (3110:archive imported myreport.doc)`
   `pgp00000.tmp:encrypt (3110:archive imported report.doc)`
   `pgp00000.tmp:encrypt (0:output file archive.pgp)`
   With the option --archive added, the two doc files are encrypted into archive.pgp.

9  `pgp -er "Bob Smith" /Users/note.txt`
   `/Users/note.txt:encrypt (0:output file /Users/note.txt.pgp)`
   In this case, you have encrypted the file note.txt, which was located in another directory.
10  `pgp -er "Bob Smith" /Users/*/txt -o MyNewArchive.pgp --archive`  
    `pgp00000.tmp:encrypt (3110:archive imported /Users/note.txt)`  
    `pgp00000.tmp:encrypt (3110:archive imported /Users/note2.txt)`  
    `pgp00000.tmp:encrypt (0:output file MyNewArchive.pgp)`  

In this case, you have encrypted multiple text files located in another directory into a new archive in your local directory.

**--export-session-key**

Exports the session key of an encrypted message. This key is used to encrypt each set of data on a transaction basis, and a different session key is used for each communication session. Output of this command is a key file with the extension .key, which contains the key fingerprint of the key used during the session that produced the encrypted file.

Using the session key, it is possible to decrypt a document without the recipient’s private key and its passphrase. Therefore, it reveals only the content of a specific message without compromising the private recipient’s key (which would reveal all messages encrypted to that key). Note that a user cannot directly specify a session key during encryption.

The usage format is:

```
pgp --export-session-key <input> [ <input2> ... ] --passphrase <pass> [ --output ]
```

Where:

- `<input>` is the encrypted file whose session key is to be exported to a separate file. It is required. Multiple files can have their session key exported as well; each encrypted file must be listed, separated by a space.

- `--passphrase` is needed for encrypted files (`--symmetric-passphrase` is used for conventionally encrypted files, but `--passphrase` will also work)

- `--output` lets you specify a different filename for the resulting file.

Refer to the descriptions of these options for information about how to use them.

Example:

1  `pgp -e report.doc -r "Bob Smith" --output BobsReport.pgp`  
    `report.doc:encrypt (0:output file BobsReport.pgp)`  

First, the file `report.doc` was encrypted into `BobsReport.pgp`.

2  `pgp --export-session-key BobsReport.pgp --passphrase "sm1t4"`  
    BobsReport.pgp:export session key (0:output file report.doc.key)`

Second, the key used for the encrypting session was exported into the file `report.doc.key`, which contains the fingerprint of the key used for the session, such as:

```
7:8F042E99E383FCD4921FD74A63C514D3
```
**--list-sda**

Lists the contents of a Self-Decrypting Archive (SDA). The entire SDA needs to be decrypted in order to list its contents, which could take up to several minutes (depending on the number and size of the files in the archive).

The usage format is:

```
pgp --list-sda <input> --passphrase <pass>
```

Where:

- `<input>` is an SDA file, such as reports.exe. Output is always the standard output.
- `<pass>` This is a passphrase or symmetric passphrase with which the SDA was encrypted.

Example:

```
pgp --list-sda reports.exe --symmetric-passphrase smlt4

reports\ reports\README.rtf
reports\README.txt
reports\report.txt
reports.exe:list SDA (0:SDA decoded successfully)
```

The archive “reports.exe” was decrypted and listed.

**--list-archive**

Lists the contents of a PGP Zip archive, which lets you add any combination of files and folders to an encrypted, compressed, portable archive.

A PGP Zip archive is an excellent way to distribute files and folders securely or back them up. Refer to “--archive” on page 140 for more information about PGP Zip archives.

The usage format is:

```
pgp --list-archive <input> [<input2> ...] --passphrase <pass>
```

Where:

- `<input>` is the PGP archive(s) whose files you want to list.
- `<pass>` is the passphrase of the archive whose files you want to list.

Example:

```
p gp --list-archive archive.pgp --passphrase smlt4

In this case, the archive is located in the local directory and no directory path is displayed.
```

```
report.txt
README.txt
```
--sign (-s)

Signs a document, without encrypting it. You can sign and encrypt a file at the same time using the command -es. Input is a standard input or a list of files; output is a standard output or a list of files.

The usage format is:

```
pgp --sign <input> [<input2> ...] --passphrase <pass>
[--signer <user>] [options]
```

Where:

- `<input>` is the name of the file to be signed. It is required. You can sign multiple files by listing them, separated by a space.
- `<pass>` is the passphrase of the private key of the signer. It is required.
- `<user>` is the user ID, portion of the user ID, or the key ID of the signer. The private key of the signer must be on the keyring. If `<user>` is not specified, the default key is used to sign.

--archive allows you to create an unencrypted signed tar file. You cannot use this archive until it is decrypted (the signature is removed). Using the option --sign with --archive, you can create a signed tar file that anyone can open.

-a, --armor. Armors the signed file.

--comment saves a comment at the beginning of the file with the header tag 'Comment'. It works only if --armor is specified as well.

--compress toggles compression.

--compression-algorithm. You can select the compression algorithm in case you are creating an attached opaque signature only (that is not encrypted), or when you are creating a conventionally encrypted and signed output.

--eyes-only. Text inputs that are processed using this option can be decrypted only to the screen.

--force. Required to use --hash.

--hash. If you use this option, the existing hash algorithm will be forcefully overridden. Note that the key preferences and algorithm lists in the SDK will be ignored, which can lead to the creation of messages that violate OpenPGP standard. You must use the option --force with --hash.

--input-cleanup cleans up the input file, depending on the arguments you specify: off (default), remove, or wipe.

--output lets you specify a different name for the signed file.

--overwrite sets the overwrite behavior when PGP Command Line tries to create an output file that already exists. This option accepts the following arguments: off (default), remove, rename, or wipe.

--temp-cleanup cleans up the temporary file(s) depending on the arguments you specify: off, remove, or wipe (default). For large encryption jobs, this option should be set to remove to speed up the process.
--text forces the input to canonical text mode. Do not use with binary files (automatic detection of file types is not supported).

-v|--verbose gives a verbose (detailed) report about the operation.

Refer to the descriptions of these options or to the man page for information about how to use these options.

Examples:

1. `pgp -s report.txt --signer "Bob Smith" --passphrase "sm1t4" report.txt:sign (0:output file report.txt.pgp)`
   Output is "report.txt.pgp" signed by Bob.

2. `pgp -es report.txt -r bob@example.com --passphrase "cam3r0n"` This command produces "report.txt.pgp," which is encrypted for Bob and signed by Alice using her passphrase (we assume that her key is the default signing key and the option --signer is not used).

3. `pgp -s report.txt --signer "Bob Smith" --passphrase "sm1t4" --compression-algorithm zip report.txt:sign (0:output file report.txt.pgp)`
   The file "report.txt.pgp" was signed by Bob and compressed using the Zip compression algorithm.

4. `pgp -s report.doc note.txt --signer "Bob Smith" --passphrase "sm1t4" -o NewArchive.pgp --archive`
   `pgp00001.tmp:sign (3110:archive imported report.doc)`
   `pgp00001.tmp:sign (3110:archive imported note.txt)`
   `pgp00001.tmp:sign (0:output file NewArchive.pgp)`
   First, both files are signed and saved as a tar file NewArchive.pgp. This file cannot be used until the signature is removed by decrypting the file. This file is just opaquely signed, and you do not need a passphrase to verify the signature:

   `pgp --decrypt NewArchive.pgp` NewArchive.pgp:decrypt (3038:signing key 0x6245273E Bob Smith <bob@example.com>)
   NewArchive.pgp:decrypt (3040:signature created 2005-11-11T16:40:42-08:00)
   NewArchive.pgp:decrypt (3035:good signature)
   NewArchive.pgp:decrypt (0:output file NewArchive.tar)
   The resulting tar file can be uncompressed with utilities that are appropriate for your platform.
--symmetric (-c)

Encrypts data using symmetric encryption, not public-key encryption.

The usage format is:

```
pgp --symmetric <input> [<input2> ...] --symmetric-passphrase <pass> [options]
```

Where:

- `<input>` is the name of the file to be symmetrically encrypted and it is required. You can encrypt multiple files by listing them, separated by a space. The default filename for an encrypted file is `<input filename>.pgp`. You can modify the filename of the encrypted file using `--output`.

- `<pass>` is the passphrase you want to use for the symmetrically encrypted file.

- `[options]` let you modify the command. Options are:
  - `--output` lets you specify a different filename for the encrypted file.
  - `--sign` lets you sign the encrypted file. If you use `--sign` with `--symmetric`, you will need both `--symmetric-passphrase` for the encryption and `--passphrase` for the signature.
  - `--armor` armors the output file. File extension is changed to `.asc`.
  - `--comment` lets you specify a comment for armored data.
  - `--text` forces the `<input>` to canonical text mode. Do not use with binary files. Automatic detection of file type is not supported.
  - `--compress` toggles compression.
  - `--compression-algorithm` specifies the compression algorithm to use for the operation. The default is Zip.
  - `--cipher` specifies the cipher to use for the operation. The default is AES256.
  - `--eyes-only` prevents the decrypted output from being saved to disk; the decrypted output can only be displayed on-screen.
  - `--encrypt-to-self` lets you encrypt to the default key.
  - `--archive` lets you combine multiple files into a single `.pgp` file.
  - `--overwrite` lets you specify what to do if a file of the same name as the output filename already exists.
  - `--input-cleanup` lets you specify what to do with `<input>` files when the operation is done. The default is off (leave them alone).
  - `--temp-cleanup` lets you specify how to handle temporary files. The default is to wipe them.
  - `--verbose (-v)` shows verbose results information.
Examples:

1. `pgp --symmetric file.txt --symmetric-passphrase "Bilbo$Frodo"`
   Encrypts a file, which will be called `file.txt.pgp`, using the passphrase 'Bilbo$Frodo' without the quotes.

2. `pgp -ec file.txt --symmetric-passphrase "Bilbo$Frodo"
   Same as above, using the short forms.

The important information about `--encrypt` also applies to `--symmetric`.

**--verify**

Verifies that data was not tampered with and tests whether PGP Command Line can process the entire file. It verifies data, signatures, and key files and works on all PGP Command Line data types. The command output describes what was verified.

The usage format is:

```
pgp --verify <input> [<input2> ...] [options]
```

Where:

- `<input>` is the file to be verified. It is required.
- `[options]` let you modify the command. Options are:
  - `--input-cleanup` cleans up the input file, depending on the arguments you specify: off (default), remove, or wipe.
  - `--passphrase | --symmetric-passphrase`. This is the passphrase that is required for encrypted files.
  - `--temp-cleanup` cleans up the temporary file(s) depending on the arguments you specify: off, remove, or wipe (default). For large encryption operations, this option should be set to remove to speed up the process.
  - `-v | --verbose` gives a verbose (detailed) report about the operation.

Refer to the descriptions of these options for information about how to use them.

Example:

```
pgp --verify report.doc.pgp --passphrase "smit4"
```

```
report.doc.pgp:verify (3111: data is a PGP archive)
report.doc.pgp:verify (3042: suggested output file name report.doc.tar)
report.doc.pgp:verify (3038: signing key 0x6245273E Bob Smith <bob@example.com>)
report.doc.pgp:verify (3040: signature created 2005-11-10T13:58:07-08:00)
report.doc.pgp:verify (3035: good signature)
report.doc.pgp:verify (0: verify complete)
```

The file `report.doc.pgp` is verified.
Key Listings

How to Get Information About Your Keys

This chapter describes the commands that list information about the PGP keys on keyrings. These commands are:

- `--fingerprint`, which lists the fingerprints of keys on your keyring, in hexadecimal numbers or biometric words (page 72).
- `--fingerprint-details`, which lists the fingerprints of keys on your keyring and their subkeys, in hexadecimal numbers or biometric words (page 72).
- `--list-key-details`, which lists the keys on the keyring and displays detailed information about those keys (page 75).
- `--list-keys`, which lists the keys on the keyring (page 76).
- `--list-keys-xml`, which lists keys in XML format (page 77).
- `--list-sig-details`, which provides detailed information about signatures on a key (page 78).
- `--list-sigs`, which lists the keys on the keyring and the user IDs and signatures on those keys (page 78).
- `--list-userids`, which lists the keys on the keyring and the user IDs on those keys (page 79).

Overview

At some point, you are going to need to know about the keys on your keyrings. The key listing commands provide those details. Using the commands in basic display mode gives you summary information about the keys on a keyring. Detailed display mode tells you everything there is to know about those keys.

Refer to Appendix A, Lists for more information about what the key and signature lists show about a key.
Commands

The key listing commands are described in the following sections.

--fingerprint

Lists the fingerprints of keys on your keyring that match the supplied criteria. If you run the command with no user or key ID information, all key fingerprints will be displayed. If you enter any user or key ID information, only key fingerprints that match will be displayed.

The usage format is:

```
pgp --fingerprint [<user1> ...] [--biometric] [--verbose]
```

Where:

- `<user1>` is the user ID, portion of a user ID, or the key ID of a key on your keyring. If you don’t supply a user ID, all fingerprints will be listed.
- `--biometric` displays biometric words instead of hexadecimal numbers.
- `--verbose` shows the key IDs under the primary user ID for each fingerprint.

Examples:

```
pgp --fingerprint Alice
```

Displays the fingerprint in hexadecimal of any keys on the keyring that match "Alice" using the format:

```
Alice Cameron <alice@example.com>
896A 4A96 9C3A 3BEC C87C EA8B 2CDB B87B 2CEB 53CC
```

```
pgp --fingerprint 0x12345678 --biometric
```

Displays the fingerprint in biometric words of the key with the specified key ID using the format:

```
Alice Cameron <alice@example.com>
aimless    photograph   goldfish    yesteryear
beeswax    corporate    crackdown   millionaire
indoors    upcoming     choking     sardonic
reward      underfoot   eyeglass    amulet
sawdust     holiness     glitter     therapist
```

1 key found
--fingerprint-details

Lists the fingerprints and subkeys of keys on your keyring that match the supplied criteria. If you run the command with no user or key ID information, all key fingerprints will be displayed. If you enter any user or key ID information, only key fingerprints that match will be displayed.

Subkey fingerprints are displayed if found on the specified key. Hash names are the same as listed in the detailed key list mode.

Fingerprints are shown with one of the following prefixes:

- Key Fingerprint indicates that the following fingerprint is for a master key.
- Subkey Fingerprint indicates that the following fingerprint is for a subkey.
- X.509 <alg> Thumbprint indicates that the following thumbprint is for an X.509 certificate, where <alg> is replaced by the hash algorithm used to create the thumbprint.

The usage format is:

```plaintext
gpg --fingerprint-details [<user1> ...] [--biometric]
```

Where:

- `<user1>` is the user ID, portion of a user ID, or the key ID of a key on your keyring. If you do not supply a user ID, all fingerprints and subkeys will be listed.
- `--biometric` displays biometric words instead of hexadecimal numbers.

Examples:

```plaintext
pgp --fingerprint-details Alice

Displays the fingerprint in hexadecimal of any keys on the keyring that match "Alice" using the format:

Alice Cameron <alice@example.com>

Key Fingerprint: 0x6D2A476D (0x7B72AAE06D2A476D)
  D2E0 23B2 53D0 49C9  6812 31AC 7B72 AAE0  6D2A 476D
Subkey Fingerprint: 0xB86FF2CF (0x0787EE48B86FF2CF)
  DAB6 570B 9411 197D  5DDF A9B2 0787 EE48  B86F F2CF
```
pgp --fingerprint-details 0xF88C6910 --biometric

Displays the key and subkey fingerprints in biometric words of the key with the specified key ID using the format:

Alice Cameron <alice@example.com>

Key Fingerprint: 0x6D2A476D (0x7B72AAE06D2A476D)

<table>
<thead>
<tr>
<th>crucial</th>
<th>performance</th>
<th>ragtime</th>
<th>adviser</th>
</tr>
</thead>
<tbody>
<tr>
<td>robust</td>
<td>molasses</td>
<td>stairway</td>
<td>sardonic</td>
</tr>
<tr>
<td>beehive</td>
<td>quantity</td>
<td>spindle</td>
<td>gravity</td>
</tr>
<tr>
<td>reform</td>
<td>monument</td>
<td>artist</td>
<td>supportive</td>
</tr>
<tr>
<td>Vulcan</td>
<td>megaton</td>
<td>gazelle</td>
<td>autopsy</td>
</tr>
</tbody>
</table>

Subkey Fingerprint: 0xB86FF2CF (0x0787EE48B86FF2CF)

<table>
<thead>
<tr>
<th>chatter</th>
<th>decimal</th>
<th>snowcap</th>
<th>caravan</th>
</tr>
</thead>
<tbody>
<tr>
<td>breadline</td>
<td>caravan</td>
<td>pupil</td>
<td>decimal</td>
</tr>
<tr>
<td>beeswax</td>
<td>Wilmington</td>
<td>tunnel</td>
<td>nebula</td>
</tr>
<tr>
<td>bombast</td>
<td>outfielder</td>
<td>endorse</td>
<td>Jupiter</td>
</tr>
<tr>
<td>prelude</td>
<td>Eskimo</td>
<td>drainage</td>
<td>sandalwood</td>
</tr>
</tbody>
</table>
**--list-key-details**

Lists the keys on a keyring in detailed output mode. If you run the command with no user or key ID information, all keys on the keyring will be displayed. If you enter any user or key ID information, only keys that match will be displayed.

The usage format is:

```
pgp --list-key-details [user1] ...
```

Where:

`<user1>` is the user ID, portion of a user ID, or the key ID of a key on your keyring.

Example:

```
pgp --list-key-details Alice
```

Lists all of the keys on your keyrings using the format:

```
Key Details: Alice Cameron <alice@example.com>
   Key ID: 0xB2726BDF (0xAAEB5E06B2726BDF)
      Type: RSA (v4) key
      Size: 2048
   Validity: Complete
      Trust: Implicit (Axiomatic)
   Created: 2006-04-22
  Expires: Never
   Status: Active
   Cipher: AES-192
   Cipher: AES-128
   Cipher: CAST5
   Cipher: TripleDES
   Cipher: Twofish-256
   Hash: SHA
   Compress: Zip (Default)
      Photo: No
   Revocable: No
      Token: No
   Keyserver: keyserver.pgp.com
      Default: No
   Prop Flags: Sign user IDs
   Prop Flags: Sign messages
   Ksrv Flags: None
  Feat Flags: Modification detection
Notations: 01 0x80000000 preferred-email-encoding@pgp.com:pgpmime

Subkey ID: 0x6F742FE6 (0x939BB8896F742FE6)
      Type: ElGamal
      Size: 2048
   Created: 2006-04-22
  Expires: Never
   Status: Active
  Revocable: No
   Prop Flags: Encrypt communications
   Prop Flags: Encrypt storage
      ADK: None
```
Revoker: None

1 key found
For more information, refer to “Detailed Key List” on page 193.

--list-keys (-l)

Lists the keys on a keyring in basic output mode. If you run the command with no user or key ID information, all keys on the keyring will be displayed. If you enter any user or key ID information, only keys that match will be displayed.

The usage format is:

    pgp --list-keys [<user1> ...]

Where:

    <user1> is the user ID, portion of a user ID, or the key ID of a key on your keyring.

Examples:

1  pgp --list-keys
   Lists all of the keys on your keyrings using the format:

   Alg  Type  Size/Type Flags   Key ID     User ID
   ---  ----  --------- ------- ---------- ------------------------
   DSS  pub  2048/1024 [-----] 0xABCD1234 Alice C <ac@example.com>

   1 key found

2  pgp -l Alice Bob Jill
   Uses the short form of the command; displays any key on the keyring with 'Alice', 'Bob', or 'Jill' in the user ID.

3  pgp -l 0x12345678
   Lists only the key with the specified key ID, if it is on the keyring.

For more information, refer to “Basic Key List” on page 187.
--list-keys-xml

When you choose to list a key in XML format, PGP Command Line will display all information including all user IDs and signatures. If you run the command with no user or key ID information, all keys on the keyring will be displayed. If you enter any user or key ID information, only keys that match will be displayed.

To list keys in XML format, you may use either the command --list-keys-xml, or a key list operation with the added option --xml, such as --list-keys user1 --xml, or --list-keys --xml.

The usage format is:

    pgp --list-keys-xml [<user1> ...]

Where:

    <user1> is the name of the specific local user whose keys you want to check.

Example:

    pgp --list-keys-xml "Jose Medina"

Here is an abbreviated key list in XML format. For more details and explanations, refer to “Key List in XML Format” on page 206.

    <?xml version="1.0"?>
    <keyList>
      <key>
        ....
        <signature>
        ...
        <subkey>
        ...
        <adk>
        ...
        <revoker>
      </key>
    </keyList>
--list-sig-details

Lists keys with their user IDs and signatures in detailed output mode.

The usage format is:

    pgp --list-sig-details <user> [<user2> ...]

Where:

    <user> is the user ID, portion of a user ID, or the key ID of a key on your keyring. You can list one or more users, with their names/IDs separated by a space. If you don’t specify a user, you will get an error message (“too many keys found”).

Example:

    pgp --list-sig-details Alice

Lists Alice’s key and shows details about her user IDs and signatures:

    Signature Details: Alice Cameron <alice@example.com>
    Signed Key ID: 0xB2726BDF (0xAAEB5E06B2726BDF)
    Signed User ID: Alice Cameron <alice@example.com>

    Signer Key ID: 0xB2726BDF (0xAAEB5E06B2726BDF)
    Signer User ID: Alice Cameron <alice@example.com>

    Type: DSA signature
    Exportable: Yes
    Status: Active
    Created: 2006-04-22
    Expires: Never
    Trust Depth: 0
    Domain: None

    1 signature found

For more information, refer to “Detailed Signature List” on page 213.

--list-sigs

Lists keys with their user IDs and signatures in basic output mode. If you run the command with no user or key ID information, all signatures on the keyring will be displayed. If you enter any user or key ID information, only signatures that match will be displayed.

The usage format is:

    pgp --list-sigs [<user> ...]

Where:

    <user> is the user ID, portion of a user ID, or the key ID of a key on the keyring.

Example:

    pgp --list-sigs 0x12345678

Lists the user IDs and signatures on the key with the specified key ID, if it is on the keyring.
--list-userids

Lists keys and their user IDs in basic output mode. The command `--list-users` is the same as `--list-userids`.

The usage format is:

```
pgp --list-userids [<user1> ...]
```

Where:

- `<user1>` is the user ID, portion of a user ID, or the key ID of a key on your keyring.

Examples:

1. `pgp --list-userids`
   Lists all of the user IDs on the keys on your keyrings.

2. `pgp --list-users`
   Same as the previous command, using the other form of the command.

3. `pgp --list-userids Alice Bob Jill`
   Lists any key on the keyring with "Alice", "Bob", or "Jill" in the user ID.
This chapter describes those commands that explain how PGP Command Line interacts with keyservers.

- `--keyserver-disable`, which disables keys on a keyserver (page 81).
- `--keyserver-recv`, which gets keys from a keyserver and imports them onto your keyring (page 82).
- `--keyserver-remove`, which removes keys from a keyserver (page 83).
- `--keyserver-search`, which searches a keyserver for keys but does not import them (page 84).
- `--keyserver-send`, which sends keys to a keyserver (page 85).
- `--keyserver-update`, which updates keys on a keyserver (page 86).

### Overview

PGP Command Line provides several commands that let you interact with keyservers. These commands help you post keys to a keyserver, import keys from a keyserver, and so on.

When using commands that require you to specify a keyserver, make sure to use the full URL to the keyserver such as `ldap://keyserver.pgp.com`, and not just `keyserver.pgp.com`.

### Commands

#### `--keyserver-disable`

Disables a key on a keyserver. Note that this command only works with the legacy PGP Keyserver product.

Requests for disabling a key must be signed. If no signer is supplied, the default signing key is used. Key disable requires an exact match on the key to be removed.

If a keyserver is specified on the command line, any keyserver listed in the PGP Command Line configuration file will not be used.

The usage format is:

```
pgp --keyserver-disable <input> [--keyserver <ks1> ...]
[|--signer <signer>] [|--passphrase <pass>] [options]
```
Where:

<input> is the user ID, portion of the user ID, or key ID of the key you want disabled on the keyserver. Key disable requires an exact match on the key to be disabled.

<ks> is the name of the keyserver where the key to be disabled is located.

You can enter more than one keyserver, separated by a space.

[options] let you modify the command. Options are:

--signer the user ID of the signer.

--passphrase the passphrase of the signer.

--keyserver-timeout sets the number of seconds until the keyserver operation times out. The default setting is 120 seconds.

--halt-on-error stops if an error occurs, if more than one keyserver is specified, or the operation stops.

Example:

```
pgp --keyserver-disable 0x12345678 --keyserver ldap://keyserver.example.com --signer "Alice Cameron <alice@example.com>" --passphrase "Bilbo*Baggins"
```

The specified key is disabled on the specified keyserver.

--keyserver-recv

Finds keys on a keyserver and imports them onto your keyring. Keyservers are searched in the order provided on the command line. As soon as a match is made on a keyserver, the operation will finish and all other keyservers on the list will be ignored.

If a keyserver is specified on the command line, any keyservers listed in the PGP Command Line configuration file will not be used. Preferred keyservers are not used. Note that you cannot search for disabled or pending keys.

The usage format is:

```
pgp --keyserver-recv <input> [<input2> ...] --keyserver <ks> [--keyserver <ks2> ...] [options]
```

Where:

<input> is the user ID, portion of the user ID, or key ID of the key you want to get onto your keyring.

To get a specific key, use the key ID. To get one or more keys, use the user ID or portion of the user ID.

<ks> is the name of the keyserver you want to search.

You can enter more than one keyserver to search, separated by a space. Only results from the first keyserver where there is a match will be returned.
[options] let you modify the command. Options are:

--keyserver-timeout sets the number of seconds until the keyserver operation times out. The default setting is 120 seconds.

--halt-on-error stops if an error occurs, if more than one keyserver is specified, or the operation stops.

For example:

`pgp --keyserver-recv 0xABCD1234 --keyserver ldap://keyserver.pgp.com`

The key with the key ID shown would be imported if it were on the specified keyserver.

`pgp --keyserver-recv Jim --keyserver http://keyserver.pgp.com`

All keys that have "Jim" in their user IDs would be found and imported.

--keyserver-remove

Removes a key from a keyserver. Note that this command only works with the legacy PGP Keyserver product.

Requests for removal must be signed. If no signer is supplied, the default signing key is used. Key removal requires an exact match on the key to be removed.

If a keyserver is specified on the command line, any keyservers listed in the PGP Command Line configuration file will not be used.

The usage format is:

```
pgp --keyserver-remove <input> [--keyserver <ks1> ...] [--signer <signer>] [--passphrase <pass>] [options]
```

Where:

<input> is the user ID, portion of the user ID, or key ID of the key you want removed from the keyserver. Key removal requires an exact match on the key to be removed.

<ks> is the name of the keyserver from which you want the key removed.

You can enter more than one keyserver, separated by a space.

[options] let you modify the command. Options are:

--signer the user ID of the signer.

--passphrase the passphrase of the signer.

--keyserver-timeout sets the number of seconds until the keyserver operation times out. The default setting is 120 seconds.

--halt-on-error stops if an error occurs, if more than one keyserver is specified, or the operation stops.
Example:

```
pgp --keyserver-remove 0x12345678 --keyserver ldap://keyserver.pgp.com --signer "bob@example.com" --passphrase "sm1t4"
```

Removes the specified key from the specified keyserver.

---

**--keyserver-search**

Searches a keyserver for keys and lists those that it finds that match the criteria; it does not import them.

Keyservers are searched in the order provided on the command line. As soon as a match is made on a keyserver, the operation finishes; all other keyservers in the list after the one that made the match will be ignored.

If a keyserver is specified on the command line, any keyservers listed in the PGP Command Line configuration file will not be used. Preferred keyservers are not used. You cannot search for disabled or pending keys.

The usage format is:

```
pgp --keyserver-search <input> [<input2> ...] --keyserver <ks> [--keyserver <ks2> ...] [options]
```

Where:

- **<input>** is the user ID, portion of the user ID, or key ID of the key for which you are searching.
  
  To find a specific key, use the key ID. To find one or more keys, use the user ID or portion of the user ID.

- **<ks>** is the name of the keyserver you want to search.
  
  You can enter more than one keyserver to search, separated by a space. Only results from the first keyserver where there is a match will be returned.

- **[options]** let you modify the command. Options are:

  - **--keyserver-timeout** sets the number of seconds until the keyserver operation times out. The default setting is 120 seconds.

  - **--halt-on-error** stops if an error occurs, if more than one keyserver is specified, or the operation stops.

Example:

```
p gp --keyserver-search example.com --keyserver ldap://keyserver.pgp.com
```

This search would return keys that have example.com in the user ID and are on keyserver.pgp.com, a public keyserver.
--keyserver-send

Posts a public key to a keyserver. If multiple keyservers are specified, in most cases only the first keyserver specified will be used. If a keyserver is specified on the command line, any keyservers listed in the PGP Command Line configuration file will not be used. Preferred keyservers are not used.

The usage format is:

```
pgp --keyserver-send <input> [input2 ...] --keyserver <ks>
    [--keyserver <ks2> ...] [options]
```

Where:

- `<input>` is the user ID, portion of the user ID, or key ID of the public key you are posting. You can list one or more users, with their names/IDs separated by a space.
- `<ks>` is the name of the keyserver to which you are posting.
- `[options]` let you modify the command. Options are:
  - `--keyserver-timeout` sets the number of seconds until the keyserver operation times out. The default setting is 120 seconds.
  - `--halt-on-error` moves to the next keyserver if an error occurs, if more than one keyserver is specified, or the operation stops.

Example:

```
pgp --keyserver-send alice@example.com --keyserver ldap://keyserver.example.com
```

If there are multiple keys on the keyring with user IDs that match the input, all of them will be posted. To make sure only a specific key is posted, use the key ID as the input.

```
pgp --keyserver-send 0x12345678 --keyserver ldap://keyserver.pgp.com
```

Only the specified key (if it is on the keyring) will be posted to ldap://keyserver.pgp.com, a public keyserver.
--keyserver-update

Updates keys that have already been uploaded to a keyserver. This ensures that the most up-to-date versions of the keys are on the keyserver.

An update consists of finding the key on the keyserver; merging that key onto the local keyring; and sending the merged key back to the keyserver on which it was found. A key must be on the local keyring to be updated.

If no keys are specified on the command line, all of the keys on the local keyring are updated, one at a time. When multiple keys are specified, they are updated one key at a time.

If a key has a preferred keyserver established, that keyserver is used for the update (only RSA and DH/DSS v4 keys can have a preferred keyserver); keyservers specified on the command line or in the configuration file are ignored. If the key being updated is not found, it is sent to the preferred keyserver; if it is found, it is updated.

If a key does not have a valid preferred keyserver established, PGP Command Line will search the keyserver specified on the command line, followed by keyservers specified in the configuration file. If the key cannot be found, an error is returned; if it is found, it is updated.

The usage format is:

```
pgp --keyserver-update <input> [ <input> ... ] [--keyserver <ks1> ... ] [options]
```

Where:

- `<input>` is the user ID, portion of the user ID, or key ID of the key for which you are searching. To find a specific key, use the key ID. To find one or more keys, use the user ID or portion of the user ID.

- `<ks>` is the name of the keyserver you want to search. You can enter more than one keyserver to search, separated by a space. Only results from the first keyserver where there is a match will be returned.

- `--keyserver-timeout` sets the number of seconds until the keyserver operation times out. The default setting is 120 seconds.

- `--halt-on-error` stops if an error occurs, if more than one keyserver is specified, or the operation stops.

Examples:

1. `pgp --keyserver-update 0x12345678 --keyserver ldap://keyserver.pgp.com`
   Updates the key with key ID 0x12345678 on keyserver.pgp.com if that key is on the local keyring and has already been uploaded to the keyserver. If either is not true, the operation returns with an error.

2. `pgp --keyserver-update 0x12345678`
   Key 0x12345678 has a preferred keyserver set, and that keyserver is used for the update.
Managing Keys
Descriptions and Examples of Key Commands

This chapter describes those commands used to manage keys with PGP Command Line. These commands are:

- **--add-adk**, which adds an ADK to a key (page 89).
- **--add-photoid**, which adds a photo ID to a key (page 90).
- **--add-preferred-cipher**, which adds the preferred cipher to a key (page 90).
- **--add-preferred-compression-algorithm**, which adds the preferred compression algorithms to a key (page 91).
- **--add-preferred-email-encoding**, which adds a preferred email encoding to a key (page 91).
- **--add-preferred-hash**, which adds the preferred hash encryption algorithm to a key (page 92).
- **--add-revoker**, which adds a revoker to a key (page 92).
- **--add-userid**, which adds a user ID to a key (page 93).
- **--cache-passphrase**, which specifically caches a passphrase (page 93).
- **--change-passphrase**, which changes the passphrase (page 95).
- **--clear-key-flag**, which clears one of the preferences flags (page 95).
- **--disable**, which disables a key (page 96).
- **--enable**, which enables a key (page 96).
- **--export** and **--export-key-pair**, which export keys or key pairs (page 97).
- **--export-photoid**, which exports a photo ID to a file (page 99).
- **--gen-key**, which generates a new key pair (page 100).
- **--gen-revocation**, which generates a revoked version of a key without actually revoking the key. The revoked version of the key is stored securely in the event the passphrase is lost, so the key can still be revoked (page 102).
- **--gen-subkey**, which generates a subkey (page 103).
- **--import**, which imports keys (page 104).
- **--join-key**, which reconstitutes a split key (page 104).
- **--join-key-cache-only**, which temporarily joins a key on the local machine (page 108).
- **--key-recon-send**, which sends PGP key reconstruction data to a PGP Universal Server (page 109).
- `--key-recon-recv-questions`, which retrieves the PGP key reconstruction questions for a specified key (page 110).
- `--key-recon-recv`, which reconstructs a key (page 111).
- `--remove`, which removes a key (page 112).
- `--remove-adk`, which removes an ADK from a key (page 112).
- `--remove-all-adks`, which remove all ADKs from a key (page 112).
- `--remove-all-photoids`, which removes all photo IDs (page 113).
- `--remove-all-revokers`, which removes all revokers (page 113).
- `--remove-expiration-date`, which removes the expiration date from a key (page 114).
- `--remove-key-pair`, which removes a key pair (page 114).
- `--remove-photoid`, which removes a photo ID from a key (page 114).
- `--remove-preferred-cipher`, which removes a preferred cipher from a key (page 115).
- `--remove-preferred-compression-algorithm`, which removes a preferred compression algorithm from a key (page 115).
- `--remove-preferred-email-encoding`, which removes a preferred email encoding from a key (page 116).
- `--remove-preferred-hash`, which removes the preferred hash from a key (page 116).
- `--remove-preferred-keyserver`, which removes a preferred keyserver from a key (page 117).
- `--remove-revoker`, which removes a revoker from a key (page 117).
- `--remove-sig`, which removes a signature (page 118).
- `--remove-subkey`, which removes a subkey (page 118).
- `--remove-userid`, which removes a user ID from a key (page 119).
- `--revoke`, which revokes a key pair (page 119).
- `--revoke-sig`, which revokes a signature (page 120).
- `--revoke-subkey`, which revokes a subkey (page 120).
- `--send-shares`, which sends shares to the server joining a key (page 121).
- `--set-expiration-date`, which sets the expiration date (page 121).
- `--set-key-flag`, which sets one of the preference flags for a key (page 122).
- `--set-preferred-ciphers`, which sets the list of preferred ciphers on a key (page 122).
--set-preferred-compression-algorithms, which sets the list of preferred compression algorithms on a key (page 123).

--set-preferred-email-encodings, which sets preferred email encodings for a key (page 124).

--set-preferred-hashes, which sets the entire list of hashes for a key (page 124).

--set-preferred-keyserver, which adds a preferred keyserver to a key (page 125).

--set-primary-userid, which sets a user ID as primary for a key (page 125).

--set-trust, which sets the trust on a key (page 126).

--sign-key, which signs all user IDs on a key (page 126).

--sign-userid, which signs a single user ID on a key (page 127).

--split-key, which splits a specified key into multiple shares (page 128).

Overview

The PGP keys that you create and those you obtain from others are stored in digital keyrings; private keys are stored on your private keyring in a file named secring.skr and public keys are stored on your public keyring in a file called pubring.pkr.

PGP Command Line provides great flexibility in what your keys can be used for. Commands that you can use to manage your keys are described in this chapter.

Commands

--add-adk

Adds an ADK to a key. Keys can support multiple ADKs, if desired.

An Additional Decryption Key (ADK) is a key that allows an authorized person, generally in an organization, to decrypt data this is from or was sent to someone in the organization if that person is unable or unwilling to do it themselves.

Only RSA and DH/DSS v4 keys can have ADKs.

The usage format is:

    pgp --add-adk <user> --adk <adk> --passphrase <pass>

Where:

    <user> is the user ID, portion of the user ID, or the key ID of the key to which the ADK is being added.
<adk> is the specific ADK to be added to the key.

<pass> is the passphrase of the key to which the ADK is being added.

Example:

```
pgp --add-adk "Bob Smith" --adk Alice --passphrase "smlt4"
0x6245273E:add ADK (0:ADKs successfully updated)
```

Adds the specified ADK to the specified key.

**--add-photoid**

Adds a photo ID to a key. You can add just one photo ID to a key using PGP Command Line. Other programs that are compatible with PGP Command Line support allow more than one photo ID added to a file; PGP Command Line can work with these extra photo IDs.

Only JPEG files can be added. For maximum picture quality, crop the picture to 120 by 144 pixels before adding it.

The usage format is:

```
pgp --add-photoid <user> --image <photo.jpg> --passphrase <pass>
```

Where:

**<user>** is the user ID, portion of the user ID, or the key ID of the key to which the photo ID is being added.

**<photo.jpg>** is the filename of the image being added.

**<pass>** is the passphrase of the key to which the photo ID is being added.

Example:

```
pgp --add-photoid Alice --image alice.jpg --passphrase "cam3r0n" 0x3E439B98:add photo ID (0:photo ID added successfully)
```

Adds the image alice.jpg to the specified key.

**--add-preferred-cipher**

Adds a preferred cipher to a key.

If the preferred cipher is already on the key, it is moved to the top of the list. Only RSA v4 and DH/DSS v4 keys can have a preferred cipher.

The usage format is:

```
pgp --add-preferred-cipher <user> --cipher <cipher> --passphrase <pass>
```

Where:
<user> is the user ID, portion of the user ID, or the key ID of the key to which the preferred cipher is being added.

<cipher> is the preferred cipher being added.

<pass> is the passphrase of the key.

Example:

```
pgp --add-preferred-cipher "Bob Smith" --cipher aes256
   --passphrase "sm1t4"
0x6245273E: add preferred cipher (0: preferred ciphers updated)
```

Adds the cipher AES256 to the specified key.

--add-preferred-compression-algorithm

Adds a preferred compression algorithm to a key.

If the preferred compression algorithm is already on the key, it is moved to the top of the list. Only RSA v4 and DH/DSS v4 keys can have a preferred compression algorithm.

The usage format is:

```
pgp --add-preferred-compression-algorithm <user>  
   --compression-algorithm <algo> --passphrase <pass>
```

Where:

<user> is the user ID, portion of the user ID, or the key ID of the key to which the preferred compression algorithm is being added.

<algo> is the preferred compression algorithm being added.

<pass> is the passphrase of the key.

Example:

```
pgp --add-preferred-compression-algorithm "bob@example.com"  
   --compression-algorithm bzip2 --passphrase "sm1t4"
0x6245273E: add preferred compression algorithm (0: preferred compression algorithms updated)
```

Adds the compression algorithm Bzip2 to the specified key.

--add-preferred-email-encoding

Adds a preferred email encoding to a key.

If the preferred email encoding is already on the key, it is moved to the top of the list. Only RSA v4 and DH/DSS v4 keys can have a preferred email encoding.

The usage format is:

```
pkp --add-preferred-email-encoding <user>  
   --email-encoding <encoding> --passphrase <pass>
```

Where:
**--add-preferred-email-encoding**

Adds the preferred email encoding to a key. The usage format is:

```
pgp --add-preferred-email-encoding <user> --email-encoding <encoding> --passphrase <pass>
```

Where:

- `<user>` is the user ID, portion of the user ID, or the key ID of the key to which the preferred email encoding is being added.
- `<encoding>` is the preferred email-encoding being added.
- `<pass>` is the passphrase of the key.

Example:

```
pgp --add-preferred-email-encoding "Bob Smith" --email-encoding pgpmime --passphrase "sm1t4"
```

Adds the email encoding pgpmime to the specified key.

**--add-preferred-hash**

Adds the preferred hash encryption algorithm to a key and lists it on the top of the hash list. The usage format is:

```
pgp --add-preferred-hash <user> --hash <hash> --passphrase <pass>
```

Where:

- `<user>` is the user ID, portion of the user ID, or the key ID of the key to which the preferred hash is being added.
- `<hash>` is the preferred hash being added to a key. You can add several preferred hashes to a key, one at a time. The newly added preferred hash will appear on top of the hash list.
- `<pass>` is the passphrase of the key.

Example:

```
pgp --add-preferred-hash "Bob Smith" --hash sha512 --passphrase "sm1t4"
```

Adds the preferred hash SHA-512 and displays it on top of the hash list.

**Hash: SHA-512**

**--add-revoker**

Adds a revoker to a key. It is possible that you might forget your passphrase or lose your private key, which would mean that you could never use it again and you would have no way of revoking it. To safeguard against this latter possibility, you can add a key to your keyring as a revoker, which could be used to revoke your key if you could not do it.

Only RSA and DH/DSS v4 keys can have revokers.

The usage format is:

```
pgp --add-revoker <user> --revoker <revoker> --passphrase <pass>
```

```
pgp --add-revoker "Bob Smith" --revoker "Anita Jones" --passphrase "passphrase123"
```

This adds a revoker to the key for Bob Smith.
Where:

- `<user>` is the user ID, portion of the user ID, or the key ID of the key to which the revoker is being added.
- `<revoker>` is the specific revoker to be added to the key.
- `<pass>` is the passphrase of the key to which the revoker is being added.

Example:

```bash
pgp --add-revoker "Bob Smith" --revoker Alice --passphrase "sm1t4"
```

0x6245273E:add revoker (0:revokers successfully updated)

Adds the specified revoker to the specified key:

Revoker: 0x3E439B98 (0xA9B1D273E439B98)
User ID: Alice Cameron <alice@example.com>

--add-userid

Adds a user ID to a key. You can add as many user IDs as you want to a key. To add a photo ID, use --add-photoid.

The usage format is:

```bash
pgp --add-userid <user> --user <newID> --passphrase <pass>
```

Where:

- `<user>` is the user ID, portion of the user ID, or the key ID of the key to which the user ID is being added.
- `<newID>` is the user ID being added to the key.
- `<pass>` is the passphrase of the key to which the user ID is being added.

Example:

```bash
pgp --add-userid "bob@example.com" --user Alice --passphrase "sm1t4"
```

Adds the specified user ID to the specified key.

--cache-passphrase

Caches the passphrase for a key for the current session. Caching your passphrase can save you time in that you do not have to enter it for those operations that require it. Passphrase caching must be enabled (using the option --passphrase-cache) for this command to work.

Make sure to log out at the end of your session (which purges the passphrase cache) or purge the passphrase cache manually using the command --purge-passphrase-cache.

The number of cached passphrases can be checked with --version in verbose mode.
The usage format is:

```plaintext
pgp --cache-passphrase <user> --passphrase <pass> [options]
```

Where:

- `<user>` is the user ID, portion of the user ID, or the key ID of the key whose passphrase is being cached.

- `<pass>` is the passphrase of the key.

- `[options]` change the behavior of the command. Options are:

  - `--passphrase-cache` enables passphrase caching. This is optional, since you can enable passphrase caching by changing the passphrase cache settings in the configuration file `PGPprefs.xml` from `false` to `true`.

  - `--passphrase-cache-timeout` sets the amount of time a passphrase can be cached, in seconds. The default is 120. If you enter 0 (zero), the passphrase cache will not timeout; it must be specifically purged.

Examples:

1. `pgp --cache-passphrase "Bob Smith" --passphrase "smit4" --passphrase-cache`
   
   ```plaintext
   0x6245273E:cache passphrase (0:key passphrase cached)
   ```
   
   Caches the passphrase of the specified key. Since no timeout is specified, the default of 120 seconds will be used.

2. `pgp --cache-passphrase "Bob Smith" --passphrase "smit4" --passphrase-cache --passphrase-cache-timeout 0`
   
   ```plaintext
   0x6245273E:cache passphrase (0:key passphrase cached)
   ```
   
   Caches the passphrase of the specified key and establishes a timeout of 0, which means the passphrase cache must be specifically purged to remove the passphrase from memory.
--change-passphrase
Changes the passphrase for a key and all subkeys (if the key has any).

The usage format is:

```
pgp --change-passphrase <user> --new-passphrase <newpass>
[--passphrase <oldpass>]
```

Where:

- `<user>` is the user ID, portion of the user ID, or the key ID of the key whose passphrase is being changed.
- `<newpass>` is the new passphrase of the key.
- `<oldpass>` is the old passphrase of the key. It is not needed if the key has no passphrase.

Example:

```
pgp --change-passphrase "Bob Smith" --passphrase "sm1t4"
--new-passphrase "b0bsm1t4"
```

0x6245273E: change passphrase (3135: master passphrase changed)

```
0x894BA6DC: change passphrase (3136: subkey passphrase changed)
0x6245273E: change passphrase (0: key passphrase changed)
```

Replaces the old passphrase `sm1t4` with the new passphrase `b0bsm1t4` for the specified key and its subkey.

--clear-key-flag
Clears one of the key’s preferences flags.

The usage format is:

```
pgp --clear-key-flag <user> [--subkey <subkeyID>] --key-flag <flag> [--passphrase <pass>]
```

Where:

- `<user>` is the user ID, portion of the user ID, or the key ID of the user whose key preferences flag is being cleared.
- `<flag>` is the key preferences flag to be cleared. See `--key-flag` for more details.
- `<subkeyID>` is the subkey ID of the key whose key preferences flag is being cleared.
- `<pass>` is the passphrase of the key for which the preferences flag is being cleared.

Example:

```
pgp --clear-key-flag Bob --key-flag encrypt --passphrase "sm1t4"
```

Clear the key preference flag "encrypt" from Bob’s key.
--disable

Disables a key or keypair.

Disabling a key or key pair prevents it from being used without deleting it. Note that you cannot disable an axiomatic key.

The usage format is:

```
pgp --disable <user>
```

Where:

- `<user>` is the user ID, portion of the user ID, or the key ID of the key being disabled.

Examples:

```
1  pgp --disable "Jose Medina"
   0xF6EFC4D9:disable key (3067: key is axiomatic)
   You cannot disable Jose's key since it is axiomatic.

2  pgp --disable "Maria Fuentes"
   0x136259CB:disable key (0: key successfully disabled)
   Maria's public key is disabled.
```

--enable

Enables a key or keypair that has been disabled.

Once enabled, you can use the key or keypair again.

The usage format is:

```
pgp --enable <user>
```

Where:

- `<user>` is the user ID, portion of the user ID, or the key ID of the key being enabled.

Example:

```
pgp --enable "Maria Fuentes"
   0x136259CB:enable key (0: key successfully enabled)
   Maria's public key is enabled.
```
**--export, --export-key-pair**

Exports keys or key pairs. You will export a key so that you can send a public key to your correspondents and/or to a public keyserver. Keys are exported as ASCII armor files (`.asc`), or in other supported export formats. Note that when you are exporting a keypair, the operation will be successful when there is only one key pair that contains the string you specify as input (see examples).

At least one key must be specified for export. Keys are exported as ASCII armor (.asc) files into the current directory. Keys can also be exported in other formats; refer to “Export Format” on page 98 for detailed information.

The command **--export** exports only public keys, while the command **--export-key-pair** exports private keys.

The usage format is:

```
pgp --export/--export-key-pair <input> [options]
```

Where:

- `<input>` is the user ID, portion of the user ID, or the key ID of the key you want to export.
- `[options]` change the behavior of the command. Options are:
  - **--output** lets you specify a different name for the exported file.
  - **--export-format** lets you specify an export format from the following list of supported formats. For more information, refer to **--export-format**.
  - **--cert**. This option is the X.509 issuer long name or the 32-bit or 64-bit key ID, if the signing key is available.
  - **--export-passphrase** specifies the passphrase to use when exporting PKCS8 and PKCS12 data. If only **--export-passphrase** is supplied, PGP Command Line does the following depending on the used argument:
    - **valid**. Exports with the export passphrase.
    - **invalid**. Gives an error.
  - **--passphrase** belongs to the key that has a certificate. If only **--passphrase** is supplied, PGP Command Line does the following depending on the used argument:
    - **valid**. Exports the key with no passphrase.
    - **invalid**. Gives an error.

To specify no passphrase, use the empty string "".

Examples:

1. **pgp --export Bob**
   
   `0x6245273E:export key (0:key exported to Bob Smith.asc)`
   
   `0xF6F83318:export key (0:key exported to Bob Reynolds.asc)`
   
   All public keys that contain the string "Bob" were exported.
2  `pgp --export-key-pair "bob@example.com"`
0x6245273E:export key pair (0:key exported to Bob Smith.asc)
Bob’s key pair was exported to the ASCII-armored file "Bob Smith.asc".

3  `pgp --export-key-pair Bob`
Bob:export key pair (2003:too many matches for key to edit)
The operation cannot be completed because there is more than one key pair that
contains the string: "Bob".

4  `pgp --export-key-pair Medina`
0xF6EFC4D9:export key pair (0:key exported to Jose Medina.asc)
This operation was successful because there is only one key pair with the string
'Medina'.

Export Format
PGP Command Line supports several export formats:

- **Complete** (default): Only ASCII-armored files are output; the default file extension is .asc. Use `Complete` to export keys in a newer format that supports all PGP features.

- **Compatible**: Only ASCII-armored files are output; the default file extension is .asc. Use `Compatible` to export keys in a format compatible with older versions of PGP software; that is, PGP software versions 7.0 and prior. Some newer PGP features are not supported when using Compatible.

- **X.509-cert**: Only ASCII-armored files are output; the default file extension is .crt. The <input> must match exactly one key, and `--cert` is required.

- **PKCS8**: Only ASCII-armored files are output; the default file extension is .p8. A signed key must be paired. The <input> must match exactly one key, `--cert` is required as well as `--passphrase`.

  The passphrase options change the passphrase of the exported key and certificate. They do not change the passphrase of the local key.

  - If only `--passphrase` is supplied, and the passphrase is valid, the key/certificate is exported with no passphrase. If the supplied passphrase is invalid, an error is generated.

  - If only `--export-passphrase` is supplied, and the passphrase is valid, the key/certificate is exported with the export passphrase. If the supplied passphrase is invalid, an error is generated.

  - If no `--passphrase` is supplied, the cache and an empty passphrase is tried.

- **PKCS12**: Only binary blocks are output; the default file extension is .p12. A signed key must be paired. The <input> must match exactly one key, `--cert` is required as well as `--passphrase`.

  The passphrase options change the passphrase of the exported key and certificate. They do not change the passphrase of the local key.
— If only `--passphrase` is supplied, and the passphrase is valid, the key is exported with no passphrase. If the supplied passphrase is invalid, an error is generated.

— If only `--export-passphrase` is supplied, and the passphrase is valid, the key is exported with the export passphrase. If the supplied passphrase is invalid, an error is generated.

— If no passphrase is supplied, the cache and an empty passphrase is tried.

- **Certificate signature request (CSR):** Only ASCII-armored blocks are output. The default file extension is `.csr`. Key must be paired. The input must match exactly one key.

  Example:

  ```
  pgp --export "Bob Smith" --export-format pkcs12 --passphrase "sm1t4" --cert 0x6245273E
  0x6245273E:export key (0: key exported to Bob Smith.p12)
  ```

  Bob’s key pair is exported to a file "Bob Smith.p12".

---

**--export-photoid**

Exports a photo ID from a key to a file. There must be a photo ID on the key for it to be exported. Only JPEG files are supported. Resulting files are saved to the current directory.

The usage format is:

```
pgp --export-photoid <user> [options]
```

Where:

- `<user>` is the user ID, portion of the user ID, or the key ID of the key from which the photo ID is being exported.

- `[options]` change the behavior of a command. Options are:

  --index specifies which photo ID on the key should be exported. 1 indicates the first photo ID, 2 the second photo, and so on.

  --output is a desired filename.

Examples:

1. `pgp --export-photoid "Alice C"`
   Exports the photo ID to filename "alice c.jpg".

2. `pgp --export-photoid "Alice C" --output photoid.jpg`
   Exports the photo ID to filename 'photoid.jpg'.

3. `pgp --export-photoid "Alice C" --index 2`
   Exports the second photo ID on the key to filename "alice c.jpg".
--gen-key

Creates a new key. It also creates a keyring pair if no keyrings exist.

The usage format is:

```
pgp --gen-key <user> --key-type <type> --encryption-bits <bits> --passphrase <pass> [--signing-bits <bits>] [options]
```

Where:

- `<user>` is the user for whom the key is being generated. A common user ID is your name and email address in the format: "Alice Cameron <alice@example.com>". If your user ID contains spaces, you must enclose it in quotation marks.

- `<type>` is the key type: rsa, rsa-legacy, rsa-sign-only, dh, or dh-sign-only.

- `--encryption-bits` is the length of the encryption subkey in bits (1024 - 4096). When generating sign-only keys (keys without a subkey), you can specify `--bits` only to define the signing key size.

- `<pass>` is a passphrase of your choice. This flag is not optional: to generate a key without a passphrase, use `--passphrase ""`.

- `--signing-bits` defines the length of the signing key in bits. The valid sizes in bits for signing keys are as follows: for RSA legacy 1024 to 2048 bits; for RSA v4 1024 to 4096 bits; and for DH the size is only 1024 bits. For RSA v4 keys, this option can be set independently from `--bits`.

- `[options]` modify the behavior of the command. Options are:
  - `--adk` specifies an ADK (Additional Decryption Key). See `--adk` for more information.
  - `--compression-algorithm` sets the compression algorithm. Note that this option does not work with public-key encryption, because in this case the recipient’s key preferences are used. The default for this option is `zip`. See `--compression-algorithm` for more information.
  - `--creation-date` changes the date of creation. The format is `yyyy-mm-dd` and it cannot be used together with `--creation-days`. Month and day do not have to be two digits if the first digit is zero.
  - `--creation-days` changes the number of days until creation ("1" equals next day, "2" equals day after next, etc.)
  - `--expiration-date` changes the date of expiration. The format is `yyyy-mm-dd`. This option cannot be used at the same time as `--expiration-days`. Month and day do not have to be two digits if the first digit is zero.
  - `--expiration-days` changes the number of days until expiration. The default is not set (no expiration).
  - `--fast-key-gen` enables fast key generation. The default is `on`.
  - `--preferred-keyserver` specifies a preferred keyserver. The keyserver must have the correct prefix: `http://`, `ldap://`, `ldaps://`, or `hkp://`. 
--revoker specifies a revoker for a key. See --revoker for more information.

Any cipher lets you specify which ciphers can be used with the key being generated; see “--set-preferred-ciphers” on page 123 for more information.

Any compression algorithm lets you specify which compression algorithms can be used with the key being generated; see “--set-preferred-compression-algorithms” on page 123 for more information.

Any preferred hash lets you specify which hashes can be used with the key being generated; see “--set-preferred-hashes” on page 124 for more information.

Any preferred email encoding lets you specify which email encodings can be used with the key being generated; see “--set-preferred-email-encodings” on page 124 for more information.

Examples:

1. `pgp --gen-key "Alice Cameron <alice@example.com>" --key-type rsa --encryption-bits 2048 --signing-bits 2048 --passphrase "cam3r0n" --expiration-date 2007-06-01`
   Creates a key pair for Alice with the expiration date June 1, 2007.

2. `pgp --gen-key "Fumiko Asako <fumiko@example.com>"
   --encryption-bits 2048 --signing-bits 2048 --key-type rsa
   --passphrase "asak0" --preferred-keyserver "ldap://keys.example.com"
   Creates a key pair for Fumiko with the preferred keyserver "ldap://keys.example.com".

3. `pgp --gen-key ... --aes256 1 --3des 2 --preferred-keyserver ldap://aes.pgp.com`
   Creates a key pair with aes256 as the preferred cipher and 3des as the secondary cipher.

**Key Types**

PGP Command Line gives you several key types to choose from: RSA, RSA-legacy, RSA-sign-only, DH, DH-sign-only. Each is described below:

- **RSA.** RSA v4 keys support all PGP key features, such as ADKs, designated revoker, preferred ciphers, multiple encryption subkeys, or photo IDs. Their size is 1024 bits to 4096 bits.

- **RSA-legacy.** This is a RSA v3 (legacy) key, for which either --bits or --signing-bits can be supplied. These keys are used only for communicating with people who are using older versions of PGP applications. Note that RSA v4 and RSA v3 (legacy) keys are not compatible. Unlike v4 keys, v3 keys do not support many features such as ADKs, designated revoker, multiple encryption subkeys, or photo IDs. RSA v3 keys can have a length of maximum 2048 bits.
- **RSA-sign-only.** These are RSA v4 keys with no automatically generated subkey. You can generate a subkey for this key later by using `--gen-subkey`. Like any other v4 keys, they support all PGP key features, such as ADKs, designated revoker, preferred ciphers, and so on.

- **DH.** Diffie-Hellman (DH/DSA) signing keys can only be 1024 bits long. Their subkeys (the encryption keys) can be longer; therefore, specifying longer bit sizes for this key type only affects the subkey size. Version 4 keys support all PGP key features, such as ADKs, designated revoker, preferred ciphers. This is a DH/DSA key with no automatically generated subkey. Since only the signing key is generated, the size cannot be larger than 1024 bits: if you enter a larger size, the key will not be generated. Version 4 keys support all PGP key features, such as ADKs, designated revoker, preferred ciphers, and so on.

- **DH-sign-only.** This is a DH/DSS key without an encryption subkey. Maximum size is 1024 bits.

**--gen-revocation**

Generates a revocation certificate for a key, but it doesn’t revoke the key on the key ring. By default, the revocation certificate is exported as if you have used the command `--export`.

The usage format is:

```
pgp --gen-revocation <user> --passphrase <pass> --force [--revoker <revoker>][--output <output>]
```

Where:

- `<user>` is the user ID, portion of the user ID, or the key ID of the key being revoked.
- `<pass>` is the passphrase of the key being revoked.
- `--force` is required to revoke a key.
- `<revoker>` is the user ID, portion of the user ID, or the key ID of the designated revoker key. When this option is used, the passphrase belongs to the revoker key. This option is not needed if you use a designated revoker or if you are doing self revocation.
- `<output>` is used to change the location of the exported certificate.

Example:

```
pgp --gen-revocation "Jose Medina" --passphrase "medina" --force
0xF6EFC4D9:generate revocation (0:key exported to Jose Medina.asc)
0xF6EFC4D9:generate revocation (2094:this key has NOT been permanently revoked)
```

Generates the revocation certificate "Jose Medina.asc".
--gen-subkey

Generates a subkey on an existing key. The key must be allowed to have subkeys or this operation fails. The subkey is always of the same type as the key to which it is being added.

The usage format is:

```
pgp --gen-subkey <user> --bits <bits> --passphrase <pass> [options]
```

Where:

- `<user>` is the user ID, portion of the user ID, or key ID of the key that is getting the subkey.
- `<bits>` specifies the length of the encryption subkey in bits. Values are 1024 to 4096.
- `<pass>` is the passphrase of the key that is getting a subkey.
- `[options]` change the behavior of the command. Options are:
  - `--creation-date` specifies the date on which the key becomes valid. You cannot use `--creation-date` and `--creation-days` for the same operation.
  - `--creation-days` specifies the number of days until creation.
  - `--expiration-date` specifies the date the key expires. You cannot use `--expiration-date` and `--expiration-days` in one operation.
  - `--expiration-days` specifies the number of days until expiration.

Example:

```
pgp --gen-subkey "bob@example.com" --bits 2048 --passphrase "b0bsmt4"

0x3D58AE31: generate subkey (0:subkey successfully generated)
```

Generates a subkey of the specified number of bits on Bob’s key:

```
Subkey ID: 0x3D58AE31 (0xAEE6484D3D58AE31)
  Type: RSA (v4)
  Size: 2048
  Created: 2005-11-18
  Expires: Never
  Status: Active
  Revocable: Yes
  Prop Flags: Encrypt communications
  Prop Flags: Encrypt storage
```
--import

Imports keys to the local keyring.

The file containing the key(s) to be imported should be in the current directory, or you must specify the fully qualified path to the file containing the keys. Note that both private and public keys will be imported, if they exist in the file. If a key being imported already exists in the local keyring, the keys are merged.

The usage format is:

```
pgp --import <input> [<input2> ...] [options]
```

Where:

- `<input>` is the filename of the key being imported. Multiple keys can also be imported by listing them, separated by a space.
- `[options]` modify the behavior of the command. Options are:
  - `--import-format` specifies the import format for the current operation. See `--import-format` for more information.
  - `--manual-import-keys` changes the behavior of PGP Command Line when keys are found during import operations. The default is all.
  - `--manual-import-key-pairs` changes the behavior of PGP Command Line when key pairs are found during an import operation.
  - `--passphrase` is the passphrase of the key being imported.

Example:

```
pgp --import "Bob Smith.asc"
```

```
Bob Smith.asc:import key (0:key imported as 0x6245273E Bob Smith <bob@example.com>)
```

Imports Bob’s key ‘Bob Smith.asc’.

--join-key

This command joins the shares of a key that was previously split.

The minimum number of share files must be on the computer where the key is being joined. The passphrase cache must be enabled for this command to work with public keys that have passphrases; no passphrase caching is required for public keys with no passphrases.

Since PGP Command Line currently cannot cache symmetrical passphrases, you need to enter all necessary symmetrical passphrases onto the command line during key joining. The symmetrical passphrases are added together with corresponding share files onto the command line.

You can also turn on automatic passphrase caching by changing the value for `CLpassphraseCache` from `false/` to `true/` in the preference file `PGPprefs.xml`, which is located in your Data directory.
Following is an overview of how PGP Command Line handles key joining:

- Local shares are always assembled before PGP Command Line begins listening on the network for remote shares.
- If the local shares are based on keys with passphrases, the passphrases must be cached.
- If the local shares are conventionally encrypted, the passphrase must be supplied on the command line.
- If there are enough local shares for reconstruction of the key, PGP Command Line does not listen on the network for remote shares.

If you are experiencing problems with your local shares, perform the `--join-key` command without `--force`; PGP Command Line will return all of the information about each local file share that it has found, including whether or not the passphrases are correct. If you find problems without `--force`, fix them. Once all problems with the local shares are fixed, add `--force` and `--skep` to have PGP Command Line listen on the network for remote shares after collecting the local shares.

The usage format is:

```
pgp --join-key <user> --passphrase <new pass> --share <share1> --share <share2> [--share <shareN> ...] [--force] [options]
```

Where:

- `<user>` is the user ID, portion of the user ID, or the key ID of the key you want to join. You must make an exact match, as you can only join one key at a time.
- `<new pass>` This is the passphrase of the newly joined key. It is given to the new key after the threshold requirement is removed: there were enough shares put together for the key to be joined.
- `<share1> <share2>` are share files given to a specific user when the key was split. When you join the key using these shares, you need to reach the threshold: the minimum number of shares needed for joining operation to succeed.

You need to supply the symmetrical passphrases incorporated with the shares for any share users who have such passphrases.

The share file format for users with symmetric passphrases (that cannot be cached for this operation) is as follows:

```
--share "<share user>-2-<split key ID>.shf:<share user's symmetric passphrase>" --share "Alice Cameron-2-Jill Johnson.shf:jill"
```

The share file format for users with asymmetric passphrases (that must be cached for this operation) is as follows:

```
--share "<share user>-1-<split key ID>.shf" --share "Alice Cameron-1-Bob Smith.shf"
```
**--force**. If you run the **--join** command without the **--force** option, PGP Command Line will not join the key: it will only list the state of the shares in the preview mode. The output will not be displayed if there are parse errors, or if a key is missing or unable to decrypt.

The key shares preview will report if there are enough shares to join the key and if there are invalid (or not cached) passphrases.

**--skep**. PGP Command Line uses this option when joining split keys over the network. It looks for split files on the network and if it doesn’t find enough of them, it continues to listen using the timeout defined by the option **--skep-timeout**.

**--skep-timeout** changes the timeout for joining keys over the network. There is no value reserved to indicate no timeout. Default is 120 seconds

**-v|--verbose** will give a detailed overview of the operation.

Examples:

1. In this example, the original key was split in 50 shares with a threshold of 40. Therefore, you need only 40 shares in order to join the key: you can take shares from two share users who together have 40 shares.

   In order to join a key, you need first to cache passphrases of the users whose shares you are joining:

   ```bash
   pgp --cache-passphrase "Bob Smith" --passphrase "sm1t4" --passphrase-cache 0x2B65A65E:cache passphrase
   (0:key passphrase cached)
   ```

   You will enter the symmetrical passphrase together with the shares onto the command line (Jill’s passphrase in this example):

   ```bash
   pgp --join-key "Alice Cameron" --passphrase "testkey" --share "Alice Cameron-1-Bob Smith.shf" --share "Alice Cameron-2-Jill Johnson.shf:ji11"
   ```

2. 

   ```bash
   pgp --join-key "Alice Cameron" --passphrase "testkey" --share "Alice Cameron-1-Bob Smith.shf" --share "Alice Cameron-2-Jill Johnson.shf:ji11" --force --skep --skep-timeout 300
   ```

   Tells the key joining operation to wait 5 minutes before it times out.

**Command output for **--join-key**

**Row 1: Split Key User Name**
Name: “Split Key User”
Value: Primary user ID of the key being split, in this case “Alice Cameron”.

**Row 2: Split Key ID**
Name: “Split Key ID”
Value: The 32-bit key ID followed by the 64-bit key ID in the format:

0xEB778BFA (0xEF20715FEB778BFA)
Row 3: Empty

Row 4: Threshold
Name: "Threshold"
Value: This is the threshold for the key being split (minimum number of shares to put the key back together).

If threshold cannot be determined when joining a key, the character ‘?’ is displayed. This can happen when PGP Command Line displays this information before it listens for network shares.

Row 5: Total Shares
Name: “Total Shares’
Value: Join. This is the number of shares being collected from the file shares.

Row 6: Total Users
Name: “Total Users”
Value: Join. This is the total number of users from whom PGP Command Line has collected file shares. When joining a key using --skep, network shares will not show here because they are collected after this information is displayed.

Row 7: Empty

Row 8-N: Share User
Name: Share User
Value: The parsed value of each share in the following format:

```
Share User: 20 0xEB910E083 Bob Smith
```
- Number of shares assigned to a specific user (3 characters, left justified).
- Key ID of the share recipient. For public key encryption, this is a key ID in standard format, while for symmetric encryption, this is the string "symmetric".
- The name of the share recipient. For public key encryption, this is the primary user ID string; for symmetric encryption, this is the name provided in the --share option.

If there are no share users specified, "N/A" is displayed. This can only happen when joining a key with the --skep option enabled.

```
pgp --join-key "Alice Cameron" --passphrase "testkey" --share "Alice Cameron-1-Bob Smith.shf" --share "Alice Cameron-2-Jill Johnson.shf:jill" --force
```

The key is joined:

```
0xEB778BFA:join key (3134:reconstructed split key passphrase is valid)
0xEB778BFA:join key (0: key joined successfully)
```
**--join-key-cache-only**

Use this command to temporarily join a key on the local machine. After the key is joined, it is not saved to the disk: instead, the key remains split and the newly joined key is cached for later use.

The passphrase cache must be enabled for this command to work with public keys that have passphrases; no passphrase caching is required for public keys with no passphrases.

The usage format is:

```
pgp --join-key-cache-only <user> --share <share1> --share <share2> [--share <shareN> ...] --force [-v|--verbose][--skep]
```

Where:

- `<user>` is the user ID, portion of the user ID, or the key ID of the key being joined.
- `<share1>` and `<share2>` are the share files given to specific users when the key was split. When you join the key using these shares, you need to reach the threshold: the minimum number of shares needed for joining operation to succeed. The minimum number of shares is two.

For more information, refer to the command **--join-key**.

**--force.** If you run the **--join-key-cache-only** command without this option, PGP Command Line will not join the key: it will only list the state of the shares in the preview mode. The output will not be displayed if there are parse errors, if a key is missing, or PGP Command Line was unable to decrypt.

The key shares preview will report if there are enough shares to join the key, and if there are invalid (or not cached) passphrases.

**-v|--verbose.** This option will give a detailed overview of the operation.

**--skep.** PGP Command Line uses this option when joining split keys: it looks for split files on the network. If it doesn’t find enough of split files, it will continue to listen on the network using the timeout defined by the option **--skep-timeout**.

Example:

Before you run **--join-key-cache-only**, refer to **--passphrase-cache** for more explanation on enabling passphrase caching.

```
pgp --join-key-cache-only "Alice Cameron" --passphrase "newkey" --share "Alice Cameron-1-Alice Cameron.shf:brapa1" --share "Alice Cameron-2-Jose Medina.shf:medina" --force
```

Split Key User: Alice Cameron

Split Key ID: 0xB910E083 (0xBCC87BD2B910E083)

Threshold: 20

Total Shares: 20

Total Users: 2

Share User: 10 symmetric Alice Cameron
Share User: 10 symmetric Jose Medina

0xB910E083:join key cache only (3134:reconstructed split key passphrase is valid)

0xB910E083:join key cache only (0:key passphrase cached)

After the key is joined, it is not saved to the disk: instead, the key remains split and the passkey is cached for later use.

**--key-recon-send**

Sends PGP key reconstruction data to a PGP Universal Server.

Key reconstruction works with PGP Universal Version 2.0 or greater (it is not supported by Version 1.x PGP Universal, nor does it work with PGP Keyserver Version 7.0).

Key reconstruction lets you store your private key and passphrase so that only you can retrieve it. It is a safety net in case you lose your private key or its passphrase.

Key reconstruction requires a PGP Universal Server that is getting user data from an account on an Active Directory server. If no reconstruction server is specified, the preferred server on the key will be used.

When setting up key reconstruction, you create five questions and answers. To reconstruct the key, you must answer three or more of the five questions correctly (the threshold of three correct answers is not configurable).

The usage format is:

```
pgp --key-recon-send <key> [--question <q1> ... --question <q5>] [--answer <a1> ... --answer <a5>] --passphrase <pass> --auth-username <auth user> --auth-passphrase <auth pass> [--recon-server <recon server>]
```

Where:

- `<key>` is the user ID, portion of the user ID, or the key ID of the key whose reconstruction data you want to send to a PGP Universal Server.
- `<q1>` is a first of five questions that only you can answer.
- `<a1>` is the answer to the first question. Answers must be at least six characters long.
- `<pass>` is the passphrase to your private key.
- `<auth user>` is your username on an Active Directory server. This username will be authenticated by the PGP Universal Server.
- `<auth pass>` is your passphrase on an Active Directory server. This passphrase will be authenticated by the PGP Universal Server.
- `<recon server>` is the PGP Universal Server on which your key reconstruction information is stored.
Examples:

```
pgp --key-recon-send 0xEB778BFA --question "First question?"  
--answer "First answer" ... --auth-username myuser  
--auth-passphrase "mypass"
```

The specified key (0xEB778BFA) is sent to the preferred server on the key  
accompanied by the five questions and answers and the authorization username and  
passphrase for the Active Directory server.

```
pgp --key-recon-send 0xEB778BFA --question "First question?"  
--answer "First answer" ... --question "Fifth question?"  
--answer "Fifth answer" --auth-username myuser  
--auth-passphrase "mypass" --recon-server 10.1.1.45
```

The specified key (0xEB778BFA) is sent to the PGP Universal Server with IP address  
of 10.1.1.45 accompanied by the five questions and answers and the authorization  
username and passphrase for the Active Directory server.

```
--key-recon-recv-questions
```

Retrieves PGP key reconstruction questions for a specified key.

In order to be retrieved, the key reconstruction questions must already reside on the PGP  
Universal Server.

PGP Command Line responds to a successful request in the following format:

```
User ID: <user>  
Key ID: <keyID>  
Question 1: <question1>  
...  
Question 5: <question5>
```

Where:

- `<user>` is the user ID of the key being reconstructed.
- `<keyID>` is key ID of the key being reconstructed.
- `<question1>` is the first of the five stored questions, `<question2>` is the second of  
The five stored questions, and so on through `<question5>`, the last of the second of  
The five stored questions.

The usage format is:

```
pgp --key-recon-recv-questions <key> --auth-username <auth  
user> --auth-passphrase <auth pass> [--recon-server <recon  
server>]
```

Where:

- `<key>` is the user ID, portion of the user ID, or the key ID of the key whose  
reconstruction data you want to send to a PGP Universal Server.
- `<auth user>` is your username on an Active Directory server. This username will  
be authenticated by the PGP Universal Server.
<auth pass> is your passphrase on an Active Directory server. This passphrase will be authenticated by the PGP Universal Server.

<recon server> is the PGP Universal Server on which your key reconstruction information is stored.

Example:

```
pgp --key-recon-recv-questions 0x3D58AE31 --auth-username myuser --auth-passphrase mypass --recon-server 10.1.1.45
```

The PGP key reconstruction questions for the specified key (0x3D58AE31) are retrieved from the specified PGP Universal Server.

---

**--key-recon-recv**

Reconstructs a private key locally, on successful completion of the five key reconstruction questions.

A new passphrase must be specified, even if it is blank (""").

The usage format is:

```
pgp --key-recon-recv <key> [--answer <a1> ... --answer <a5>] --new-passphrase <newpass> --auth-username <auth user> --auth-passphrase <auth pass> [--recon-server <recon server>]
```

Where:

- `<key>` is the user ID, portion of the user ID, or the key ID of the key being reconstructed.
- `<a1>` is the answer to the first question of the five questions that only you can answer. Answers must be at least six characters long.
- `<newpass>` is the new passphrase for your reconstructed private key.
- `<auth user>` is your username on an Active Directory server. This username will be authenticated by the PGP Universal Server.
- `<auth pass>` is your passphrase on an Active Directory server. This passphrase will be authenticated by the PGP Universal Server.
- `<recon server>` is the PGP Universal Server on which your key reconstruction information is stored.
- `<force>` is required.

Example:

```
pgp --key-recon-recv 0x3D58AE31 --answer "Answer 1" ... --answer "Answer 5" --new-passphrase "cam3r0n-Alic&" --auth-username myuser --auth-passphrase "mypass" --recon-server 10.1.1.45
```

The answers to the questions stored for the specified key (0x3D58AE31) on the specified PGP Universal Server are provided and the key is reconstructed.
--remove

Removes a public key (not private keys) from the local keyring.

The usage format is:

```
pgp --remove <input>
```

Where:

- `<input>` is the user ID, portion of the user ID, or the key ID of the key that is being removed from the keyring.

Example:

```
pgp --remove 0x12345678
```

Removes the specified key from the keyring.

--remove-adk

Removes a specific ADK from a key.

You can remove an ADK by name if the ADK is present on the local keyring. Otherwise, you must use the key ID.

The usage format is:

```
pgp --remove-adk <user> --adk <adk> --passphrase <pass>
```

Where:

- `<user>` is the user ID, portion of the user ID, or the key ID of the key from which the ADK is being removed.
- `<adk>` is the specific ADK to be removed from the key.
- `<pass>` is the passphrase of the key from which the ADK is being removed.

Example:

```
pgp --remove-adk "Bob Smith" --adk Alice --passphrase "b0bsm1t4"
```

```
0x6245273E:remove ADK (0:ADKs successfully updated)
```

Removes the specified ADK from Bob’s key.

--remove-all-adks

Removes all ADKs from a key.

The usage format is:

```
pgp --remove-adks <user> --passphrase <pass>
```

Where:

- `<user>` is the user ID, portion of the user ID, or the key ID of the key whose ADKs are being removed.
<pass> is the passphrase of the key.

Example:

```
pgp --remove-all-adks alice@example.com --passphrase "cam3r0n"
0x3E439B98: remove all ADKs (0: ADKs successfully updated)
```

Removes all ADKs from Alice’s key.

--remove-all-photoids

Removes all photo IDs from a key. PGP Command Line can add only one photo ID, but it can remove multiple photo IDs that exist on a key.

The usage format is:

```
pgp --remove-all-photoids <user>
```

Where:

- `<user>` is the user ID, portion of the user ID, or the key ID of the user whose photo IDs are being removed.

Example:

```
pgp --remove-all-photoids Alice
0xD0EA20A7: remove all photo IDs (0: removed photo IDs, 1)
```

All photo IDs are removed from Alice’s key.

--remove-all-revokers

Removes all revokers from a key.

The usage format is:

```
pgp --remove-all-revokers <user> --passphrase <pass>
```

Where:

- `<user>` is the user ID, portion of the user ID, or the key ID of the key whose revokers are being removed.

- `<pass>` is the passphrase of the key.

Example:

```
pgp --remove-all-revokers alice@example.com --passphrase "cam3r0n"
0x3E439B98: remove all revokers (0: revokers successfully updated)
```

Removes all revokers from Alice’s key.
--remove-expiration-date

Removes the expiration date from a key.

The usage format is:

   pgp --remove-expiration-date <user> --passphrase <pass>

Where:

   <user> is the user ID, portion of the user ID, or the key ID of the key whose expiration date is being removed.

   <pass> is the passphrase of the key.

Example:

   pgp --remove-expiration-date Cameron --passphrase "cam3r0n"
   0x3E439B98:remove expire date (0:expiration date successfully updated)
   Removes the expiration date from Alice’s key.

--remove-key-pair

Removes a key pair from the local keyring. The option --force is required to make it more difficult to accidentally remove a key pair.

The usage format is:

   pgp --remove-key-pair <input> --force

Where:

   <input> is the user ID, portion of the user ID, or the key ID of the key pair that is being removed from the keyring.

Example:

   pgp --remove-key-pair "Jose Medina" --force
   0xF6EFC4D9:remove key pair (0:key successfully removed)
   Removes Jose’s key pair from the keyring.

--remove-photoid

Removes a photo ID from a key. There must be a photo ID on the key for it to be removed.

The usage format is:

   pgp --remove-photoid <user> [options]

Where:

   <user> is the user ID, portion of the user ID, or the key ID of the key from which the photo ID is being removed.

   --index specifies which photo ID on the key should be exported. 1 indicates the first photo ID, 2 the second photo, and so on.
Examples:

1. `pgp --remove-photoid "Bob Smith"
   0x6245273E: remove photo ID (0: successfully removed photo ID)
   Removes the photo ID from Bob's key.

2. `pgp --remove-photoid 0x12345678 --index 2`
   Removes only the second photo ID from the specified key.

--remove-preferred-cipher

Removes a preferred cipher from a key.

The usage format is:

```
pgp --remove-preferred-cipher <user> --cipher <cipher>
   --passphrase <pass>
```

Where:

- `<user>` is the user ID, portion of the user ID, or the key ID of the key from which the preferred cipher is being removed.
- `<cipher>` is the preferred cipher being removed.
- `<pass>` is the passphrase of the key.

Example:

```
pgp --remove-preferred-cipher "Bob Smith" --cipher blowfish
   --passphrase "b0bsmlt4"
   0x6245273E: remove preferred cipher (0: preferred ciphers updated)
   Removes the cipher Blowfish from Bob's key.
```

--remove-preferred-compression-algorithm

Removes a preferred compression algorithm from a key.

The usage format is:

```
pgp --remove-preferred-compression-algorithm <user>
   --compression-algorithm <algo> --passphrase <pass>
```

Where:

- `<user>` is the user ID, portion of the user ID, or the key ID of the key from which the preferred compression algorithm is being removed.
- `<algo>` is the preferred compression algorithm being removed.
- `<pass>` is the passphrase of the key.
Example:

```
pgp --remove-preferred-compression-algorithm "Bob Smith"
--compression-algorithm bzip2 --passphrase "b0bsm1t4"
0x6245273E:remove preferred compression algorithm (0:preferred
compression algorithms updated)
```

Removes the compression algorithm Bzip2 from Bob’s key.

--remove-preferred-email-encoding

Removes the preferred email encoding from a key.

A key must be at least v4 to have a preferred email encoding.

The usage format is:

```
pgp --remove-preferred-email-encoding <user> --email-encoding
<encoding> --passphrase <pass>
```

Where:

- `<user>` is the user ID, portion of the user ID, or the key ID of the key from which the preferred email encoding is being removed.
- `<encoding>` is the preferred email encoding being removed from a key. You can remove several preferred email encodings from a key, one at a time.
- `<pass>` is the passphrase of the key from which the preferred email encodings are being removed.

Example:

```
pgp --remove-preferred-email-encoding "Bob Smith" --email-encoding
pgpmime --passphrase sm1t4
```

Removes the preferred email encoding pgpmime from Bob’s key.

--remove-preferred-hash

Removes the preferred hash from a key. Note that a key must be at least v4 to have preferred hashes.

The usage format is:

```
pgp --remove-preferred-hash <user> --hash <hash> --passphrase <pass>
```

Where:

- `<user>` is the user ID, portion of the user ID, or the key ID of the key from which the preferred hash is being removed.
- `<hash>` is the preferred hash being removed from a key. You can remove several preferred hashes from a key, one at a time.
- `<pass>` is the passphrase of the key from which the preferred hashes are being removed.
Example:

```
pgp --remove-preferred-hash "Bob Smith" --hash md5
    --passphrase sm1t4
```

Removes the preferred hash MD5 from Bob's key.

---

**--remove-preferred-keyserver**

Removes the preferred keyserver from a key.

The usage format is:

```
pgp --remove-preferred-keyserver <user> --passphrase <pass>
```

Where:

- `<user>` is the user ID, portion of the user ID, or the key ID of the key from which the preferred keyserver is being removed.
- `<pass>` is the passphrase of the key.

Example:

```
pgp --remove-preferred-keyserver "Bob Smith" --passphrase "b0bsm1t4"
```

0x6245273E:remove preferred keyserver (0:preferred keyserver removed)

The preferred keyserver is removed from Bob's key.

---

**--remove-revoker**

Removes a specific revoker from a key. You can remove a revoker by name if the revoker is present on the local keyring; otherwise use the key ID.

The usage format is:

```
pgp --remove-revoker <user> --revoker <revoker> --passphrase <pass>
```

Where:

- `<user>` is the user ID, portion of the user ID, or the key ID of the key from which the revoker is being removed.
- `<revoker>` is the specific revoker to be removed from the key.
- `<pass>` is the passphrase of the key from which the revoker is being removed.

Examples:

```
pgp --remove-revoker Smith --revoker Alice --passphrase "sm1t4"
```

0x6245273E:remove revoker (0:revokers successfully updated)

Removes the specified revoker from Bob's key.
--remove-sig

Removes a signature from your public key.

You can remove a signature from any key on the local keyring. The signature will be merged back into the key when it is updated from the keyserver.

If you have posted your public key to a keyserver with the signature you are removing, first remove your public key from the keyserver, remove the signature on your local public key, and then post your key back to the keyserver. This will prevent the signature from being merged back in on update.

The usage format is:

```
pgp --remove-sig <user> --sig <signature>
```

Where:

- `<user>` is the user ID, portion of the user ID, or the key ID of the public key that holds the signature you want to remove. Be specific since there can be multiple signatures from the same user on different user IDs of the same key.

- `<sig>` is the user ID or key ID of the key of the signature you are removing from your public key. You must match this ID exactly.

Example:

```
pgp --remove-sig "Bob Smith" --sig 0x3E439B98
0x6245273E:remove signature (0:removed signature by user Alice Cameron <alice@example.com>)
```

Removes a specific signature (0x3E439B98) from Bob’s key.

--remove-subkey

Removes a subkey from a key on the local keyring.

The only way to specify the subkey is by its key ID. The --force option is required to make it more difficult to accidentally remove a subkey. No passphrase is required.

The usage format is:

```
pgp --remove-subkey <user> --subkey <subkey> --force
```

Where:

- `<user>` is the user ID, portion of the user ID, or the key ID of the key from which the subkey is being removed.

- `<subkey>` is the key ID of the subkey being removed.

Example:

```
pgp --remove-subkey bob@example.com --subkey 0x3D58AE31 --force
0x3D58AE31:remove subkey (0:subkey successfully removed)
```

The specified subkey (0x3D58AE31) is removed from Bob’s key.
--remove-userid

Removes a user ID from a key. If a key has only one user ID, you cannot remove it; also, when removing user IDs, you cannot remove the last user ID. You cannot have a key with only a photo ID. This command does not remove photo IDs; refer to the --remove-photoid command.

If you remove the primary user ID on a key, the next one below it becomes primary; to establish a different primary user ID, use --set-primary-userid.

The usage format is:

```
pgp --remove-userid <user> --user <userID>
```

Where:

- `<user>` is the user ID, portion of the user ID, or the key ID of the key from which the user ID is being removed.
- `<userID>` is the user ID being removed from the key.

Examples:

```
pgp --remove-userid "Bob Smith" --user Alice
0x6245273E:remove user ID (0:successfully removed Alice)
```

Removes the user ID “Alice” from Bob’s key.

--revoke

Revokes a key on the local keyring.

If for some reason you cannot trust a key pair, you can revoke it, which tells the world to stop using your public key to encrypt data to you. The best way to circulate a revoked key is to put it onto a public keyserver after you have revoked it.

--force is required to make it more difficult to accidentally revoke a key.

The usage format is:

```
pgp --revoke <user> [--revoker <revoker>] --passphrase <pass> --force
```

Where:

- `<user>` is the user ID, portion of user ID, or the key ID of the key being revoked.
- `<pass>` is the passphrase to the key being revoked.
- `<revoker>` is the user ID, portion of the user ID, or the key ID of the designated revoker key. When this option is used, the passphrase belongs to the revoker key. This option is not needed if you use a designated revoker or if you are doing self revocation.

Examples:

```
1 pgp --revoke "Bob Smith" --passphrase "b0bsm1t4" --force
0x6245273E:revoke key (0: key successfully revoked)
```
Revoke Bob’s key from the local keyring.

2. `pgp --revoke "Bob Smith" --revoker "Maria Fuentes <maria@example.com>" --passphrase "fu3nt3s" --force`

Maria Fuentes, the designated revoker, revokes Bob’s key.

---

**--revoke-sig**

Revokes your signature on a public key that you have previously signed. The public key that you signed and whose signature you now want to revoke must be on the local keyring to be revoked.

The usage format is:

```
pgp --revoke-sig <user> --sig <sig> --passphrase <pass>
[options]
```

Where:

- `<user>` is the user ID, portion of the user ID, or the key ID of the public key you signed and whose signature you now want to revoke. Be as specific as possible, as there can be multiple signatures from the same user on different user IDs of the same key.

- `<sig>` is the user ID or key ID of the key of the person who is revoking their signature.

- `<pass>` is the passphrase of the private key of the person revoking their signature.

Options:

- `<photo>` is
- `<index>` is
- `<force>` is required to revoke a signature.

Example:

```
pgp --revoke-sig Fumiko --sig 0x3E439B98 --passphrase "cam3r0n"
0x5571A08B: revoke signature (0: revoked signature by user Alice Cameron <alice@example.com>)
```

Alice removed her signature from Fumiko’s key using Alice’s passphrase.

---

**--revoke-subkey**

Revokes a subkey on a key on the local keyring.

The option `--force` is required to make it more difficult to accidentally revoke a subkey.

The usage format is:

```
pgp --revoke-subkey <user> --subkey <subkey> --passphrase <pass> --force
```
--revoke-subkey

Where:

<user> is the user ID, portion of the user ID, or the key ID of the key on which the subkey is being revoked.

<subkey> is the key ID of the subkey being revoked.

<pass> is the passphrase of the key on which the subkey is being revoked.

Example:

```bash
pgp --revoke-subkey fumiko@example.com --subkey 0x29D55ACE --passphrase "asak0" --force
0x29D55ACE:revoke subkey (0:subkey successfully revoked)
```

The specified subkey on Fumiko’s key is revoked.

--send-shares

Sends key shares to a server that is joining a key and allows you to join a key over the network. If shares are protected by a key with a passphrase, this passphrase must be cached before sending the shares.

For more information, refer to the command --join-key.

The usage format is:

```bash
pgp --send-shares --share <share> --share-server <server> [--signer <signer>][--passphrase <pass>]
```

Where:

<share> is the specific share you want to send to the server.

<server> is the URL of the server that is joining the shares

<signer> is the name of the key used to authenticate the connection.

<pass> is the passphrase of the signer authenticating the connection.

Example:

```bash
pgp --send-shares --share "Alice Cameron-1-Bob Smith.shf" --share-server 172.30.100.51 --signer admin --passphrase "adminpass"
```

This command sends the share of Alice’s key assigned to Bob Smith to the server 172.30.100.51, where the connection is authenticated by the signer’s key "admin" and the passphrase "adminpass".

--set-expiration-date

Establishes an expiration date for a key.

The usage format is:

```bash
pgp --set-expiration-date <user> (--expiration-date <date>) --passphrase <pass>
```
Where:

- `<user>` is the user ID, portion of the user ID, or the key ID of the key whose expiration date is being set.
- `<date>` is the date on which the key expires.
- `<pass>` is the passphrase of the key.

Examples:

```
pgp --set-expiration-date 0x12345678 --expiration-date 2006-12-27 --passphrase "Merry#Pippen"
```
Sets the expiration date for the specified key to December 27, 2006.

```
pgp --set-expiration-date 0x12345678 --expiration-days 365 --passphrase "Saturday&Sunday"
```
Sets the specified key to expire in 365 days.

---

**--set-key-flag**

Sets one of the key preferences flags.

The usage format is:

```
pgp --set-key-flag <user> [--subkey <subkeyID>] --key-flag <flag> [--passphrase <pass>]
```

Where:

- `<user>` is the user ID, portion of the user ID, or the key ID of the user whose key preferences flag is being set.
- `<flag>` is the key preferences flag to be set.
- `<subkeyID>` is the subkey ID of the key whose key preferences flag is being set.
- `<pass>` is the passphrase of the key for which the preferences flag is being set.

Example:

```
p gp --set-key-flag Bob --key-flag private-shared --passphrase "sm1t4"
```

0x2B65A65E: set key flag (0:flags updated successfully)
You have successfully set the properties preference flag on Bob's key to 'private-shared'.

Prop Flags: Private shared
--set-preferred-ciphers
Sets the entire list of preferred ciphers on a key. Only RSA and DH/DSS v4 keys can have preferred ciphers.

The numbering of the ciphers in the command determines which cipher is used first, which is used second, and so on. The cipher set as 1 is the preferred cipher.

The usage format is:

```
pgp --set-preferred-ciphers <user> --passphrase <pass> <ciphers>
```

Where:

- `<user>` is the user ID, portion of the user ID, or the key ID of the key to which the preferred ciphers are being added.
- `<pass>` is the passphrase of the key.
- `<ciphers>` is one or more preferred ciphers.

Example:

```
pgp --set-preferred-ciphers 0x12345678 --passphrase "bicycling#is*fun" --aes256 1 --cast5 2
```

Specifies that only the ciphers AES256 and CAST5 should be used for the specified key, in that order.

--set-preferred-compression-algorithms
The `--set-preferred-compression-algorithms` command sets the entire list of preferred compression algorithms on a key. Only RSA and DH/DSS v4 keys can have preferred compression algorithms.

The numbering of the compression algorithms in the command determines which algorithm is used first, which is used second, and so on.

The usage format is:

```
pgp --set-preferred-compression-algorithms <user> --passphrase <pass> <algos>
```

Where:

- `<user>` is the user ID, portion of the user ID, or the key ID of the key to which the preferred compression algorithms are being added.
- `<pass>` is the passphrase of the key.
- `<algos>` is one or more preferred compression algorithms.

Example:

```
pgp --set-preferred-compression-algorithms 0x12345678 --passphrase "Goo^Goo^Dolls" --bzip2 1 --zlib 2
```
Specifies that only the compression algorithms BZip2 and Zlib should be used for the specified key, in that order.

**--set-preferred-email-encodings**

Sets the entire list of preferred email encodings on a key. Only RSA and DH/DSS v4 keys can have preferred email encodings.

The numbering of the email encodings in the command determines which email encoding is used first, which is used second, and so on. The email encoding set as 1 is the preferred email encoding.

The usage format is:

```
pgp --set-preferred-email-encodings <user> --passphrase <pass> <email encodings>
```

Where:

- `<user>` is the user ID, portion of the user ID, or the key ID of the key to which the preferred ciphers are being added.
- `<pass>` is the passphrase of the key.
- `<email encodings>` is one or more preferred email encodings.

Example:

```
pgp --set-preferred-email-encodings 0x12345678 --passphrase "bicycling#is*fun" --pgpmime 1 --partitioned 2
```

Specifies that the email encodings pgpmime and partitioned should be used for the specified key, in that order.

**--set-preferred-hashes**

Sets the entire list of hashes for a key (which can be only a v4 key).

The usage format is:

```
pgp --set-preferred-hashes <user> --passphrase <pass> <hash> 1
[<hash> 2...]
```

Where:

- `<user>` the user ID, portion of the user ID, or the key ID of the key for which the preferred hashes are being set.
- `<hash>` is the preferred hash being set. The number following this option defines the place on the hash list: the first hash (1) is always the preferred hash, and other numbers are entered for conflict resolution.
- `<pass>` is the passphrase of the key on which the preferred ciphers are being set.

Example:

```
pmpz --set-preferred-hashes "Bob Smith" --passphrase "smlt4" --md5 1 --sha256 2 --sha384 3
```
0x2B65A65E: set preferred hashes (0:preferred hashes updated)
Sets MD5, SHA-256, and SHA-384 as preferred hashes for Bob's key.

<table>
<thead>
<tr>
<th>Hash</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Md5</td>
<td>MD5</td>
</tr>
<tr>
<td>SHA-256</td>
<td>SHA-256</td>
</tr>
<tr>
<td>SHA-384</td>
<td>SHA-384</td>
</tr>
</tbody>
</table>

**--set-preferred-keyserver**

Sets a preferred keyserver for a key. Only RSA and DH/DSS v4 keys can have a preferred keyserver, and it can be only one preferred keyserver. The full URL of the keyserver must be specified, such as ldap://keyserver.pgp.com.

The usage format is:

```
pgp --set-preferred-keyserver <user> --preferred-keyserver <ks> --passphrase <pass>
```

Where:

- `<user>` is the user ID, portion of the user ID, or the key ID of the key to which the preferred keyserver is being set.
- `<ks>` is the keyserver being set.
- `<pass>` is the passphrase of the key.

Example:

```
pgp --set-preferred-keyserver 0x12345678 ldap://keyserver.pgp.com --passphrase "asdfg*98765"
```

Sets ldap://keyserver.pgp.com as the preferred keyserver for the specified key.

**--set-primary-userid**

Sets a new primary user ID on a key.

Photo IDs cannot be set as the primary user ID.

The usage format is:

```
pgp --set-primary-userid <user> --user <newID> --passphrase <pass>
```

Where:

- `<user>` is the user ID, portion of the user ID, or the key ID of the key to which the new primary user ID is being added.
- `<newID>` is the new primary user ID for the key.
- `<pass>` is the passphrase of the key to which the new primary user ID is being added.
Example:

```
pgp --set-primary-userid 0x12345678 --user "Alice Cameron
<acameron@example.com>" --passphrase "jrrtolkien"
```

Adds the user ID "Alice Cameron <acameron@example.com>" to the specified key and makes it the primary user ID.

---

**--set-trust**

Sets the trust setting for a key.

Private keys can have trust settings of None or Implicit (for those for which you are the owner). Public keys can have trust settings of None (Untrusted), Marginal, or Complete (Trusted).

The usage format is:

```
pgp --set-trust <user> --trust <trust>
```

Where:

- `<user>` is the user ID, portion of the user ID, or the key ID of the key whose trust is being set.
- `<trust>` is trust setting you want to assign to the key. Options for private keys are `none` and `implicit`. Options for public keys are `none`, `marginal`, and `complete`.

Examples:

```
pgp --set-trust 0x12345678 --trust implicit
```

Trust is set to Implicit for the specified private key.

```
pgp --set-trust 0xABCD1234 --trust marginal
```

Trust is set to Marginal for the specified public key.

---

**--sign-key**

Signs every user ID on a key.

To sign a photo ID, use the `--photo` option. To sign just one photo ID among many, use the `--index` option.

The usage format is:

```
pgp --sign-key <user> --signer <signer> --sig-type <type> --passphrase <pass> [options]
```

Where:

- `<user>` is the user ID, portion of the user ID, or the key ID of the key you are signing.
- `<pass>` is the passphrase of the signer of the key.

[options] modify the behavior of the command. Options are:
--signer is the user ID, portion of the user ID, or the key ID of the signer of the key. If no signer is specified, the default key is used for signing.

--sig-type is the signature type: local, exportable, meta-introducer, or trusted introducer.

**Signature Types**

PGP Command Line supports several signature types:

- **local** means the signature is non-exportable, which means it cannot be sent with the key to a keyserver or exported in any way. Use this signature when you believe the key is valid, but you don’t want others to rely on your opinion of the key.

- **exportable** means the signature is exportable, which means that the signature can be sent with the key to a keyserver or exported with the key. Use this signature when you believe the key is valid and you want others to be able to rely on your opinion of the key. They are not obligated to rely on your opinion, however.

- **meta-introducer** means this is a non-exportable meta-introducer, which means that this key and any keys signed by this key with a trusted introducer validity assertion are fully trusted introducers to you. This signature type is not exportable.

- **trusted-introducer** means that you certify that this key is valid and that the owner of the key should be completely trusted to vouch for other keys. This signature type is exportable.

--trust-depth for meta-introducers and trusted introducers, you can specify how many levels of trust your signature applies to. The default for meta-introducer is 2, the default for trusted introducers is 1. The maximum depth for both is 8.

--regular-expression lets you establish a domain restriction for trusted introducers. This limits the trusted introducer’s certificate validation capabilities to the domain you enter. For example, example.com.

Examples:

```
pgp --sign-key "Bob Smith" --signer "alice@example.com" --sig-type exportable --passphrase "cam3r0n"
0x6245273E: sign key (0: certified user ID Bob Smith <bob@example.com>)
```

Signs Bob’s key with an exportable signature.

**--sign-userid**

Signs a user ID on a key on the local keyring.

To sign a single user ID, specify that user ID uniquely. To sign a photo ID, use the --photo option. To sign just one photo ID among many, use the --index option.
The usage format is:

```
pgp --sign-userid <user> --signer <signer> --sig-type <type> --passphrase <pass> [options]
```

Where:

- `<user>` is the user ID, portion of the user ID, or the key ID of the user ID you are signing.
- `<signer>` is the user ID, portion of the user ID, or the key ID of the signer of the user ID.
- `<type>` is the signature type: local, exportable, meta-introducer, or trusted introducer. See “Signature Types” on page 127 for complete descriptions.
- `<pass>` is the passphrase of the signer of the user ID.

[options] modify the behavior of the command. Options are:

- `--trust-depth` for meta-introducers and trusted introducers, you can specify how many levels of trust your signature applies to. The default for meta-introducer is 2, the default for trusted introducers is 1. The maximum depth for both is 8.
- `--regular-expression` lets you establish a domain restriction for trusted introducers. This limits the trusted introducer’s certificate validation capabilities to the domain you enter. For example, example.com.
- `--photo` lets you sign a photo ID.
- `--index` lets you sign one photo ID on a key when there are many. Specify 1 for the first photo ID on the key, 2 for the second, and so on.

Examples:

```
pgp --sign-userid "specific user" --signer me --sig-type exportable
Sign a specific user ID.

pgp --sign-userid key --photo --signer me ...
Sign the specified photo ID.
```

--split-key

Splits a key into two or more share files, called shares.

When you split a key, you split it between a group of shareholders. Each shareholder is assigned a certain number of shares in their share file; each shareholder can be assigned a different number of shares.

You specify the number of shares required to reconstitute the key so that it can be used (the threshold). For example, you could split a key into three shares with a threshold of two. Two of the three share files would be required before the key could be used.

Key splitting is a way to protect an important key, like a Corporate Signing Key, so that no one person can use the key unilaterally.
You must reconstitute a key using the `--join-key` command before you can use it again; refer to "--join-key" on page 104 for more information.

You can only split one key at a time, and a key cannot be split more than once. The number of people who get shares of a key (called shareholders) must be from two to 99. The maximum number of shares for a key is 255. A shareholder can have more than one share.

You can encrypt a share to a public key or you can use the name of the shareholder, in which case the share will be conventionally encrypted to a passphrase you specify.

Running the `--split-key` command without the `--force` option causes PGP Command Line to list the share information rather than split the key; refer to "--split-key Preview Mode" on page 131 for more information.

If the key you specify to be split is missing or not valid (revoked, disabled, and so on) or there is an error in the entering of the command, preview mode will not work nor will the key be split (depending on whether or not the `--force` option was used).

The share files are created based on the following:

- If `--output` is not used, the share filenames use the following format:

  `<split key common name>-#<recipient common name>.shf`

- If `--output` is a file, the share filenames use the following format:

  `<output>-#<recipient common name>.shf`

- If `--output` is a directory, the share filenames use the following format:

  `<output>/<split key common name>-#<recipient common name>.shf`

Where:

- `#` is the number of this share. The first share being a 1, the second a 2, and so on. The number is a single digit if the number of shareholders is fewer than 10 or double digits with zero padding from 10 to 99 (04, 09, 55, for example).

The usage format is:

```
pgp --split-key <user> --threshold <number> --share <share1> --share <share2> [--share <shareN> ...] --passphrase pass --force [--output]
```

Where:

- `<user>` is the user ID, portion of the user ID, or the key ID of the key you want to split. You must make an exact match, as you can only split one key at a time. Maximum number of share users is 99 (inclusive).

- `--threshold` is the threshold for the key being split: a minimum number of shares you need to put the split key back together (or to sign or decrypt with the key). It must be between 1 and the total number of shares (inclusive).
<share1> is the information that identifies share1, <share2> is the information that identifies share2, and so on. Restrictions on the shares are as follows: minimum number of shares per user is 1; maximum total number of shares (given to all users) is 255.

--force. If you run --split-key without the option --force, you will be able to see the preview mode before the actual key splitting occurs. There will be no output if there are parse errors or if the specific key is missing or invalid (revoked, disabled, etc.).

--passphrase specifies the passphrase of the key being split. It can be omitted if the key has no passphrase.

There is one option that can be user with the command --split-key:

--output lets you specify a different name for the share file. If output is not used, share filenames look as follows:

Alice Cameron-1-Bob Smith.shf
<common name of the split key user>-<number of share users>-<common name of the recipient>.shf
If output is a file, share filenames look as follows:

shares-1-Bob Smith.shf
<output file name>-<number of share users>-<common name of the recipient>.shf
If output is a directory, share filenames look as follows:

shares/Alice Cameron-1-Bob Smith.shf
<output file name>/<common name of the split key user>-<number of share users>-<common name of the recipient>.shf
The number of share users is presented with a single digit for less than 10 users, and a double digit for 10 to 99 users (which is the limit).

Example:

pgp --split-key "Alice Cameron" --threshold 40 --share "20:BobSmith" --share "20:Jill Johnson" --share "10:Mary Smith" --passphrase "cam3r0n"

Since you did not use --force, you will get the preview mode that gives you information such as follows:

Split Key User: Alice Cameron <alice@example.com>

Split Key ID: 0xEB778BFA (0xEF20715FEB778BFA)

Threshold: 40
Total Shares: 50
Total Users: 3
Share User: 20 0x2B65A65E Bob Smith <bob@example.com>
Share User: 20 0x17452786 Jill Johnson <jill@example.com>
Share User: 10 0x17452786 Mary Smith <mary@example.com>

0xEB778BFA:split key (3108:permission denied, force option required)

--split-key Preview Mode
If you use the --split-key command without the --force option, the specified key will not be split. Instead, the information about the split that would have happened if you had used --force is displayed; a preview mode.

This preview lets you check the split information you have entered to make sure it is correct before you actually split the key.

**Row 1: Split Key User Name**
Name: "Split Key User"
Value: Primary user ID of the key being split, in this case "Alice Cameron".

**Row 2: Split Key ID**
Name: "Split Key ID"
Value: The 32-bit key ID followed by the 64-bit key ID in the format:

0xEB778BFA (0xEF20715FEB778BFA)

**Row 3: Empty**

**Row 4: Threshold**
Name: "Threshold"
Value: Threshold for the key being split (minimum number of shares to put the key back together).
If threshold cannot be determined when joining a key, the character "?" is displayed. This can happen when PGP Command Line displays this information before it listens for network shares.

**Row 5: Total Shares**
Name: "Total Shares"
Value: Split. This is the total number of shares being divided among all users.
Join: This is the number of shares being collected from the file shares.

**Row 6: Total Users**
Name: "Total Users"
Value: Split. This is the total number of users who are getting the split key shares. Users can be public key recipients as well as conventionally encrypted recipients.

Row 7: Empty

Row 8-N: Share User
Name: “Share User”
Value: The parsed value of each share in the following format:

Share User: 20 0xB910E083 Bob Smith
- Number of shares assigned to a specific user (3 characters, left justified).
- Key ID of the share recipient. For public key encryption, this is a key ID in standard format, while for symmetric encryption, this is the string "symmetric".
- The name of the share recipient. For public key encryption, this is the primary user ID string, while for symmetric encryption, this is the name provided in the --share option.

If there are no share users specified, "N/A" is displayed. This can only happen when joining a key with the --skep option enabled.

Example:

pgp --split-key "Alice Cameron" --threshold 50 --share "25:Bob Smith" --share "25:Jill Johnson" --share 25:0x4EF05026 --passphrase "cam3r0n" --force

This time, the key was split successfully and the following message is displayed:

Split Key User: Alice Cameron <alice@example.com>
Split Key ID: 0xEB778BFA (0xEF20715FEB778BFA)
Threshold: 50
Total Shares: 50
Total Users: 2

Share User: 25 0x2B65A65E Bob Smith <bob@example.com>
Share User: 25 0x17452786 Jill Johnson <jill@example.com>

Alice Cameron-1-Bob Smith.shf:split key (2065:share file)
Alice Cameron-2-Jill Johnson.shf:split key (2065:share file)
0xEB778BFA:split key (0:key split successfully)
10 Miscellaneous Commands
Descriptions and Examples of Miscellaneous Commands

Overview

This chapter covers those PGP Command Line commands that don’t fit nicely into any other category. These commands are:

- **--create-keyrings**, which creates a pair of empty keyrings (page 133)
- **--help**, which displays the banner message and the built-in help message (page 134)
- **--license-authorize**, which activates PGP Command Line after receiving user’s data and license number (page 134)
- **--purge-all-caches**, which purges the passphrase and keyring caches (page 134)
- **--purge-keyring-cache**, which purges the keyring cache (page 135)
- **--purge-passphrase-cache**, which purges the passphrase cache (page 135)
- **--speed-test**, which runs a suite of PGP SDK speed tests (page 135)
- **--version**, which displays the version of PGP Command Line you are using and the banner message (page 136)
- **--wipe**, which wipes files off of your system (page 136)
- **--check-sigs**, which checks the signatures on all keys on the keyring (page 137)
- **--check-userids**, which checks the user IDs on specified keys to make sure they conform to the conventional naming standard (page 137)

**--create-keyrings**

Creates a pair of empty keyrings. Several commands create keyrings automatically as part of the command; **--gen-key**, **--import**, and **--keyserver-recv**, for example. You only need to use **--create-keyrings** if you want to create empty keyrings.

PGP Command Line will try to create the keyrings in the default location for the operating system: C:\Documents and Settings\<current user>\My Documents\PGP on Windows, $HOME/Documents/PGP on Mac OS X, and $HOME/.pgp/ on UNIX. If the PGP portions of these directories do not exist, PGP Command Line attempts to create them.

If the home directory is set and keyrings are not specified, PGP Command Line will try to create the keyrings in the default home directory location. No paths will be created in this case; they must already exist. If the keyrings are specified, they are relative to the current directory. Use a full path in this case.

The usage format is:
pgp --create-keyrings [--home-dir <path1>] [--public-keyring <path2>] [--private-keyring <path3>]

Where:

<path1> is the path to the home directory.

<path2> is the path to the public keyring file. You can specify a single file (which is relative to the current directory), a relative path (relative to the current directory), or a full path (the recommended usage).

<path3> is the path to the private keyring file. You can specify a single file (which is relative to the current directory), a relative path (relative to the current directory), or a full path (the recommended usage).

Example:

pgp --create-keyrings --home-dir /test/
Create keyrings using /test as the home directory.

--help (-h)

Displays the banner message and the built-in help message, which provides a brief description of the commands and options in PGP Command Line.

The usage format is:

pgp --help

--license-authorize

You cannot use PGP Command Line normally until is licensed.

Refer to Chapter 3, Licensing for a complete description of how to license PGP Command Line.

--purge-all-caches

Purges both the passphrase cache and the keyring cache. This is a security risk, so PGP Command Line makes it easy for you to purge the passphrase and keyring caches at any time.

The usage format is:

pgp --purge-all-caches

Example:

pgp --purge-all-caches
Purges both the passphrase and the keyring cache.
--purge-keyring-cache

Purges the keyring cache, which stores keyrings in memory so that they do not have to be retrieved each time they are needed. This is a security risk, so PGP Command Line makes it easy for you to purge the keyring cache at any time. The option --purge-keyring-cache is not used unless specifically enabled.

The usage format is:

```bash
pgp --purge-keyring-cache
```

Example:

```bash
pgp --purge-keyring-cache
```

Purges the keyring cache.

--purge-passphrase-cache

Purges the global (shared) passphrase cache, which stores in memory passphrases you enter so that you do not have to enter them every time you need them. This is a security risk, so PGP Command Line makes it easy for you to purge the passphrase cache at any time.

--purge-passphrase-cache is not used unless specifically enabled.

The usage format is:

```bash
pgp --purge-passphrase-cache
```

Example:

```bash
pgp --purge-passphrase-cache
```

Purges the passphrase cache.

--speed-test

Runs a suite of PGP SDK speed tests, which both identify the version of the PGP SDK that PGP Command Line is using and returns test results for several tests: hash, cipher, and public key, for example.

Running --speed-test forces PGP Command Line into local mode. Running --speed-test in FIPS mode (--fips-mode) runs the tests with the PGP SDK in FIPS mode, which runs a slightly different set of tests.

The usage format is:

```bash
pgp --speed-test [--fips-mode]
```

Example:

```bash
pgp --speed-test
```

Runs the suite of PGP SDK speed tests.
--version

Tells you what version of PGP Command Line you are using and displays the banner message.

The usage format is:

```
pgp --version [options]
```

Where:

[options] let you modify the command. Options are:

--verbose, which displays additional information about PGP Command Line, including passphrase cache information, time zone information, PGP SDK information, public key algorithms, symmetric ciphers, hashes, and compression.

Examples:

```
pgp --version
```

Displays version information and the banner message in the format:

```
PGP Command Line 9.6
Copyright (C) 2007 PGP Corporation
All rights reserved.
```

--wipe

Wipes a file off of your system.

The --wipe command exceeds the media sanitization requirements of Department of Defense 5220.22-M at three passes. Security continues to increase up to approximately 28 passes.

The usage format is:

```
pgp --wipe <input> [<input> ...] [options]
```

Where:

<input> is the file or files you want to wipe.

[options] let you modify the command. Options are:

--wipe-passes, which lets you specify how many wipe passes are made. Available values are 1 through 49. The default is 3.

--recursive, which lets you select subdirectories and files in subdirectories.

--verbose, which provides extra information about the progress.

Examples:

```
1 pgp --wipe secretreport.txt
```

Wipes the file secretreport.txt from your system using the default number of passes, three.
2  `pgp --wipe secret.doc --wipe-passes 8`
   Wipes the file secret.doc from your system using the number of passes specified
   with the `--wipe-passes` option, eight.

--check-sigs
Checks the signatures on all keys on your keyring. If errors are found, they are displayed.

   The usage format is:
     `pgp --check-sigs`

   Example:

1  `pgp --check-sigs`
   Checks the signatures of all keys on your keyring.

--check-userids
Checks the user IDs on specified keys to make sure they conform to the conventional
naming standard. The acceptable form for a user ID is:

   - More than 1 character but fewer than 256 characters.
   - Common Name <contact information>. For example, "Alice Cameron <acameron
     @example.com>" or "Ming Pa <AIM: 12345678>".

     Common Name does not have to be the name of an individual. On an ADK, for
     example, it could be a company name.

     <contact information> cannot be empty, but it does not have to be an email address
     or viable contact information.

   - The GPG format "Common Name (Comment) <contact information>" is invalid.

If no invalid user IDs are found, a successful status message ("0:signatures checked
successfully") appears. If invalid user IDs are found, each is listed as an error status
message and the exit code is returned.

   The usage format is:
     `pgp --check-userids [<user1> ...]

   Where:

     <user1> is the user ID, portion of a user ID, or the key ID of a key on your keyring.

   Examples:

1  `pgp --check-userids`
   Checks the user IDs of all keys on your keyring.

2  `pgp --check-userids acameron`
   Checks the user IDs of all keys on your keyring with "acameron" in the user ID or key
   ID of the key.
11 Options

Descriptions and Examples of PGP Command Line Options

This chapter lists and describes PGP Command Line options. Options are listed in alphabetical order within their sections.

The descriptions of some options in PGP Command Line mention that they are 'secure,' as in "This option is not secure" or "--auth-passphrase is secure". In this context, "secure" means that the option's argument is saved in non-pageable memory (when that option is available to applications). Options that are not "secure" are saved in normal system memory.

There are certain options that can change PGP Command Line behavior. For example, the options --archive and --sda will change how an encryption command works.

For example, if you wish to encrypt multiple files and you specify an output file without the option --archive, you will get an error message:

```
pgp -er "Bob Smith" note.txt report.doc -o bobsarchive.pgp
pgp:encrypt (3028:multiple inputs cannot be sent to a single output file)
```

If you enter the option --archive, the command will succeed:

```
pgp -er "Bob Smith" note.txt report.doc -o bobsarchive.pgp
--archive
pgpo0001.tmp:encrypt (3110:archive imported note.txt)
pgo0001.tmp:encrypt (3110:archive imported report.doc)
pgo0001.tmp:encrypt (0:output file bobsarchive.pgp)
```

PGP Command Line options are described in the following sections:

- "Boolean Options" on page 140
- "Integer Options" on page 150
- "Enumeration Options" on page 160
- "String Options" on page 170
- "List Options" on page 180
- "File Descriptors" on page 184
Boolean Options

Boolean options are settings that support only on and off conditions. To enable a Boolean option, just specify the flag on the command line. To disable a Boolean option, specify the flag with the --no prefix. For example:

- To enable local mode, use --local-mode on the command line.
- To disable local mode, use --no-local-mode on the command line.

Boolean arguments are never secure.

--always-trust

Assumes all keys used are trusted. The default is off. This setting is not reflected in key list operations.

--archive

This option enables or disables the archive mode. When set, PGP Command Line lets you encrypt/sign multiple files or entire directories into a PGP Zip output archive that is encrypted and compressed.

A PGP Zip archive is an excellent way to distribute files and folders securely or back them up.

The usage format is:

```
pgp -e/-c <input1> <input2> [inputN...] --archive/ --no-archive
```

Where:

- `<input>` is the file being encrypted

Examples:

1. `pgp -er <bob@example.com> note.txt README.txt -o archive.pgp --archive`
   When archiving several files, you have to separate them with spaces. This command creates `archive.pgp` with the following contents:
   ```
   pgp000000.tmp:encrypt (3110:archive imported note.txt)
   pgp000000.tmp:encrypt (3110:archive imported README.txt)
   pgp000000.tmp:encrypt (0:output file archive.pgp)
   ```

   When encrypting multiple files using `*`, the output will be different depending on whether the archive mode is enabled or disabled:

2. `pgp -er "Bill Brown" *.txt --archive`
   This gives an error:
   ```
   pgp:encrypt (3029:no output specified)
   ```

3. `pgp -er "Bill Brown" *.txt --no-archive`
All files ending with .txt are encrypted:

- `note.txt:encrypt (0:output file note.txt.pgp)`
- `README.txt:encrypt (0:output file README.txt.pgp)`
- `report.txt:encrypt (0:output file report.txt.pgp)`

```bash
pgp -er "Bill Brown" *.txt -o newarchive.pgp --archive
```

All files ending with .txt are encrypted into the file "newarchive.pgp".

```bash
pgp00000.tmp:encrypt (3110:archive imported note.txt)
pgp00000.tmp:encrypt (3110:archive imported README.txt)
pgp00000.tmp:encrypt (3110:archive imported report.txt)
p gp00000.tmp:encrypt (0:output file newarchive.pgp)
```

```bash
pgp -er "Bill Brown" *.txt -o newarchive.pgp --no-archive
```

This gives an error:

With the option `--no-archive` set, you cannot produce an archive.

---

**--banner**

Changes how the PGP Command Line banner displays.

The PGP Command Line banner is automatically turned on for certain operations; `--version` and `--help`, for example. The default is off.

Example:

```
pgp --list-keys --banner
```

List keys with the PGP Command Line banner at the top.

---

**--biometric**

Causes output to be in biometric format. Used only with `--fingerprint`. The default is off.

Example:

```
pgp --fingerprint 0xABCD5678 --biometric
```

Displays the fingerprint of the specified key using biometric words, not hexadecimal numbers.

---

**--buffered-stdio**

Enables buffered `stdio` (standard input and output).

Some platforms, such as Win32, AIX, and HP-UX, require the use of buffered `stdio`. Note that large operations may become slower because the data must be stored in memory.

Other platforms may optionally use `/dev/stdin` and `/dev/stdout` as files. This speeds up I/O since PGP Command Line has direct file access to `stdin` and `stdout`. 
Default for Win32, AIX, HP-UX is **TRUE**.
Default for Linux, Solaris, Mac OS X is **FALSE**.

Examples:

1. **pgp -er user file --output**
   Writes directly to `/dev/stdout` as if it were a file.

2. **pgp -er user file --output --buffered-stdio**
   First stores data in memory and then writes it to `stdout`.

---

**--compress, --compression**

Toggles compression, which is **on** by default.

When enabled, compression behaves as follows:

- **Public-key encryption**: The preferred compression algorithm of the recipient is used. If no preferred compression algorithms are set, Zip is used.
- **Symmetric encryption**: If a preferred compression algorithm is supplied, it is used; otherwise, Zip is used.

When compression is disabled, any preferred compression algorithms are ignored.

Example:

```
pgp -er "Bill Brown" README.txt --compress
```

The file README.txt was compressed using the preferred compression algorithm of the recipient.

```
README.txt:encrypt (0:output file README.txt.pgp)
```

---

**--encrypt-to-self**

Encrypts to the default key. The default is **off**. A warning is generated if the default key cannot encrypt. The default is **off**.

Example:

```
pgp -er Alice file.txt --encrypt-to-self
```

Encrypts the file to the specified recipient and also to the default key on the keyring. If the default key cannot encrypt, a warning is generated (this doesn’t correspond to an error condition, since the default key is technically the default signing key).

---

**--eyes-only**

Specifies that encryption should be for 'eyes only,' which means the recipient must view the decrypted output on screen; the sender, the person encrypting the file, specifies that the file is encrypted 'eyes only.' The default is **off**. The option **--eyes-only** should be used for text inputs.
When a message is sent “eyes only,” the decrypted output is only kept in secure memory and is never written to disk. The recipient can only view the decrypted data on screen. The recipient must use `--eyes-only` on decrypt.

⚠️ While “eyes only” can prevent a file from being written to disk, it cannot prevent the recipient from saving the data some other way; by writing it down or by doing a screen capture, for example.

Example:

```
pgp -er "Alice@example.com" report.txt --eyes-only
```

Output is the file report.txt.pgp, which is encrypted so that Alice can view it on her screen (for her eyes only).

---

**--fast-key-gen**

Enables fast key generation. The default is **on**.

The key generation process is made faster by using a previously calculated set of prime numbers rather than going through the process of creating them from scratch. Although it would be unlikely for anyone to crack your key based on their knowledge of these canned prime numbers, you may want to spend the extra time to create a key pair with the maximum level of security.

Example:

```
pgp --gen-key <bob@example.com> --key-type rsa --encryption-bits 1024 --passphrase "" --fast-key-gen
```

Generate this key in fast key generation mode.

---

**--fips-mode, --fips**

FIPS (Federal Information Processing Standards) is a series of standards, from which FIPS 140-1 and FIPS 140-2 are both worldwide de facto standards for the implementation of cryptographic modules

This option enables FIPS-compliant mode. The default is **off**.

Example:

```
pgp --speed-test --fips-mode
```

Performs the `--speed-test` command with the PGP SDK in FIPS mode.

---

**--force (-f)**

Required for certain operations to continue. Because there is no user interaction once a command has been issued, **--force** is used to ensure that the user really wants to issue the command.

This option is required for the following operations: `--remove-key-pair`, `--remove-subkey`, `--revoke`, `--revoke-subkey`, `--split-key`, and `--join-key`. 
For more details, refer to these commands. The default is \texttt{off}.

Examples:

1. \texttt{pgp --remove-key-pair Alice}
   Returns an error; \texttt{--force} is required.

2. \texttt{pgp --remove-key-pair Alice --force}
   Operation works.

\texttt{--halt-on-error}

Causes PGP Command Line to stop processing on error when multiple input/output files are being used. The default is \texttt{off}. Does not apply to some operations.

Use \texttt{--halt-on-error} if you want processing to stop when an error occurs. If you do not use \texttt{--halt-on-error}, PGP Command Line will keep trying all the files in the list until there are not any more, then return a partial failure.

\texttt{--keyring-cache}

Enables the keyring cache. The default is \texttt{off}. This option does not work with \texttt{--local-mode}.

\texttt{--large-keyrings}

Checks keyring signatures only when necessary. This option will improve performance of PGP Command Line when dealing with large keyrings, since keyring signatures won’t be verified.

This option is ignored when the following commands are used: \texttt{--verify}, \texttt{--export}, \texttt{--export-key-pair}, and \texttt{--revoke}.

The default is \texttt{FALSE}.

Example:

\texttt{pgp --list-keys --large-keyrings}

This command will list all keys, but it will skip the signatures check.

\texttt{--license-recover}

Enables email support for license recovery.

If you are re-licensing PGP Command Line and the information entered (licensee name and organization) does not match the information for which the existing authorization was issued, you will get an error.

In such a case, an email message will be sent to you with the correct information if the license recover feature is enabled.

The default is enabled.
Examples:

```
pgp --license-authorize --license-name "Alice Cameron"
   --license-email "alice@example.com" --license-organization
   "Example Corporation" --license-number
   "D45T4-TXXWZ-FNPVB-LP6MJ-12NWJ-ZYA" authorization.txt --force
   --license-recover
```

In this case you will get an error since the file "authorization.txt" was issued for the data that does not match the data entered in the above command. The option --license-recover is enabled by default and can be omitted on the command line.

```
pgp --license-authorize --license-name "Alice Cameron"
   --license-email "alice@example.com" --license-organization
   "Example Corporation" --license-number
   "D45T4-TXXWZ-FNPVB-LP6MJ-12NWJ-ZYA" authorization.txt --force
   --no-license-recover
```

In this case you will also get an error since the file "authorization.txt" was issued for the data that does not match the data entered in the above command. Since you used --no-license-recover, you will not get an email from the license server.

---

**--local-mode**

Forces the PGP SDK to initialize and run in local mode. The default is off.

Running in local mode means passphrase and keyring caches are not enabled or used. Entropy generation can be affected in some cases as well.

Example:

```
pgp --list-keys --local-mode
```

Performs the --list-keys command in local mode.

---

**--marginal-as-valid**

Treat keys with marginal validity as fully valid. The default is off.

---

**--pass-through**

Pass through non-PGP data during decode. The default is off.

The option --pass-through is useful for decrypting an email, for example, and preserving the headers.

⚠️ If there is data outside a signature and you are using --pass-through, there is no way to tell what was originally signed.

Example:

```
pgp --decrypt file ... --pass-through
```

Decrypt a file with pass through enabled.
--passphrase-cache

Enables the passphrase cache. The default is off.

This option does not work with --local-mode.

--photo

Specifies that PGP Command Line is to match a photo ID when searching for users to match. The default is off.

This option is implemented for --sign-userid, --remove-sig, and --revoke-sig.

Example:

    pgp --sign-userid jasonskey --user mykey --photo
    Sign the photo ID on Jason's key.

--quiet (-q)

Limits the messages that PGP Command Line writes out to errors (in other words, warnings are suppressed). The default message level is normal.

Example:

    pgp --version -q
    Runs the --version command in quiet mode.

--recursive

Enables recursive mode, which is used to select items in subdirectories for archiving and wiping.

This option is automatically enabled for --archive and --sda; it cannot be disabled for these commands.

Example:

    pgp --wipe *
    pgp --wipe * --recursive
    The first command wipes just the files at the specified location; subdirectories and files in those subdirectories are not wiped. The second command, with --recursive, wipes the files at the specified location and all subdirectories and all files in those subdirectories.

--reverse-sort, --reverse

Causes lists to be sorted backwards. The default is off.

Example:

    pgp --list-keys --sort userid --reverse-sort
    Lists keys on the keyring in reverse order, sorted by user ID.
--sda

This option is used with --encrypt or --decrypt to encode or decode a Self-Decrypting Archive (SDA).

An SDA is an encrypted archive that contains the code needed to decrypt it, but the recipient does not need to have PGP Desktop or PGP Command Line on their system to open the SDA. Because of this, you must be able to securely communicate the passphrase of the SDA to the person who is going to be decrypting it.

To specify the target platform for the output file, see --target-platform for more details. The extension .exe will be added also on all UNIX platforms in order to differentiate the new SDA from the original file.

The default is FALSE.

Examples:

1  pgp --encrypt newreports --symmetric-passphrase sm1t4 --sda --target-platform win32
   pgp00001.tmp:encrypt (0:output file newreports.exe)
   When encrypting only one file or directory, you do not need to specify the output file: it will be created with the extension .exe by default.

2  pgp --encrypt reports newreports -o allreports.exe --symmetric-passphrase sm1t4 --sda --target-platform win32
   pgp00001.tmp:encrypt (0:output file allreports.exe)
   When encrypting more files or directories into one SDA, you must specify the output file with the extension (allreports.exe).

--skep

PGP Command Line uses this option when joining split keys over the network. It looks for split files on the network and if it does not find enough of them, it continues to listen using the timeout defined by the option --skep-timeout.

The default is FALSE.

This option is used with the commands --join-key and --join-key-cache-only.

Example:

   pgp --join-key "Alice Cameron" --passphrase testkey --share "Alice Cameron-1-Bob Smith.shf" --share "Alice Cameron-2-Jill Johnson.shf:jill" --force --skep --skep-timeout 300
   Tells the key joining operation to wait 5 minutes before it times out (the default for --skep-timeout is 120 seconds).

--text-mode, --text (-t)

Forces the input to canonical text mode. The default is off.

This option should not be used with binary files, because they will not decode properly. Auto detection of file type is currently not supported.
Example:

```
pgp -er user file.txt -t
The file.txt will decrypt properly on systems with alternate line endings.
```

---verbose (-v)

Enables verbose messages. The default message level is normal.

Example:

```
pgp --version -v
Runs the --version command in verbose mode, which displays more information than the default message level.
```

--warn-adk

Enables warning messages for ADKs. The default is off. See also --enforce-adk, as some warnings are not affected by this option.

You can also enable this option in the PGP Command Line configuration file; see “Configuration File” on page 36 for more information.

ADK warning messages are issued based on:

- If --enforce-adk is set to require and --warn-adk is enabled, PGP Command Line will issue a warning when adding an ADK.
- If --enforce-adk is set to attempt and --warn-adk is enabled, PGP Command Line will issue a warning when adding an ADK.
- If --enforce-adk is set to off and --warn-adk is enabled, PGP Command Line will issue a warning when an ADK is not found and when skipping an ADK.

--xml

This option is used to list key information in XML format. PGP Command Line will display all information including all user IDs and signatures in this format. You can list all keys or specify a single key for this operation.

To list keys in XML format, you may use either the command --list-keys-xml, or a key list operation with the added option --xml, such as --list-keys user1 --xml, or --list-keys --xml.

The default is FALSE.

This option is used with the following commands: --list-keys, --list-key-details, --list-userids, --list-sigs, --list-sig-details, --list-users, and other key listing commands such as --keyserver-search.

Example:

```
pgp --list-keys Bob --xml
<?xml version="1.0"?>
```
This command displays output in XML format.

Refer to the command `--list-keys-xml` to see the complete XML output.
Integer Options

Integer options are options that take a single number as an argument. Currently PGP Command Line does not support these options with negative values. The argument is required in all cases.

Integer arguments are never secure.

--3des

Specifies the precedence for the 3DES cipher algorithm. The default is not set.

This option takes as argument any number between 1 and the total number of ciphers (currently eight). The cipher set to 1 is the preferred cipher.

Examples:

1. `pgp --set-preferred-cipher user --3des 1`
   Sets 3DES to be the only preferred cipher.

2. `pgp --set-preferred-cipher user --3des 1 --aes256 2`
   Sets 3DES and AES256 to be preferred ciphers.

--aes128, --aes192, --aes256

Specifies the precedence for the AES128, AES192, or AES256 cipher algorithm. The default is not set.

This option takes as argument any number between 1 and the total number of ciphers (currently eight). The cipher set to 1 is the preferred cipher.

Examples:

1. `pgp --set-preferred-cipher user --aes128 1`
   Sets AES128 to be the only preferred cipher.

2. `pgp --set-preferred-cipher user --aes128 1 --aes256 2`
   Sets AES128 and AES256 to be preferred ciphers.

3. `pgp --set-preferred-cipher user --aes192 1 --aes256 2`
   Sets AES192 and AES256 to be preferred ciphers.

--bits, --encryption-bits

Specifies the size of the encryption key for generation. This option is required for all key types.

Valid sizes for RSA legacy are 1024 to 2048 bits, RSA v4 are 1024 to 4096 bits, DH are 1024 to 4096 bits.

- For RSA legacy keys, either `--bits` or `--signing-bits` can be supplied.
For RSA-sign-only keys, this option is mapped to `--signing-bits`, if not already supplied.

For DH-sign-only keys, this option is mapped to `--signing-bits`, if not already supplied.

Neither `--encryption-bits` nor `--bits` is a required option for RSA-sign-only keys if `--signing-bits` is set.

Neither `--encryption-bits`, `--bits`, nor `--signing-bits` is required for DH-sign-only keys, as the only valid setting is 1024 bits (specifying `--bits` or `--signing-bits` for a DH-sign-only key with a size other than 1024 returns an error).

Refer to the command “--gen-key” on page 100 for more details.

**--blowfish**

The algorithm Blowfish is deprecated and should not be set for new encryption keys. Due to concerns over security, PGP Command Line does not allow you to create new encryption keys with Blowfish specified as the preferred cipher, but it can be used either to decrypt messages encrypted using Blowfish, or to encrypt messages to existing PGP keys that specify Blowfish as their preferred cipher.

The only action you can take with PGP Command Line in regards to Blowfish is to remove it as a preferred cipher from a key.

Example:

```
pgp --remove-preferred-ciphers user --cipher blowfish
   --passphrase pass
```

Removes Blowfish as the preferred cipher.

**--bzip2**

Specifies the precedence of the BZip2 compression algorithm. The default is not set.

Takes a number between one and the total number of compression algorithms (currently three). The compression algorithm set to 1 is the preferred cipher.

Example:

```
pgp --set-preferred-compression-algorithms --bzip2 1 --zip 2
```

Sets BZip2 and Zip to be the preferred compression algorithms.

**--cast5**

Specifies the precedence for the CAST5 cipher algorithm. The default is not set.

Takes a number between 1 and the total number of ciphers (currently eight). The cipher set to 1 is the preferred cipher.
Examples:

1. `pgp --set-preferred-cipher user --cast5 1`
   Sets CAST5 to be the only preferred cipher.

2. `pgp --set-preferred-cipher user --cast5 1 --aes256 2`
   Sets CAST5 and AES256 to be preferred ciphers.

**--creation-days**

Changes the number of days until creation (1 equals tomorrow, 2 equals the next day, and so on). The default is today. See **--creation-date** for more information.

The option **--creation-days** is used only with **--gen-key** and **--gen-subkey**. It cannot be used on the same operation as **--creation-date**.

Using **--creation-days** changes the behavior of **--expiration-days**.

Example:

```
pgp --gen-key test ... --creation-days 31
```
Key will be valid starting in 31 days.

**--expiration-days**

Changes the number of days until expiration. The default is not set (no expiration). See **--expiration-date** for more information.

Days are interpreted as days from creation. If no creation is specified (with a date or number of days), **--expiration-days** is days from today (1 equals tomorrow, 2 equals the next day, and so on).

This option cannot be used on the same operation as **--expiration-date**. It is used only with the commands **--gen-key** and **--gen-subkey**.

If **--creation-date** is set, this becomes number of days from the creation date. If **--creation-days** is set, this becomes number of days from the creation date.

Examples:

1. `pgp --gen-key test ... --expire-days 31`
   Key valid for 31 days.

2. `pgp --gen-key test ... --creation-date 2006-01-01 --expire-days 31`
   Key valid in January of 2006.

**--idea**

Specifies the precedence for the IDEA cipher algorithm. The default is not set. It takes a number between 1 and the total number of ciphers (currently eight). The cipher set to 1 is the preferred cipher.
Example:

```
pgp --set-preferred-cipher user --idea 1 --aes256 2
```
Set IDEA and AES256 to be preferred ciphers.

---

**--index**

Specifies which object to use if multiple objects are found. The default is not set. If there is only one match, then the first item is returned. If there are multiple matches, then an error is returned.

This option requires an integer value greater than zero. This option works only with **--photo** to specify which photo ID is to be acted on. PGP Command Line lets you add only one photo ID to a key. Other applications with which PGP Command Line is compatible allow users to add more than one photo ID to a key; **--index** lets you work with these keys.

Examples:

1. `pgp --remove-photoid bobs-key`
   Removes the first, and only, photo ID on bobs-key.

2. `pgp --remove-photoid bobs-key --index 1`
   Remove the first photo ID on bobs-key when there is more than one.

3. `pgp --remove-photoid bills-key --index 2`
   Removes the second photo ID on bills-key when there are two or more.

4. `pgp --remove-photoid bills-key`
   Error, bills-key has two photo IDs on it.

---

**--keyring-cache-timeout**

Sets the number of seconds after which the keyring cache will time out. If set to zero (0), the keyring will not time out unless the cache is specifically purged. If timeout is greater than zero (>0), the keyring will time out after the specified number of seconds.

This option requires **--keyring-cache** to work. The default time for keyring cache is 120 seconds.

Example:

```
pgp --cache-passphrase 0x73CC6D8F --passphrase cam3r0n
--keyring-cache --keyring-cache-timeout 0
```
Cache the specified keyring with no timeout.

---

**--keyserver-timeout**

Sets the number of seconds until a keyserver operation times out. The default is 120 seconds and the minimum setting is 1 second.
The option \texttt{--keyserver-timeout} applies to a single keyserver operation; when searching multiple servers, the timeout increases. The update operation can use multiple keyservers, as well.

Example:

\begin{verbatim}
pgp --keyserver-search user --keyserver-timeout 30
\end{verbatim}
Search with a 30 second timeout.

\textbf{--md5}

This option is used to specify precedence of MD5 hash algorithm. Note that only v4 keys have preferred hashes.

- Digest length: 16 bytes
- Block size: 64 bytes
- Max. final block size: 55 bytes
- State size: 16 bytes
- Default: UNSET

This option is used with the following commands: \texttt{--add-preferred-hash}, \texttt{--set-preferred-hashes}, and \texttt{--remove-preferred-hash}.

Example:

\begin{verbatim}
pgp --add-preferred-hash Bob --hash md5 --passphrase sm1t4
\end{verbatim}
Adds the preferred hash algorithm MD5 to Bob's key.

\textbf{--passphrase-cache-timeout}

Specifies the number of seconds a passphrase lasts when cached. The default is 120 seconds.

Using a setting of zero means the passphrase cache will not time out, unless the cache is purged. A number greater than zero means the passphrase cache will time out after the specified number of seconds.

This option requires \texttt{--passphrase-cache}.

Examples:

1. \begin{verbatim}
pgp --passphrase-cache --passphrase-cache-timeout 0
   --cache-passphrase user --passphrase pass
\end{verbatim}
The passphrase cache will not time out until the cache is purged.

2. \begin{verbatim}
pgp --cache-passphrase 0x73CC6D8F --passphrase "cam3r0n"
   --passphrase-cache --passphrase-cache-timeout 0
\end{verbatim}
Cache the specified passphrase with no timeout.
--partitioned

Specifies the precedence of the partitioned email encoding scheme on a key.

The value can be a number between 1 and the total number of available email encodings (currently two: pgpmime and partitioned).

The default is unset.

Example:

```
pgp --set-preferred-email-encodings ... --partitioned 1 --pgpmime 2
```

Establishes partitioned as the preferred email encoding scheme for the key and pgpmime as secondary.

--pgpmime

Specifies the precedence of the pgpmime email encoding scheme on a key.

The value can be a number between 1 and the total number of available email encodings (currently two: pgpmime and partitioned).

The default is unset.

Example:

```
pgp --set-preferred-email-encodings ... --pgpmime 1 --partitioned 2
```

Establishes pgpmime as the preferred email encoding scheme for the key and partitioned as secondary.

--ripemd160

This option is used to specify precedence of RIPEMD hash algorithm. Note that only v4 keys have preferred hashes.

- Digest length: 20 bytes
- Block size: 64 bytes
- Max. final block size: 55 bytes
- State size: 20 bytes
- Default: UNSET

This option is used with the following commands: `--add-preferred-hash`, `--set-preferred-hashes`, and `--remove-preferred-hash`.

Example:

```
pgp --add-preferred-hash Bob --hash ripemd160 --passphrase "smit4"
```

Adds the preferred hash algorithm RIPEMD160 to Bob’s key.
2  `pgp --set-preferred-hashes Bob --passphrase smlt4 --ripemd160 1 --sha256 2 --sha384 3`
   Sets first RIPEMD160 and then SHA-256 and SHA-384 as preferred hashes for Bob’s key.

3  `pgp --remove-preferred-hash Bob --hash ripemd160 --passphrase "smlt4"
    Removes the preferred hash algorithm RIPEMD160 from Bob’s key.

`--sha, --sha256, --sha384, --sha512`
These options are used to specify precedence of the specified hash algorithm. Note that only v4 keys have preferred hashes. The default is unset. These options are used with the following commands: `--add-preferred-hash`, `--set-preferred-hashes`, and `--remove-preferred-hash`.

**SHA-1**
- Digest length: 20 bytes
- Block size: 64 bytes
- Max. final block size: 55 bytes
- State size: 20 bytes

**SHA-256**
- Digest length: 32 bytes
- Block size: 64 bytes
- Max. final block size: 55 bytes
- State size: 32 bytes

**SHA-384**
- Digest length: 32 bytes
- Block size: 64 bytes
- Max. final block size: 55 bytes
- State size: 32 bytes

**SHA-512**
- Digest length: 64 bytes
- Block size: 128 bytes
- Max. final block size: 111 bytes
- State size: 64 bytes
Examples:

1. `pgp --add-preferred-hash Bob --hash md5 --passphrase "sm1t4"`
   Adds the preferred hash algorithm MD5 to Bob’s key.

2. `pgp --set-preferred-hashes Bob --passphrase sm1t4 --md5 1 --sha256 2 --sha384 3`
   Sets first MD5 and then SHA-256 and SHA-384 as preferred hashes for Bob’s key.

3. `pgp --remove-preferred-hash "Bob Smith" --hash md5 --passphrase "sm1t4"`
   Removes the preferred hash algorithm MD5 from Bob’s key.

---

**--signing-bits**

Specifies the size of the master key for generation.

Valid bit ranges for signing keys are: RSA legacy, 1024 to 2048 bits; RSA v4, 1024 to 4096 bits; DH, 1024 bits. For RSA legacy keys, either `--bits` or `--signing-bits` can be supplied.

This option is required for RSA legacy keys. For RSA v4 keys, this option can be set independently of `--bits`. For DH keys, this option is automatically set to 1024.

For detailed explanation, refer to the command “--gen-key” on page 100.

---

**--skep-timeout**

Changes the timeout for joining keys over the network. There is no value reserved to indicate no timeout. The default is 120 seconds.

This option is used with the command `--join-key`.

Example:

```
pgp --join-key "Alice Cameron" --passphrase testkey --share "Alice Cameron-1-Bob Smith.shf" --share "Alice Cameron-2-Jill Johnson.shf;jill" --force --skep --skep-timeout 300
```

Tells the key joining operation to wait 5 minutes before it times out.

---

**--threshold**

Establishes the minimum share threshold required when reconstituting a split key. The default is not set. Refer to “--split-key” on page 128 for more information splitting a key.

Requires a value greater than zero and less than or equal to the total number of shares.

Example:

```
pgp --split-key 0x1234abcd --threshold 5 --share share1...
```

Establishes a threshold of 5 shares for the key being split.
--trust-depth

Sets the trust depth to use when creating meta-introducer and trusted-introducer signatures. The default for meta-introducer signatures is 2. The default for trusted-introducer signatures is 1.

For meta-introducer signatures, available values are 2 to 8, inclusive. For trusted-introducer signatures, 1 to 8, inclusive.

Example:

```
pgp --sign-key ... --trust-depth 4
```

Sets the trust depth to 4.

--twofish

Specifies the precedence for the Twofish cipher algorithm. The default is not set. It takes a number between 1 and the total number of ciphers (currently eight). The cipher set to 1 is the preferred cipher.

Example:

```
pgp --set-preferred-cipher user --twofish 1
```

Sets Twofish to be the only preferred cipher.

--wipe-input-passes

This option sets the number of wipe passes when wiping the input file. This number must be between 1 and 49 (inclusive). The default is 3.

This option requires --input-cleanup to be set for wipe following one of the file generating commands: --armor, --cleartext, --decrypt, --detached, --encrypt, and --sign.

Example:

```
pgp -er alice report.txt --input-cleanup wipe --wipe-input-passes 8
```

Encrypt the file report.txt and wipe the original with 8 passes.

--wipe-passes

Sets the number of passes to use with --wipe (between 1 and 49 inclusive). This command exceeds the media sanitization requirements of DoD 5220.22-M at 3 passes (which is the default for this option). The default is 3.

Example:

```
pgp --wipe README.txt --wipe-passes 6
```

Wipes the file README.txt with 6 passes.
--wipe-temp-passes

Sets the number of wipe passes to use when wiping temporary files. The default is 3. The
number of passes must be from 1 to 49, inclusive.

This option requires --temp-cleanup to be set for wipe following one of the file
generating commands: --armor, --clearsign, --decrypt, --detached, --encrypt, and --sign.

Example:

```
pgp -er Alice report.txt --input-cleanup wipe
--wipe-temp-passes 8
```

Encrypt file, then wipe the temporary file with 8 passes.

--wipe-overwrite-passes

This option sets the number of wipe passes to use when overwriting an existing output
file. The number of passes must be between 1 and 49 (inclusive). The default is 3.

This option requires --overwrite to be set for wipe following one of the file generating
commands: --armor, --clearsign, --decrypt, --detached, --encrypt, and --sign.

Example:

```
pgp -er Bob report.txt --overwrite wipe --wipe-overwrite-passes 12
```

Encrypt "report.txt" and then wipe the output file with 12 passes.

--zip

Specifies the precedence of the Zip compression algorithm. The default is not set. It takes
a number between one and the total number of compression algorithms (currently three).
The compression algorithm set to 1 is the preferred cipher.

Example:

```
pgp --set-preferred-compression-algorithms --zip 1 --zlib 2
```

Sets Zip and Zlib to be the preferred compression algorithms.

--zlib

Specifies the precedence of the Zlib compression algorithm. The default is not set. It takes
a number between one and the total number of compression algorithms (currently three).
The compression algorithm set to 1 is the preferred cipher.

Example:

```
pgp --set-preferred-compression-algorithms --zlib 1 --zip 2
```

Sets Zlib and Zip to be the preferred compression algorithms.
Enumeration Options

Enumeration options are options that take one of a specific set of strings that get converted internally to values. Each option has its own set of arguments. The argument is always required.

Enumeration arguments are never secure.

--auto-import-keys

Changes the behavior of PGP Command Line when keys are found during non-import operations. The default is all.

Options are:

- off (do not automatically import keys)
- merge (only merge the key if it already exists on the local keyring)
- new (import the key if it does not exist on the local keyring)
- all (automatically import / merge all keys found)

Examples:

1. `pgp --decrypt file-with-keys.pgp --auto-import-keys off`
   Skips keys.
2. `pgp --decrypt file-with-keys.pgp --auto-import-keys new`
   Gets any new keys.

--cipher

Specifies a cipher to use with certain operations. The default is unset. AES256 is used for those operations that require a cipher to be set. Symmetric encryption defaults to AES256.

This operation has no affect in certain cases; refer to --set-preferred-ciphers for more information. Blowfish is deprecated.

Options are as follows:

- idea (IDEA cipher)
- 3des (3DES cipher)
- cast5 (CAST5 cipher)
- blowfish (Blowfish cipher)
- aes128 (AES128 cipher)
- aes192 (AES192 cipher)
- aes256 (AES256 cipher)
- **twofish** (Twofish 256 cipher)

Examples:

1. `pgp -c report.txt --symmetric-passphrase sm1t4 --cipher cast5`
   Conventionally encrypts the file for the recipient Bob using the CAST5 cipher.

2. `pgp --add-preferred-cipher Bill --cipher idea --passphrase "bill2"
   Adds the cipher IDEA as the preferred cipher for Bill’s key.

---

**--compression-algorithm**

Sets the compression algorithm. Note that this option doesn’t work with public key encryption, because in this case the recipient’s key preferences are used. Mainly for This option is used mainly with symmetric encryption; it can be used also with the public key encryption, which is an advanced feature (see --encrypt for more information).

This option can be used with the following arguments:

- **zip**. ZIP compression (default for SDK)
- **zlib**. ZLIB compression
- **bzip2**. BZIP2 compression

Examples:

1. `pgp -s report.txt --signer Bob --passphrase sm1t4
   --compression-algorithm zip`
   An opaque attached signature (sign only) is created by Bob.

2. `pgp -cs report.txt --symmetric-passphrase sympass --signer "Bob Smith" --passphrase "sm1t4" --compression-algorithm zlib
   pgp -c report.txt --symmetric-passphrase sympass
   --compression-algorithm zip
   Two conventionally encrypted and signed files are created using the option --compression-algorithm.

3. `pgp --add-preferred-compression-algorithm "Bill Brown"
   --compression-algorithm zlib --passphrase "bill2"
   Adds the preferred compression algorithms zlib to Bill’s key:

   Compress: ZLIB
   Compress: Zip

---

**--compression-level**

Sets the compression level for the current operation. The choices are as follows:

- **default**. Use the default compression level.
- **fastest**. Use the least compression.
- **balanced**. Optimize compression for size and speed.
- **smallest**. Use the most compression.

The default is **balanced**.

This option currently valid only for SDA creation.

Example:

```
pgp --encrypt newreports -o newreports.exe
   --symmetric-passphrase "sm1t4" --sda --compression-level fastest
pgp00001.tmp:encrypt (0:output file newreports.exe)
```

This command produced a self-decrypting archive "newreports.exe" using the least amount of compression.

--- **email-encoding**

Specifies the email encoding to use with certain operations, such as editing the preferred email encoding for a key, for example.

The choices are as follows:

- **pgpmime**. Use PGP-MIME encoding.
- **partitioned**. Use partitioned encoding (formerly known as PGP Legacy encoding).

The default is unset.

Example:

```
pgp --add-preferred-email-encoding ... --email-encoding pgpmime
```

Specifies pgpmime as the preferred email encoding for the key.

--- **enforce-adk**

Changes the ADK enforcement policy. The default is attempt.

Options are: **off** (do not enforce any ADKs), **attempt** (attempt to enforce all ADKs), and **require** (require all ADKs).

When **off** is specified, warnings are only generated when **--warn-adk** is enabled. When attempt is specified, a non-suppressible warning is generated if an ADK is not found or if an ADK is not valid. Also when attempt is specified, if **--warn-adk** is enabled, a warning is generated when adding an ADK to the recipient set.

When **require** is specified, an error will be generated if an ADK is not found or an ADK is not valid. When require is specified, if **--warn-adk** is enabled, a warning is generated when adding an ADK to the recipient set.
**Examples:**

1. `pgp -er user file --enforce-adk require`
   Require all ADKs; error otherwise.

2. `pgp -er user file --enforce-adk off`
   Ignore all ADKs.

3. `pgp -er user file --enforce-adk off --warn-adk`
   Ignore all ADKs, but show them.

**--export-format**

This option lets you specify an export format.

Choose the export format from the following list of supported formats:

- **complete** (default format). Only armored blocks are output; the default file extension is `.asc`.
- **compatible**. Only armored blocks are output; the default file extension is `.asc`. Use compatible to export keys in the format compatible with older versions of PGP software (Versions 7.0 and prior).
- **x509-cert**. Only armored blocks are output; the default file extension is `.crt`. In this case, input must match exactly one key and **--cert** is required.
- **pkcs8**. Only binary blocks are output; the default file extension is `.p8`; a signed key must be paired; and input must match exactly one key. In this case, **--cert** is required.
- **pkcs12**. Only binary blocks are output; the default file extension is `.p12`; a signed key must be paired; and input must match exactly one key. In this case, **--cert** is required.
- **csr**. This option generates a certificate signature request (CSR). Only armored blocks are output and the default file extension is `.csr`. In this case, user must match exactly one key and key must be paired.

Example:

```
pgp --export-key-pair "Bill Brown" --export-format complete --passphrase ""
```

Bill’s key pair is exported to the ASCII-armored file "Bill Brown.asc" with no passphrase.

**--hash**

Used with operations that need to specify a single hash algorithm. The default is `unset`.

Choose from the following list of hashes:

- **md5**. MD5 hash
- **ripemd160**. RIPEMD-160 hash
- **sha**. SHA-1 hash
- **sha256**. SHA-256 hash
- **sha384**. SHA-384 hash
- **sha512**. SHA-512 hash

This option is used with the following commands: **--add-preferred-hash**, **--remove-preferred-hash**, and **--s/-sign** (see **--sign** for more information)

Example:

1. `pgp --add-preferred-hash "Bob Smith" --hash md5 --passphrase "sm1t4"
   Adds the preferred hash algorithm MD5 to Bob's key.

2. `pgp -s report.txt --signer Bob --passphrase "sm1t4" --hash md5
   The file "report.txt.asc" is signed by Bob using the hash algorithm MD5.

---

**--import-format**

Specifies the import format for the current operation. Choose one of the following supported import formats:

- **auto**. Auto detect import format, which is the default. When using auto detect, PGP Command Line will key off the file extension:
  - .crt,.pem for x509-cert
  - .asc,.pgp for pgp
  - .p7,.p7b for pkcs7
  - .p12,.pfx for pkcs12

If the format cannot be determined from the file extension, PGP Command Line will also look at the file header.

- **pgp**. PGP key
- **x509-cert**. PEM encoded X.509 certificate
- **pkcs7**. PKCS7 data
- **pkcs12**. PKCS12 data. The option **--passphrase** is required when importing PKCS12 data, even if it is an empty string.

Examples:

1. `pgp --import "Bob Smith.asc" --import-format pgp
   Bob Smith.asc: import key (0:key imported as 0x6245273E Bob Smith <bob@example.com>)`
Import Bob’s key using the PGP file format.

```
2 pgp --import "Bob Smith.asc" --import-format auto

Bob Smith.asc:import key (0:key imported as 0x6245273E Bob
  Smith <bob@example.com>)
In this case, the import format was detected automatically.
```

---

**--input-cleanup**

Determines what to do with input files when an operation has finished with them. The default is `off`. Input can be plaintext or ciphertext. See `--wipe-input-passes` for more information.

Options are:
- **off** (leave input files alone)
- **remove** (delete input files)
- **wipe** (wipe input files)

Example:

```
pgp -er user file.txt --input-cleanup wipe
```

Encrypts a file and then wipes the original when done.

---

**--key-flag**

Specifies the key preference flag. These flags specify how a key will encrypt or sign and are grouped by their function into key usage flags, keyserver preference flags, and key feature flags.

This option is used with the commands `--set-key-flag` and `--clear-key-flag`. The default is **unset**.

The key preference flags are:

**Key usage flags:**
- **sign-user-ids**. When this flag is specified, the key can sign user IDs.
- **sign-messages**. When this flag is specified, the key can sign messages.
- **encrypt-communications**. When this flag is specified, the key can encrypt communications.
- **encrypt-storage**. When this flag is specified, the key can encrypt for storage.
- **private-shared**. When this flag is specified, the private key is in the possession of a third party (group bit)
- **sign**. This flag specifies all signing flags at the same time.
- **encrypt**. This flag specifies all encryption flags at the same time.
- **encrypt-and-sign.** This flag specifies all signing and encryption flags at the same time.

**Keyserver preferences**
- **no-modify.** This flag requests that only the owner may modify the key on the server.

Examples:

```plaintext
pgp --set-key-flag Bob --key-flag private-shared --passphrase "sm1t4"
0x2B65A65E: set key flag (0: flags updated successfully)
You have successfully set the preference flag on Bob's key to "private-shared".
```

**--manual-import-key-pairs**
Changes the behavior of PGP Command Line when key pairs are found during import.

The manual key import can be set as follows:

- **off.** Do not import key pairs
- **public.** Imports public keys only
- **pair.** Imports key pairs

The default is **pair.**

Example:

```plaintext
pgp --import "Bob Smith.asc" --manual-import-key-pairs public
Bob Smith.asc: import key (0: key imported as 0x6245273E Bob Smith <bob@example.com>
Only Bob's public key was imported.
```

**--key-type**
Specifies a key type when generating keys. This option is required when **--gen-key** is used.

Options are:

- **rsa-legacy** (the older RSA v3 key format)
- **rsa** (the newer RSA v4 key format)
- **rsa-sign-only** (the newer RSA v4 key format with no automatically generated subkey)
- **dh** (the Diffie-Hellman/DSS v4 key format)
- **dh-sign-only** (the Diffie-Hellman/DSS v4 key format with no automatically generated subkey).
--manual-import-keys
Changes the behavior of PGP Command Line when keys are found during import operations. The default is all. The available settings are:

- **off** (do not import keys)
- **merge** (only merge the key if it already exists on the local keyring)
- **new** (import the key if it does not exist on the local keyring)
- **all** (import/merge all keys found)

Example:

```
pgp --import key.asc --manual-import-keys merge
```
Merge existing keys only.

--overwrite
Determines what to do when an operation tries to create an output file but it exists. The default is off.

Options are:

- **off** (return an error if the file exists)
- **remove** (delete the existing file)
- **rename** (rename the current output file and try again; existing files are left alone)
- **wipe** (wipe the existing file)

When the rename option is in use, PGP Command Line renames files by adding a number to the filename (for example, /dir/file.ext becomes /dir/file.x.ext, where x is a number from 1 to 10,000). If 10,000 renamed files is surpassed, an error is returned.

--sig-type
Specifies the signature type when signing user IDs. Default is local. See --sign-key and --sign-userid for more information.

Options are:

- **local** (non-exportable signature)
- **exportable** (exportable signature)
- **meta-introducer** (non-exportable meta-introducer signature)
- **trusted-introducer** (exportable trusted introducer signature)

--sort-order, --sort
Changes the sort order for writing key lists. This option accepts the following arguments:
- **any.** Key order is not changed at all.
- **creation.** Sort by creation date.
- **email.** Sort by email address of the primary user ID.
- **expiration.** Sort by expiration date.
- **keyid.** Sort by key ID.
- **keyszie.** Sort by key size.
- **subkeyszie.** Sort by subkey size.
- **trust.** Sort by trust.
- **userid.** Sort by primary user ID.
- **validity.** Sort by validity.

Key ID sorting does not work as expected, because keys are sorted by their 64-bit key IDs while PGP Command Line generally shows the 32-bit key ID.

Example:

```
pgp --list-keys --sort-order email
RSA4 pair 2048/2048 [VI--A] 0x3E439B98 Alice Cameron
  <alice@example.com>
RSA4 pair 2048/2048 [VI--A] 0x6245273E Bob Smith <bob@example.com>
RSA4 pair 2048/2048 [VI--A] 0x5571A08B Fumiko Asako
  <fumiko@example.com>
RSA4 pair 2048/2048 [VI--A] 0xF6EFC4D9 Jose Medina
  <jose@example.com>
```

---

**--tar-cache-cleanup**

Specifies how PGP Command Line removes a temporary TAR cache file.

TAR cache files are stored encrypted, so leaving them on the system is a minimal security risk. If wipe is used, the number of passes is taken from **--wipe-temp-passes**.

Options are:

- **off.** leaves the TAR cache file on the system.
- **remove.** removes any TAR cache files from the system.
- **wipe.** securely wipes any TAR cache files from the system.

The default is **remove**.

Example:

```
pgp --decrypt --archive.pgp ... --tar-cache-cleanup off
```

The temporary TAR cache files are left on the system.
--target-platform

Specifies the platform on which a SDA can decrypt itself.

The default is current platform. This option is used with --encrypt and --sda, such as:

```
pgp --encrypt <SDA> --sda --target-platform <platform>
```

The OS platforms for which the files can be encrypted are:

- win32 (Windows)
- linux (Linux)
- solaris (Solaris)
- aix (AIX)
- hpux (HP-UX)
- osx (Mac OS X)

Example:

```
pgp -e report.txt -r Bob --passphrase "smit4" --target-platform hpux report.txt:encrypt (0:output file report.txt.pgp)
```

This command produced the encrypted file "report.txt.pgp" prepared for the HP-UX platform.

--temp-cleanup

Determines what to do when an operation tries to remove a temporary file. The default is wipe.

Options are: off (leave temporary files behind), remove (remove temporary files), and wipe (wipe temporary files).

The remove option is recommended for large encryptions, as it will speed up the process.

Removing temporary files does not occur under some circumstances. It will occur if the output from an operation could not be moved into place or if the output file is on another file system than the temporary file.

--trust

Sets the trust for the current operation. This option is required when --set-trust is used.
See --set-trust for more information.

Trust options are: never (the key is never trusted), marginal (the key is marginally trusted), complete (the key is fully trusted), implicit (the key has ultimate trust).

Example:

```
pgp --set-trust key --trust complete
```
String Options

String options are options that take a single string as an argument. This argument is required in all cases.

In certain cases, white space is required in an argument; in these cases, double quotes must be used to enclose the entire argument.

--auth-username / --auth-passphrase

Specifies an authentication username and passphrase for logging in to a server that is not a keyserv er nor a proxy server. These options are generally used to log into a PGP Universal Server for key reconstruction. The default is not set.

--auth-passphrase is secure, --auth-username is not secure.

Example:

```plaintext
pgp --key-recon-recv ... --auth-username "acameron"
--auth-passphrase "Acam$r0n-Alic@
```

The login credentials of username “acameron” and passphrase ‘Acam$r0n-Alic@” are being used to log in to a PGP Universal Server for key reconstruction.

--city, --common-name, --contact-email, --country

Specifies the data when making a certificate signing request (CSR). Used with --export and --export-key-pair.

--comment

Specifies a comment string to be used in armored output blocks. The default is not set. This option is not secure.

Strings with spaces in them must be in quotes. When this option is not set, an empty comment header is not shown.

You can also set this option in the PGP Command Line configuration file; see “Configuration File” on page 36 for more information.

Example:

```plaintext
pgp ... --comment "Insert this comment..."
```

Calls for a comment of “Insert this comment...” in the current operation.

--creation-date

Changes the date of creation for the current operation. The default is unset (today). This option is not secure. See --creation-days for more information.

Dates must be in the format YYYY-MM-DD (month and day can be a single digit; no leading zero is required). You cannot use --creation-date and --creation-days for the same operation. Using --creation-date changes the behavior of --expiration-days. Dates beyond 2037-12-31 are not allowed.
Examples:

1. `pgp --gen-key test ... --creation-date 2006-12-27`
   Key will be valid starting on Dec. 27, 2004.

2. `pgp --gen-key test ... --creation-date 2006-7-14`
   Key will be valid starting on July 4, 2005.

**--default-key**

Specifies the default key to use for **--sign** and for **--encrypt-to-self**. As this is a signing key, it must be able to sign. The ability to encrypt is good, but not required. If the key can encrypt, it will be used for **--encrypt-to-self**. If it can’t encrypt, a warning is generated.

If a default key is not specified, PGP Command Line searches for a key to use as the default. PGP Command Line looks for the most recently created that can sign; encryption is not required. This option is not secure.

You can specify the default key in either of several ways:

- User ID: a case insensitive substring search of all user IDs on the local keyring. Not recommended, as you must match exactly one key.
- 32-bit key ID
- 64-bit key ID

You must make an exact match to exactly one key. The matched key must be able to encrypt and sign.

**--expiration-date**

Changes the date of expiration for the current operation. The default is not set (no expiration). This option is not secure. See **--expiration-days** for more information.

Dates must be in the format YYYY-MM-DD (month and day can be a single digit; no leading zero is required). Dates beyond 2037-12-31 are not allowed.

You cannot use **--expiration-date** and **--expiration-days** for the same operation.

Example:

```
pgp --gen-key test ... --expiration-date 2005-1-16
```
Key expires on Jan. 16, 2005.

**--export-passphrase**

Specifies the passphrase to use when exporting PKCS12 data. The default is not set. This option is secure.

To specify no passphrase, use the empty string in double quotes: "". See **--export** for more information.
Example:

```bash
pgp --export key --sig cert --export-format pkcs12 --passphrase "keypass" --export-passphrase "newpass"
```

Specifies to use an export passphrase of "newpass".

---

**--home-dir**

Establishes where PGP Command Line looks for preference files, keyring files, and the random seed file. This option is not secure.

The default on Solaris and Linux is $HOME/.pgp/. On Windows, keyring files are stored in C:\Documents and Settings\<current user>\My Documents\PGP\ and data files (the random seed file and the configuration file) are stored in C:\Documents and Settings\<current user>\Application Data\PGP Corporation\PGP\. If you specify **--home-dir**, all PGP Command Line files will be stored in the directory you specify.

To use **--home-dir**, enter the path to the new home directory (with or without a trailing directory separator).

All files except preferences can be overridden.

Example:

```bash
pgp --list-keys --home-dir other-pgp-files/
```

Changes the home directory for this command to "other-pgp-files/".

---

**--local-user (-u), --user**

Specifies a local user to use for the current operation. The default is not set. This option is not secure.

This option can be specified in one of several ways:

- When matching keys:
  - User ID (a case insensitive substring search of all user IDs on the local keyring)
  - 32-bit key ID
  - 64-bit key ID

- When matching signatures:
  - User ID of the signer (if PGP Command Line has the signing key). User ID match is a case insensitive substring search.
  - 32-bit key ID
  - 64-bit key ID

- When matching X.509 certificates:
  - X.509 issuer long name
– 32-bit key ID (if PGP Command Line has the signing key)
– 64-bit key ID (if PGP Command Line has the signing key)

Example:

```
pgp --sign-key gold --signer "my test user" --passphrase "test"
```

Specifies the user "my test user" for this operation.

---license-name, --license-number, --license-organization, --license-email

These options specify various licensee information when requesting a license authorization.

The default is unset. These options are used with the command

```
--license-authorize.
```

- **--license-name** is the name of the person for whom the software is licensed
- **--license-number** is the number a user receives from PGP Corporation
- **--license-organization** is the organization of the licensee
- **--license-email** is the email of the person for whom the software is licensed.

This number is used to send license recovery emails and it cannot be changed once the license is authorized: if you don’t specify an email during licensing, the license recovery won’t be possible.

Be sure to enter these options correctly and also to write them down: if you need to update your license, you will need to enter the identical information again. To get more information, refer to the command **--license-authorize**.

Example:

```
pgp --license-authorize --license-name "Alice Cameron" --license-email "alice@example.com" --license-organization "Example Corporation" --license-number "5555-KMKM-44444-33MMM-MM000-000" authorization.txt
```

This command will generate a license for the user Alice with the given license number, using manual authorization and the previously saved license authorization file.

---new-passphrase

Specifies the new passphrase to use when changing a passphrase. The default is not set. This option is secure.

To specify no passphrase, specify an empty string in double quotes: "".

Example:

```
pgp --change-passphrase user --passphrase "oldpass" --new-passphrase "newpass"
```

Specifies a new passphrase of "newpass".
--organization, --organization-unit

Specifies the organization when making a certificate signing request (CSR). Used with --export and --export-key-pair.

--output (-o)

Specifies the output location/object for the current operation. The default is not set; if a location/object cannot be determined from the input, an error is returned. This option is not secure.

Operations that require an output filename or directory and do not get it return an error. The exception to this rule is decoding files that have a suggested filename embedded in them. User-supplied output filenames will not be modified. You can specify the following:

- **File**, specify a file for output.
- **Directory**, specify to output the file into the directory named.
- **"-"**, a special keyword that means use the standard output.

Examples:

1. `pgp -er user file -o new`
   
   Output is an encrypted file called "new".

2. `pgp -er user file -o new.pgp`
   
   Output is an encrypted file called "new.pgp".

--output-file

Sets a file to use for output messages. The file name can be supplied with or without path information. The output file is created when PGP Command Line is initialized, even if no date is written to it. If you want to override the preferences settings and write to file to the default location, use the value "-" for the output file name.

Default is unset (output messages are written to stdout by default).

Examples:

1. `pgp --list-keys --output-file output.txt`
   
   The file containing key listing is written to "output.txt"

2. `pgp --list-keys --output-file -`
   
   In this case, the key list is displayed on the screen.

--passphrase

Specifies a passphrase to use for the current operation. The default is not set. This option is secure. To specify no passphrase, specify an empty string in double quotes: "".

Example:

```
pgp --decrypt file.txt.pgp --passphrase "test"
```
Specifies a passphrase of 'test' for this operation.

--preferred-keyserver

Specifies a preferred keyserver. The default is not set. This option is not secure. To remove a keyserver, use `--remove-preferred-keyserver`.

Prefixes supported are:

- `http://`
- `https://`
- `ldap://`
- `ldaps://`
- `ldapx509://`
- `ldapsx509://`

Example:

```
pgp --add-preferred-keyserver user --preferred-keyserver ldap://keyserver.pgp.com
```

Specifies ldap://keyserver.pgp.com as the preferred keyserver.

--private-keyring

Changes the location of the private keyring file. The default order for keyring search is: specified in configuration file, then home directory/secring.skr. This option is not secure.

This option always specifies a file. Relative or absolute path information can be included, but the target must still be a file.

You can also set the location in the PGP Command Line configuration file; refer to “Configuration File” on page 36 for more information.

You can specify a single file, relative path, or full path:

- **File**, relative to the personal directory
- **Relative path**, relative to the current directory
- **Absolute path**, recommended usage

Examples:

1. `pgp --private-keyring /home/dave/.pgp/secring-backup.skr`
   Absolute path to the private keyring file.

2. `pgp --private ./secring.skr`
   Relative path to the private keyring file.
--proxy-password, --proxy-server, --proxy-username
These options specify login credentials for a proxy server and are used with
--license-authorize. The default is unset.
- --proxy-password specifies login credentials to a proxy server.
- --proxy-server specifies a proxy server for certain network operations. If this
  server is not supplied, PGP Command Line makes a direct connection.
- --proxy-username specifies login credentials for a proxy server.
Example:
```
pgp --license-authorize --license-name "Alice Cameron"
    --license-email "alice@example.com" --license-organization
    "Example Corporation" --license-number
    "5555-KMKM-44444-33MMM-MM000-000" --proxy-server "http://
192.168.1.98:9000/" --proxy-username alice --proxy-password
alice2
```
The user Alice has licensed her copy of PGP Command Line 9.0 over the proxy
server at http://192.168.1.98:9000, using her proxy user name and
password.

--public-keyring
Changes the location of the public keyring file. The default order for keyring search is:
specified in configuration file, then home directory/pubring.pkr. This option is not secure.
This option always specifies a file. Relative or absolute path information can be included,
but the target must still be a file.
You can also set the location in the PGP Command Line configuration file; refer to
"Configuration File" on page 36 for more information.
You can specify a single file, relative path, or full path:
- File, relative to the personal directory
- Relative path, relative to the current directory
- Absolute path, recommended usage
Examples:
1. `pgp --public-keyring /home/dave/.pgp/pubring-backup.pkr`
   Absolute path to the public keyring file.
2. `pgp --keyring ./pubring.pkr`
   Relative path to the public keyring file.

--recon-server
Specifies a PGP Universal Server to use for key reconstruction.
If a reconstruction server is not established, PGP Command Line uses the preferred keyserver for the key. This option is not secure.

The default is not set.

Example:

```
pgp --key-recon-send ... --recon-server 10.1.1.45
```

Uses the PGP Universal Server with IP address 10.1.1.45 for key reconstruction.

**--regular-expression**

Specifies a regular expression. The default is not set. This option is not secure. Regular expressions are attached to trusted-introducer signatures as domain restrictions.

Example:

```
pgp --sign-key 0x12345678 --signer "Alice C" --sig-type trusted-introducer --passphrase "Sam_Gamgee" --regular-expression example.com.
```

Restricts trusted introducer signatures to the domain example.com.

**--random-seed**

Sets the location of the random seed file. The default random seed file is randseed.rnd, located in the home directory. This option is not secure. You can specify a single file, relative path, or full path:

- **File**, relative to the home directory
- **Relative path**, relative to the current directory
- **Absolute path**, recommended usage

If the path specified does not exist, the file will not be created. No warning or error is generated in this case.

Example:

```
pgp --list-keys --random-seed /home/user/.pgp-other/randseed.rnd
```

Specifies a directory location for the random seed file.

**--root-path**

Specifies a root path (directory path information) when creating SDAs and archives. The root path will be removed from any input files added to SDAs and archives. The default is unset.

If the files `root/path/dir/file` and `root/path/dir/file2` are added with root path set to "root/path", you will get these files in the archive: `dir/file` and `dir/file2`. 
--share-server

Specifies a server to use when sending split key shares over the network and is used with --send-shares. The default is unset.

For more information, refer to --send-shares.

--state

Specifies the state when making a certificate signing request (CSR). Used with --export and --export-key-pair.

--status-file

Sets a file to use for status messages. The status file is posted in the current working directory, unless a specific path information is added to the file name. This file is created on initialization even if no data is written to it. The special value of "-" can be used to override the preferences setting and to write to the default location.

Note that success messages are sent to the same location as error messages. The default is unset.

Examples:

1. `pgp -er "Bob Smith" newnote.txt --status-file status.log`
   The file "status.log" was created in the home directory. If you open this file, you will find the error message for the operation, which in this case is the following one:

   newnote.txt:encrypt (3013:no keys found)

2. `pgp -er "Bob Smith" newnote.txt --status-file logs\status1.log`
   In this case, the file "status1.log" was created in the directory "logs." If you open this file, you will find the same error message as above:

   newnote.txt:encrypt (3013:no keys found)

3. `pgp -er "Bob Smith" newnote.txt --status-file -`
   By using the value "-" as the status file name, you will get the error message displayed on the screen (which is the default location in this case).

--symmetric-passphrase

Specifies the symmetric passphrase to use for encryption, decryption, or verification. The default is not set. This option is secure.

You must enter a passphrase.

When decrypting, PGP Command Line will try all passphrases before giving up. This means that a symmetric passphrase specified with --passphrase will work correctly. This does not work for encryption, because PGP Command Line might need the normal passphrase to sign the data.
Examples:

1. `pgp -c file.txt --symmetric-passphrase "weak"
   Specifies a symmetric passphrase of "weak" for the specified file.

2. `pgp -c file.txt --symmetric-passphrase "this is a much $+r0ng3r pass code"
   Specifies a symmetric passphrase of "this is a much $+r0ng3r pass code" for the specified file.

--temp-dir

Specifies a temporary directory for PGP Command Line to use.

Setting --temp-dir to a different file system is not recommended for large operations.

This option is not secure. The default is the current directory.

You can specify a relative or absolute path:

- **Relative path**, relative to the current directory
- **Absolute path**, recommended usage

Example:

   `pgp --er user file --temp-dir /tmp`

   Specifies the use of /tmp as a temporary directory.
List Options

Lists are special cases of string options. They follow all the same rules, but there can be more than one of them defined at any given time.

--additional-recipient

Specifies an additional recipient for encryption. This option is configurable in the PGP Command Line configuration file; see “Configuration File” on page 36 for more information. The default is not set. This option is not secure.

If additional recipients are specified, they are required; if not found, an error is generated. A 32- or 64-bit key ID must be specified.

--adk

Specifies an ADK (Additional Decryption Key) and is used with --add-adk, --remove-adk, and --gen-key. The default is unset.

Example:

```
pgp --add-adk bob@example.com --adk jose@example.com
--passphrase "sm1t4"
0x6245273E:add ADK (0:ADKs successfully updated)
```

You have added an ADK (Jose Medina) to Bob’s key using Bob’s passphrase. If you check Bob’s key now, it will display the following:

```
pgp --list-key-details bob@example.com

………………

ADK: 0xF6EFC4D9 (0x90AC8366F6EFC4D9)
User ID: Jose Medina <jose@example.com>
Enforced: Yes
```

--input (-i)

Specifies the input location/object for the current operation. The default is not set (in some cases the default can be determined from the input; if not, an error is returned). This option is not secure.

The flag itself is optional. You can just specify the input on the command line without using the flag. If an operation requires input but does not get it, an error is returned.

The input can be as follows:

- File. Simply specify the file.
- Directory. Specify to put the file into the specified directory.
- "-". This is a special keyword that means use the standard input.

For operations that require input and get nothing, an error is returned.
Examples:

1. **pgp --verify file.txt.sig**
   
   The input, file.txt.sig, is entered on the command line without the flag.

2. **pgp --decrypt --input - --passphrase test < file.txt.pgp**
   
   Use the standard input, which is file.txt.pgp.

---

**--question / --answer**

Specify questions and answers for the key reconstruction feature.

The maximum length for a question is 95 characters; the maximum length for an answer is 255 characters. The minimum length for an answer is six characters. Both questions and answers should be in quotes.

--question is not secure; --answer is secure. The default is not set.

Example:

```
pgp --key-recon-send ... --question "What day were you born?"
--question "What is your mother’s maiden name?" ... --answer "Friday the 13th" --answer "Cameron"
```

Two questions and their answers are sent to the key reconstruction server.

---

**--keyserver**

Specifies a keyserver for the current operation. The default is not set. This option is not secure.

The basic format for **--keyserver** is `protocol://hostname:port/`. If you supply a keyserver on the command line, keyservers specified in the configuration file are ignored.

Depending on how your network is configured, certain ports in your corporate firewall may need to be opened to allow PGP Command Line to access external keyservers.

Supported protocols are:

- LDAP and LDAPPGP: LDAP PGP keyserver
- LDAPS and LDAPSPGP: LDAPS PGP keyserver
- HTTP: HTTP (hkp) keyserver
- LDAPX509: LDAP X.509 keyserver
- LDAPSX509: LDAPS X.509 keyserver

The hostname can be a hostname or an IP address. Port is optional; if not supplied, the default port for the protocol is used. The defaults are: LDAP, 389; LDAPS, 636; HTTP, 11371.
Example:

```
pgp --keyserver-send alice@example.com --keyserver ldap://keyserver.pgp.com
```

Use the public LDAP keyserver at pgp.com. No port is specified, so the default for the protocol will be used.

**--recipient (-r)**

Specifies a recipient for an encrypted message. The default is not set. This option is not secure.

Recipient lists support the same format as user IDs; see **--local-user** for more information.

Examples:

1. `pgp -er "ben" file.dat`
   Encrypt file file.dat to recipient Ben using the short forms of the commands.

2. `pgp --encrypt --recipient "dave" file.dat`
   Encrypt file file.dat to recipient Dave using the long forms of the commands.

3. `pgp -er "mike" -r "jim" -r "glen" file*.dat`
   Encrypt all files that match "file*.dat" to recipients Mike, Jim, and Glen.

**--revoker**

Specifies a revoker for a key and is used with the commands **--add-revoker**, **--remove-revoker**, **--gen-key**, and **--revoke** (third party revocation).

The default is **unset**.

Example:

```
pgp --add-adk bob@example.com --adk jose@example.com --passphrase "sm1t4"
0x6245273E:add ADK (0:ADKs successfully updated)
```

You added a revoker (Jose Medina) to Bob’s key by using Bob’s passphrase. If you check Bob’s key now, it will display the following:

```
pgp --list-key-details bob@example.com
```

```
```
Revoker: 0xF6EFC4D9 (0x90AC8366F6EFC4D9)
User ID: Jose Medina <jose@example.com>
```

--share

Specifies a share when splitting a key. The default is not set. This option is secure because a passphrase may be entered. Refer to “--split-key” on page 128 and “--join-key” on page 104 for more information about --share.

Usage:

Key split: <number of shares>:<user>[[:passphrase]]

Key join: <share file name>[[:passphrase]]

Where:

<number of shares> is required and must be one or more. This is the number of shares in the share file that counts towards the threshold when the key is being reconstituted. You can make all share files include one share, all share files include multiple shares, or you can assign different numbers of shares to different share files.

$user$ is required and can be specified by user ID, portion of the user ID, or key ID for a public key or by name if you want to conventionally encrypt the share. If a username includes a colon (:), it must be preceded by a backslash (\).

<share file name> is required; you can rename a share file if you wish. If a share file name includes a colon (:), it must be preceded by a backslash (\).

[:passphrase] is optional and is used to provide a passphrase for a conventionally encrypted share.

Examples:

pgp --split-key ... --share 1:0x1234abcd --share "1:Alice Cameron" --share 1:John

Specifies three shares to the specified key (not shown), one share to public key 0x1234abcd, one to the public key of Alice Cameron (which is shown in quotes as there is a space in the name), and one share to the public key of John. If an exact match to public keys is not made, the key will not be split.

pgp --split-key ... --share 1:conventionaluser:passphrase --share "2:Alice Cameron" --share 1:0x1234abcd --share "1:Ming Pa <mingp@example.com>"

Specifies five shares to the specified key (not shown), two to "conventionaluser", one to Alice Cameron, and two to public key 0x1234abcd. If the threshold were three, then Alice Cameron could reconstitute the key with any of the others; if Alice's share wasn't available, then all three of the others would need to provide their shares.

pgp --join-key ... --share ming-1-recip1.shf --share alice-2-recip2.shf --share maria-3-recip3.shf

Specifies the three files that need to be joined to reconstitute the key that has been split (not shown).
File Descriptors

These options are very similar to the integer options except that PGP Command Line reads from the file descriptor supplied.

--auth-passphrase-fd, --auth-passphrase-fd8
Sets --auth-passphrase to the data that is read from a descriptor. The default is not set. These options are secure. Requires a positive integer.

These options read double byte characters on Windows and UTF-8 on UNIX. The version of this option that ends with “8” will read UTF-8 on Windows, but has no effect on UNIX since UTF-8 is already being read there.

Example:

```plaintext
pgp ... --auth-passphrase-fd 7
Read authorization passphrase from file descriptor 7.
```

--export-passphrase-fd, --export-passphrase-fd8
Sets --export-passphrase to the data that is read from a descriptor. The default is unset. This option is secure. Requires a positive integer.

These options read double byte characters on Windows and UTF-8 on UNIX. The version of this option that ends with “8” will read UTF-8 on Windows, but has no effect on UNIX since UTF-8 is already being read there.

Example:

```plaintext
pgp ... --export-passphrase-fd 7
Read export passphrase from file descriptor 7.
```

--new-passphrase-fd, --new-passphrase-fd8
Sets --new-passphrase to the data read from a file descriptor. The default is not set. This option is secure. Requires a positive integer.

Reads double-byte characters on Windows and UTF-8 on UNIX. The version of the option that ends with “8” reads UTF-8 on Windows; this has no effect on UNIX, as UTF-8 is already being read there.

Example:

```plaintext
pgp ... --new-passphrase-fd 7
Read new passphrase from file descriptor 7.
```

--passphrase-fd, --passphrase-fd8
Sets --passphrase to the data read from a file descriptor. The default is not set. This option is secure. Requires a positive integer.
Reads double-byte characters on Windows and UTF-8 on UNIX. The version of the option that ends with "8" reads UTF-8 on Windows; this has no effect on UNIX, as UTF-8 is already being read there.

Example:

```
pgp ... --passphrase-fd 7
```

Read passphrase from file descriptor 7.

**--proxy-passphrase-fd, --proxy-passphrase-fd8**

Sets **--proxy-passphrase** to the data that is read from a descriptor. The default is not set. These options are secure. Requires a positive integer.

These options read double-byte characters on Windows and UTF-8 on UNIX. The version of this option that ends with "8" will read UTF-8 on Windows, but has no effect on UNIX since UTF-8 is already being read there.

Example:

```
pgp ... --proxy-passphrase-fd 7
```

Read proxy passphrase from file descriptor 7.

**--symmetric-passphrase-fd, --symmetric-passphrase-fd8**

Sets **--symmetric-passphrase** to the data that is read from a file descriptor. The default is unset. This option is secure. Requires a positive integer.

These options read double-byte characters on Windows and UTF-8 on UNIX. The version of this option that ends with "8" will read UTF-8 on Windows; this has no effect on UNIX, as UTF-8 is already being read there.

Example:

```
pgp ... --symmetric-passphrase-fd 7
```

Read symmetric passphrase from file descriptor 7.
Lists
How to Understand PGP Command Line Listings

This appendix provides details about the information that PGP Command Line displays in the following lists:

- the basic key list (page 187)
- the detailed key list (page 193)
- the detailed key list in XML format (page 206)
- the detailed signature list (page 213)

Basic Key List

Three PGP Command Line commands display information about the keys on the local keyring in basic output mode: --list-keys, --list-userids, and --list-sigs.

- --list-keys displays the primary user IDs of keys that match the input
- --list-userids displays all user IDs of keys that match the input
- --list-sigs displays all user IDs and signatures of keys that match the input

If you run any of these commands with no user ID or key ID information, all keys on the keyring will be displayed. If you enter any user or key ID information, only keys that match that some or all of that information will be displayed.

For example, enter the following command:

```bash
pgp --list-sigs "bob@example.com"
```

PGP Command Line responds with information about the key that has a key ID of 0x1234ABCD if that key is on the local keyring. If the key with that key ID is not on the local keyring, PGP Command Line responds with “0 keys found.”

If the key is found, PGP Command Line responds with something like:

```
Alg  Type Size/Type Flags   Key ID     User ID
----- ---- --------- ------- ---------- -------
RSA4 pair 2048/2048 [VI--A] 0x6245273E Bob Smith <bob@example.com>
   RSA sig                  [ -- ] 0x6245273E Bob Smith <bob@example.com>
```

1 key found

This response is a basic output mode listing showing the primary user ID, a secondary user ID, and a signature for one key. This section tells you what this information is and what it means.
The Default Key Column

The very first character in the display is called the default key column. It has no heading text.

For the primary user ID, the default key column can have an asterisk (*) or be blank:

- An asterisk (*) in the default key column indicates this key is the default key on the keyring.
- Nothing in the default key column ("") indicates this key is not the default key on the keyring.

The default key column is always blank for secondary user IDs and signatures.

The Algorithm Column

Characters 2 through 5 are the algorithm column. The heading text is “Alg.”

For the primary user ID, the algorithm column can display:

- **DSS** to indicate a DH/DSS key.
- **RSA1** to indicate a v1 RSA key (a very old version).
- **RSA2** to indicate a v2 RSA key (a very old version).
- **RSA** to indicate a v3 RSA key, also called an RSA Legacy key.
- **RSA4** to indicate a v4 RSA key.
- **RSAe** to indicate an RSA encrypt-only key.
- **RSAe** to indicate an RSA sign-only key.
- **RSA?** to indicate an RSA key of unknown version.
- **ECe** to indicate an elliptic curve encryption key (not currently supported)
- **ECs** to indicate elliptic curve signing key (not currently supported)
- **0xXY** to indicate an unknown key algorithm < 256 (YY is the algorithm ID in hexadecimal).
- **UNK** to indicate an unknown key algorithm >= 256.

For the secondary user IDs, the algorithm column is always blank.

For a signature, the algorithm column can display the following:

- **X509** to indicate an X.509 signature.
- **DSS** to indicate a DSS signature.
- **RSA** to indicate an RSA signature.
- **0xXY** to indicate an unknown key algorithm < 256 (YY is the algorithm ID in hexadecimal)
**The Type Column**

Characters 7 through 10 are the type column. The heading is "Type".

For the primary user ID, the type column can display:

- **pub** to indicate a public key
- **pair** to indicate a key pair.
- **split** to indicate a split key.

For the secondary user IDs, the type column always shows **uid**.

For a signature, the type column can display:

- **sig** to indicate a signature in which the signer’s key is known (on the local keyring).
- **sig?** to indicate a signature in which the signer’s key is unknown.
- **sigX** to indicate a corrupt or damaged signature.

**The Size/Type Column**

Characters 12 through 20 are the size/type column. The heading is "Size/Type".

For the primary user ID, the size/type column can display:

- DSS key with no subkey, shows the size of the signing DSS key.
- RSA v4 key with no subkey shows:
  - **ssss** indicates signing key bits greater than or equal to 1,000.
  - **sss** indicates signing key bits less than 1,000.
  - **ssss** indicates signing key bits greater than or equal to 10,000.
  
  The "s" characters are replaced with actual values.
- DSS or RSA v4 key with subkey present shows:
  - **eeeee/ssss** indicates encryption key (subkey) bits followed by signing key bits.
  - **eeeee/ssss** if encryption key bits are less than 1,000.
  - **eeeee/sss** if signing key bits are greater than 1,000.
  - **eeeee/sss** if both bits are greater than 1,000.
  - ******/ssss** if encryption key bits are greater than or equal to 10,000.
  - **eeeee/****** if signing key bits are greater than or equal to 10,000.
  - ******/****** if both bits are greater than or equal to 10,000.
The "s" and "e" characters are replaced with actual values.

- RSA non-v4 key shows:
  - bbb if key bits are greater than or equal to 1,000.
  - bbb if key bits are less than 1,000.
  - bbbbb if key bits are greater than or equal to 10,000.

The "b" characters are replaced with actual values.

For the secondary user IDs, the size/type column can display:

- Blank for a normal user ID.
- photo for a photo user ID.

For a signature, the size/type column can display:

- Blank for an exportable signature or a meta- or trusted-introducer signature.
- private for a non-exportable signature or a meta- or trusted-introducer signature.

**The Flags Column**

Characters 22 through 28 are the flags column. The header is "Flags".

The --marginal-as-valid setting does not affect this display.

For the primary user ID, the secondary user IDs, and a signature, the flags column can display:

- Column 1: Delimiter
  - [ is always shown.

- Column 2: Validity
  - V indicates a fully valid key.
  - v indicates a marginally valid key.
  - - indicates an invalid key
  - ? indicates unknown validity.

- Column 3: Trust
  - I indicates an implicitly trusted key.
  - T indicates a fully trusted key.
  - t indicates a marginally trusted key.
  - - indicates an untrusted key.
  - ? indicates unknown trust.
\*\*Column 4: Revoked\*

\* R \* indicates a revoked key.

\* r \* indicates a unverified revoked key.

\* - \* indicates a non-revoked key.

\* Column 5: Disabled/Expired\*

\* E \* indicates an expired key (or an expired and disabled key).

\* D \* indicates a disabled key.

\* - \* indicates an active key.

\* Column 6: ADK\*

\* A \* indicates ADKs present on the key

\* - \* indicates an ADK is absent

\* Column 7: Delimiter\*

\* ] \* is always shown.

\*\*Note: To see the value affected by the option \*--marginal-as-valid\*, use the command \*--list-key-details\*.\*

**The Key ID Column**

Characters 30 through 39 are the key ID column. The header is "Key ID".

For the primary user ID, the key ID column displays:

\* The 32-bit hexadecimal key ID with an "0x" prefix and numbers and/or capital letters. For example: \*0xB2726BDF\*.

For the secondary user IDs, the key ID column is always blank.

For a signature, the key ID column displays:

\* For the key ID of the signer, which is always available, the 32-bit hexadecimal signing key ID with an "0x" prefix and numbers and/or capital letters.

\* For an X.509 signature when the signing key is found, the 32-bit hexadecimal signing key ID with an "0x" prefix and numbers and/or capital letters.

\* For an X.509 signature where the signing key is not found, the column is blank.
The User ID Column

Characters 41 through the end of the line are the user ID column. The heading is "User ID".

For the primary user ID, the user ID column displays the primary user ID. For example: Alice Cameron <ac@example.com>.

For the secondary user IDs, the user ID column displays the user ID string. For example, Alice C <alice@example.com>.

For a signature, the user ID column displays:

- For a PGP signature where the signing key has been found:
  User ID of the signer.

- For a PGP signature where the signing key has not been found:
  Blank if the signer is unknown.

- For an X.509 signature, which is always available:
  Long name of the issuer.
Detailed Key List

The `--list-key-details` command provides detailed information about the specified key.

If you run `--list-key-details` with no user or key ID information, all keys on the keyring are displayed. If you enter user or key ID information, only keys that match that some or all of that information will be displayed.

For example, enter the following command:

```
pgp --list-key-details "Bob Smith"
```

PGP Command Line responds with detailed information about Bob’s key. If that key is not on the local keyring, PGP Command Line responds with "0 keys found".

If the key is found, PGP Command Line responds with something like:

```
Key Details: Bob Smith <bob@example.com>
  Key ID: 0x6245273E (0xB9C0F8856245273E)
    Type: RSA (v4) key pair
    Size: 2048
  Validity: Complete
    Trust: Implicit (Axiomatic)
  Created: 2006-10-27
  Expires: Never
  Status: Active
  Cipher: AES-128
  Cipher: AES-192
  Cipher: AES-256
  Cipher: TripleDES
    Hash: SHA-256
    Hash: SHA-512
  Compress: Zip (Default)
  Photo: No
  Revocable: Yes
  Token: No
  Keyserver: None
  Default: No
  Wrapper: No
  Prop Flags: Sign user IDs
  Prop Flags: Sign messages
  Ksrv Flags: None
  Feat Flags: Modification detection
```
Notation: 01 preferred-email-encoding@example.com=pgpmime

Subkey ID: 0x894BA6DC (0xBAABB613894BA6DC)
  Type: RSA (v4)
  Size: 2048
  Created: 2004-10-27
  Expires: Never
  Status: Active
  Revocable: Yes
  Prop Flags: Encrypt communications
  Prop Flags: Encrypt storage
    ADK: 0xF6EFC4D9 (0x90AC8366F6EFC4D9)
    User ID: Jose Medina <jmedina@example.com>
    Enforced: Yes

    Revoker: 0xF6EFC4D9 (0x90AC8366F6EFC4D9)
    User ID: Jose Medina <jmedina@example.com>

1 key found

Unlike the basic key list, the detailed key list displays information in rows, not columns. The detailed key list is divided into four sections: main key details, subkey details, ADK details, and revoker details.

Main Key Details

Row 1: Primary User ID Name
Name: Key Details
Value: The primary user ID of the key.

Row 2: Key ID
Name: Key ID
Value: The 32-bit key ID followed by the 64-bit key ID in the format:

  0x12341234 (0x12341234ABCDABCD)

Key ID hexadecimal letters are always uppercase (except for the x in 0x).

Row 3: Key Type
Name: Type
First value:

- DSA means this is a DSA signing key (with or without subkeys).
- **RSA legacy (v1)** means this is an RSA v1 key.
- **RSA legacy (v2)** means this is an RSA v2 key.
- **RSA legacy (v3)** means this is an RSA v3 key (RSA legacy key).
- **RSA (v4)** means this is an RSA v4 key.
- **RSA encrypt only** means this is an RSA encrypt-only key.
- **RSA sign only** means this is an RSA sign-only key.
- **RSA (version unknown)** means this is an RSA key of unknown version.
- **Unknown algorithm ID 0xYY** means this is an unknown key algorithm (YY is the algorithm ID in hexadecimal)

Second Value:

- **public key** means this is a public key.
- **key pair** means this is a key pair (or private key only).
- **split key** means this is a split key pair.
  The second value string is appended to the first separated by a space.

**Row 4: Key Size**

Name: Size

Values:

- For keys that have a master key, the size in bits of that key.
- For legacy keys, the size in bits of the key.

There is no length restriction here as there is in basic mode.

**Row 5: Validity**

Name: Validity

Values:

- **Complete** means this is a valid key.
- **Marginal** means this is a marginally valid key.
- **Invalid** means the key is invalid.
- **Unknown** means the key has unknown validity.
- **Unknown 0xYY** means the key has a validity value that is not not handled by command line (YY is the value in hexadecimal)

Values (effective):

- **Complete** means this is a valid key.
- Invalid means the key is invalid.

Notes: For marginally valid keys, PGP Command Line displays two validity settings, the actual and the effective validity.

For example, the Marginal validity in the actual setting will depend on --marginal-as-valid in its effective setting. In most cases, there will be just one validity shown (the actual value).

**Row 6: Trust**

Name: Trust

Values:
- **Implicit** means this is an implicitly trusted key.
- **Complete** means this is a completely trusted key.
- **Marginal** means this is a marginally trusted key.
- **Never** means this is an untrusted key.
- **Undefined** means this key has an undefined trust value.
- **Unknown** means this is a key with an unknown trust value.
- **Unknown 0xYY** means this is a key with a trust value not handled by command line (YY is the value in hexadecimal)

Only key pairs can have implicit trust. The implicit and never states will have a suffix if the key is paired.

The **Implicit** and **Never** states will have a suffix if the key is paired, such as:
- **(Axiomatic)** when the key is axiomatic
- **(Not axiomatic)** when the key is not axiomatic

The normal states are
- **Implicit (Axiomatic)** and
- **Never (Not axiomatic)**

Other states are possible, but not common: they are caused by errors and can be fixed by changing the key trust, and then changing it back.

**Row 7: Creation Date**

Name: Created

Value:
- **yyyy-mm-dd** is the key’s creation date.

**Row 8: Expiration Date**

Name: Expires
Value:

- **never** means the key doesn’t expire.
- **yyyy-mm-dd** is the key’s expiration date.
- **unknown** means the expiration date of the key is unknown.

**Row 9: Status Fields**

Name: Status

Values:

- **Disabled** means this key is disabled.
- **Expired** means this key is expired.
- **Revoked** means this key has been revoked.
- **Unverified Revocation** means this key has been revoked, but the revocation is unverified.
- **Third Party Revocation** means the key was revoked by a third party.
- **Active** means the key has no status. If a key is active, there will be no other status lines.

One or more status characteristics can be shown one after the other if they apply. Revoked and unverified revocation are mutually exclusive.

**Row 10: Preferred Cipher**

Name: Cipher

The first preferred cipher row is the "preferred cipher."

Values:

- **IDEA** means IDEA is the preferred cipher for this key.
- **TripleDES** means 3DES is the preferred cipher for this key.
- **CAST5** means CAST5 is the preferred cipher for this key.
- **Blowfish** means Blowfish is the preferred cipher for this key.
- **AES-128** means AES 128 is the preferred cipher for this key.
- **AES-192** means AES 192 is the preferred cipher for this key.
- **AES-256** means AES 256 is the preferred cipher for this key.
- **Twofish-256** means Twofish 256 is the preferred cipher for this key.
- **Unknown 0xYY** means an unknown cipher (YY is the cipher algorithm ID in hexadecimal)
If a key has no preferred ciphers the default is used. For keys with versions less than 4 this is IDEA. For all other keys this is CAST5. One or more ciphers can be shown one after the other if they are set in the list.

**Row 11: Preferred Hash**

Name: Hash

Values:

- **MD5** means MD5 is the hash being used for this key.
- **SHA** means SHA is the hash being used for this key.
- **RIPEMD-160** means RIPEMD 160 is the hash being used for this key.
- **SHA-256** means SHA 256 is the hash being used for this key.
- **SHA-384** means SHA 384 is the hash being used for this key.
- **SHA-512** means SHA 512 is the hash being used for this key.
- **Unknown 0xYY** is an unknown hash (YY is the hash algorithm ID in hex)

If a key has no preferred hashes, the following default is used:

- **MD5** for keys with versions less than 4
- **SHA-1** for all other keys

In the case where the default is used, PGP Command Line appends the string "(Default)" to the hash.

One or more hashes can be shown one after the other if set on the list.

**Row 12: Preferred Compression Algorithm**

Name: Compress

Values:

- **Zip** means Zip is the preferred compression algorithm.
- **Zlib** means Zlib is the preferred compression algorithm.
- **Bzip2** means Bzip2 is the preferred compression algorithm.
- **Unknown. 0xYY** is an unknown compression algorithm (YY is the compression algorithm ID in hexadecimal)

If a key has no preferred compression algorithm, the default is used (Zip is the default in all cases). In this case, PGP Command Line appends the string "(Default)" to the compression algorithm.

One or more compression algorithms can be shown one after the other if they are set in the list.

**Row 13: Photo ID**
<table>
<thead>
<tr>
<th>Row</th>
<th>Name</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>14</td>
<td><strong>Revocable</strong></td>
<td>Name: Revocable</td>
</tr>
<tr>
<td></td>
<td>Values:</td>
<td>- Yes means one of the keys on the keyring can revoke this key.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- No means none of the key on the keyring can revoke this key.</td>
</tr>
<tr>
<td>15</td>
<td><strong>Token</strong></td>
<td>Name: Token</td>
</tr>
<tr>
<td></td>
<td>Values:</td>
<td>- Yes means part of all of this key is on a token</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- No means no part of this key is on a token</td>
</tr>
<tr>
<td>16</td>
<td><strong>Preferred Keyserver</strong></td>
<td>Name: Keyserver</td>
</tr>
<tr>
<td></td>
<td>Values:</td>
<td>- None means no preferred keyserver is set.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Keyserver name if there is a preferred keyserver set.</td>
</tr>
<tr>
<td>17</td>
<td><strong>Default Key</strong></td>
<td>Name: Default</td>
</tr>
<tr>
<td></td>
<td>Values:</td>
<td>- Yes means this is the default key for encrypting and signing.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- No means this is not the default key.</td>
</tr>
<tr>
<td>18</td>
<td><strong>X.509 Wrapper Key</strong></td>
<td>Name: Wrapper</td>
</tr>
<tr>
<td></td>
<td>Values:</td>
<td>- Yes if the key was created to contain an imported X.509 certificate</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- No if the key is normal</td>
</tr>
</tbody>
</table>
Row 19: Key Properties Flags

Name: Prop Flags

Values:

- **Sign user IDs** when the key can sign other user IDs
- **Sign messages** when the key can sign messages
- **Encrypt communications** when the key can encrypt communications
- **Encrypt storage** when the key can encrypt storage
- **Private split** when the private key is split
- **Private shared** when the private key is in the possession of a third party (group bit)
- **None** when the key has no properties flags set
- **Unknown (0xNNNNNNNN)** when one or more unknown key properties flags are set

If enabled, one or more properties can be shown one after the other in the following way:

- **Unknown** may be shown with other properties or by itself
- **None** will only be shown if there are no flags set
- If **Unknown** flags are set, they are shown in hexadecimal
- Any known flags are stripped before PGP Command Line displays the hexadecimal number

Row 20: Key Server Preferences Flags

Name: Ksrv Flags

Values:

- **No modify** when the key should not be modified except by the owner
- **None** when the key has no keys server preferences flags set
- **Unknown (0xNNNNNNNN)** when one or more unknown keys server preferences flags are set

If enabled, one or more preferences can be shown one after the other in the following way:

- **Unknown** may be shown with other properties or by itself
- **None** will only be shown if there are no flags set
- If unknown flags are set, they are shown in hexadecimal
- Any known flags are stripped before PGP Command Line displays the hexadecimal number
Note that there is currently only one flag.

**Row 21: Key Features Flags**

Name: Feat Flags

Value:

- **Modification detection**
- **None** when the key has no features flags set
- **Unknown (0xNNNNNNNN)** when one or more unknown key features flags are set

If enabled, one or more features can be shown one after the other in the following way:

- **Unknown** may be shown with other properties or by itself
- **None** will only be shown if there are no flags set
- If unknown flags are set, they are shown in hexadecimal
- Any known flags are stripped before PGP Command Line displays the hexadecimal number

Note that there is currently only one flag.

**Row 22: Notation Packs**

Name: Notations

Value:

- **None**

  ZZ 0xNNNNNNNN <name>=<value>

  ZZ 0xNNNNNNNN <name>=<binary data, length <length>>

Notes:

- One of more notations can be shown one after the other if they exist.
- **None** is displayed if there are no notation packets for the current key.
- **ZZ** is the index of the notation packet (starting with 01, 02, etc.).
- **0xNNNNNNNN** is the value of the flags portion of the notation packet.
- **<name>** and **<value>** are substituted for the actual data.
- The name is always printable UTF-8.
- If value is not printable then the second value line above is used.
- The value portion of this line is literal except that **<length>** is substituted.
Subkey Details

The subkey details section has either one or N rows:

**Row 1: Subkey ID**

Name: Subkey ID

Values:

- **N/A** indicates the key type does not support subkeys.
- **None** means the current key does not have any subkeys.
- 32-bit and 64-bit subkey IDs in the same format as for main key details

If the key type does not support subkeys or there are no subkeys on the current key, then no additional rows are shown.

**Row 2: Type**

Name: Type

Values:

- **ElGamal** means an Elgamal encryption key.
- **RSA (v4)** means an RSA v4 encryption key.
- **Unknown algorithm ID 0xYY** means an unknown subkey algorithm ID (YY is the ID in hexadecimal)

**Row 3: Size**

Name: Size

Value:

- Subkey size in bits.

There is no length restriction here as there is in the basic key list view.

**Row 4: Creation Date**

Name: Created

Value:

- Creation date (same format as for main key details)

**Row 5: Expiration Date**

Name: Expires

Value:

- Expiration date (same format as for main key details).

**Row 6: Status Fields**
Name: Status

Values:

- **Expired** means an expired key.
- **Revoked** means a revoked key.
- **Unverified Revocation** means an unverified revoked key.
- **Active** means an active key.

If a subkey has no status, it shows as active. One or more status characteristics can be shown one after the other, if they apply. Revoked and unverified revocation are mutually exclusive.

**Row 7: Revocable**

Name: Revocable

Values:

- **Yes** if one of the keys on the keyring can revoke this subkey.
- **No** if none of the key on the keyring can revoke this subkey

**Row 8: Key Properties Flags**

Name: Prop Flags

Values:

- **Sign user IDs** when the key can sign other user IDs
- **Sign messages** when the key can sign messages
- **Encrypt communications** when the key can encrypt communications
- **Encrypt storage** when the key can encrypt storage
- **Private split** when the private key is private split
- **Private shared** when the private key is in the possession of a third party (group bit)
- **None** when the key has no properties flags set
- **Unknown (0xNNNNNNNNN)** when one or more unknown key properties flags are set

If enabled, one or more properties can be shown one after the other in the following way:

- **Unknown** may be shown with other properties or by itself
- **None** will only be shown if there are no flags set
- If unknown flags are set, they are shown in hexadecimal
- Any known flags are stripped before PGP Command Line displays the hexadecimal number
ADK Details

ADK details uses either one or three rows. If there is no ADK on the key, then you see just one row: **ADK: None**.

If there is an ADK on the key, you see three rows:

**Row 1: ADK Key ID**

Name: ADK

Values:
- 32-bit subkey ID
- 64-bit subkey IDs

**Row 2: ADK Primary User ID**

Name: User ID

Values:
- Primary User ID of the ADK.
- Blank if the ADK is not found on the local keyring.

**Row 3: Enforced**

Name: Enforced

Values:
- **Yes** if the ADK is set to be enforced.
- **No** if the ADK is not be enforced.
- **Unknown 0xNN** if the ADK has some other unknown setting.

Revoker Details

Revoker details uses either one or two rows. If there is no revoker on the key, then you see just one row: **Revoker: None**.

If there is a revoker on the key, you see two rows:

**Row 1: Revoker Key ID**

Name: Revoker

Values:
- 32-bit subkey ID
- 64-bit subkey IDs

**Row 2: Revoker Primary User ID**

Name: User ID
Values:

- Primary User ID of the revoker.
- Blank if the key is not found on the local keyring.
Key List in XML Format

When you choose to list a key in XML format, PGP Command Line will display all information including all user IDs and signatures. You can also specify a single key to view in XML format.

To list keys in XML format, you may use either the command `--list-keys-xml`, or a key list operation with the added option `--xml`, such as `--list-keys user1 --xml`, or `--list-keys --xml`.

If no users are specified, the command lists all keys on the local keyring.

Example:

```
pgp --list-keys-xml "Jose Medina"
```

Here is a typical key list (for the user Jose Medina) in XML format, with short explanations in brackets. Elements with several fixed choices are listed after the example.

```xml
<?xml version="1.0"?>  (exactly one element)
<keyList>  (exactly one element)
<key>  (zero or more elements)
  <keyID>0xCCFA35EC</keyID>
  <keyID64>0x3A76B511CCFA35EC</keyID64>
  <algorithm>RSA</algorithm>
  <version>4</version>
  <type>pair</type>
  <size>2048</size>
  <validity>complete</validity>
  <trust>implicit</trust>
  <creation>2004-10-19</creation>
  <expiration/>
  <revoked>false</revoked>
  <unverifiedRevocation>false</unverifiedRevocation>
  <thirdPartyRevocation>false</thirdPartyRevocation>
  <expired>false</expired>
  <disabled>false</disabled>
  <revocable>true</revocable>
  <preferredKeyserver/>
  <preferredCipherAlgorithms>
    <cipher>  (one or more elements)
      <name>AES-128</name>
      <value>7</value>
    </cipher>
  </preferredCipherAlgorithms>
</key>
</keyList>
```
<priority>1</priority>
<default>false</default>
</cipher>
</preferredCipherAlgorithms>
<preferredHashAlgorithms> (one or more elements)
<hashAlgorithm>
  <name>SHA-256</name>
  <value>8</value>
  <priority>1</priority>
  <default>false</default>
</hashAlgorithm>
</preferredHashAlgorithms>
<preferredCompressionAlgorithms> (one or more elements)
<compressionAlgorithm>
  <name>Zip</name>
  <value>1</value>
  <priority>1</priority>
  <default>true</default>
</compressionAlgorithm>
</preferredCompressionAlgorithms>
</token>
<defaultKey>false</defaultKey>
<X509WrapperKey>false</X509WrapperKey>
<fingerprint>C984E2FB2BAAB8A02F61B8273A76B511CCFA35EC</fingerprint>
</keyProperties>
<signUserIDs>true</signUserIDs>
<signMessages>true</signMessages>
<encryptCommunications>false</encryptCommunications>
<encryptStorage>false</encryptStorage>
<privateSplit>false</privateSplit>
<privateShared>false</privateShared>
<unknown>0x00000000</unknown> (same rules as --list-key-details)
</keyProperties>
</keyServerPreferences>
<noModify>false</noModify>
<unknown>0x00000000</unknown>
</keyServerPreferences>

<keyFeatures>
  <modificationDetection>true</modificationDetection>
  <unknown>0x00000000</unknown>  (same rules as --list-key-details)
</keyFeatures>

<userID>(one or more elements)
  <name>Jose Medina</name>
  <commonName>Jose Medina</commonName>
  <contactName/>
  <type>primary</type>
  <validity>complete</validity>
  <revoked>false</revoked>
  <signature>
    <signerKeyID>0xCCFA35EC</signerKeyID>
    <signerKeyID64>0x3A76B511CCFA35EC</signerKeyID64>
    <signerName>Jose Medina</signerName>
    <signerCommonName>Jose Medina</signerCommonName>
    <signerContactName/>
    <algorithm>RSA</algorithm>
    <type>signature</type>
    <exportable>true</exportable>
    <revoked>false</revoked>
    <expired>false</expired>
    <corrupt>false</corrupt>
    <creation>2004-10-19</creation>
    <expiration/>
    <trustDepth>0</trustDepth>
    <domainRestriction/>
  </signature>
</userID>

<subkey> (zero or more elements)

<subkeyID>0x0E948D0B</subkeyID>
  <subkeyID64>0x152393F70E948D0B</subkeyID64>
  <algorithm>RSA</algorithm>
  <version>4</version>
  <size>2048</size>
<creation>2004-10-19</creation>
<expiration/>
<revoked>false</revoked>
<unverifiedRevocation>false</unverifiedRevocation>
<expired>false</expired>
<revocable>true</revocable>

<subkeyProperties>
  <signUserIDs>false</signUserIDs>
  <signMessages>false</signMessages>
  <encryptCommunications>true</encryptCommunications>
  <encryptStorage>true</encryptStorage>
  <privateSplit>false</privateSplit>
  <privateShared>false</privateShared>
  <unknown>0x00000000</unknown>  (same rules as --list-key-details)
</subkeyProperties>

<subkey>
</subkey>
<adk>  (zero or more elements)
  <keyID>0xAF3D2BB8</keyID>
  <keyID64>0x183ED5C6AF3D2BB8</keyID64>
  <name>Example Corp Additional Decryption Key</name>
  <commonName>Example Corp Additional Decryption Key</commonName>
  <contactName/>
  <class>
    <setting>not enforced</setting>
    <value>0x00</value>
  </class>
</adk>
<revoker>  (zero or more elements)
  <keyID>0x14A96E62</keyID>
  <keyID64>0x4B2AA68CE14A96E62</keyID64>
  <name/>
  <commonName/>
  <contactName/>
</revoker>
</key>
</keyList
Elements with fixed settings

Algorithm
Key encryption algorithms appear in the following sections:

<key> section
   RSA | DSS

<signature> section
   RSA | DSS | X.509

<subkey> section
   RSA | Elgamal

For more details about key encryption algorithms refer to `--list-key-details`.

Type
Key types appear in the following sections:

<key> section
   public | split | pair

<userID> section
   primary | secondary | photo

<signature> section
   signature | trusted-introducer | meta-introducer

For more details about key types refer to `--list-key-details`.

Validity
Key validity types appear in the following sections:

<key> section
   complete | marginal | invalid | unknown

<userID> section
   complete | marginal | invalid | unknown

For more details about key validity refer to `--list-key-details`.

Trust
Key trust types appear as follows:

   implicit | complete | marginal | never | undefined | unknown | invalid
For more details about key trust refer to **--list-key-details**.

**Hash**

Key hash algorithm types appear as follows:

| MD5  | SHA  | RIPEMD-160 | SHA-256 | SHA-384 | SHA-512 | invalid | unknown |

For more details about key hash algorithms refer to **--list-key-details**.

**Cipher**

Key cipher algorithm types appear as follows:

<cipher> section

| none | IDEA | TripleDES | CAST5 | Blowfish | AES-128 | AES-192 | AES-256 | Twofish-256 | unknown |

**Key compression algorithm types appear as follows:**

<compressionAlgorithm> section

| Zip   | ZLIB | BZIP2     |

For more details about compression algorithms refer to **--compression-algorithm**.

**Setting**

Key settings appear as follows:

<class> section (in the <adk> section)

| not enforced | enforce | unknown |

**X.509 Signatures**

For X.509 signatures there are additional items under the <signature> heading. Currently these are:

- x509Name
- x509Issuer
- thisCRL
- nextCRL

Example:

This is an abbreviated example of an X.509 signature. Note that the signer key ID and signer name may not be known.

```xml
<?xml version="1.0"?>
<keyList>
```
<key>
...
</key>

<signature>
...
</signature>

<x509Name>CN=www.example.com, O=Example.com Inc., L=San Jose, ST=California, C=US</x509Name>
<x509Issuer>OU=Secure Server Certification Authority, O="RSA Data Security, Inc.", C=US</x509Issuer>
<thisCRL>1969-12-31</thisCRL>
<nextCRL>1969-12-31</nextCRL>
</signature>
</userID>
</key>
</keyList>
### Detailed Signature List

The `--list-sig-details` command provides detailed information about the signatures on the specified key.

When you run `--list-sig-details`, enter either the key ID or enough of the user ID so that only one key from the local keyring is specified. If more than one fits the criteria you enter, an error message will be returned.

For example, enter the following command:

```
pgp --list-sig-details "Bob Smith"
```

PGP Command Line responds with detailed information about the signatures on Bob's key. If the specified key is found, PGP Command Line responds with something like:

```
Signature Details: Bob Smith <bob@example.com>
  Signed Key ID: 0x6245273E (0xB9C0F8856245273E)
  Signed User ID: Bob Smith <bob@example.com>

  Signer Key ID: 0x6245273E (0xB9C0F8856245273E)
  Signer User ID: Bob Smith <bob@example.com>
    Type: RSA signature
    Hash: SHA-256
    Exportable: Yes
    Status: Active
    Created: 2004-11-09
    Expires: Never
    Trust Depth: 0
    Domain: None

1 signature found
```

Like the detailed key list, the detailed signature list displays information in rows.

#### Row 1: Primary User ID Name of the signed key

**Name:** Signature Details  

**Value:**

- The primary user ID of the key that contains the signature.

#### Row 2: Signed Key ID

**Name:** Signed Key ID  

**Value:**

- The 32-bit key ID followed by the 64-bit key ID in the format:
  
  0x12341234 (0x12341234ABCDABCD)

Key ID hexadecimal letters are always uppercase (except for the x in 0x).
**Row 3: Signed User ID**

Name: Signed User ID

Value:
- The name of the user ID to which the current signature belongs.

**Row 4: Signer Key ID**

Name: Signer Key ID

Value:
- PGP Signature (always available):
  The 32-bit key ID followed by the 64-bit key ID in the format:
  0x12341234 (0x12341234ABCDABCD)
- X.509 Signature (signing key found):
  The 32-bit key ID followed by the 64-bit key ID in the format:
  0x12341234 (0x12341234ABCDABCD)
- X.509 Signature (signing key not found):
  Empty

Key ID hexadecimal letters are always upper case (except for the x in 0x).

**Row 5: Signer User ID**

Name: Signer User ID

Value:
- PGP Signature (signing key found):
  The primary user ID of the signing key
- PGP Signature (signing key not found):
  Empty
- X.509 Signature (signing key found):
  The primary user ID of the signing key
- X.509 Signature (signing key not found):
  Empty

**Row 6: Signature type**

Name: Type

Value (algorithm ID):
- **DSA** means a signature by a DH/DSS key
- **RSA** means a signature by an RSA key
- **Unknown algorithm ID 0xYY** means a signature by an unknown algorithm ID (YY is the ID in hexadecimal)

Value (signature type):

- **signature** means a regular signature
- **trusted-introducer signature** means a trusted-introducer signature
- **meta-introducer signature** means a meta-introducer signature

Values are added together, with the algorithm ID first and the signature type second, such as:

- **DSA signature**
- **RSA trusted-introducer-signature**

**Row 7: Hash Algorithm**

Name: Hash

Values:

- **MD5** means MD5.
- **SHA-1** means SHA-1.
- **RIPEMD-160** means RIPEMD-160.
- **SHA-256** means SHA-256.
- **SHA-384** means SHA-384.
- **SHA-512** means SHA-512.
- **Invalid** indicates an invalid hash.
- **Unknown 0xYY** means unknown hash, where YY is the hash algorithm ID in hex.

**Row 8: Exportable Status**

Name: Exportable

Values:

- **Yes** means the signature is marked exportable.
- **No** means the signature is local to this keyring.

Trusted-introducer signatures are always exportable. Meta-introducer signatures are always local; that is, they are not exportable.

**Row 9: Signature Status**

Name: Status
Value:

- **Expired** means the signature is expired.
- **Revoked** means the signature is revoked.
- **Corrupt** means verification of the signature failed for some reason.
- **Active** means this is a verified good signature.

**Row 10: Creation date**

Name: Created

Value:

- yyyy-mm-dd is the date the signature was created.

**Row 11: Expiration date**

Name: Expires

Value:

- yyyy-mm-dd is the expiration date of the signature.
- **Never** means the signature does not expire.

**Row 12: Trust depth**

Name: Trust Depth

Value:

- A number, zero or greater

Regular signatures always have a trust depth of zero.

**Row 13: Domain restriction**

Name: Domain

Value:

- Regular expression domain restriction for this signature.

Domain restrictions can only be set for trusted-introducer signatures.

**Row 14: X509 Long Name**

Name: X509 Name

Value:

- X.509 Signature (always available): the DN used for the X.509 certificate

This row is not displayed for PGP signatures.

**Row 15: X509 Issuer**
Name: X509 Issuer
Values:
   - X.509 Signature (always available): the DN used for the issuer of the X.509 certificate

The row is not displayed for PGP signatures

**Row 16: This CRL**
Name: This CRL
Values:
   - yyyy-mm-dd. Date of the current CRL
   - N/A. No current CRL

The row is not displayed for PGP signatures

**Row 17: Next CRL**
Name: Next CRL
Values:
   - yyyy-mm-dd. Date of the next CRL
   - N/A. No next CRL

The row is not displayed for PGP signatures

**Row 18: Serial Number**
Name: Serial Number
Value:
   - Serial number bytes converted to a string
     - Each byte is represented by two characters (00-FF)
     - One space is added every two bytes
     - One space is added every eight bytes
     - Format: XXYY XXYY XXYY XXYY XXYY XXYY XXYY XXYY ...
Usage Scenarios
How PGP Command Line Works in the Real World

This appendix describes some of the ways PGP Command Line can be used in your organization.

Secure Off-Site Backup

A data warehouse administrator for Acme Corporation creates a nightly hot backup of a database containing sensitive corporate data so that it can be securely stored off-site.

The file, AcmeCorpData.db, is encrypted to Acme Corporation's official archival key, Archival Key, and is then transferred to the secure, off-site backup location. The file is stored encrypted in the appropriate directory on the archival machine.

After the file is transferred, it must be securely wiped off of the main database server so that it cannot be retrieved. Acme Corporation uses PGP Command Line's --wipe command at six passes, three more passes than required by the media sanitization requirements of the U.S. Department of Defense specification 5220.22-M.

Acme Corporation's use of PGP Command Line to secure its nightly off-site backup ensures that their sensitive corporate data is protected by proven PGP encryption both while in transit and while stored on the archival machine. Wiping the original file ensures that the file will not be recoverable from the main database server.

The PGP Command Line solution is:

```
pgp --encrypt AcmeCorpData.db --recipient "Archival Key"
scp AcmeCorpData.db.pgp archiveuser@172.30.100.90:~/<current date>/AcmeCorpData.db
pgp --wipe AcmeCorpData.db --wipe-passes 6
```
A system administrator with Acme Corporation wants to create a script that will automate the process of creating a PGP key for new employees for their use with PGP Desktop, used by all Acme Corporation employees.

The key needs to be a 2048-bit, RSA v4 key that includes the Acme Corporation Additional Decryption Key (ADK) so that the employee’s encrypted email or files can be decrypted after they leave the company, if they forget their password, or if they cannot decrypt the message/file themselves.

Each new key must be signed with the company’s employee certification key so that outside users are assured that messages/files encrypted and/or signed by this key are, without doubt, from an Acme Corporation employee.

To make the process of creating the key as user-friendly as possible, the new employee should only be required to enter his or her name and passphrase on the internal corporate Web site; the script should handle the rest.

The use of PGP Command Line to assist with the creation of keys for use with PGP Desktop leverages the batch processing capabilities of PGP Command Line and the ease-of-use of PGP Desktop.

The following PGP Command Line commands would be added to the script:

```
pgp --gen-key $NEWUSER --bits 2048 --key-type rsa --passphrase "$USER_PASSPHRASE" --adk $ACMECORP_ADK_ID
pgp --sign-key $NEWUSER --user $ACMECORP_CERT_KEY_ID --passphrase "$ACMECORP_KEY_PASSPHRASE"
```

The variable names shown are examples.
Compression Saves Money

Acme Corporation’s Engineering department performs a weekly download of the Widget1000 engineering drawings and schematics to the Manufacturing department located in another state over a leased line. Manufacturing uses the drawings and schematics to create prototype boards that are sent to the Quality Assurance department for testing.

The files are copied to a specific directory, which is made into a PGP archive for transfer to Manufacturing. The files are compressed with BZip2, one of the three compression formats supported by PGP Command Line (Zip and ZLib are the other two), which reduces the size of the archive by approximately 80%.

Creating a PGP Archive using PGP Command Line and using BZip2 compression means gives the Engineering department an easy-to-transfer file that is significantly smaller than all of the files taken together, and thus saves Acme Corporation money by speeding the transfer over the leased lines.

The PGP Command Line solution is:

```
a  pgp --set-preferred-compression-algorithms 0x1234ABCD
    --bzip2 1 --zlib 2

b  pgp --encrypt c:\drawings\ --recipient 0xABCD1234 --archive
    --output drawings.pgp
```

Step a sets BZip2 as the preferred compression algorithm for the key that will be used to encrypt (the default key), Step b creates the PGP archive.
Surpasses Legal Requirements

Acme Corporation’s Human Resources (HR) department uses PGP Command Line to encrypt and sign employee records that it sends over the Internet, an insecure medium, to its benefits partners.

Because information in these records includes medical information about employees, it’s important to Acme Corporation that they remain fully protected while in transit. Using strong PGP encryption also ensures that Acme Corporation is in compliance with the Health Insurance Portability and Accountability Act (HIPAA), which was passed by the U.S. Congress in 1996 and is required by the Department of Health and Human Services to, among other things, implement security standards to protect the confidentiality and integrity of all “individually identifiable health information.”

Prior to any employee records being sent over the Internet, the data is encrypted to the public key of the benefits partner it is being sent to, then the data is transferred to the partner. The benefits partner reverses the process on their end, decrypting the employee data with their private key and routing it to the appropriate personnel.

Using PGP Command Line to encrypt their employee data protects it during transfer over the Internet and ensures compliance with HIPAA.

The PGP Command Line solution is:

```
pgp -es employee42.doc -r 0xABCD1234 --signer "Alice Cameron"
--passphrase "<Alice'sPrivateKeyPassphrase>"
```
This appendix lists all of PGP Command Line’s commands, options, and environment variables.

## Commands

### Miscellaneous
- **---create-keyrings**
  - Creates empty keyring files.
- **---help (-h)**
  - Shows basic help information.
- **---license-authorize**
  - Authorizes a license number for use with PGP Command Line.
- **---list-archive**
  - Lists the contents of a PGP archive.
- **---purge-all-caches**
  - Purges all caches.
- **---purge-keyring-cache**
  - Purges the keyring cache.
- **---purge-passphrase-cache**
  - Purges the passphrase cache.
- **---speed-test**
  - Runs the PGP SDK speed tests.
- **---version**
  - Shows version information.
- **---wipe (-w)**
  - Wipes a file.

### Cryptographic
- **---armor (-a)**
  - Armors a file.
- **---clearsign**
  - Creates a clear signature.
- **---decrypt**
  - Decrypts.
- **---detached (-b)**
  - Creates a detached signature.
- **---dump-packets**
  - Dumps the packets in a PGP message.
- **---encrypt (-e)**
  - Encrypts data.
- **---export-session-key**
  - Exports the session key of an encrypted message.
- **---list-packets**
  - Lists the packets in a PGP message.
- **---list-sda**
  - Lists the contents of an SDA.
- **---sign (-s)**
  - Signs data.
- **---symmetric (-c)**
  - Encrypts using a symmetric cipher.
- **---verify**
  - Verifies data.

### Key Listings
- **---fingerprint**
  - Shows fingerprint.
- **---list-keys (-l)**
  - Shows key list in basic mode.
- **---list-key-details**
  - Shows key list in detailed mode.
- **---list-sigs**
  - Shows signatures in basic key list.
- --list-sig-details Shows signature details.
- --list-keys-xml Shows keys in XML format.
- --list-userids, --list-users Shows user IDs in a basic key list.

### Key Editing

- --add-adk Adds an ADK to a key.
- --add-photoid Adds a photo ID to a key.
- --add-preferred-cipher Adds/updates the preferred cipher on a key.
- --add-preferred-compression-algorithm Adds/updates the preferred compression algorithm on a key.
- --add-preferred-email-encoding Adds / updates the preferred email encoding on a key.
- --add-preferred-hash Adds / updates the preferred hash on a key.
- --add-revoker Adds a revoker to a key.
- --add-userid Adds a user ID to a key.
- --cache-passphrase Caches a passphrase.
- --change-passphrase Changes the passphrase of a key.
- --clear-key-flag Clears one of the key’s preferences flags.
- --disable Disables key.
- --enable Enables key.
- --export Exports keys.
- --export-key-pair Exports key pair.
- --export-photoid Exports a photo ID to a file.
- --gen-key Generates a new key pair.
- --gen-subkey Generates subkey.
- --import Imports keys.
- --join-key Rejoins a split key so it can be used.
- --join-key-cache-only Temporarily joins a previously split key
- --key-recon-reccv Reconstructs a key locally.
- --key-recon-reccv-questions Receives reconstruction questions for a specified key.
- --key-recon-send Sends reconstruction data to a server.
- --remove Removes key.
- --remove-adk Removes an ADK from a key.
- --remove-all-adks Removes all ADKs from a key.
- --remove-all-photoids Removes all photo IDs from a key.
- --remove-all-revokers Removes all revokers from a key.
- --remove-expiration-date Removes the expiration date from a key.
- --remove-key-pair Removes key pair.
- --remove-photoid Removes a photo ID from a key.
- --remove-preferred-cipher Removes a preferred cipher from a key.
- --remove-preferred-compression-algorithm Removes a preferred compression algorithm from a key.
--remove-preferred-email-encoding  Removes the preferred email encoding from a key.
--remove-preferred-hash  Removes the preferred hash from a key.
--remove-preferred-keyserver  Removes a preferred keyserver from a key.
--remove-revoker  Removes a revoker from a key.
--remove-sig  Removes signature.
--remove-subkey  Removes subkey.
--remove-userid  Removes a user ID from a key.
--revoke  Revokes key pair.
--revoke-sig  Revokes signature.
--revoke-subkey  Revokes subkey.
--send-shares  Sends shares to the server which is joining a key.
--set-expiration-date  Sets the expiration date of a key.
--set-key-flag  Sets one of the preference flags for a key.
--set-preferred-ciphers  Sets the list of preferred ciphers on a key.
--set-preferred-compression-algorithms  Sets the list of preferred compression algorithms on a key.
--set-preferred-email-encodings  Sets the list of preferred email encodings for a key.
--set-preferred-hashes  Sets the list of preferred hashes for a key.
--set-preferred-keyserver  Sets the list of preferred keyservers for a key.
--set-primary-userid  Sets a user ID as primary for a key.
--set-trust  Sets the trust on a key.
--sign-key  Signs all user IDs on a key.
--sign-userid  Signs a single user ID on a key.
--split-key  Splits a key into multiple shares.

**Keyserver**

--keyserver-disable  Disables a key on a keyserver.
--keyserver-recv  Gets keys from a keyserver.
--keyserver-remove  Removes keys from a keyserver.
--keyserver-search  Searches for keys on a keyserver, lists results.
--keyserver-send  Sends keys to a keyserver.
--keyserver-update  Updates keys with respect to a keyserver.
--recv-keys  Gets keys from a keyserver (GPG synonym for --keyserver-recv)
--send-keys  Sends keys to a keyserver (GPG synonym for --keyserver-send)

**Options**

**Boolean**

--always-trust  Always trust all keys used.
--archive  Sets encode and decode to use archive mode.
<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>--banner</td>
<td>Toggles the banner display for every operation.</td>
</tr>
<tr>
<td>--biometric</td>
<td>Uses biometric output format.</td>
</tr>
<tr>
<td>--buffered-stdio</td>
<td>Buffers stdin / stdout operations.</td>
</tr>
<tr>
<td>--compress</td>
<td>Toggles compression.</td>
</tr>
<tr>
<td>--encrypt-to-self</td>
<td>Always encrypt to the default key.</td>
</tr>
<tr>
<td>--eyes-only</td>
<td>Specifies encryption for your-eyes-only.</td>
</tr>
<tr>
<td>--fast-key-gen</td>
<td>Uses fast key generation.</td>
</tr>
<tr>
<td>--fips-mode, --fips</td>
<td>Enables FIPS mode in the PGP SDK.</td>
</tr>
<tr>
<td>--force (-f)</td>
<td>Forces certain dangerous operations to continue.</td>
</tr>
<tr>
<td>--halt-on-error</td>
<td>Stops on error for multiple I/O operations.</td>
</tr>
<tr>
<td>--keyring-cache</td>
<td>Enables the keyring cache.</td>
</tr>
<tr>
<td>--large-keyrings</td>
<td>Checks keyring signatures only when necessary.</td>
</tr>
<tr>
<td>--license-recover</td>
<td>Enables the license recovery e-mail option during authentication</td>
</tr>
<tr>
<td>--local-mode</td>
<td>Forces the PGP SDK to run in local mode.</td>
</tr>
<tr>
<td>--marginal-as-valid</td>
<td>Treats marginal keys as valid.</td>
</tr>
<tr>
<td>--pass-through</td>
<td>Passes through non-PGP data on decode.</td>
</tr>
<tr>
<td>--passphrase-cache</td>
<td>Enables the passphrase cache.</td>
</tr>
<tr>
<td>--photo</td>
<td>Specifies that we want to match a photo user ID.</td>
</tr>
<tr>
<td>--quiet (-q)</td>
<td>Quiet mode.</td>
</tr>
<tr>
<td>--recursive</td>
<td>Enables recursive mode.</td>
</tr>
<tr>
<td>--reverse-sort, --reverse</td>
<td>Reverses the sorting order.</td>
</tr>
<tr>
<td>--sda</td>
<td>Enables SDA (Self Decrypting Archive) creation</td>
</tr>
<tr>
<td>--skep</td>
<td>Checks file shares first when joining split keys.</td>
</tr>
<tr>
<td>--textmode, text (-t)</td>
<td>Forces the input to canonical text mode.</td>
</tr>
<tr>
<td>--verbose (-v)</td>
<td>Shows verbose information.</td>
</tr>
<tr>
<td>--warn-adk</td>
<td>Warns when enforcing ADKs.</td>
</tr>
<tr>
<td>--xml</td>
<td>Displays information in XML format.</td>
</tr>
</tbody>
</table>

**Integer**

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>--3des</td>
<td>Precedence of the 3DES cipher algorithm.</td>
</tr>
<tr>
<td>--aes128</td>
<td>Precedence of the AES128 cipher algorithm.</td>
</tr>
<tr>
<td>--aes192</td>
<td>Precedence of the AES192 cipher algorithm.</td>
</tr>
<tr>
<td>--aes256</td>
<td>Precedence of the AES256 cipher algorithm.</td>
</tr>
<tr>
<td>--bits, --encryption-bits</td>
<td>Encryption key bits.</td>
</tr>
<tr>
<td>--blowfish</td>
<td>Precedence of the Blowfish cipher algorithm (deprecated).</td>
</tr>
<tr>
<td>--bzip2</td>
<td>Precedence of the Bzip2 compression algorithm.</td>
</tr>
<tr>
<td>--cast5</td>
<td>Precedence of the CAST5 cipher algorithm</td>
</tr>
<tr>
<td>--creation-days</td>
<td>Number of days until creation.</td>
</tr>
<tr>
<td>--expiration-days</td>
<td>Number of days until expiration.</td>
</tr>
</tbody>
</table>
--idea Precedence of the IDEA cipher algorithm.
--index Matches a specific index (if more than one object is found).
--keyring-cache-timeout Number of seconds keyrings are cached.
--keyserver-timeout Number of seconds until a keyserver operation times out.
--md5 Precedence of the MD5 hash algorithm
--passphrase-cache-timeout Number of seconds passphrases are cached.
--ripemd160 Precedence of the CAST5 hash algorithm
--sha Precedence of the SHA-1 hash algorithm
--sha256 Precedence of the SHA-256 hash algorithm
--sha384 Precedence of the SHA-384 hash algorithm
--sha512 Precedence of the SHA-512 hash algorithm
--signing-bits Signing key bits.
--skep-timeout Timeout for joining keys over the network
--threshold Defines the minimum share threshold when splitting a key.
--trust-depth Trust depth when creating meta and trusted-introducer sigs.
--twofish Precedence of the Twofish cipher algorithm.
--wipe-input-passes Number of wipe passes for input files.
--wipe-passes Number of wipe passes for normal files.
--wipe-temp-passes Number of wipe passes for temp files.
--wipe-overwrite-passes Number of wipe passes for moving existing output files.
--zip Precedence of the Zip compression algorithm.
--zlib Precedence of the Zlib compression algorithm.

Enumeration
--auto-import-keys How to handle keys found during non-import operations.
--cipher Specifies a cipher algorithm to use with certain operations.
--compression-algorithm Sets the compression algorithm.
--compression-level Sets the compression level.
--enforce-adk Specifies how to handle ADKs.
--export-format Specifies the export format to use.
--hash Sets the hash algorithm.
--import-format Specifies the import format.
--input-cleanup How to deal with input files when done with them.
--key-flag Specifies one of the key preference flags.
--key-type Sets key type.
--manual-import-keys How to handle keys found during import.
--manual-import-key-pairs Specifies how to handle key pairs found during import.
--overwrite Sets the overwrite behavior.
--sig-type Sets the signature type.
<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>--sort-order, --sort</td>
<td>Sets the sort ordering for the current operation.</td>
</tr>
<tr>
<td>--target-platform</td>
<td>Specifies the target platform for SDAs</td>
</tr>
<tr>
<td>--temp-cleanup</td>
<td>How to deal with temp files when done with them.</td>
</tr>
<tr>
<td>--trust</td>
<td>Sets the current trust level.</td>
</tr>
<tr>
<td>--city</td>
<td>Specifies a city.</td>
</tr>
<tr>
<td>--comment</td>
<td>Specifies a comment for armored blocks.</td>
</tr>
<tr>
<td>--common-name</td>
<td>Specify a common name</td>
</tr>
<tr>
<td>--contact-email</td>
<td>Specifies a contact e-mail address.</td>
</tr>
<tr>
<td>--country</td>
<td>Specifies a country.</td>
</tr>
<tr>
<td>--creation-date</td>
<td>Number of days until creation in a date format.</td>
</tr>
<tr>
<td>--default-key</td>
<td>Sets default key for signing (also used for --encrypt-to-self).</td>
</tr>
<tr>
<td>--expiration-date</td>
<td>Number of days until expiration in a date format.</td>
</tr>
<tr>
<td>--export-passphrase</td>
<td>Passphrase to use when exporting PKCS12 data.</td>
</tr>
<tr>
<td>--home-dir</td>
<td>Location of the home directory (~/.pgp).</td>
</tr>
<tr>
<td>--license-email</td>
<td>E-mail address of the licensed user</td>
</tr>
<tr>
<td>--license-name</td>
<td>Name of the licensed user</td>
</tr>
<tr>
<td>--license-number</td>
<td>License number</td>
</tr>
<tr>
<td>--license-organization</td>
<td>Organization of the licensed user</td>
</tr>
<tr>
<td>--local-user (-u), --user</td>
<td>Local user to use for an operation.</td>
</tr>
<tr>
<td>--new-passphrase</td>
<td>Passphrase to use when changing a passphrase.</td>
</tr>
<tr>
<td>--organization</td>
<td>Specifies an organization.</td>
</tr>
<tr>
<td>--organizational-unit</td>
<td>Specifies an organizational unit.</td>
</tr>
<tr>
<td>--output (-o)</td>
<td>Specifies an output object.</td>
</tr>
<tr>
<td>--output-file</td>
<td>Sets a file to use for output messages</td>
</tr>
<tr>
<td>--passphrase</td>
<td>Passphrase to use for the current operation.</td>
</tr>
<tr>
<td>--preferred-keyserver</td>
<td>Specifies a preferred keyserver.</td>
</tr>
<tr>
<td>--private-keyring</td>
<td>Private keyring file.</td>
</tr>
<tr>
<td>--proxy-password</td>
<td>Proxy server password</td>
</tr>
<tr>
<td>--proxy-server</td>
<td>Proxy server to use for certain network operations</td>
</tr>
<tr>
<td>--proxy-username</td>
<td>Proxy server username</td>
</tr>
<tr>
<td>--public-keyring</td>
<td>Public keyring file.</td>
</tr>
<tr>
<td>--random-seed</td>
<td>Specifies a random seed file.</td>
</tr>
<tr>
<td>--regular-expression</td>
<td>Specifies a regular expression.</td>
</tr>
<tr>
<td>--root-path</td>
<td>Root path used to create SDAs and archives</td>
</tr>
<tr>
<td>--share-server</td>
<td>Server to use for split key operations</td>
</tr>
<tr>
<td>--state</td>
<td>Specifies a state.</td>
</tr>
<tr>
<td>--status-file</td>
<td>Sets a file to use for status messages</td>
</tr>
</tbody>
</table>
--symmetric-passphrase Specifies a passphrase to use with conventional encryption.
--temp-dir Specifies a temporary directory for PGP Command Line to use.

List
--additional-recipient Specifies additional (required) recipients.
--adk Specifies an ADK
--input (-i) Specifies an input object.
--keyserver Specifies a keyserver.
--recipient (r) Specifies a recipient.
--revoker Specifies a revoker.
--share Specifies a share when splitting a key.

File Descriptors
--auth-passphrase-fd Reads --auth-passphrase from a file descriptor.
--auth-passphrase-fd8 Reads --auth-passphrase from a file descriptor (in UTF8).
--export-passphrase-fd Reads --export-passphrase from a file descriptor.
--export-passphrase-fd8 Reads --export-passphrase from a file descriptor (in UTF8).
--new-passphrase-fd Reads --new-passphrase from a file descriptor.
--new-passphrase-fd8 Reads --new-passphrase from a file descriptor (in UTF8).
--passphrase-fd Reads --passphrase from a file descriptor.
--passphrase-fd8 Reads --passphrase from a file descriptor (in UTF8).
--proxy-passphrase-fd Reads --proxy-passphrase from a file descriptor.
--proxy-passphrase-fd8 Reads --proxy-passphrase from a file descriptor (in UTF8).
--symmetric-passphrase-fd Reads --symmetric-passphrase from a file descriptor.
--symmetric-passphrase-fd8 Reads --symmetric-passphrase from a file descriptor (in UTF8).

Environment Variables
PGP_LOCAL_MODE Forces PGP Command Line to run in local mode (Boolean).
PGP_HOME_DIR Overrides the default home directory (String).
PGP_FIPS_MODE Forces PGP SDK to run in a FIPS-compliant mode (Boolean).
PGP_PASSPHRASE Lets you set your passphrase (String).
PGP_NEW_PASSPHRASE Lets you set a new passphrase (String).
PGP_SYMMETRIC_PASSPHRASE Lets you set a passphrase for symmetric encryption (String).
PGP_EXPORT_PASSPHRASE Lets you set the export passphrase (String).
PGP_PROXY_PASSPHRASE Lets you set the proxy passphrase in the environment (String).
PGP_AUTH_PASSPHRASE Lets you set the auth passphrase in the environment (String).
PGP_TEMP_DIR Lets you set the temporary directory in the environment (String).
PGP_SOURCE_CODE_PAGE Lets you set the source code page in the environment (String).
## Configuration File Variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLlicenseAuthorization</td>
<td>String</td>
<td>License Authorization</td>
<td>Specifies the license authorization.</td>
</tr>
<tr>
<td>CLlicenseName</td>
<td>String</td>
<td>License Name</td>
<td>Specifies the name of the licensee.</td>
</tr>
<tr>
<td>CLlicenseNumber</td>
<td>String</td>
<td>License Number</td>
<td>Specifies the license number.</td>
</tr>
<tr>
<td>CLlicenseOrganization</td>
<td>String</td>
<td>License Organization</td>
<td>Specifies the organization of the licensee.</td>
</tr>
<tr>
<td>CLstatusFile</td>
<td>String</td>
<td>Status File</td>
<td>Specifies the status file used for status messages.</td>
</tr>
<tr>
<td>CLoutputFile</td>
<td>String</td>
<td>Output File</td>
<td>Specifies the output file.</td>
</tr>
<tr>
<td>CLtempDir</td>
<td>String</td>
<td>Temp Directory</td>
<td>Specifies a temporary directory.</td>
</tr>
<tr>
<td>rngSeedFile</td>
<td>String</td>
<td>Random seed filename</td>
<td>Sets the location of the random seed file.</td>
</tr>
<tr>
<td>privateKeyringFile</td>
<td>String</td>
<td>Private keyring file</td>
<td>Sets filename or path and filename to the private keyring file.</td>
</tr>
<tr>
<td>publicKeyringFile</td>
<td>String</td>
<td>Public keyring file</td>
<td>Sets filename or path and filename to the public keyring file.</td>
</tr>
<tr>
<td>commentString</td>
<td>String</td>
<td>Comment</td>
<td>Specifies a comment string to be used in armored output blocks.</td>
</tr>
<tr>
<td>CLDefaultKey</td>
<td>String</td>
<td>Default signing key</td>
<td>Specifies a key to be used by default for signing.</td>
</tr>
<tr>
<td>adkWarning</td>
<td>Boolean</td>
<td>ADK warning level</td>
<td>Enables warning messages for ADK actions.</td>
</tr>
<tr>
<td>fastKeyGen</td>
<td>Boolean</td>
<td>Fast keygen</td>
<td>Sets fast key generation setting.</td>
</tr>
<tr>
<td>marginalIsInvalid</td>
<td>Boolean</td>
<td>Marginal is invalid</td>
<td>Sets minimum number of marginally trusted signatures.</td>
</tr>
<tr>
<td>encryptToSelf</td>
<td>Boolean</td>
<td>Encrypt to self</td>
<td>Files/messages you encrypt are also encrypted to your key.</td>
</tr>
<tr>
<td>CLpassphraseCache</td>
<td>Boolean</td>
<td>Passphrase cache</td>
<td>Saves your passphrase in memory.</td>
</tr>
<tr>
<td>CLkeyringCache</td>
<td>Boolean</td>
<td>Keyring cache</td>
<td>Stores keyrings in memory for each access.</td>
</tr>
<tr>
<td>CLhaltOnError</td>
<td>Boolean</td>
<td>Halt on error</td>
<td>Halts operations when an error occurs.</td>
</tr>
<tr>
<td>CLlargeKeyrings</td>
<td>Boolean</td>
<td>Large Keyrings</td>
<td>Checks keyring signatures only when necessary.</td>
</tr>
<tr>
<td>fileWipePasses</td>
<td>Integer</td>
<td>Number of wipe passes</td>
<td>Sets passes used by the --wipe command.</td>
</tr>
<tr>
<td>CLfileWipeInputPasses</td>
<td>Integer</td>
<td>Number of wipe input passes</td>
<td>Sets wipe passes for input files.</td>
</tr>
<tr>
<td>Variable</td>
<td>Type</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>-------------------</td>
<td>-----------------</td>
<td>------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>CLfileWipeTempPasses</td>
<td>Integer</td>
<td>Number of wipe temp passes</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Sets wipe passes for temporary files.</td>
<td></td>
</tr>
<tr>
<td>CLfileWipeOverwritePasses</td>
<td>Integer</td>
<td>Number of wipe overwrite passes</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Sets wipe passes when overwriting an existing output file.</td>
<td></td>
</tr>
<tr>
<td>CLpassphraseCacheTimeout</td>
<td>Integer</td>
<td>Passphrase cache timeout</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Sets seconds a passphrase stays cached.</td>
<td></td>
</tr>
<tr>
<td>CLkeyringCacheTimeout</td>
<td>Integer</td>
<td>Keyring cache timeout</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Sets seconds a keyring stays cached in memory.</td>
<td></td>
</tr>
<tr>
<td>CLkeyserverTimeout</td>
<td>Integer</td>
<td>Keyserver timeout</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Sets seconds to wait before a keyserver operation times out.</td>
<td></td>
</tr>
<tr>
<td>CLcompressionLevel</td>
<td>Enumeration</td>
<td>Compression Level</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Sets the compression level for the current operation.</td>
<td></td>
</tr>
<tr>
<td>CLmanualImportKeyPairs</td>
<td>Enumeration</td>
<td>Manual import key pairs</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Establishes behavior when key pairs are found during import</td>
<td></td>
</tr>
<tr>
<td>CLsortOrder</td>
<td>Enumeration</td>
<td>Sort order</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Changes the sort order for writing key lists.</td>
<td></td>
</tr>
<tr>
<td>CLinputCleanup</td>
<td>Enumeration</td>
<td>Input cleanup</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Sets behavior with input files after they have been used.</td>
<td></td>
</tr>
<tr>
<td>CLOverwrite</td>
<td>Enumeration</td>
<td>Overwrite</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Sets behavior when an output file already exists.</td>
<td></td>
</tr>
<tr>
<td>CLEnforceADK</td>
<td>Enumeration</td>
<td>Enforce ADK</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Sets the ADK enforcement policy.</td>
<td></td>
</tr>
<tr>
<td>CLautoImportKeys</td>
<td>Enumeration</td>
<td>Automatic import of keys</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Sets behavior when keys are found in non-import operations.</td>
<td></td>
</tr>
<tr>
<td>CLmanualImportKeys</td>
<td>Enumeration</td>
<td>Manual import of keys</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Sets behavior when keys are found during an import.</td>
<td></td>
</tr>
<tr>
<td>alwaysEncryptToKeys</td>
<td>List</td>
<td>Always encrypt to keys</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Specifies an additional recipient for encryption.</td>
<td></td>
</tr>
<tr>
<td>keyservers</td>
<td>List</td>
<td>Default keyserver</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Specifies a default keyserver.</td>
<td></td>
</tr>
</tbody>
</table>
This appendix lists many of the critical commands of PGP Command Line and their counterparts in GnuPG and the McAfee E-Business Server.

<table>
<thead>
<tr>
<th>Operation</th>
<th>PGP Command Line</th>
<th>GnuPG</th>
<th>E-Business Server</th>
</tr>
</thead>
<tbody>
<tr>
<td>Encrypt</td>
<td>-e, --encrypt</td>
<td>-e, --encrypt</td>
<td>--encrypt</td>
</tr>
<tr>
<td>Symmetric</td>
<td>-c, --symmetric</td>
<td>-c, --symmetric</td>
<td>--conventional</td>
</tr>
<tr>
<td>Decrypt</td>
<td>--decrypt</td>
<td>--decrypt</td>
<td>--decrypt</td>
</tr>
<tr>
<td>Sign</td>
<td>-s, --sign</td>
<td>-s, --sign</td>
<td>--sign</td>
</tr>
<tr>
<td>Verify</td>
<td>--verify</td>
<td>--verify</td>
<td>Supported. No verify command as such.</td>
</tr>
<tr>
<td>Armor</td>
<td>-a, --armor</td>
<td>-a, --armor</td>
<td>--armor</td>
</tr>
<tr>
<td>Export session key</td>
<td>--export-session-key</td>
<td>--show-session-key</td>
<td>Not supported.</td>
</tr>
<tr>
<td>Wipe</td>
<td>-w, --wipe</td>
<td>Not supported.</td>
<td>--wipe</td>
</tr>
<tr>
<td>Split/join key</td>
<td>--split-key, --join-key</td>
<td>Not supported.</td>
<td>--key-split, --key-join</td>
</tr>
<tr>
<td>Benchmarking (speed) tests</td>
<td>--speed-test</td>
<td>Not supported.</td>
<td>Not supported.</td>
</tr>
<tr>
<td>Archive files</td>
<td>--archive</td>
<td>Not supported.</td>
<td>Not supported.</td>
</tr>
<tr>
<td>Help</td>
<td>-h, --help</td>
<td>-h, --help</td>
<td>--help</td>
</tr>
<tr>
<td>Version</td>
<td>--version</td>
<td>--version</td>
<td>--version</td>
</tr>
<tr>
<td>List keys</td>
<td>-l, --list-keys</td>
<td>--list-keys</td>
<td>--key-list</td>
</tr>
<tr>
<td>Generate a key</td>
<td>-gen-key</td>
<td>--gen-key</td>
<td>--key-gen</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(interactive only, no unattended key generation. Does not generate RSA v4 sign and encrypt keys without recompiling)</td>
<td></td>
</tr>
<tr>
<td>Import a key</td>
<td>--import</td>
<td>--import</td>
<td>--key-add</td>
</tr>
<tr>
<td>Export a key</td>
<td>--export</td>
<td>--export, --export-secret-key</td>
<td>Not supported.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(cannot export an entire key pair at once, requires two operations)</td>
<td></td>
</tr>
<tr>
<td>Revoke key</td>
<td>--revoke</td>
<td>--edit-key, revkey or revsig command (interaction required)</td>
<td>--key-edit ...--revoke</td>
</tr>
<tr>
<td>Sign all user IDs on a key</td>
<td>--sign-key</td>
<td>--edit-key, sign/lsign/nrsign/nrlsign commands (interaction required)</td>
<td>--key-sign</td>
</tr>
<tr>
<td>Sign one user ID on a key</td>
<td>--sign-userid</td>
<td>Not supported.</td>
<td>Not supported.</td>
</tr>
<tr>
<td>Set trust on a key</td>
<td>--set-trust</td>
<td>--edit-key, trust command (interaction required)</td>
<td>--key-edit ...--trust</td>
</tr>
<tr>
<td>Feature</td>
<td>Command Line</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>------------------------------------</td>
<td>--------------</td>
<td>-----------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>Enable a key</td>
<td><code>--enable</code></td>
<td><code>--edit-key</code>, enable command (interaction required)`</td>
<td></td>
</tr>
<tr>
<td>Disable a key</td>
<td><code>--disable</code></td>
<td><code>--edit-key</code>, enable command (interaction required)`</td>
<td></td>
</tr>
<tr>
<td>Add photo ID to a key</td>
<td><code>--add-photoid</code></td>
<td><code>--edit-key</code>, addphoto command (interaction required)`</td>
<td></td>
</tr>
<tr>
<td>Add revoker to a key</td>
<td><code>--add-revoker</code></td>
<td><code>--edit-key</code>, addrevoker command (interaction required)`</td>
<td></td>
</tr>
<tr>
<td>Add ADK to a key</td>
<td><code>--add-adk</code></td>
<td>Not supported.</td>
<td></td>
</tr>
<tr>
<td>Change passphrase</td>
<td><code>--change-passphrase</code></td>
<td><code>--edit-key</code>, passwd command (interaction required)`</td>
<td></td>
</tr>
<tr>
<td>Set preferred keyserver on a key</td>
<td><code>--set-preferred-keyserver</code></td>
<td>Not supported.</td>
<td></td>
</tr>
<tr>
<td>Set preferred ciphers on a key</td>
<td><code>--set-preferred-ciphers</code></td>
<td><code>--edit-key</code>, setpref and updpref commands (interaction required). Also requires mapping all cipher names to internal code numbers.</td>
<td></td>
</tr>
<tr>
<td>Set preferred compression algorithms on a key</td>
<td><code>--set-preferred-compression-algorithms</code></td>
<td><code>--edit-key</code>, setpref and updpref commands (interaction required). Also requires mapping all cipher names to internal code numbers.</td>
<td></td>
</tr>
<tr>
<td>Eyes only</td>
<td><code>--eyes-only</code></td>
<td>Not supported.</td>
<td></td>
</tr>
<tr>
<td>Fast key generation</td>
<td><code>--fast-key-gen</code></td>
<td>Not supported. Configuration file.</td>
<td></td>
</tr>
<tr>
<td>Ciphers</td>
<td><code>3DES, AES 128, AES 192, AES 256, Blowfish, CAST5, IDEA, Twofish</code></td>
<td>All supported. IDEA requires an extra module be compiled in that is not included with the base install.</td>
<td></td>
</tr>
<tr>
<td>Compression algorithms</td>
<td><code>BZip2, Zip, Zlib</code></td>
<td>Zip and Zlib fully supported. BZip2 supported for read and write as of GnuPG 1.4.</td>
<td></td>
</tr>
</tbody>
</table>
Codes and Messages
How to Understand PGP Command Line Codes and Error Messages

This appendix lists and describes the numeric codes and descriptive messages generated by PGP Command Line.

A code of 0 (zero) means the operation was concluded successfully. The accompanying message provides additional information.

A numeric code other than zero means the operation did not conclude successfully. The accompanying message provides additional information.

ℹ️ Some non-zero status codes are informational and do not indicate an error condition. Exit codes always indicate an error.

Status messages use the form:
<source>,<operation> (<code>,<description>)

For example, in the case of a file that is not found:

    file.txt,encrypt (3001,input file not found)
## Messages Without Codes

<table>
<thead>
<tr>
<th>Message</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>unknown</td>
<td>An unknown error occurred.</td>
</tr>
<tr>
<td>unknown description</td>
<td>An error with an unknown description occurred.</td>
</tr>
<tr>
<td>unknown err [number]</td>
<td>An error with an unknown error number occurred.</td>
</tr>
<tr>
<td>unknown time zone</td>
<td>PGP Command Line is unable to determine the current time zone.</td>
</tr>
<tr>
<td>PGP SDK running in local mode.</td>
<td>The PGP SDK is running in Local Mode.</td>
</tr>
<tr>
<td>PGP SDK running in forced local mode.</td>
<td>The PGP SDK is running in Forced Local Mode.</td>
</tr>
<tr>
<td>PGP SDK running in FIPS mode.</td>
<td>The PGP SDK is running in FIPS Mode.</td>
</tr>
<tr>
<td>FIPS mode initialization failed.</td>
<td>FIPS Mode failed to initialize.</td>
</tr>
<tr>
<td>Unable to determine current time zone.</td>
<td>PGP Command Line was unable to determine the current time zone from the host computer.</td>
</tr>
<tr>
<td>operation cancelled</td>
<td>The operation was cancelled.</td>
</tr>
<tr>
<td>no application data directory found</td>
<td>PGP Command Line was unable to locate its application data directory.</td>
</tr>
<tr>
<td>no personal documents directory found</td>
<td>PGP Command Line was unable to locate its personal documents directory.</td>
</tr>
</tbody>
</table>

## Messages With Codes

### Parser

<table>
<thead>
<tr>
<th>Code</th>
<th>Message</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>9000</td>
<td>invalid flag “flag”</td>
<td>An invalid flag was used.</td>
</tr>
<tr>
<td>9001</td>
<td>no match for enum argument “argument”</td>
<td>There was no match for the listed enumeration argument.</td>
</tr>
<tr>
<td>9002</td>
<td>invalid primary operation</td>
<td>The primary operation is invalid.</td>
</tr>
<tr>
<td>9003</td>
<td>you cannot specify multiple operations</td>
<td>Multiple operations cannot be specified.</td>
</tr>
<tr>
<td>9004</td>
<td>preferred cipher list contains gaps or duplicates</td>
<td>The list of preferred ciphers includes gaps or duplicate ciphers.</td>
</tr>
<tr>
<td>9005</td>
<td>Blowfish cipher has been deprecated</td>
<td>The Blowfish cipher has been deprecated; you cannot select. If a key already uses it, however, PGP Command Line will work with it.</td>
</tr>
<tr>
<td>9006</td>
<td>no preferred ciphers specified</td>
<td>No preferred ciphers have been specified.</td>
</tr>
<tr>
<td>9007</td>
<td>preferred cipher list contains overlaps</td>
<td>The list of preferred ciphers has overlaps.</td>
</tr>
<tr>
<td>Code</td>
<td>Message</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>---------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>9008</td>
<td>no preferred cipher specified</td>
<td>A preferred cipher was not specified.</td>
</tr>
<tr>
<td>9009</td>
<td>invalid cipher options specified</td>
<td>Invalid cipher options were specified.</td>
</tr>
<tr>
<td>9010</td>
<td>unable remove the only preferred cipher</td>
<td>PGP Command Line is unable to remove the only preferred cipher.</td>
</tr>
<tr>
<td>9011</td>
<td>preferred compression list contains overlaps</td>
<td>The list of preferred compression algorithms has overlaps.</td>
</tr>
<tr>
<td>9012</td>
<td>preferred compression list contains gaps or duplicates</td>
<td>The list of preferred compression algorithms has gaps or overlaps.</td>
</tr>
<tr>
<td>9013</td>
<td>no preferred compression algorithms specified</td>
<td>No preferred compression algorithms have been specified.</td>
</tr>
<tr>
<td>9014</td>
<td>no preferred compression algorithm specified</td>
<td>A preferred compression algorithm was not specified.</td>
</tr>
<tr>
<td>9015</td>
<td>invalid compression algorithm options specified</td>
<td>An invalid compression algorithm option was specified.</td>
</tr>
<tr>
<td>9016</td>
<td>unable remove the only preferred compression algorithm</td>
<td>PGP Command Line is unable to remove the only preferred compression algorithm.</td>
</tr>
<tr>
<td>9017</td>
<td>invalid file descriptor</td>
<td>An invalid file descriptor was used.</td>
</tr>
<tr>
<td>9018</td>
<td>missing argument for option “option”</td>
<td>An argument is missing for the specified option.</td>
</tr>
</tbody>
</table>
### Keyrings

<table>
<thead>
<tr>
<th>Code</th>
<th>Message</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1001</td>
<td>could not open keyrings, file not found</td>
<td>PGP Command Line could not open the keyring file because it was not found.</td>
</tr>
<tr>
<td>1002</td>
<td>could not open keyrings, file locked</td>
<td>PGP Command Line could not open the keyring file because it is locked.</td>
</tr>
<tr>
<td>1003</td>
<td>default key does not exist</td>
<td>The default key does not exist.</td>
</tr>
<tr>
<td>1004</td>
<td>too many matches for default key</td>
<td>There were too many matches for the default key.</td>
</tr>
<tr>
<td>1005</td>
<td>invalid default key specified</td>
<td>An invalid default key was specified.</td>
</tr>
<tr>
<td>1006</td>
<td>public keyring</td>
<td>An informational message that displays the location of the public keyring file. Displays in verbose mode only.</td>
</tr>
<tr>
<td>1007</td>
<td>private keyring</td>
<td>An informational message that displays the location of the private keyring file. Displays in verbose mode only.</td>
</tr>
<tr>
<td>1008</td>
<td>keyring already exists</td>
<td>The keyring already exists.</td>
</tr>
<tr>
<td>1009</td>
<td>unable to open prefs file</td>
<td>PGP Command Line cannot open the preferences file.</td>
</tr>
</tbody>
</table>

### Wipe

<table>
<thead>
<tr>
<th>Code</th>
<th>Message</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>file wiped successfully</td>
<td>The file was successfully wiped.</td>
</tr>
<tr>
<td>0</td>
<td>file removed successfully</td>
<td>The file was successfully removed.</td>
</tr>
<tr>
<td>0</td>
<td>directory removed successfully</td>
<td>The directory was successfully removed.</td>
</tr>
<tr>
<td>0</td>
<td>symbolic link removed successfully</td>
<td>The symbolic link was successfully removed.</td>
</tr>
<tr>
<td>0</td>
<td>directory wiped successfully</td>
<td>The directory was successfully wiped.</td>
</tr>
<tr>
<td>0</td>
<td>symbolic link wiped successfully</td>
<td>The symbolic link was wiped successfully.</td>
</tr>
<tr>
<td>1010</td>
<td>invalid number of wipe passes specified</td>
<td>An invalid number of wipe passes was specified.</td>
</tr>
<tr>
<td>1011</td>
<td>invalid file permissions</td>
<td>The wipe failed because of invalid file permissions.</td>
</tr>
<tr>
<td>1013</td>
<td>wipe failed</td>
<td>The wipe failed.</td>
</tr>
<tr>
<td>1014</td>
<td>file locked</td>
<td>The wipe failed because the file was locked.</td>
</tr>
</tbody>
</table>
### Encrypt

<table>
<thead>
<tr>
<th>Code</th>
<th>Message</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1030</td>
<td>key added to recipient list</td>
<td>The key was added to the recipient list.</td>
</tr>
<tr>
<td>1031</td>
<td>default key not suitable for encryption</td>
<td>The default key is not suitable for encryption.</td>
</tr>
<tr>
<td>1032</td>
<td>text mode is not applicable in archive mode</td>
<td>Text mode is not applicable in PGP Archive mode.</td>
</tr>
</tbody>
</table>

### Sign

<table>
<thead>
<tr>
<th>Code</th>
<th>Message</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1050</td>
<td>key added as signer</td>
<td>The key was added as a signer.</td>
</tr>
<tr>
<td>1051</td>
<td>default key added as signer</td>
<td>The default key was added as a signer.</td>
</tr>
<tr>
<td>1052</td>
<td>no signing key specified</td>
<td>No signing key was specified.</td>
</tr>
<tr>
<td>1053</td>
<td>signing key not found</td>
<td>The signing key was not found.</td>
</tr>
<tr>
<td>1054</td>
<td>too many matches for signing key</td>
<td>There were too many matches to the signing key.</td>
</tr>
<tr>
<td>1055</td>
<td>SDA is not applicable when signing</td>
<td>A self-decrypting archive (SDA) is not applicable when signing.</td>
</tr>
</tbody>
</table>

### Decrypt

<table>
<thead>
<tr>
<th>Code</th>
<th>Message</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>SDA decoded successfully</td>
<td>The SDA was successfully decoded.</td>
</tr>
<tr>
<td>0</td>
<td>packet dump complete</td>
<td>The packet dump is complete.</td>
</tr>
<tr>
<td>1080</td>
<td>no private key could be found for decryption</td>
<td>No private key could be found to use for decryption.</td>
</tr>
<tr>
<td>1081</td>
<td>detached signature not found</td>
<td>The detached signature was not found.</td>
</tr>
<tr>
<td>1082</td>
<td>detached signature target file</td>
<td>Displays the file PGP Command Line believes is the target file when verifying or decrypting a detached signature.</td>
</tr>
<tr>
<td>1083</td>
<td>pass through is not applicable for archive data</td>
<td>Passthrough is not applicable for archive data.</td>
</tr>
<tr>
<td>1084</td>
<td>signature date precedes key creation date</td>
<td>The signature date precedes the key creation date.</td>
</tr>
<tr>
<td>1085</td>
<td>invalid SDA</td>
<td>The SDA you are trying to decrypt is invalid.</td>
</tr>
<tr>
<td>1086</td>
<td>only one passphrase allowed</td>
<td>You can only enter one passphrase when decrypting.</td>
</tr>
<tr>
<td>1087</td>
<td>SDA is not encrypted to any ADKs</td>
<td>The SDA is not encrypted to the ADK you specified.</td>
</tr>
<tr>
<td>1088</td>
<td>PGP self-decrypting archive</td>
<td>The file you are trying to decrypt is a PGP SDA.</td>
</tr>
</tbody>
</table>
# Speed Test

<table>
<thead>
<tr>
<th>Code</th>
<th>Message</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>speed test successful</td>
<td>The speed test was successful.</td>
</tr>
</tbody>
</table>

# Key edit

<table>
<thead>
<tr>
<th>Code</th>
<th>Message</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>key imported as</td>
<td>The key was imported as specified.</td>
</tr>
<tr>
<td>0</td>
<td>X.509 certificate imported to</td>
<td>The X.509 certificate was imported as specified.</td>
</tr>
<tr>
<td>0</td>
<td>key exported to</td>
<td>The key was exported as specified.</td>
</tr>
<tr>
<td>0</td>
<td>key successfully generated</td>
<td>The key was generated.</td>
</tr>
<tr>
<td>0</td>
<td>subkey successfully generated</td>
<td>The subkey was generated.</td>
</tr>
<tr>
<td>0</td>
<td>key successfully removed</td>
<td>The key was removed.</td>
</tr>
<tr>
<td>0</td>
<td>key successfully revoked</td>
<td>The key was revoked.</td>
</tr>
<tr>
<td>0</td>
<td>subkey successfully removed</td>
<td>The subkey was removed.</td>
</tr>
<tr>
<td>0</td>
<td>subkey successfully revoked</td>
<td>The subkey was revoked.</td>
</tr>
<tr>
<td>0</td>
<td>certified user ID</td>
<td>The user ID was certified.</td>
</tr>
<tr>
<td>0</td>
<td>removed signature by user</td>
<td>The signature of the specified user was removed.</td>
</tr>
<tr>
<td>0</td>
<td>revoked signature by user</td>
<td>The signature of the specified user was revoked.</td>
</tr>
<tr>
<td>0</td>
<td>trust set successfully</td>
<td>Trust was successfully set.</td>
</tr>
<tr>
<td>0</td>
<td>key successfully enabled</td>
<td>The key was enabled.</td>
</tr>
<tr>
<td>0</td>
<td>key successfully disabled</td>
<td>The key was disabled.</td>
</tr>
<tr>
<td>0</td>
<td>user ID added successfully</td>
<td>The user ID was added.</td>
</tr>
<tr>
<td>0</td>
<td>successfully removed</td>
<td>The specified item was removed.</td>
</tr>
<tr>
<td>0</td>
<td>photo ID added successfully</td>
<td>The photo ID was added.</td>
</tr>
<tr>
<td>0</td>
<td>successfully removed photo ID</td>
<td>The photo ID was removed.</td>
</tr>
<tr>
<td>0</td>
<td>photo ID exported to</td>
<td>The photo ID was exported as specified.</td>
</tr>
<tr>
<td>0</td>
<td>new primary user ID</td>
<td>The specified user ID is now primary.</td>
</tr>
<tr>
<td>0</td>
<td>revokers successfully updated</td>
<td>Revokers were updated.</td>
</tr>
<tr>
<td>0</td>
<td>ADKs successfully updated</td>
<td>ADKs were updated.</td>
</tr>
<tr>
<td>0</td>
<td>verify complete</td>
<td>The verify is complete.</td>
</tr>
<tr>
<td>0</td>
<td>expiration date successfully updated</td>
<td>Expiration date was updated.</td>
</tr>
<tr>
<td>0</td>
<td>key passphrase changed</td>
<td>The key passphrase was changed.</td>
</tr>
<tr>
<td>0</td>
<td>subkey passphrase changed</td>
<td>The subkey passphrase was changed.</td>
</tr>
<tr>
<td>Code</td>
<td>Message</td>
<td>Description</td>
</tr>
<tr>
<td>------</td>
<td>--------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>0</td>
<td>key passphrase cached</td>
<td>The key passphrase was cached.</td>
</tr>
<tr>
<td>0</td>
<td>preferred keyserver updated</td>
<td>The preferred keyserver was updated.</td>
</tr>
<tr>
<td>0</td>
<td>preferred keyserver removed</td>
<td>The preferred keyserver was removed.</td>
</tr>
<tr>
<td>0</td>
<td>preferred ciphers updated</td>
<td>The preferred ciphers were updated.</td>
</tr>
<tr>
<td>0</td>
<td>preferred compression algorithms updated</td>
<td>The preferred compression algorithms were updated.</td>
</tr>
<tr>
<td>0</td>
<td>key split successfully</td>
<td>The key was split.</td>
</tr>
<tr>
<td>0</td>
<td>key joined successfully</td>
<td>The key was joined.</td>
</tr>
<tr>
<td>0</td>
<td>new primary user ID numbers</td>
<td>New primary user ID numbers have been created.</td>
</tr>
<tr>
<td>0</td>
<td>flags updated successfully</td>
<td>Flags were successfully updated.</td>
</tr>
<tr>
<td>0</td>
<td>shares successfully sent</td>
<td>Shares were successfully sent.</td>
</tr>
<tr>
<td>0</td>
<td>preferred hashes updated</td>
<td>Preferred hashes were successfully updated.</td>
</tr>
<tr>
<td>0</td>
<td>notation packet removed</td>
<td>A notation packet was removed.</td>
</tr>
<tr>
<td>0</td>
<td>removed notation packets</td>
<td>Multiple notation packets were removed.</td>
</tr>
<tr>
<td>0</td>
<td>notation packet added</td>
<td>A notation packet was added.</td>
</tr>
<tr>
<td>0</td>
<td>notation packet updated</td>
<td>A notation packet was updated.</td>
</tr>
<tr>
<td>0</td>
<td>preferred email encodings updated</td>
<td>Preferred email encodings were updated.</td>
</tr>
<tr>
<td>2000</td>
<td>editing key</td>
<td>Displays the key found for the edit operation. Displays in verbose mode only.</td>
</tr>
<tr>
<td>2001</td>
<td>you must specify a key to edit</td>
<td>A key to edit must be specified.</td>
</tr>
<tr>
<td>2002</td>
<td>key to edit not found</td>
<td>The key to edit was not found.</td>
</tr>
<tr>
<td>2003</td>
<td>too many matches for key to edit</td>
<td>There were too many matches for the key to edit.</td>
</tr>
<tr>
<td>2004</td>
<td>filter didn’t match any keys</td>
<td>The filter didn’t match any keys.</td>
</tr>
<tr>
<td>2005</td>
<td>cannot edit key</td>
<td>The key cannot be edited.</td>
</tr>
<tr>
<td>2020</td>
<td>key already enabled</td>
<td>The key is already enabled.</td>
</tr>
<tr>
<td>2021</td>
<td>key already disabled</td>
<td>The key is already disabled.</td>
</tr>
<tr>
<td>2022</td>
<td>unable to remove the last user ID</td>
<td>PGP Command Line is unable to remove the last user ID.</td>
</tr>
<tr>
<td>2023</td>
<td>cannot set trust on invalid key</td>
<td>PGP Command Line cannot set trust on an invalid key.</td>
</tr>
<tr>
<td>2024</td>
<td>key pair trust setting can only be never or implicit</td>
<td>The trust setting on the key pair can only be Never or Implicit.</td>
</tr>
<tr>
<td>2025</td>
<td>public key trust setting cannot be implicit</td>
<td>The trust setting on a public key cannot be Implicit.</td>
</tr>
<tr>
<td>2026</td>
<td>no revoker specified</td>
<td>No revoker was specified.</td>
</tr>
<tr>
<td>2027</td>
<td>revoker not found</td>
<td>No revoker was found.</td>
</tr>
<tr>
<td>Code</td>
<td>Message</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>--------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>2028</td>
<td>too many revokers found</td>
<td>Too many revokers were found.</td>
</tr>
<tr>
<td>2029</td>
<td>revoker found</td>
<td>Displays the revoker found when adding a revoker to a key. Displays in verbose mode only.</td>
</tr>
<tr>
<td>2030</td>
<td>no ADK specified</td>
<td>No ADK (additional decryption key) was specified.</td>
</tr>
<tr>
<td>2031</td>
<td>ADK not found</td>
<td>The specified ADK was not found.</td>
</tr>
<tr>
<td>2032</td>
<td>too many ADKs found</td>
<td>Too many ADKs were found.</td>
</tr>
<tr>
<td>2033</td>
<td>ADK found</td>
<td>Displays the ADK found when adding an ADK to a key. Displays in verbose mode only.</td>
</tr>
<tr>
<td>2034</td>
<td>preferred keyserver not specified</td>
<td>A preferred keyserver was not specified.</td>
</tr>
<tr>
<td>2035</td>
<td>invalid preferred keyserver</td>
<td>There is a formatting error on the preferred keyserver.</td>
</tr>
<tr>
<td>2036</td>
<td>certification exists for user ID</td>
<td>Certification exists for the specified user ID.</td>
</tr>
<tr>
<td>2037</td>
<td>unwilling to remove key pair</td>
<td>The key pair was not removed. Use <code>--remove-key-pair</code> to remove a key pair.</td>
</tr>
<tr>
<td>2038</td>
<td>no private key found to remove</td>
<td>A request was made to remove a key pair, but a public key was specified.</td>
</tr>
<tr>
<td>2039</td>
<td>no private key found to export</td>
<td>No private key was found to export.</td>
</tr>
<tr>
<td>2040</td>
<td>cannot revoke key, no private key present</td>
<td>No private key is present, so the key cannot be revoked.</td>
</tr>
<tr>
<td>2041</td>
<td>cannot remove a self signature</td>
<td>The self-signature cannot be removed.</td>
</tr>
<tr>
<td>2042</td>
<td>cannot remove photo ID</td>
<td>A photo ID cannot be removed with <code>--remove-userid</code>. Use <code>--remove-photoid</code>.</td>
</tr>
<tr>
<td>2043</td>
<td>creation cannot be specified</td>
<td>When trying to specify an expiration date, a creation date was also specified.</td>
</tr>
<tr>
<td>2044</td>
<td>expiration in date format is required</td>
<td>An expiration date in date format is required.</td>
</tr>
<tr>
<td>2045</td>
<td>trust not specified</td>
<td>Trust was not specified.</td>
</tr>
<tr>
<td>2046</td>
<td>photo ID too large</td>
<td>The photo ID is too large.</td>
</tr>
<tr>
<td>2047</td>
<td>photo ID format invalid</td>
<td>The format of the photo ID is invalid.</td>
</tr>
<tr>
<td>2048</td>
<td>too many photo IDs</td>
<td>Too many photo IDs specified.</td>
</tr>
<tr>
<td>2049</td>
<td>too many keys found</td>
<td>Too many keys were found.</td>
</tr>
<tr>
<td>2050</td>
<td>passphrase cache disabled</td>
<td>The passphrase cache is disabled.</td>
</tr>
<tr>
<td>2051</td>
<td>revoker already present</td>
<td>The specified revoker is already present on the key, and thus cannot be added.</td>
</tr>
<tr>
<td>2052</td>
<td>ADK already present</td>
<td>The ADK is already present on the key, and thus cannot be added.</td>
</tr>
<tr>
<td>2053</td>
<td>unable to set export passphrase</td>
<td>PGP Command Line is unable to set an export passphrase.</td>
</tr>
<tr>
<td>Code</td>
<td>Message</td>
<td>Description</td>
</tr>
<tr>
<td>-------</td>
<td>----------------------------------------------</td>
<td>------------------------------------------------------------------</td>
</tr>
<tr>
<td>2054</td>
<td>too many matches for X.509 certificate</td>
<td>There are too many matches for the X.509 certificate.</td>
</tr>
<tr>
<td>2055</td>
<td>X.509 certificate not found</td>
<td>The X.509 certificate was not found.</td>
</tr>
<tr>
<td>2056</td>
<td>one or more attribute value pairs are required</td>
<td>One or more attribute value pairs are required.</td>
</tr>
<tr>
<td>2057</td>
<td>only one X.509 certificate can be imported at a time</td>
<td>Only one X.509 certificate can be imported at one time.</td>
</tr>
<tr>
<td>2058</td>
<td>key does not match X.509 certificate</td>
<td>The key does not match the X.509 certificate.</td>
</tr>
<tr>
<td>2059</td>
<td>error decoding X.509 certificate</td>
<td>An error occurred during decoding of the X.509 certificate.</td>
</tr>
<tr>
<td>2060</td>
<td>no shares specified</td>
<td>No shares were specified.</td>
</tr>
<tr>
<td>2061</td>
<td>invalid share</td>
<td>One of the specified shares is invalid.</td>
</tr>
<tr>
<td>2062</td>
<td>threshold must be between 1 and the total number of shares inclusive</td>
<td>The threshold setting must be between 1 and the total number of shares being created.</td>
</tr>
<tr>
<td>2063</td>
<td>there must be at least 2 recipients</td>
<td>There must be at least the specified number of recipients when splitting a key.</td>
</tr>
<tr>
<td>2064</td>
<td>split key cannot be a share recipient</td>
<td>The key being split cannot be its own recipient.</td>
</tr>
<tr>
<td>2065</td>
<td>share file</td>
<td>Displays the share file name for every recipient of a share when the key is split. Informational.</td>
</tr>
<tr>
<td>2066</td>
<td>there can only be X recipients</td>
<td>There can only be the specified number of recipients.</td>
</tr>
<tr>
<td>2067</td>
<td>there can only be 255 total shares</td>
<td>There can only be 255 total shares when splitting a key.</td>
</tr>
<tr>
<td>2068</td>
<td>this key is already a share recipient</td>
<td>The specified key is already a share recipient.</td>
</tr>
<tr>
<td>2069</td>
<td>this user is already a share recipient</td>
<td>The specified user is already a share recipient.</td>
</tr>
<tr>
<td>2070</td>
<td>could not open share file</td>
<td>PGP Command Line could not open the share file.</td>
</tr>
<tr>
<td>2071</td>
<td>share file key ID does not match split key</td>
<td>The key ID of the share file does not match that of the split key.</td>
</tr>
<tr>
<td>2072</td>
<td>share file threshold does not match split key</td>
<td>The threshold of the share file does not match that of the split key.</td>
</tr>
<tr>
<td>2073</td>
<td>share file owner not found</td>
<td>The key the share file is encrypted to was not found. This error cannot happen to conventionally encrypted shares.</td>
</tr>
<tr>
<td>2074</td>
<td>not enough shares collected for split key</td>
<td>Not enough shares were collected to reconstitute the split key.</td>
</tr>
<tr>
<td>2075</td>
<td>invalid passphrase for user X Y</td>
<td>An invalid passphrase was entered for the specified share file.</td>
</tr>
<tr>
<td>Code</td>
<td>Message</td>
<td>Description</td>
</tr>
<tr>
<td>-------</td>
<td>----------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>2076</td>
<td>invalid passphrase for X</td>
<td>An invalid passphrase was entered for a conventionally encrypted share file.</td>
</tr>
<tr>
<td>2077</td>
<td>duplicate shares detected</td>
<td>Duplicate share files were detected on key join.</td>
</tr>
<tr>
<td>2078</td>
<td>non-standard user ID</td>
<td>A non-standard user ID was detected. User IDs not in the form “common name &lt;contact&gt;” generate a warning.</td>
</tr>
<tr>
<td>2079</td>
<td>the primary user ID cannot be a photo ID</td>
<td>You cannot specify a photo ID as the primary user ID for a key.</td>
</tr>
<tr>
<td>2080</td>
<td>unknown input format</td>
<td>PGP Command Line encountered unknown input format</td>
</tr>
<tr>
<td>2081</td>
<td>no key flag specified</td>
<td>No key flag was specified.</td>
</tr>
<tr>
<td>2082</td>
<td>subkeys do not support keyserver preferences</td>
<td>Subkeys do not support keyserver preferences.</td>
</tr>
<tr>
<td>2083</td>
<td>subkeys do not support feature flags</td>
<td>Subkeys do not support feature flags.</td>
</tr>
<tr>
<td>2084</td>
<td>only one share can be sent at a time</td>
<td>You can only send one share at a time.</td>
</tr>
<tr>
<td>2085</td>
<td>connected to share server</td>
<td>You are connect to a share server.</td>
</tr>
<tr>
<td>2086</td>
<td>invalid SKEP timeout</td>
<td>PGP Command Line encountered an invalid SKEP timeout.</td>
</tr>
<tr>
<td>2087</td>
<td>network share key ID does not match split key</td>
<td>The network share key ID does not match that of the split key.</td>
</tr>
<tr>
<td>2088</td>
<td>network share threshold does not match split key</td>
<td>The network share threshold does not match that of the split key.</td>
</tr>
<tr>
<td>2089</td>
<td>timeout waiting for network shares</td>
<td>A timeout was exceeded waiting for network shares.</td>
</tr>
<tr>
<td>2090</td>
<td>no share server specified</td>
<td>No share server was specified.</td>
</tr>
<tr>
<td>2091</td>
<td>connected to share client</td>
<td>You are connected to a share client.</td>
</tr>
<tr>
<td>2092</td>
<td>SKEP authenticated with user x</td>
<td>SKEP authenticated with the specified user.</td>
</tr>
<tr>
<td>2093</td>
<td>shares received, x</td>
<td>The specified number of shares were received.</td>
</tr>
<tr>
<td>2094</td>
<td>this key has NOT been permanently revoked</td>
<td>The specified key has not been permanently revoked.</td>
</tr>
<tr>
<td>2095</td>
<td>non-standard user ID</td>
<td>PGP Command Line encountered a non-standard user ID.</td>
</tr>
<tr>
<td>2096</td>
<td>the MDC flag cannot be cleared</td>
<td>PGP Command Line cannot clear an MDC flag.</td>
</tr>
</tbody>
</table>

**Keyservers**

244
<table>
<thead>
<tr>
<th>Code</th>
<th>Message</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>key imported as X</td>
<td>The key was imported as specified.</td>
</tr>
<tr>
<td>0</td>
<td>key uploaded to X</td>
<td>The key was uploaded to the specified keyserver.</td>
</tr>
<tr>
<td>0</td>
<td>key removed from X</td>
<td>The key was removed from the specified keyserver.</td>
</tr>
<tr>
<td>0</td>
<td>key disabled on X</td>
<td>The key was disabled on the specified keyserver.</td>
</tr>
<tr>
<td>2500</td>
<td>no keyserver specified</td>
<td>No keyserver was specified.</td>
</tr>
<tr>
<td>2501</td>
<td>invalid keyserver specified</td>
<td>An error was detected on the specified keyserver.</td>
</tr>
<tr>
<td>2502</td>
<td>keyserver operation timed out</td>
<td>The keyserver operation timed out.</td>
</tr>
<tr>
<td>2503</td>
<td>invalid keyserver timeout value</td>
<td>An invalid keyserver timeout value was encountered.</td>
</tr>
<tr>
<td>2504</td>
<td>successful search</td>
<td>Displays the keyserver that matched the search. Informational.</td>
</tr>
<tr>
<td>2505</td>
<td>keyserver error: X</td>
<td>The specified keyserver error was encountered.</td>
</tr>
<tr>
<td>2506</td>
<td>skipping invalid preferred keyserver</td>
<td>The preferred keyserver is invalid, so it was skipped.</td>
</tr>
<tr>
<td>2507</td>
<td>key not found on any keyserver</td>
<td>The specified key was not found on any keyserver.</td>
</tr>
<tr>
<td>2508</td>
<td>too many matches found</td>
<td>The search timed out while still receiving results from the keyserver.</td>
</tr>
<tr>
<td>2509</td>
<td>keyserver error</td>
<td>Lists the keyserver that caused the error.</td>
</tr>
<tr>
<td>2510</td>
<td>unsuccessful search</td>
<td>The search was unsuccessful; no keys matched the search criteria.</td>
</tr>
</tbody>
</table>
## Key Reconstruction

<table>
<thead>
<tr>
<th>Code</th>
<th>Message</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>reconstruction data sent successfully</td>
<td>The key reconstruction data was sent successfully.</td>
</tr>
<tr>
<td>0</td>
<td>reconstruction questions received successfully</td>
<td>The key reconstruction questions were received successfully.</td>
</tr>
<tr>
<td>0</td>
<td>key reconstructed successfully</td>
<td>The key was reconstructed.</td>
</tr>
<tr>
<td>2600</td>
<td>no reconstruction server found for this key</td>
<td>There is no reconstruction server associated with the specified key.</td>
</tr>
<tr>
<td>2601</td>
<td>reconstruction server on port x</td>
<td>There is no reconstruction server on the specified port.</td>
</tr>
<tr>
<td>2602</td>
<td>five questions must be specified for key reconstruction</td>
<td>You must specify five questions to set up key reconstruction.</td>
</tr>
<tr>
<td>2603</td>
<td>empty reconstruction question</td>
<td>Not all key reconstruction questions were submitted.</td>
</tr>
<tr>
<td>2604</td>
<td>five answers must be specified for key reconstruction</td>
<td>Not all key reconstruction answers were submitted.</td>
</tr>
<tr>
<td>2605</td>
<td>empty reconstruction answer</td>
<td>A key reconstruction answer held no data.</td>
</tr>
<tr>
<td>2606</td>
<td>reconstruction question too long</td>
<td>A key reconstruction question was too long.</td>
</tr>
<tr>
<td>2607</td>
<td>reconstruction answer too long</td>
<td>A key reconstruction answer was too long.</td>
</tr>
<tr>
<td>2608</td>
<td>reconstruction server name too long</td>
<td>The key reconstruction server name was too long.</td>
</tr>
<tr>
<td>2609</td>
<td>invalid reconstruction server</td>
<td>An invalid reconstruction server was specified.</td>
</tr>
<tr>
<td>2610</td>
<td>key reconstruction data not found on server</td>
<td>No key reconstruction data was found on the specified server.</td>
</tr>
<tr>
<td>2611</td>
<td>key reconstruction answers are not valid with this key</td>
<td>The specified key reconstruction answers aren’t valid for the specified key.</td>
</tr>
<tr>
<td>2612</td>
<td>invalid key reconstruction data</td>
<td>The submitted key reconstruction data is invalid.</td>
</tr>
</tbody>
</table>

## Licensing

<table>
<thead>
<tr>
<th>Code</th>
<th>Message</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>license authorized</td>
<td>Your PGP Command Line license has been authorized.</td>
</tr>
<tr>
<td>0</td>
<td>license recovery email requested</td>
<td>A PGP Command Line license recovery email was requested.</td>
</tr>
<tr>
<td>2700</td>
<td>no license name specified</td>
<td>No Name was specified in the license request.</td>
</tr>
<tr>
<td>Code</td>
<td>Message</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>------------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>2701</td>
<td>no license email address specified</td>
<td>No Email Address was specified in the license request.</td>
</tr>
<tr>
<td>2702</td>
<td>no license organization specified</td>
<td>No Organization was specified in the license request.</td>
</tr>
<tr>
<td>2703</td>
<td>no license number specified</td>
<td>No license number was specified in the license request.</td>
</tr>
<tr>
<td>2704</td>
<td>invalid license number</td>
<td>An invalid license number was submitted.</td>
</tr>
<tr>
<td>2705</td>
<td>this license is for a different PGP product</td>
<td>The submitted license is for a different product line from PGP Corporation.</td>
</tr>
<tr>
<td>2706</td>
<td>PGP Command Line already has a license</td>
<td>This copy of PGP Command Line is already licensed.</td>
</tr>
<tr>
<td>2707</td>
<td>invalid license authorization</td>
<td>An invalid license authorization was submitted.</td>
</tr>
<tr>
<td>2708</td>
<td>the current license is expired - please contact support</td>
<td>Your PGP Command Line license has expired; please contact PGP Corporation.</td>
</tr>
<tr>
<td>2709</td>
<td>license authorization failed</td>
<td>The license authorization failed. Try again later.</td>
</tr>
<tr>
<td>2710</td>
<td>days left in current license, x</td>
<td>The specified number of days are left on the current license.</td>
</tr>
<tr>
<td>2711</td>
<td>could not store license information</td>
<td>PGP Command Line could not store the license information.</td>
</tr>
<tr>
<td>2712</td>
<td>invalid license</td>
<td>The PGP Command Line license is invalid.</td>
</tr>
<tr>
<td>2713</td>
<td>no license has been entered</td>
<td>No license was entered.</td>
</tr>
<tr>
<td>2714</td>
<td>encrypt / sign not allowed with this license</td>
<td>Encrypting and signing are not supported by your current license.</td>
</tr>
<tr>
<td>2715</td>
<td>decrypt / verify not allowed with this license</td>
<td>Decrypting and verifying are not supported by your current license.</td>
</tr>
<tr>
<td>2716</td>
<td>number of CPUs not allowed with the current license</td>
<td>The number of CPUs on the computer hosting PGP Command Line is not supported by the current license.</td>
</tr>
</tbody>
</table>
### PGP Universal Server

<table>
<thead>
<tr>
<th>Code</th>
<th>Message</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2800</td>
<td>could not connect to server</td>
<td>PGP Universal Server could not connect to the specified PGP Universal Server.</td>
</tr>
<tr>
<td>2801</td>
<td>server authentication failed</td>
<td>PGP Universal Server could not authenticate to the specified PGP Universal Server.</td>
</tr>
<tr>
<td>2802</td>
<td>server responded with request</td>
<td>The specified PGP Universal Server responded that the request failed.</td>
</tr>
</tbody>
</table>

### General

<table>
<thead>
<tr>
<th>Code</th>
<th>Message</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>output file X</td>
<td>The specified file was output.</td>
</tr>
<tr>
<td>0</td>
<td>output symbolic link X</td>
<td>The specified symbolic link was output.</td>
</tr>
<tr>
<td>0</td>
<td>output of archive files</td>
<td>The archive files were output.</td>
</tr>
<tr>
<td></td>
<td>successful</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>file created successfully</td>
<td>The file was created.</td>
</tr>
<tr>
<td>0</td>
<td>directory created successfully</td>
<td>The directory was created.</td>
</tr>
<tr>
<td>0</td>
<td>cache purge successful</td>
<td>The cache was purged.</td>
</tr>
<tr>
<td>0</td>
<td>created symbolic link to X</td>
<td>A symbolic link to the specified item was created.</td>
</tr>
<tr>
<td>3000</td>
<td>no input file specified</td>
<td>No input file was specified.</td>
</tr>
<tr>
<td>3001</td>
<td>input file not found</td>
<td>The input file was not found.</td>
</tr>
<tr>
<td>3002</td>
<td>invalid argument for wipe</td>
<td>PGP Command Line encountered an invalid argument for wipe input passes.</td>
</tr>
<tr>
<td></td>
<td>input passes</td>
<td></td>
</tr>
<tr>
<td>3003</td>
<td>invalid argument for wipe</td>
<td>PGP Command Line encountered an invalid argument for wipe temp passes.</td>
</tr>
<tr>
<td></td>
<td>temp passes</td>
<td></td>
</tr>
<tr>
<td>3004</td>
<td>stdin cannot be used with</td>
<td>Standard input/output (stdin) cannot be used with input files.</td>
</tr>
<tr>
<td></td>
<td>input files</td>
<td></td>
</tr>
<tr>
<td>3005</td>
<td>no recipients specified</td>
<td>No recipients were specified.</td>
</tr>
<tr>
<td>3006</td>
<td>ADK added to recipients</td>
<td>The ADK was added to the recipients. Informational, not an error.</td>
</tr>
<tr>
<td>3007</td>
<td>ADK not found</td>
<td>The ADK was not found. Indicates an error; based on the setting of --enforce-adk.</td>
</tr>
<tr>
<td>3008</td>
<td>skipping ADK</td>
<td>The ADK was not enforced.</td>
</tr>
<tr>
<td>3009</td>
<td>ADK not found</td>
<td>The ADK was not found. Indicates a warning; based on the setting of --enforce-adk.</td>
</tr>
<tr>
<td>3010</td>
<td>no symmetric passphrase</td>
<td>No symmetric passphrase was specified.</td>
</tr>
<tr>
<td></td>
<td>specified</td>
<td></td>
</tr>
<tr>
<td>3011</td>
<td>invalid passphrase specified</td>
<td>An invalid passphrase was specified.</td>
</tr>
<tr>
<td>Code</td>
<td>Message</td>
<td>Description</td>
</tr>
<tr>
<td>-------</td>
<td>----------------------------------------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>3012</td>
<td>could not create output file</td>
<td>PGP Command Line could not create the output file.</td>
</tr>
<tr>
<td>3083</td>
<td>could not create output file X</td>
<td>PGP Command Line could not create the specified output file.</td>
</tr>
<tr>
<td>3013</td>
<td>no keys found</td>
<td>No keys were found.</td>
</tr>
<tr>
<td>3014</td>
<td>no keys specified</td>
<td>No keys were specified.</td>
</tr>
<tr>
<td>3015</td>
<td>failed with error X</td>
<td>The operation failed with the specified error number; error text not available.</td>
</tr>
<tr>
<td>3090</td>
<td>operation failed: X</td>
<td>The operation failed with the specified error text.</td>
</tr>
<tr>
<td>3092</td>
<td>operation warning: X</td>
<td>Operation encountered the specified warning condition.</td>
</tr>
<tr>
<td>3016</td>
<td>invalid user ID specified</td>
<td>An invalid user ID was specified; it cannot be used.</td>
</tr>
<tr>
<td>3017</td>
<td>user ID already exists</td>
<td>The specified user ID already exists.</td>
</tr>
<tr>
<td>3018</td>
<td>user ID not found</td>
<td>The specified user ID not found.</td>
</tr>
<tr>
<td>3019</td>
<td>file operation failed</td>
<td>The file operation failed.</td>
</tr>
<tr>
<td>3020</td>
<td>photo ID not found</td>
<td>The specified photo ID was not found.</td>
</tr>
<tr>
<td>3021</td>
<td>revokers are not supported with this key</td>
<td>Revokers are not supported with this key.</td>
</tr>
<tr>
<td>3022</td>
<td>ADKs are not supported with this key</td>
<td>ADKs are not supported with this key.</td>
</tr>
<tr>
<td>3023</td>
<td>key expired</td>
<td>The key is expired.</td>
</tr>
<tr>
<td>3024</td>
<td>key revoked</td>
<td>The key is revoked.</td>
</tr>
<tr>
<td>3025</td>
<td>key disabled</td>
<td>The key is disabled.</td>
</tr>
<tr>
<td>3026</td>
<td>key is not paired</td>
<td>The key is not paired.</td>
</tr>
<tr>
<td>3027</td>
<td>file locked</td>
<td>The file is locked.</td>
</tr>
<tr>
<td>3028</td>
<td>multiple inputs cannot be sent to a single output file</td>
<td>Multiple inputs cannot be sent to a single output file.</td>
</tr>
<tr>
<td>3029</td>
<td>no output specified</td>
<td>No output was specified.</td>
</tr>
<tr>
<td>3030</td>
<td>cannot output to a directory when reading from stdin</td>
<td>PGP Command Line cannot output to a directory when reading from standard input.</td>
</tr>
<tr>
<td>3031</td>
<td>input does not contain PGP data</td>
<td>The input does not contain any PGP data.</td>
</tr>
<tr>
<td>3032</td>
<td>input contains unknown data</td>
<td>The input contains unknown data.</td>
</tr>
<tr>
<td>3033</td>
<td>no passphrase specified</td>
<td>No passphrase was specified.</td>
</tr>
<tr>
<td>3034</td>
<td>file is marked for your eyes only, ignoring output</td>
<td>The specified file is marked “eyes only;” the output is being ignored.</td>
</tr>
<tr>
<td>3035</td>
<td>good signature</td>
<td>The signature is good.</td>
</tr>
<tr>
<td>3036</td>
<td>bad signature</td>
<td>The signature is bad.</td>
</tr>
<tr>
<td>Code</td>
<td>Message</td>
<td>Description</td>
</tr>
<tr>
<td>-------</td>
<td>---------------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>3037</td>
<td>cannot verify signature</td>
<td>PGP Command Line cannot verify the signature because the signing key was not found on the local keyring.</td>
</tr>
<tr>
<td>3038</td>
<td>signing key [key ID] [primary user ID]</td>
<td>Informational message when verifying the signature on a key; displays the key ID and primary user ID of the key used to verify with.</td>
</tr>
<tr>
<td>3039</td>
<td>signing key [key ID]</td>
<td>Informational message when verifying the signature on a key; displays the key ID of the key used to verify with.</td>
</tr>
<tr>
<td>3040</td>
<td>signature created [date]</td>
<td>Informational message that shows the date the signature was created.</td>
</tr>
<tr>
<td>3041</td>
<td>output not applicable</td>
<td>The --output option is not applicable; when doing a verify, for example.</td>
</tr>
<tr>
<td>3042</td>
<td>suggested output file name X</td>
<td>The suggested output filename is as specified.</td>
</tr>
<tr>
<td>3043</td>
<td>data is marked for your eyes only</td>
<td>Data is marked “eyes only.”</td>
</tr>
<tr>
<td>3044</td>
<td>subkey ID X belongs to Y</td>
<td>If the owner of the subkey is available, it is displayed; otherwise, just the subkey is displayed.</td>
</tr>
<tr>
<td>3093</td>
<td>data is encrypted to subkey ID X</td>
<td>The data is encrypted to the specified subkey ID.</td>
</tr>
<tr>
<td>3045</td>
<td>data is conventionally encrypted</td>
<td>The data is conventionally encrypted.</td>
</tr>
<tr>
<td>3046</td>
<td>preferred keyservers are not supported with this key</td>
<td>Preferred keyservers are not supported with this key; they are only supported on RSA and DH/DSS v4 keys.</td>
</tr>
<tr>
<td>3047</td>
<td>no new passphrase specified</td>
<td>No new passphrase was specified.</td>
</tr>
<tr>
<td>3048</td>
<td>data encrypted with cipher X</td>
<td>The data is encrypted with the specified cipher.</td>
</tr>
<tr>
<td>3049</td>
<td>key unsuitable for signing</td>
<td>The key is unsuitable for signing.</td>
</tr>
<tr>
<td>3050</td>
<td>too many user IDs found</td>
<td>Too many user IDs were found.</td>
</tr>
<tr>
<td>3051</td>
<td>trust level for meta-introducers must be from 2 to 8 inclusive</td>
<td>The trust level you specify for meta-introducers must be from 2 to 8.</td>
</tr>
<tr>
<td>3052</td>
<td>trust level for trusted-introducers must be from 1 to 8 inclusive</td>
<td>The trust level you specify for trusted-introducers must be from 1 to 8.</td>
</tr>
<tr>
<td>3053</td>
<td>too many signatures found</td>
<td>Too many signatures were found.</td>
</tr>
<tr>
<td>3054</td>
<td>no signatures found</td>
<td>No signatures were found.</td>
</tr>
<tr>
<td>3055</td>
<td>data contains the key X</td>
<td>Data contains the specified key.</td>
</tr>
<tr>
<td>3056</td>
<td>key import off, skipping key X</td>
<td>Error occurred during import; the import failed.</td>
</tr>
<tr>
<td>3057</td>
<td>key is not revocable</td>
<td>You cannot revoke the key.</td>
</tr>
<tr>
<td>3058</td>
<td>subkey not found</td>
<td>The subkey was not found.</td>
</tr>
<tr>
<td>Code</td>
<td>Message</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>----------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>3059</td>
<td>subkeys are not supported with this key</td>
<td>The specified key does not support subkeys.</td>
</tr>
<tr>
<td>3060</td>
<td>no subkey specified</td>
<td>No subkey was specified.</td>
</tr>
<tr>
<td>3061</td>
<td>data not encrypted</td>
<td>The data is not encrypted.</td>
</tr>
<tr>
<td>3062</td>
<td>could not create file, X</td>
<td>PGP Command Line could not create a file because of the specified error.</td>
</tr>
<tr>
<td>3063</td>
<td>key unable to encrypt</td>
<td>The key is unable to encrypt.</td>
</tr>
<tr>
<td>3064</td>
<td>key invalid</td>
<td>The key is invalid.</td>
</tr>
<tr>
<td>3065</td>
<td>signing key invalid</td>
<td>The signing key is invalid.</td>
</tr>
<tr>
<td>3066</td>
<td>key cannot be an ADK</td>
<td>The key cannot be an ADK.</td>
</tr>
<tr>
<td>3067</td>
<td>key is axiomatic</td>
<td>The key is axiomatic. You cannot disable a key pair until you set trust to Never.</td>
</tr>
<tr>
<td>3068</td>
<td>invalid key type</td>
<td>The key type is invalid.</td>
</tr>
<tr>
<td>3069</td>
<td>RSA legacy key size must be between A and Z</td>
<td>The key size of RSA Legacy keys must be between the specified values.</td>
</tr>
<tr>
<td>3070</td>
<td>RSA legacy key type does not support signing bits</td>
<td>The RSA Legacy key type does not support signing bits.</td>
</tr>
<tr>
<td>3071</td>
<td>too many user IDs specified</td>
<td>Too many user IDs specified.</td>
</tr>
<tr>
<td>3072</td>
<td>RSA key size must be between A and Z</td>
<td>The key size of RSA keys must be between the specified values.</td>
</tr>
<tr>
<td>3073</td>
<td>RSA signing key size must be between A and Z</td>
<td>The signing key size of RSA keys must be between the specified values.</td>
</tr>
<tr>
<td>3074</td>
<td>DH key size must be between A and Z</td>
<td>The key size of Diffie-Hellman keys must be between the specified values.</td>
</tr>
<tr>
<td>3075</td>
<td>DH signing key size must be X</td>
<td>The signing key size of Diffie-Hellman keys must be the specified size.</td>
</tr>
<tr>
<td>3076</td>
<td>encryption key size cannot be specified with sign only key type</td>
<td>Encryption key size cannot be specified with sign-only key types.</td>
</tr>
<tr>
<td>3077</td>
<td>out of entropy</td>
<td>PGP Command Line is out of entropy.</td>
</tr>
<tr>
<td>3078</td>
<td>could not create directory, X</td>
<td>PGP Command Line could not create a directory, because of the specified error.</td>
</tr>
<tr>
<td>3080</td>
<td>invalid index</td>
<td>The index is invalid.</td>
</tr>
<tr>
<td>3082</td>
<td>invalid date</td>
<td>The date is invalid.</td>
</tr>
<tr>
<td>3084</td>
<td>stdin not applicable</td>
<td>Standard input/output is not applicable.</td>
</tr>
<tr>
<td>3085</td>
<td>no signature specified</td>
<td>No signature was specified when matching signatures on user IDs (not signature files).</td>
</tr>
<tr>
<td>3086</td>
<td>skipping directory</td>
<td>The directory is being skipped.</td>
</tr>
<tr>
<td>Code</td>
<td>Message</td>
<td>Description</td>
</tr>
<tr>
<td>-------</td>
<td>----------------------------------------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>3087</td>
<td>could not remove file, X</td>
<td>PGP Command Line could not remove a file because of the specified error.</td>
</tr>
<tr>
<td>3088</td>
<td>invalid passphrase cache timeout</td>
<td>An invalid passphrase cache timeout was encountered.</td>
</tr>
<tr>
<td>3089</td>
<td>preferred ciphers are not supported with this key</td>
<td>The key does not support preferred ciphers.</td>
</tr>
<tr>
<td>3091</td>
<td>skipping non-regular file</td>
<td>An irregular (device, fifo, and so on) file is being skipped.</td>
</tr>
<tr>
<td>3100</td>
<td>signing key expired</td>
<td>The signing key is expired.</td>
</tr>
<tr>
<td>3101</td>
<td>signing key revoked</td>
<td>The signing key is revoked.</td>
</tr>
<tr>
<td>3102</td>
<td>signing key disabled</td>
<td>The signing key is disabled.</td>
</tr>
<tr>
<td>3103</td>
<td>photo IDs are not supported with this key</td>
<td>The key does not support photo IDs.</td>
</tr>
<tr>
<td>3104</td>
<td>could not read file</td>
<td>PGP Command Line could not read the file.</td>
</tr>
<tr>
<td>3105</td>
<td>cipher not applicable</td>
<td>The --cipher option is not applicable, not a specific cipher.</td>
</tr>
<tr>
<td>3106</td>
<td>preferred compression algorithms are not supported with this key</td>
<td>The key does not support preferred compression algorithms.</td>
</tr>
<tr>
<td>3107</td>
<td>compression algorithm not applicable</td>
<td>The --compression-algorithm option is not applicable, not a specific compression algorithm.</td>
</tr>
<tr>
<td>3108</td>
<td>permission denied, force option required</td>
<td>The --force option is required for this operation.</td>
</tr>
<tr>
<td>3109</td>
<td>output cannot be a directory, it must be a file</td>
<td>The output cannot be a directory, it must be a file.</td>
</tr>
<tr>
<td>3110</td>
<td>archive imported X</td>
<td>The specified archive was imported, where X is the file or directory just added to the archive. This is a progress message.</td>
</tr>
<tr>
<td>3111</td>
<td>data is a PGP archive</td>
<td>The data is a PGP Archive.</td>
</tr>
<tr>
<td>3112</td>
<td>input does not contain PGP archive data</td>
<td>The input does not contain PGP Archive data.</td>
</tr>
<tr>
<td>3113</td>
<td>data is armored</td>
<td>The data is ASCII-armored.</td>
</tr>
<tr>
<td>3114</td>
<td>ADK not valid for use</td>
<td>The ADK is not valid for use; it cannot encrypt (this is an error message).</td>
</tr>
<tr>
<td>3115</td>
<td>ADK not valid for use</td>
<td>The ADK is not valid for use; it cannot encrypt (this is a warning message).</td>
</tr>
<tr>
<td>3116</td>
<td>invalid additional recipient</td>
<td>The additional recipient is invalid.</td>
</tr>
<tr>
<td>3117</td>
<td>additional recipient not found</td>
<td>The additional recipient was not found.</td>
</tr>
<tr>
<td>3118</td>
<td>X.509 operations require a single key</td>
<td>The X.509 operation requires a single key.</td>
</tr>
<tr>
<td>Code</td>
<td>Message</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------------------------------------------------------------------</td>
<td>---------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>3119</td>
<td>no local key for merge, skipping key X Y</td>
<td>Because there was no local key for the merge, the specified keys were skipped; depends on the setting of manual import keys.</td>
</tr>
<tr>
<td>3120</td>
<td>local key exists, skipping key X Y</td>
<td>The local key exists, but the specified keys are being skipped; depends on the setting of manual import keys.</td>
</tr>
<tr>
<td>3121</td>
<td>automatically imported key [key ID] [primary user ID]</td>
<td>The specified keys were automatically imported.</td>
</tr>
<tr>
<td>3122</td>
<td>PGP Command Line Beta has expired - please update to the latest release</td>
<td>The Beta version of PGP Command Line that you are using has expired. You need to get a more recent version.</td>
</tr>
<tr>
<td>3123</td>
<td>could not remove directory, X</td>
<td>PGP Command Line could not remove a directory because of the specified error.</td>
</tr>
<tr>
<td>3124</td>
<td>permission denied</td>
<td>Permission is denied.</td>
</tr>
<tr>
<td>3125</td>
<td>input is not a regular file</td>
<td>The input is not a regular file.</td>
</tr>
<tr>
<td>3126</td>
<td>invalid input</td>
<td>The input is invalid.</td>
</tr>
<tr>
<td>3127</td>
<td>private key is already split</td>
<td>The private key is already split.</td>
</tr>
<tr>
<td>3128</td>
<td>output must be a directory</td>
<td>The output must be a directory.</td>
</tr>
<tr>
<td>3129</td>
<td>path too long</td>
<td>The path is too long.</td>
</tr>
<tr>
<td>3130</td>
<td>could not create symbolic link, X</td>
<td>PGP Command Line could not create a symbolic link because of the specified error.</td>
</tr>
<tr>
<td>3131</td>
<td>multiple PGP blocks found in single input stream</td>
<td>Multiple PGP blocks were encountered in a single input stream.</td>
</tr>
<tr>
<td>3132</td>
<td>reconstructed split key passphrase is invalid</td>
<td>The reconstructed split key passphrase is invalid.</td>
</tr>
<tr>
<td>3133</td>
<td>key unable to decrypt</td>
<td>The key is unable to decrypt.</td>
</tr>
<tr>
<td>3134</td>
<td>reconstructed split key passphrase is valid</td>
<td>The reconstructed split key passphrase is valid.</td>
</tr>
<tr>
<td>3135</td>
<td>master passphrase changed</td>
<td>The master passphrase has changed.</td>
</tr>
<tr>
<td>3136</td>
<td>subkey passphrase changed</td>
<td>The subkey passphrase has changed.</td>
</tr>
<tr>
<td>3137</td>
<td>eyes only option not specified, discarding output</td>
<td>The output is being discarded because the --eyes-only option was not specified.</td>
</tr>
<tr>
<td>3138</td>
<td>error opening console</td>
<td>There was an error opening the console; for direct writing (--eyes-only option).</td>
</tr>
<tr>
<td>3139</td>
<td>error writing to console</td>
<td>There was an error writing to the console; for direct writing (--eyes-only option).</td>
</tr>
<tr>
<td>3140</td>
<td>private key is not split</td>
<td>The private key is not split.</td>
</tr>
<tr>
<td>3141</td>
<td>operation warning: Y</td>
<td>The operation generated the specified warning.</td>
</tr>
<tr>
<td>3142</td>
<td>data is encrypted to key ID X</td>
<td>Data is encrypted to an RSA Legacy key, which do not have subkeys. Data is encrypted to the specified key ID.</td>
</tr>
<tr>
<td>Code</td>
<td>Message</td>
<td>Description</td>
</tr>
<tr>
<td>------</td>
<td>---------</td>
<td>-------------</td>
</tr>
<tr>
<td>3143</td>
<td>key belongs to X Y</td>
<td>Data is encrypted to an RSA Legacy key, which do not have subkeys. Specified key ID is matched to the specified primary user ID.</td>
</tr>
<tr>
<td>3144</td>
<td>data is encrypted to unknown ID X</td>
<td>PGP Command Line could not find a key, so the specified ID is unknown.</td>
</tr>
<tr>
<td>3145</td>
<td>invalid argument for wipe overwrite passes</td>
<td>PGP Command Line encountered an invalid argument for wipe overwrite passes.</td>
</tr>
<tr>
<td>3146</td>
<td>error [number] importing key X</td>
<td>The specified error occurred; the specified key is being imported.</td>
</tr>
<tr>
<td>3147</td>
<td>key pair import off, skipping key x</td>
<td>The specified key was skipped because key pair import is off.</td>
</tr>
<tr>
<td>3148</td>
<td>importing only public key x y</td>
<td>Just the specified public keys are being imported.</td>
</tr>
<tr>
<td>3149</td>
<td>no target platform specified</td>
<td>No target platform was specified.</td>
</tr>
<tr>
<td>3150</td>
<td>unknown file type</td>
<td>PGP Command Line encountered an unknown file type.</td>
</tr>
<tr>
<td>3151</td>
<td>only one input is allowed</td>
<td>Only one input is allowed.</td>
</tr>
<tr>
<td>3152</td>
<td>stdout not applicable</td>
<td>Standard output is not applicable.</td>
</tr>
<tr>
<td>3153</td>
<td>connection failed</td>
<td>The connection failed.</td>
</tr>
<tr>
<td>3154</td>
<td>invalid keyring cache timeout</td>
<td>An invalid keyring cache timeout was specified.</td>
</tr>
<tr>
<td>3155</td>
<td>preferred hashes are not supported with this key</td>
<td>Preferred hashes are not supported on the specified key.</td>
</tr>
<tr>
<td>3156</td>
<td>hash not applicable</td>
<td>The specified hash is not applicable.</td>
</tr>
<tr>
<td>3157</td>
<td>current local time x</td>
<td>The current local time is as specified.</td>
</tr>
<tr>
<td>3158</td>
<td>current UTC time x</td>
<td>The current UTC time is as specified.</td>
</tr>
<tr>
<td>3159</td>
<td>multiple revokers not allowed</td>
<td>Multiple revokers are not allowed.</td>
</tr>
<tr>
<td>3160</td>
<td>root path not found in input object</td>
<td>The object input did not include the root path.</td>
</tr>
<tr>
<td>3161</td>
<td>root path invalid with input object</td>
<td>The object input does not supported a root path.</td>
</tr>
<tr>
<td>3162</td>
<td>no auth username specified</td>
<td>No authorization username was specified.</td>
</tr>
<tr>
<td>3163</td>
<td>no auth passphrase specified</td>
<td>No authentication passphrase was specified.</td>
</tr>
<tr>
<td>3164</td>
<td>only one notation value may be specified</td>
<td>You can only specify one notation value.</td>
</tr>
<tr>
<td>3165</td>
<td>notation packet not found</td>
<td>A notation packet could not be found.</td>
</tr>
<tr>
<td>3166</td>
<td>invalid notation packet search parameters</td>
<td>There was an invalid notation packet in the search parameters.</td>
</tr>
<tr>
<td>3167</td>
<td>invalid notation packet</td>
<td></td>
</tr>
<tr>
<td>3168</td>
<td>could not change owner, x</td>
<td>The specified packet owner could not be changed.</td>
</tr>
<tr>
<td>Code</td>
<td>Message</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-----------------------------</td>
<td>-------------------------------------------------------</td>
</tr>
<tr>
<td>3169</td>
<td>could not change permissions, x</td>
<td>The specified permission could not be changed.</td>
</tr>
<tr>
<td>3170</td>
<td>signature hash x</td>
<td>There’s a problem with the specified signature hash.</td>
</tr>
<tr>
<td>3171</td>
<td>libxml error - x, y</td>
<td>A structured error has occurred.</td>
</tr>
<tr>
<td>3172</td>
<td>libxml error - x</td>
<td>A generic error has occurred.</td>
</tr>
<tr>
<td>3173</td>
<td>libxml error - unknown</td>
<td>An unknown error has occurred.</td>
</tr>
</tbody>
</table>
Exit Codes

Exit codes are returned by PGP Command Line on exit from the application. Depending on the shell or script being used, these exit codes may or may not be displayed on-screen.

<table>
<thead>
<tr>
<th>Code</th>
<th>Message</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Success</td>
<td>PGP Command Line exited successfully.</td>
</tr>
<tr>
<td>64</td>
<td>Usage</td>
<td>Parser error.</td>
</tr>
<tr>
<td>71</td>
<td>OSError</td>
<td>Bad data was received from the operating system at startup.</td>
</tr>
<tr>
<td>128</td>
<td>InternalError</td>
<td>An internal error occurred.</td>
</tr>
<tr>
<td>129</td>
<td>InitFailed</td>
<td>An initialization failure occurred on startup.</td>
</tr>
<tr>
<td>130</td>
<td>Interrupt</td>
<td>A user interrupt occurred.</td>
</tr>
<tr>
<td>145</td>
<td>PurgeCache</td>
<td>Error purging a cache: passphrase, keyring, or both.</td>
</tr>
<tr>
<td>146</td>
<td>CreateKeyrings</td>
<td>Error creating keyring files.</td>
</tr>
<tr>
<td>147</td>
<td>SpeedTest</td>
<td>Error during a speed test operation.</td>
</tr>
<tr>
<td>160</td>
<td>Wipe</td>
<td>Complete failure during a file wipe.</td>
</tr>
<tr>
<td>161</td>
<td>WipePartial</td>
<td>Partial fail, partial success during a file wipe (one file wiped, one not, for example).</td>
</tr>
<tr>
<td>162</td>
<td>Encode</td>
<td>Complete failure during an encode.</td>
</tr>
<tr>
<td>163</td>
<td>EncodePartial</td>
<td>Partial failure during an encode.</td>
</tr>
<tr>
<td>164</td>
<td>Decode</td>
<td>Complete failure during a decode.</td>
</tr>
<tr>
<td>165</td>
<td>DecodePartial</td>
<td>Partial failure during a decode.</td>
</tr>
<tr>
<td>210</td>
<td>KeyList</td>
<td>Error during one of the key list operations.</td>
</tr>
<tr>
<td>220</td>
<td>Key Maintenance</td>
<td>Error during key maintenance.</td>
</tr>
<tr>
<td>221</td>
<td>CheckSigs</td>
<td>Error when checking signatures.</td>
</tr>
<tr>
<td>222</td>
<td>CheckUserIDs</td>
<td>Error when checking user IDs.</td>
</tr>
<tr>
<td>230</td>
<td>KeyEdit</td>
<td>Error during one of the key edit operations.</td>
</tr>
<tr>
<td>231</td>
<td>KeyEdit Partial</td>
<td>Error during import of a key.</td>
</tr>
<tr>
<td>240</td>
<td>Keyserver</td>
<td>Error during one of the keyserver operations.</td>
</tr>
<tr>
<td>245</td>
<td>License</td>
<td>Error with supplied license.</td>
</tr>
<tr>
<td>250</td>
<td>BetaExpired</td>
<td>Returned if the software is expired due to beta timeout.</td>
</tr>
<tr>
<td>251</td>
<td>LicenseExpired</td>
<td>License is expired.</td>
</tr>
<tr>
<td>255</td>
<td>Unknown</td>
<td>An unknown error occurred.</td>
</tr>
</tbody>
</table>
Frequently Asked Questions
About PGP Command Line

This appendix lists some frequently asked questions about PGP Command Line and how it is used.

Q. How do I determine the key to which a file was encrypted?
A. Use the command `--verify` and the encrypted file name, such as:

```
pgp --verify report.pgp
```
You will get a report about the encryption subkey used to encrypt this file:

```
report.pgp:verify (3093: data is encrypted to subkey ID 0x894BA6DC)
report.pgp:verify (3044: subkey ID 0x894BA6DC belongs to 0x6245273E Bob Smith <bob@example.com>)
report.pgp:verify (3033: no passphrase specified)
```

Q. I imported my partner's public key to my keyring, but every time I encrypt to it, PGP Command Line gives me an error “3064: key invalid”! What does this mean?
A. The problem is that a key is not considered valid unless it is either signed by you or someone you trust, which ensures that you’re encrypting only to public key that has been confirmed to belong to the person with whom you wish to communicate.

You can simply sign the public key with your private key. Here is the whole key import and signing procedure:

1. Import the public key. If the public key is in a file called Alice.asc, use:

```
pgp --import "Alice Cameron.asc"
```
Alice Cameron.asc: import key (0: key imported as 0xD0EA20A7 Alice Cameron)

2. View the public key’s fingerprint. If this is Bob’s public key, use this command:

```
pgp --fingerprint "Alice Cameron"
```
Alice Cameron <alice@example.com>
6DE3 5CB2 DF01 8CF2 5569 971E A9B1 D272 3E43 9B98
1 key found

You can also use the biometric option to view the key:

```
pgp --fingerprint "Alice Cameron" --biometric
```
Alice Cameron <alice@example.com>
goggles  torpedo  escape  pioneer
talon    adviser  offload  vagabond
edict    guitarist  preshrunk  Burlington
revenge  photograph  standard  holiness
concert  decimal  puppy  narrative
1 key found
Now call Alice and verify that this is the correct public key by having her read her key’s fingerprint. If the fingerprints match, then you know you have the correct public key.

3 Sign the public key. If the public key is for a user called Alice, and your local private key is for a user called Bob, use:

```plaintext
pgp --sign-key "alice@example.com" --signer Smith --passphrase sm1t4
0x3E439B98:sign key (0:certified user ID Alice Cameron <alice@example.com>)
```

Alice's public key will now be valid for encryption operations.

Note that larger organizations normally establish a corporate key, sign all partner keys, and store them in a PGP keyserver. Individual Desktop or PGP Command Line installations then need only to validate and trust the corporate key. Because you trust the corporate key, PGP software knows that you also trust any key signed by the corporate key, meaning any partner key signed by the corporate key is automatically considered valid.

Q. What is the maximum size of file that PGP Command Line can encrypt?

A. There is no hard limit on the size of file you can encrypt using Command Line, where blocks of data are read from the input file, encrypted, and written to a temporary file. Once the encryption is complete, the temporary file is renamed to the proper output destination filename. Therefore, the output file is not loaded into memory at once and encrypted there before being written out to the output file.

There are some operating system and function-specific caveats:

- On Windows, AIX, and HP-UX the standard input stream works differently and PGP Command Line actually reads the whole file into memory: the user will be limited by the memory of the system and the swap file size. Hence, it’s preferable not to use standard input as the source of input for the encryption if you’re encrypting large files.

- Archiving: when using the `--archive` option, PGP Command Line first creates a compressed tar file of the input files/directories, and then encrypts that tar file. Therefore, you need to have available on the working drive two to three times the size of the file being encrypted.

The only limitation for PGP Command Line is the size of the hard drive on which you’ll be performing an operation.

Q. Can I use PGP Command Line with VB/.NET/Perl/Python/other languages?

A. Yes. You can call PGP Command Line via any programming language that allows you to call executables and pass parameters to the executable.
Q. How do I use file redirection with PGP Command Line?

A. PGP Command Line writes different data to several different places by default. Any user output generated by PGP Command Line is written to standard output (stdout), including version information, key list data, etc. Any status information generated by command line is sent to standard error (stderr).

When encrypting and decrypting, PGP Command Line reads and writes files by default. These files can be overridden with the special argument "-" to either --input or --output. This behavior is set so that PGP Command Line doesn't have to wait for input if you forget something: it will generate an error that you can detect.

The behavior of PGP Command Line changes depending on the operating system you are using, while the syntax changes depending on the shell.

When you work with PGP Command Line, you can use standard input (stdin) in two ways: by redirecting an existing file, or by typing (pasting in) data. See “Standard Input, Output, and Error” on page 42 for more information.

Q. What's the best way to protect a passphrase when I'm using PGP Command Line to automate encryption processes?

A. There are several ways to pass the passphrase into PGP Command Line: via a command-line option --passphrase, via PGP_PASSPHRASE environment variable, or via the passphrase cache.

- Passing the passphrase in via the command-line option. This is probably the least desirable, as it requires the script calling PGP Command Line to cache the passphrase. This may also be risky, especially if multiple users have access to the account responsible for running the script, as those users will be able to see the passphrase for private keys responsible for signing or decrypting data.

  To enter the passphrase onto the command line, you will use the option --passphrase combined with <passphrase>. Refer to “--cache-passphrase” on page 93 for more information.

- Using the environment variable PGP_PASSPHRASE.

  To set a passphrase environment variable PGP_PASSPHRASE, enter it in the way it's required for the platform you are using (refer to “Environment Variables” on page 41 for more information).

  You can add only one passphrase using this procedure. Note also that anyone who has access to your machine and the environment variables location can read your passphrase. This option is not recommended in any situation where other people can see your environment variable data.

- Using the passphrase cache. To change the passphrase cache settings using the configuration file, do the following:

  a  Open the PGPPrefs.xml file, which is located in the Application Data directory on Windows platform, or in the $HOME directory on any UNIX platform. For more information, refer to “Configuration File” on page 36.

  b  Find the text:
In addition, if you want to change the passphrase cache timeout to a value other than the default (120 seconds), find the text:

```xml
<key>CLpassphraseCacheTimeout</key>
<integer>120</integer>
```

and change the value to another, longer timeout.

If the machine is rebooted, the passphrase will need to be set in the cache again. This has the advantage that the passphrase is not exposed on the system. There is a slight risk that someone with access to the user account into which the passphrase has been cached will be able to perform operations using the private key (as operations requiring a passphrase for the private key will automatically pull the passphrase from the cache).
## Glossary

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ADK (Additional Decryption Key)</strong></td>
<td>An ADK is another key to which you encrypt a file or email message. Encrypting to an ADK means that two private keys can now decrypt the file or message: the private key of the recipient and the private key of the ADK. ADKs are generally used by companies to ensure another method of decryption for files or messages encrypted to the key of an employee who is unable or unwilling to decrypt the file/message. To make sure that the decryption capabilities of ADKs are not misused, ADKs are generally split after creation so that they are more difficult to use. A company’s security policy should include guidelines for using ADKs.</td>
</tr>
<tr>
<td><strong>AES (Advanced Encryption Standard)</strong></td>
<td>The NIST-approved encryption standard. The underlying cipher is Rijndael, a block cipher designed by Joan Daemen and Vincent Rijmen. The AES replaces the previous standard, the Data Encryption Standard (DES).</td>
</tr>
<tr>
<td><strong>algorithm (encryption)</strong></td>
<td>A set of mathematical rules (logic) used in the processes of encryption and decryption.</td>
</tr>
<tr>
<td><strong>algorithm (hash)</strong></td>
<td>A set of mathematical rules (logic) used in the processes of message digest creation and key/signature generation.</td>
</tr>
<tr>
<td><strong>anonymity</strong></td>
<td>Of unknown or undeclared origin or authorship, concealing an entity’s identification.</td>
</tr>
<tr>
<td><strong>ANSI (American National Standards Institute)</strong></td>
<td>Develops standards through various Accredited Standards Committees (ASC). The X9 committee focuses on security standards for the financial services industry.</td>
</tr>
<tr>
<td><strong>ASCII-armored text</strong></td>
<td>Binary information that has been encoded using a standard, printable, 7-bit ASCII character set, for convenience in transporting the information through communication systems. In the PGP program, ASCII armored text files are given the default filename extension, and they are encoded and decoded in the ASCII radix-64 format.</td>
</tr>
<tr>
<td><strong>asymmetric keys</strong></td>
<td>A separate but integrated user key-pair, comprised of one public key and one private key. Each key is one way, meaning that a key used to encrypt information can not be used to decrypt the same data.</td>
</tr>
<tr>
<td><strong>authentication</strong></td>
<td>The determination of the origin of encrypted information through the verification of someone’s digital signature or someone’s public key by checking its unique fingerprint.</td>
</tr>
<tr>
<td><strong>authorization certificate</strong></td>
<td>An electronic document to prove one’s access or privilege rights, also to prove one is who they say they are.</td>
</tr>
<tr>
<td><strong>authorization</strong></td>
<td>To convey official sanction, access or legal power to an entity.</td>
</tr>
</tbody>
</table>
backdoor  A cipher design fault, planned or accidental, which allows the apparent strength of the design to be easily avoided by those who know the trick. When the design background of a cipher is kept secret, a back door is often suspected.

blind signature  Ability to sign documents without knowledge of content, similar to a notary public.

block cipher  A symmetric cipher operating on blocks of plain text and cipher text, usually 64 bits.

bzip2  A freely available, patent free, high-quality data compression format. It runs on most 32- and 64-bit computer platforms.

CA (Certificate Authority)  A trusted third party (TTP) who creates certificates that consist of assertions on various attributes and binds them to an entity and/or to their public key.

CAST  A 64-bit block cipher using 64-bit key, six S-boxes with 8-bit input and 32-bit output, developed in Canada by Carlisle Adams and Stafford Tavares.

certificate (digital)  An electronic document attached to a public key by a trusted third party, which provides proof that the public key belongs to a legitimate owner and has not been compromised.

certification  Endorsement of information by a trusted entity.

certify  To sign another person's public key.

certifying authority  One or more trusted individuals who are assigned the responsibility of certifying the origin of keys and adding them to a common database.

ciphertext  Plaintext converted into a secretive format through the use of an encryption algorithm. An encryption key can unlock the original plaintext from ciphertext.

clear-signed message  Messages that are digitally signed but not encrypted.

clear text  Characters in a human readable form or bits in a machine-readable form (also called plain text).

command line interface  An interface where you type commands at a command prompt. PGP Command Line uses a command-line interface. DOS and UNIX use command-line interfaces. More recent operating systems, such as Windows and the Macintosh, use a graphical user interface.

common access cards (CACs)  Read-only smartcards used by the U.S. Department of Defense. CACs include two separate certificates, one for signing and one for encrypting. PGP Desktop filters the two certificates based on intended usage; for example, only the signing certificate is presented on the file signing dialog.

compression function  A compression function takes a fixed-sized input and returns a shorter, fixed sized output.
<table>
<thead>
<tr>
<th>Term</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>configuration file</strong></td>
<td>A file on your system read by PGP Command Line each time it is run; it allows PGP Command Line behavior to be changed via the settings in the file. Configuration file settings take precedence over environment variables, but can be overridden by the command line. Settings in the configuration file are internal PGP variables only; they have no effect on any other application on your system.</td>
</tr>
<tr>
<td><strong>conventional encryption</strong></td>
<td>Encryption that relies on a common passphrase instead of public-key cryptography. The file is encrypted using a session key, which encrypts using a passphrase you will be asked to choose.</td>
</tr>
<tr>
<td><strong>corporate signing key</strong></td>
<td>A public key that is designated by the security officer of a corporation as the system-wide key that all corporate users trust to sign other keys.</td>
</tr>
<tr>
<td><strong>cryptanalysis</strong></td>
<td>The art or science of transferring cipher text into plain text without initial knowledge of the key used to encrypt the plain text.</td>
</tr>
<tr>
<td><strong>cryptography</strong></td>
<td>The art and science of creating messages that have some combination of being private, signed, unmodified with non-repudiation.</td>
</tr>
<tr>
<td><strong>cryptosystem</strong></td>
<td>A system comprised of cryptographic algorithms, all possible plain text, cipher text, and keys.</td>
</tr>
<tr>
<td><strong>data integrity</strong></td>
<td>A method of ensuring information has not been altered by unauthorized or unknown means.</td>
</tr>
<tr>
<td><strong>decryption</strong></td>
<td>A method of unscrambling encrypted information so that it becomes legible again. The recipient’s private key is used for decryption.</td>
</tr>
<tr>
<td><strong>DES (Data Encryption Standard)</strong></td>
<td>A 64-bit block cipher, symmetric algorithm also known as Data Encryption Algorithm (DEA) by ANSI and DEA-1 by ISO. Widely used for over 20 years, adopted in 1976 as FIPS 46.</td>
</tr>
<tr>
<td><strong>dictionary attack</strong></td>
<td>A calculated brute force attack to reveal a password by trying obvious and logical combinations of words.</td>
</tr>
<tr>
<td><strong>Diffie-Hellman</strong></td>
<td>The first public key algorithm, invented in 1976, using discrete logarithms in a finite field.</td>
</tr>
<tr>
<td><strong>direct trust</strong></td>
<td>An establishment of peer-to-peer confidence.</td>
</tr>
<tr>
<td><strong>digital signature</strong></td>
<td>See signature.</td>
</tr>
<tr>
<td><strong>encryption</strong></td>
<td>A method of scrambling information to render it unreadable to anyone except the intended recipient, who must decrypt it to read it.</td>
</tr>
<tr>
<td><strong>entropy</strong></td>
<td>In cryptography, a measure of randomness. It specifically relates to the difficulty in determining a passphrase or key. The greater the amount of entropy, the more difficult something is to determine. For example, if you were to pick a number from zero to 9, you would have a one in 10 chance, which works out to certain amount of entropy. If you were to pick a letter in the English alphabet, from A to Z, then you would have a one in 26 chance, a far greater amount of entropy.</td>
</tr>
</tbody>
</table>
environment variables

PGP Command Line behavior can be changed using environment variables. Environment variables cannot be disabled; if they are present, they are implemented, unless overridden by settings in the configuration file or entered on the command line. To disable the effects of an environment variable, remove it.

fingerprint

A uniquely identifying string of numbers and characters used to authenticate public keys. This is the primary means for checking the authenticity of a key. See Key Fingerprint.

FIPS (Federal Information Processing Standard)

A U.S. government standard published by NIST.

firewall

A combination of hardware and software that protects the perimeter of the public/private network against certain attacks to ensure some degree of security.

hash function

A one way function that takes an input message of arbitrary length and produces a fixed length digest.

hierarchical trust

A graded series of entities that distribute trust in an organized fashion, commonly used in ANSI X.509 issuing certifying authorities.

HTTP (HyperText Transfer Protocol)

A common protocol used to transfer documents between servers or from a server to a client.

hexadecimal

Hexadecimal describes a base-16 number system. That is, it describes a numbering system containing 16 sequential numbers as base units (including 0) before adding a new position for the next number. (Note that we’re using “16” here as a decimal number to explain a number that would be “10” in hexadecimal.) The hexadecimal numbers are 0-9 and then use the letters A-F.

IDEA (International Data Encryption Standard)

A 64-bit block symmetric cipher using 128-bit keys based on mixing operations from different algebraic groups. Considered one of the strongest algorithms.

implicit trust

Implicit trust is reserved for keypairs located on your local keyring. If the private portion of a keypair is found on your keyring, PGP Desktop assumes that you are the owner of the keypair and that you implicitly trust yourself.

integrity

Assurance that data is not modified (by unauthorized persons) during storage or transmittal.

introducer

A person or organization who is allowed to vouch for the authenticity of someone’s public key. You designate an introducer by signing their public key.

ISO (International Organization for Standardization)

Responsible for a wide range of standards, like the OSI model and international relationship with ANSI on X.509.

key

A digital code used to encrypt and sign and decrypt and verify messages and files. Keys come in keypairs and are stored on keyrings.
<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>key escrow/recovery</strong></td>
<td>A practice where a user of a public key encryption system surrenders their private key to a third party thus permitting them to monitor encrypted communications.</td>
</tr>
<tr>
<td><strong>key exchange</strong></td>
<td>A scheme for two or more nodes to transfer a secret session key across an unsecured channel.</td>
</tr>
<tr>
<td><strong>key fingerprint</strong></td>
<td>A uniquely identifying string of numbers and characters used to authenticate public keys. For example, you can telephone the owner of a public key and have him or her read the fingerprint associated with their key so you can compare it with the fingerprint on your copy of their public key to see if they match. If the fingerprint does not match, then you know you have a bogus key.</td>
</tr>
<tr>
<td><strong>key ID</strong></td>
<td>A legible code that uniquely identifies a keypair. Two keypairs may have the same user ID, but they will have different Key IDs.</td>
</tr>
<tr>
<td><strong>key length</strong></td>
<td>The number of bits representing the key size; the longer the key, the stronger it is.</td>
</tr>
<tr>
<td><strong>key management</strong></td>
<td>The process and procedure for safely storing and distributing accurate cryptographic keys; the overall process of generating and distributing cryptographic key to authorized recipients in a secure manner.</td>
</tr>
<tr>
<td><strong>keypair</strong></td>
<td>A public key and its complimentary private key. In public-key cryptosystems, like the PGP program, each user has at least one keypair.</td>
</tr>
<tr>
<td><strong>keyring</strong></td>
<td>A set of keys. Each user has two types of keyrings: a private keyring and a public keyring.</td>
</tr>
<tr>
<td><strong>keyserver</strong></td>
<td>A database that holds keys. There are many public keyservers (that is, keyservers that allow anyone to post their public keys); the PGP Global Directory, at ldap://keyserver.pgp.com, for example, is a public keyserver. Many companies also host their own keyservers. You post your public key to a keyservers so that others can find your public key and send encrypted files and/or email to you.</td>
</tr>
<tr>
<td><strong>key splitting or “secret sharing”</strong></td>
<td>The process of dividing up a private key into multiple pieces, and share those pieces among a group of people. A designated number of those people must bring their shares of the key together to use the key.</td>
</tr>
<tr>
<td><strong>LDAP (Lightweight Directory Access Protocol)</strong></td>
<td>A simple protocol that supports access and search operations on directories containing information such as names, phone numbers, and addresses across otherwise incompatible systems over the Internet.</td>
</tr>
<tr>
<td><strong>MD5 (128 bits)</strong></td>
<td>A legacy hash algorithm provided only for backwards compatibility. Deprecated.</td>
</tr>
<tr>
<td><strong>message digest</strong></td>
<td>A compact “distillate” of your message or file checksum. It represents your message, such that if the message were altered in any way, a different message digest would be computed from it.</td>
</tr>
<tr>
<td><strong>meta-introducer</strong></td>
<td>A trusted introducer of trusted introducers.</td>
</tr>
</tbody>
</table>
MIME (Multipurpose Internet Mail Extensions)  A freely available set of specifications that offers a way to interchange text in languages with different character sets, and multimedia email among many different computer systems that use Internet mail standards.

non-repudiation  Preventing the denial of previous commitments or actions.

one-way hash  A function of a variable string to create a fixed length value representing the original pre-image, also called message digest, fingerprint, message integrity check (MIC).

passphrase  An easy-to-remember phrase used for better security than a single password. A passphrase can generally use non-alphanumeric characters such as *, +, or ~. Because passphrases are generally longer than passwords and use a wider variety of characters, they are more secure than passwords.

password  A sequence of characters or a word that a subject submits to a system for purposes of authentication, validation, or verification. Passwords are generally restricted to letters and numbers.

PGP/MIME  An IETF standard (RFC 2015) that provides privacy and authentication using the Multipurpose Internet Mail Extensions (MIME) security content types described in RFC1847, currently deployed in PGP 5.0 and later versions.

PKCS (Public Key Crypto Standards)  A set of de facto standards for public key cryptography developed in cooperation with an informal consortium (Apple, DEC, Lotus, Microsoft, MIT, RSA, and Sun) that includes algorithm-specific and algorithm-independent implementation standards. Specifications defining message syntax and other protocols controlled by RSA Data Security, Inc.

PKI (Public Key Infrastructure)  A widely available and accessible certificate system for obtaining an entity’s public key with some degree of certainty that you have the “right” key and that it has not been revoked.

plaintext  Normal, legible, un-encrypted, unsigned text.

private key  The secret portion of a keypair; used to sign and decrypt information. A user’s private key should be kept secret, known only to the user.

private keyring  A set of one or more private keys, all of which belong to the owner of the private keyring.

public key  One of two keys in a keypair-used to encrypt information and verify signatures. A user’s public key can be widely disseminated to colleagues or strangers. Knowing a person’s public key does not help anyone discover the corresponding private key.

public keyring  A set of public keys. Your public keyring includes your own public key(s).

public-key cryptography  Cryptography in which a public and private keypair is used, and no security is needed in the channel itself.
**random number**  
An important aspect to many cryptosystems, and a necessary element in generating a unique key(s) that are unpredictable to an adversary. True random numbers are usually derived from analog sources, and usually involve the use of special hardware.

**revocation**  
Retraction of certification or authorization.

**RFC (Request for Comment)**  
An IETF document, either FYI (For Your Information) RFC sub-series that are overviews and introductory or STD RFC sub-series that identify specify Internet standards. Each RFC has an RFC number by which it is indexed and by which it can be retrieved (www.ietf.org).

**Rijndael**  
A block cipher designed by Joan Daemen and Vincent Rijmen, chosen as the new Advanced Encryption Standard (AES). It is considered to be both faster and smaller than its competitors. The key size and block size can be 128-bit, 192-bit, or 256-bit in size and either can be increased by increments of 32 bits.

**RIPEMD-160 (160 bits)**  
An independent hash algorithm; it provides up to 80 bits of brute force resistance.

**RSA**  
Short for RSA Data Security, Inc.; or referring to the principals: Ron Rivest, Adi Shamir, and Len Adleman; or referring to the algorithm they invented. The RSA algorithm is used in public-key cryptography and is based on the fact that it is easy to multiply two large prime numbers together, but hard to factor them out of the product.

**script**  
A set of instructions written in a scripting language. PGP Command Line commands can be added to scripts so that PGP technology can be added to automated tasks.

**secure channel**  
A means of conveying information from one entity to another such that an adversary does not have the ability to reorder, delete, insert, or read (SSL, IPSec, whispering in someone’s ear).

**self-signed key**  
A public key that has been signed by the corresponding private key for proof of ownership.

**session key**  
The secret (symmetric) key used to encrypt each set of data on a transaction basis. A different session key is used for each communication session.

**SHA-1**  
A second-generation hash algorithm; it provides up to 80 bits of brute force resistance. Partially deprecated.

**SHA-2 (256 bits)**  
A third-generation hash algorithm; it provides up to 128 bits of brute force resistance.

**SHA-2 (384 bits)**  
A third-generation hash algorithm; it provides up to 192 bits of brute force resistance.

**SHA-2 (512 bits)**  
A third-generation hash algorithm; it provides up to 256 bits of brute force resistance.

**sign**  
To apply a signature.
<p>| <strong>signature</strong> | A digital code created with a private key. Signatures allow authentication of information by the process of signature verification. When you sign a message or file, the PGP program uses your private key to create a digital code that is unique to both the contents of the message and your private key. Anyone can use your public key to verify your signature. |
| <strong>S/MIME (Secure Multipurpose Mail Extension)</strong> | A proposed standard developed by Deming software and RSA Data Security for encrypting and/or authenticating MIME data. S/MIME defines a format for the MIME data, the algorithms that must be used for interoperability (RSA, RC2, SHA-1), and the additional operational concerns such as ANSI X.509 certificates and transport over the Internet. |
| <strong>SSL (Secure Socket Layer)</strong> | Developed by Netscape to provide security and privacy over the Internet. Supports server and client authentication and maintains the security and integrity of the transmission channel. Operates at the transport layer and mimics the “sockets library,” allowing it to be application independent. Encrypts the entire communication channel and does not support digital signatures at the message level. |
| <strong>symmetric algorithm</strong> | Also known as conventional, secret key, and single key algorithms; the encryption and decryption key are either the same or can be calculated from one another. Two sub-categories exist: Block and Stream. |
| <strong>subkey</strong> | A subkey is a Diffie-Hellman encryption key that is added as a subset to your master key. Once a subkey is created, you can expire or revoke it without affecting your master key or the signatures collected on it. |
| <strong>tar file</strong> | A general purpose archive format (originally developed on UNIX, but generally readable on Windows). Tar is now commonly used for packaging files together into a single archive for distribution, frequently via the Internet. The resulting archive is called a “tar file.” PGP Command Line uses the tar file format as the format for PGP archives. |
| <strong>text</strong> | Standard, printable, 7-bit ASCII text. |
| <strong>timestamping</strong> | Recording the time of creation or existence of information. |
| <strong>TLS (Transport Layer Security)</strong> | An IETF draft, version 1 is based on the Secure Sockets Layer (SSL) version 3.0 protocol, and provides communications privacy over the Internet. |
| <strong>Triple DES</strong> | An encryption configuration in which the DES algorithm is used three times with three different keys. |
| <strong>trusted</strong> | A public key is said to be trusted by you if it has been validated by you or by someone you have designated as an introducer. |
| <strong>trusted introducer</strong> | Someone whom you trust to provide you with keys that are valid. When a trusted introducer signs another person’s key, you trust that the person’s key is valid, and you do not need to verify the key before using it. |
| <strong>Twofish</strong> | A 256-bit block cipher, symmetric algorithm. Twofish was one of five algorithms that the U.S. National Institute of Standards and Technology (NIST) considered for the Advanced Encryption Standard (AES). |</p>
<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>user ID</td>
<td>A text phrase that identifies a keypair. For example, one common format for a user ID is the owner’s name and email address. The user ID helps users (both the owner and colleagues) identify the owner of the keypair.</td>
</tr>
<tr>
<td>validity</td>
<td>Indicates the level of confidence that the key actually belongs to the alleged owner.</td>
</tr>
<tr>
<td>verification</td>
<td>The act of comparing a signature created with a private key to its public key. Verification proves that the information was actually sent by the signer, and that the message has not been subsequently altered by anyone else.</td>
</tr>
<tr>
<td>web of trust</td>
<td>A distributed trust model used by PGP technology to validate the ownership of a public key where the level of trust is cumulative, based on the individuals’ knowledge of the introducers.</td>
</tr>
<tr>
<td>X.509</td>
<td>An ITU-T digital certificate that is an internationally recognized electronic document used to prove identity and public key ownership over a communication network. It contains the issuer’s name, the user’s identifying information, and the issuer’s digital signature, as well as other possible extensions.</td>
</tr>
<tr>
<td>zip</td>
<td>Zip is a compression and file packaging/archive utility. Zip is useful for packaging a set of files for distribution, for archiving files, and for saving disk space by temporarily compressing unused files or directories. Zip puts one or more compressed files into a single zip archive, along with information about the files. An entire directory structure can be packed into a zip archive with a single command. Zip has one compression method (deflation) and can also store files without compression. Zip automatically chooses the better of the two for each file.</td>
</tr>
<tr>
<td>zlib</td>
<td>Zlib is a free, general-purpose, legally unencumbered, lossless data-compression library for use on virtually any computer hardware and operating system. The zlib data format is itself portable across platforms.</td>
</tr>
</tbody>
</table>
Index

Numerics

--3des 150

A

--add-adk 89
--additional-recipient 180
--add-photoid 90
--add-preferred-cipher 90
--add-preferred-compression-algorithm 91
--add-preferred-email-encoding 91
--add-revoker 92
--add-userid 93
ADK 8
--adk 180
--aes128 150
AIX
  change home directory 14
  how to install 13
--always-trust 140
--answer 181
--archive 140
Arguments
  about 34
arguments
  Boolean 34
  enumerations 35
  file descriptors 35
  integers 34
  lists 35
  no parent 36
  strings 35
--armor (-a) 54
--auth-passphrase 170
--auth-passphrase-fd 184, 185
--auth-passphrase-fd8 184, 185
--auth-username 170
--auto-import-keys 160

B

--banner 141

--biometric 141
--blowfish 151
Boolean arguments 34
--buffered-stdio 141
--bzip2 151

C

--cache-passphrase 93
--cast5 151
certificate signature request (CSR) 99
--change-passphrase 95
--check-sigs 137
--check-userids 137
--cipher 160
--city 170
--clearsign 56
command line
  environment variables 41
command line interface 7
  flags and arguments 33
  overview 31
--comment 170
--compress 142
--compression-algorithm 161
--compression-level 161
Configuration file 36
--create-keyrings 47, 133
creating
  keypair 46
  SDA 147
--creation-date 170
--creation-days 152

D

--decrypt 57
decrypt
  eyes-only 142
decrypting
  defined 53
--default-key 171
Department of Defense 5220.22-M 136
--detached (-b) 59
--disable 96
distributing public key 48
distributing your public key 48

E
--email-encoding 162
--enable 96
encrypt
eyes-only 142
--encrypt (-e) 61
encrypting
defined 53
--encryption-bits 150
--encrypt-to-self 142
--enforce-adk 162
enumeration arguments 35
Environment variables 41
environment variables 7, 41
PGP_HOME_DIR 41
PGP_LOCAL_MODE 41
PGP_NEW_PASSPHRASE 41
PGP_NO_BANNER 41
PGP_PASSPHRASE 41
PGP_SYMMETRIC_PASSPHRASE 41
--expiration-date 171
--expiration-days 152
--export 49, 97
export formats 98
export public key to file 49
--export-format 163
--export-passphrase 171
--export-passphrase-fd 184
--export-passphrase-fd8 184
--export-photoid 99
--export-session-key 64
--eyes-only 142

F
--fast-key-gen 143

Fedora Core
change home directory 19
how to install 18
uninstalling 19
file descriptor arguments 35
File redirection 42
finding a public key on a keyserver 50
--fingerprint 52, 72, 73
--fips-mode 143
Flags
about 33
--force (-f) 143

G
--gen-key 46, 47, 100
--gen-revocation 102
--gen-subkey 103
getting public keys 50

H
--halt-on-error 144
--hash 163
--help (-h) 134
--home-dir 172
HP-UX
change home directory 16
how to install 15

I
--idea 152
If 51
--import 104
import public key from keyserver 51
--import-format 164
importing a public key from a keyserver 51
--index 153
--input (-i) 180
--input-cleanup 165
integer arguments 34

J
--join-key 104
--join-key, command output 106
K
key types 101
--key-flag 165
keypair
  creating 46
  --key-recon-recv 111
  --key-recon-recv-questions 110
  --key-recon-send 109
  --keyring-cache 144
  --keyring-cache-timeout 153
Keyserver
  configuration file settings 40
  --keyserver 181
  --keyserver-disable 81
  --keyserver-recv 51, 82
  --keyserver-remove 83
  --keyserver-search 50, 84
  --keyserver-send 48, 85
  --keyserver-timeout 153
  --keyserver-update 86
  --key-type 166

L
--license-email 173
--license-name 173
--license-number 173
--license-organization 173
--license-recover 144
licensing
  license authorization 28
  license number 27
  license recovery 26
  overview 25
  re-licensing 29
  through proxy server 30
Linux
  change home directory 19
  how to install 18
  uninstalling 19
list arguments 35
  --list-archive 65
  --list-key-details 75
  --list-keys 51
  --list-keys (-l) 76
  --list-sig-details 78
  --list-sigs 78
  --list-users 79
  --local-mode 145
  --local-user (-u) 172

M
Mac OS X
  change home directory 17
  how to install 16
  --manual-import-key-pairs 166
  --manual-import-keys 167
  --marginal-as-valid 145
  --md5 154

N
  --new-passphrase 173
  --new-passphrase-fd 184
  --new-passphrase-fd8 184
  no parent arguments 36

O
  --organization 174
  --output (-o) 174
  --overwrite 167

P
  --partitioned 155
  --passphrase 174
  --passphrase-cache 146
  --passphrase-cache-timeout 154
  --passphrase-fd 184
  --passphrase-fd8 184
  --pass-through 145
PGP Command Line
  defined 7
PGP_HOME_DIR environment variables 41
PGP_LOCAL_MODE environment variable 41
PGP_NEW_PASSPHRASE environment variables 41
PGP_NO_BANNER environment variables 41
PGP_PASSPHRASE environment variables 41
PGP_SYMMETRIC_PASSPHRASE environment variables 41
--pgpmime 155
--photo 146

platforms
    supported 9
post public key to keyserver 48
--preferred-keyserver 175
--private-keyring 175
protecting private key 47
--proxy-password 176
--proxy-server 176
--proxy-username 176
public key
    distributing 48
    finding on keyserver 50
    importing 51
public keys
    verifying 52
--public-keyring 176
--purge-all-caches 134
--purge-keyring-cache 135
--purge-passphrase-cache 135

Q
--question 181
--quiet (-q) 146

R
--random-seed 177
--recipient (-r) 182
--recursive 146
Red Hat Enterprise Linux
    change home directory 19
    how to install 18
    uninstalling 19
--regular-expression 177
--remove 112
--remove-adk 112
--remove-all-adks 112
--remove-all-revokers 113
--remove-expiration-date 114
--remove-key-pair 114
--remove-photoid 114
--remove-preferred-cipher 115
--remove-preferred-compression-algorithm 115
--remove-preferred-email-encoding 116
--remove-preferred-keyserver 117
--remove-revoker 117
--remove-sig 118
--remove-subkey 118
--remove-userid 119
--reverse-sort 146
--revoke 119
--revoker 182
--revoke-sig 120
--revoking-subkey 120
--ripemd160 155
--root-path 177

S
scripts 7
--sda 147
Self-Decrypting Archive (SDA) 147
--set-expiration-date 121
--set-preferred-ciphers 122, 123, 124
--set-preferred-compression-algorithms 123
set-preferred-hashes 124
--set-preferred-keyserver 125
--set-primary-userid 125
--set-trust 126
--sha 156
--sha256 156
--sha384 156
--sha512 156
--share 183
--share-server 178
--sign (-s) 66
signature types 127
signing
    defined 53
--signing-bits 157
--sign-key 126
--sign-userid 127
--sig-type 167
--skep 147
--skep-timeout 157
Solaris
    change home directory 21
    how to install 20
--sort-order 167
--speed-test 135
--split-key 128
--split-key, preview mode 131
standard error 42
standard input 42
standard output 42
--state 178
stderr 42
stdin 42
stdout 42
string arguments 35
supported platforms 9
--symmetric (-c) 68
--symmetric-passphrase 178
--symmetric-passphrase-fd 185
--symmetric-passphrase-fd8 185
system requirements 10

Windows
   change home directory 22
   how to install 22
--wipe 136
wipe
   Department of Defense 5220.22-M 136
--wipe-input-passes 158
--wipe-overwrite-passes 159
--wipe-passes 158
--wipe-temp-passes 159

X
--xml 148

Z
--zip 159
--zlib 159

T
--tar-cache-cleanup 168
--target-platform 169
--temp-cleanup 169
--temp-dir 179
--text (-t) 147
--threshold 157
--trust 169
--trust-depth 158
--twofish 158

U
--user 172

V
--verbose (-v) 148
--verify 69
verifying
   defined 53
verifying public keys 52
--version 136

W
--warn-adk 148