

PKWARE[®]

*Getting Started with
PKWARE[®] Key Maker[™]
for IBM i*

PKWARE Inc.

Contents

Welcome to PKWARE Key Maker	2
Introduction to OpenPGP	2
Preparing To Install Key Maker On IBM i	3
Environment Settings.....	3
Testing your Java Environment	3
Strong Cryptography in Java	4
Installing Key Maker	4
Generating OpenPGP Keys and Self-Signed X.509 Certificates.....	5
generate	5
Options.....	5
Sample commands	6
Converting Key Formats.....	6
convert	6
Options.....	7
Sample commands	7
Signing OpenPGP Keys	8
sign.....	8
Options.....	8
Sample command	8
Copying OpenPGP Keys From An Existing Keyring	9
copy.....	9
Options.....	9
Sample commands	9
Getting Information About Keys In A Ring	10
list.....	10
Options.....	10
Identify info that is displayed for an OpenPGP key	10
Sample commands	11
Working with OpenPGP Key Servers	11
keyserver-send	11
Options.....	11
Sample command	11
keyserver-search	11
Options.....	12
Sample commands	12
Making Changes to OpenPGP Keys	12
edit	12
Options.....	12
Sample commands	13
Deleting Keys From A Ring	13
delete	13
Options.....	14
Sample commands	14
Displaying Help Information.....	14
help	14

Displaying Key Maker Version Information	14
version.....	14
Return Codes	15
User Help and Contact Information	15

Welcome to PKWARE Key Maker

Organizations that rely on files encrypted with OpenPGP need a fast, reliable way to encrypt and decrypt OpenPGP files. They also need a method of ensuring the people who handle OpenPGP files can easily create and open these files. OpenPGP users identify themselves, and develop trust through public and private keys.

PKWARE provides SecureZIP to encrypt and decrypt strongly-encrypted files using passphrases, X.509 certificates and OpenPGP keys. SecureZIP Server eBusiness Edition includes PKWARE Key Maker to allow you to create and manage OpenPGP keys. This guide will walk you through the basics of using PKWARE Key Maker.

For more information about SecureZIP, see <http://www.pkware.com/software/securezip/>

Use of PKWARE Key Maker is covered under the terms and conditions of your SecureZIP license agreement.

Introduction to OpenPGP

Some organizations use encryption tools based on the OpenPGP standard, rather than X.509. OpenPGP uses the same basic Public Key Infrastructure principles for exchanging encrypted files, but uses a decentralized “Web of Trust” method of authenticating signatures.

SecureZIP extracts and decrypts files that comply with the OpenPGP specification defined by the Internet Engineering Task Force RFC 4880. SecureZIP can also create OpenPGP-compliant files and sign files with OpenPGP keys.

OpenPGP keys are typically created by individuals, and authenticated by other individuals. In the real world, you have friends who can vouch that you are who you say you are. If you walk into a room full of strangers, your friend can introduce you to the people he knows. Since you trust that your friend is correctly identifying his friends and acquaintances, your trust extends to his friends too.

When you translate the above experience to the electronic, OpenPGP world, it works this way: You create an OpenPGP key to identify yourself. When a friend comes to visit, display the key. The friend can now sign your key (often called “key signing”) and certify that this key represents you. Now everyone who trusts the person who signed your key can also trust that your key is authentic. A Web of Trust is developed as more people authenticate each key. Everyone in the Web of Trust can also exchange messages in the OpenPGP format.

In order to use OpenPGP keys with SecureZIP, they must first be generated and stored in an OpenPGP compliant key repository. Typically, this repository is a keyring file. OpenPGP public keys are stored in a public keyring file. While not required by the OpenPGP standard, or by PKWARE Key Maker, public keyring files usually have a file extension of **.pkr**. OpenPGP secret keys are stored in a secret keyring file. Secret keyring files usually have a file extension of **.skr**. Other file extensions may be used for keyring files. PKWARE recommends using the **.pkr** and **.skr** file extensions respectively when referencing public

and secret keyring files, but other keyring file extensions can be used with this program. The PKWARE Key Maker program provides a means of creating OpenPGP keys and keyring files for use with SecureZIP.

Preparing To Install Key Maker On IBM i

PKWARE Key Maker runs on Java™ v6 and later. Before installing Key Maker, you must have a Java Virtual Machine (JVM) installed and configured for use. Java commands are commonly run using the QShell Interpreter. QShell provides a command-line environment for Java that is compatible with the POSIX 1003.2 X/Open Command, and Utilities Issue 4, Version 2 standard and is also similar to commands used under AIX. Make sure you have access to this command-line environment.

Determine if you have QShell installed by typing GO LICPGM on the operating system command line and then pressing the Enter key. Select Option 10 and press Enter again. Press F11 from the column views and check that the Installed Status is COMPATIBLE. Visit <http://www-01.ibm.com/support/docview.wss?uid=nas16e6a58b238b2c3788625722d00544d5c> for additional information on installing and using QShell.

To start QShell, enter the **STRQSH** command.

Determine if you have the IBM Developer Kit for Java installed by typing GO LICPGM on the operating system command line and then pressing the Enter key. Select Option 10 and press Enter again. If the licensed program 5761-JV1 *BASE is not installed, visit <http://publib.boulder.ibm.com/infocenter/iserics/v6r1m0/index.jsp?topic=/rzaha/multjdk.htm> for information on installing the IBM Developer Kit for Java.

Environment Settings

Java relies on several environment settings that identify locations for Java modules used by the JVM. Values for these settings can be configured using ADDENVVAR.

JAVA_HOME - The <JAVA_HOME> value sets the absolute path to the installation directory of your Java environment.

ADDENVVAR ENVVAR(JAVA_HOME) VALUE('<path_to_your_java_environment>')

CLASSPATH – The CLASSPATH setting is used by the JVM to locate Java classes used during runtime operation of Java commands.

Multiple versions of Java may exist at the same time on the IBM i platform. Make sure to set your JAVA_HOME environment value to the appropriate path for the version you will use. Common paths for JAVA_HOME on the IBM i include:

/QIBM/ProdData/Java400/<vers> where <vers> identifies your Java version such as jdk6.

/QOpenSys/QIBM/ProdData/JavaVM/<vers>/<bits> where <vers> identifies your Java version such as jdk60 and <bits> identifies either 32bit or 64 bit.

Testing your Java Environment

After you have completed the setup of your Java environment you can test that it is operational using the sample command below. This command will report the version of Java you are configured to use.

Type `java -version`

To run Java with Key Maker complete the steps for installing Key Maker and then enter the `java` command with parameters for using Key Maker as illustrated in this document from within your active QShell.

If you are unable to access or use your Java environment, contact your System Administrator, or your IBM Support resource for assistance.

Strong Cryptography in Java

The Java Virtual Machine environment for IBM i ships with limited support for strong cryptography using symmetric encryption algorithms with key lengths exceeding 64 bits, due to import control restrictions for some countries. See the US Commerce Department encryption controls site for more information: <http://www.bis.doc.gov/index.php/policy-guidance/encryption>.

Residents of eligible countries (including the United States) can obtain the IBM Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy File through the IBM SDK Policy Files website: <https://www14.software.ibm.com/webapp/iwm/web/preLogin.do?source=jcesdk>. IBM produces the unrestricted JCE files to permit eligible users to use strong encryption in the Java environment.

When the download is complete, replace these files with the unlimited JCE files as outlined in the [IBM SDK Policy Files site](http://www-03.ibm.com/systems/z/os/zos/tools/java/products/javasec.html#jsechw) ((<http://www-03.ibm.com/systems/z/os/zos/tools/java/products/javasec.html#jsechw>)):

<JAVA_HOME>/lib/security/local_policy.jar

<JAVA_HOME>/lib/security/US_export_policy.jar

Installing Key Maker

When you have prepared your Java environment in UNIX System Services, installing PKWARE Key Maker is simply a matter of copying two JAR files provided in the Key Maker install package to your system.

- PKKeyM.jar contains all the Key Maker program files. This file can be placed anywhere. You must include the full IFS path, such as `/temp/mystuff/PKKeyM.jar`, in your Key Maker commands.
- Key Maker uses the Bouncy Castle API¹ to support various encryption algorithms used by OpenPGP unavailable with other JCE providers, including CAST5. Place the `bcprov-jdk15on-150.jar` in the JRE `/lib/ext` directory. This extension folder is part of the IBM Java architecture and is recommended by IBM to extend core Java capabilities.

After installing the key generation application, confirm that `bcprov*.jar` is installed in the JRE `/lib/ext` directory. Note that if you have multiple side by side versions of Java installed, copy the cryptographic provider JAR file to each instance of `lib/ext` only in

¹ Portions of this software include technology from the Legion of the Bouncy Castle provided under the following terms: THE [Bouncy Castle] SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

the Java installation(s) where you will use Key Maker.

The cryptographic API supports Java 1.6 and above.

Additional files included with Key Maker are:

- This document, *Getting Started with PKWARE Key Maker*, KeyMaker_GettingStarted_IBMi.pdf
- PKWARE, Inc., *Third-Party License Summary for PKWARE Key Maker*, ThirdPartyLicense.pdf

Generating OpenPGP Keys and Self-Signed X.509 Certificates

Key Maker is designed to operate from a command interface and can be easily used with shell scripts, batch files.

generate

This command creates a new OpenPGP public/private key pair. If you specify an existing OpenPGP keyring, Key Maker will attempt to add the new key to that ring.

You can also create a self-signed X.509 certificate with this command.

Options

Option	Description
-armor	Use ASCII armor for a new OpenPGP output file.
-earmor	Use this command to use EBCDIC ASCII Armor. Possible values are 0x0a, 0x0d, 0x15, and 0x25.
-email	Email address for X.509 certificate creation. This email address will be written as a subject alternative name extension within the certificate.
-expire	Expiration period (in days) for the key or certificate being created. If this option is not specified than the default expiration will be used. The default option is dependent upon key type. Unless you include the expire option, OpenPGP keys will not expire, and X.509 certificates will expire one year from creation.
-hash	Hash algorithm to use when signing X.509 certificates
-keySize	RSA key length when generating keys. For RSA, possible values are 1024, 2048 (default) and 4096. DSA will use the same values and defaults as RSA but they will apply to the El Gamal encryption key – not the DSA signing key. The DSA signing key will be fixed at 1024. You can specify different lengths for signing and encryption keys in OpenPGP by using (for example) -keySize 1024/2048 . The first value represents the signing key size, the second is used for encryption. If you only use one value, Key Maker will use that size for both key types.
-keyType <type>	Determines the type of key to create. Possible values for OpenPGP are RSA (default) and DSA. Only value for X.509 is RSA.
-outPublicPGP <file>	Output file name for OpenPGP public key, used as primary output for command to specify where the public key will be placed. This is a required option.

Option	Description
-outSecretPGP <file>	Output file name for OpenPGP private (secret) key, used as primary output for command to specify where the private key will be placed. This is a required option.
-outPKCS7 <file>	Output file name for one or more X.509 certificates. Creates a PKCS#7 certificates-only message.
-outPKCS12 <file>	Output file name for an X.509 certificate and its private key.
-outCert <file>	Output file name for an X.509 certificate (without private key) in DER format.
-outPass <pw>	Output passphrase, used to protect generated private key. This is a required option.
-singleKey	Output a single OpenPGP signing and encryption key.
-subject	Subject to be used in X.509 certificate creation. The subject string must contain a valid X.500 name using known X.500 tokens for the various values. This option is required when generating an X.509 certificate or converting from OpenPGP to X.509.
-userid	OpenPGP userid to be used in OpenPGP key creation or for locating an OpenPGP key in a keyring. This is a required option. This value can contain a name, email address and comment; for example: Tom <tom@example.com>

Sample commands

In this and the other samples in this guide, replace *<path>* with the full path to the PKKeyM.jar file, such as */temp/mystuff/PKKeyM.jar*.

Create new OpenPGP RSA keyring

Create new OpenPGP RSA public/private keyring using the default 2048 key size and no expiration date.

```
java -jar <path> generate -outPublicPGP test.pkr -outSecretPGP test.skr -outPass password -userid "User <user@example.com>"
```

Create new OpenPGP DSA keyring

Create new OpenPGP DSA public/private keyring using no-default values.

```
java -jar <path> generate -KeyType DSA -KeySize 2048 -outPublicPGP test.pkr -outSecretPGP test.skr -outPass password -userid "User <user@example.com>"
```

Create new X.509 Certificate

Generate a new X.509 Certificate with 4096 key for John Doe which will expire in 1 year.

```
java -jar <path> generate -keySize 4096 -keyType RSA -subject "CN=John Doe" -email john.doc@example.com -outPKCS12 cert.p12 -outPass password
```

Converting Key Formats

convert

Converts keys between OpenPGP and X.509 formats. You can convert these types of keys:

- PKCS#12 to OpenPGP Public Ring

- PKCS#12 to OpenPGP Public and OpenPGP Secret Ring
- OpenPGP Secret Key to PKCS#12, PKCS#7 and/or X.509 certificate
- OpenPGP Public Key to PKCS#7 and/or X.509 certificate

Options

Option	Description
-armor	Use ASCII armor for a new OpenPGP output file.
-earmor	Use this command to use EBCDIC ASCII Armor. Possible values are 0x0a, 0x0d, 0x15, and 0x25.
-expire	Expiration period (in days) for the key being created. If this option is not specified then the default expiration will be used. The default option is dependent upon key type. Unless you include the expire option, OpenPGP keys will not expire, and X.509 certificates will expire one year from creation.
-in <file>	Input file name. This is a required option.
-inPass	Input passphrase. This option can be used when converting a private OpenPGP key to an X.509 certificate or converting from an X.509 certificate and private key to OpenPGP. This is a required option.
-inType	Indicates the input type to expect. Possible values: cert, pgp, pkcs7, and pkcs12. This option is required when one -in is used.
-keyID	Used to identify a particular OpenPGP key by its unique key ID. This option can be specified multiple times on the command line. You can use the complete short or long version of the key ID with this option.
-outPublicPGP <file>	Output file name for OpenPGP public keyring, used as primary output for command to specify where the public key will be placed.
-outSecretPGP <file>	Output file name for OpenPGP private (secret) keyring, used as primary output for command to specify where the private key will be placed.
-outPKCS7 <file>	Output file name for one or more X.509 certificates. Creates a PKCS#7 certificates-only message.
-outPKCS12 <file>	Output file name for an X.509 certificate and its private key.
-outCert <file>	Output file name for an X.509 certificate (without private key) in DER format.
-outPass <pass>	Output passphrase, used to protect generated private key. This is a required option when the target includes a private key.
-userid	OpenPGP userid to be used in OpenPGP key creation or for locating an OpenPGP key in a keyring. This is a required option. This value can contain a name, email address and comment; for example: Tom <tom@example.com>. This option is required when converting an X.509 certificate to OpenPGP.

Sample commands

Convert PKCS#12 to OpenPGP Public/Private

Convert an X.509 certificate and private key in PKCS#12 file to an OpenPGP public/private key ring.

```
java -jar <path> convert -inType pkcs12 -in test.p12 -inPass password -outPublicPGP test.pkr
-outSecretPGP test.skr -outPass changeit -userid "User <user@example.com>"
```


Convert PKCS#12 to OpenPGP Public Only

Convert an X.509 certificate with private key in a PKCS#12 file to an OpenPGP public key ring.

```
java -jar <path> convert -inType pkcs12 -in test.p12 -inPass password -outPublicPGP test.pkr -userid "User <user@example.com>"
```

Convert OpenPGP Secret to PKCS#12

Convert an OpenPGP Secret Key identified by the key ID AFEAB87F932ACF4A to an X.509 format.

```
java -jar <path> convert -inType pgp -in test.skr -inPass password -keyID AFEAB87F932ACF4A -outPKCS12 test.p12 -outPass password -subject "CN=User" -email user@example.com
```

Convert a OpenPGP Secret Key for the userid "User" to an X.509 certificate.

```
java -jar <path> convert -inType pgp -in test.skr -inPass password -outPKCS12 test.p12 -outPass password -userid "User <user@example.com>" -subject "CN=User" -email user@example.com
```

Signing OpenPGP Keys

sign

Establish trust relationships with other OpenPGP keys with the *sign* command.

Options

Option	Description
-inPass	Input passphrase. Use this option with -signwith to unlock the OpenPGP secret key you are signing with.
-inPublicPGP <file>	(Optional) Specify the public keyring location for the key(s) you want to sign.
-inSecretPGP <file>	(Optional) Specify the private keyring location for the key(s) you want to sign.
-keyID	Identify a particular OpenPGP key to sign by its unique key ID. You must identify a key by its keyID or userid. You can use the complete short or long version of the key ID with this option. Sign multiple keys by specifying this option multiple times on the command line.
-signwith	Identify the OpenPGP secret key to sign the key with.
-userid	Identify a particular OpenPGP userid to sign. You must identify a key by its keyID or userid. The userid value can contain a name, email address and comment; for example: Tom <tom@example.com>.
-multi	Allow multiple keys to be selected.
-verbose	Enable verbose output.

Sample command

Sign OpenPGP key 0x1234568 with OpenPGP key 0x87654321 where both are in the default key rings.

```
java -jar <path> sign -signWith 0x87654321 -inPass <passphrase> -keyid 0x12345678
```

Copying OpenPGP Keys From An Existing Keyring

copy

Copies one or more keys from one OpenPGP keyring to another. Allows copying of a public key(s) or a keyring to another public keyring, or copying of a secret key(s) or keyring to another secret keyring. You can use this command to import and export keys and keyrings from one location to another as well.

Options

<i>Option</i>	<i>Description</i>
-armor	Use ASCII armor for OpenPGP output file when the target keyring does not yet exist.
-earmor	Use this command to use EBCDIC ASCII Armor. Possible values are 0x0a, 0x0d, 0x15, and 0x25.
-in <file>	Input file name. This is a required option.
-keyID	Used to identify a particular OpenPGP key by its unique key ID. This option can be specified multiple times on the command line. Either -keyID or -userid is required when selecting specific keys. You can use the complete short or long version of the key ID with this option.
-outPublicPGP <file>	Output file name for OpenPGP public key, used as primary output for command to specify where the public key will be placed. This is a required option.
-outSecretPGP <file>	Output file name for OpenPGP private (secret) key, used as primary output for command to specify where the private key will be placed. This is a required option.
-userid <regex>	OpenPGP userid to be used to locate an OpenPGP key in a keyring. This value can contain a name, email address and comment; for example: Tom <tom@example.com>. The <regex> parameter is a Java Regular Expression. Java regular expressions are virtually identical to Perl. Either -keyID or -userid is required when selecting specific keys.

Sample commands

Copy OpenPGP Public/Secret Keys

Copy a single OpenPGP public key from one OpenPGP public keyring to another. Only the OpenPGP key that matches the specified key ID will be copied (multiple **-keyID** values can be specified).

```
java -jar <path> copy -inType pgp -in source.pkr -outPublicPGP target.pkr -keyid 085F4A9D5AA93E4D
```

Export all of the OpenPGP public keys from the Source public keyring, and import those keys to the Target public keyring.

```
java -jar <path> copy -inType pgp -in source.pkr -outPublicPGP target.pkr
```

Copy all users in the example.com domain:

```
java -jar <path> copy -inType pgp -in source.pkr -outPublicPGP target.pkr -userid ".*<.*@example.com>"
```

Getting Information About Keys In A Ring

list

Display information on keys within a key file

Options

Option	Description
-expire <N>	Expiration period (in days) for the key or certificate. Include this option if you want to learn what keys will expire within the specified number <N> of days from today.
-in <file>	Input file name
-inType	Indicates the input type to expect. Values include PGP, CERT, PKCS7 and PKCS12 (with -inPass).
-keyID	Used to identify a particular OpenPGP key by its unique key ID. This option can be specified multiple times on the command line. Either <i>-keyID</i> or <i>-userid</i> is required when selecting specific keys. You can use the complete short or long version of the key ID with this option.
-sort	Sort the list based on one of the default columns: Values include: none, create, keyid, keysize, expire, userid
-userid	OpenPGP userid to be used to locate an OpenPGP key in a keyring. This value can contain a name, email address and comment; for example: Tom <tom@example.com>. The <regex> parameter is a Java Regular Expression. Java regular expressions are virtually identical to Perl. Either <i>-keyID</i> or <i>-userid</i> is required when selecting specific keys.
-verbose	Enable verbose output.
-withSig	Include OpenPGP signatures in output
-withUserIDs	Include OpenPGP userid in output

Identify info that is displayed for an OpenPGP key

- *User ID*
- *Key ID*
- *Key Type*
- *Key Size*
- *Date Created*
- *Expiration*
- *Thumbprint*
- *Signature, including signature date, end date, keyid of creator, and name*

Sample commands

List OpenPGP Public

List the OpenPGP keys within an OpenPGP public keyring.

```
java -jar <path> list -inType pgp -in test.pkr
```

List the PGP keys within a PGP public key ring that will expire within the next 120 days.

```
java -jar <path> list -inType pgp -in test.pkr -expire 120
```

List OpenPGP Secret

List the OpenPGP keys within an OpenPGP secret keyring.

```
java -jar <path> list -inType pgp test.skr
```

Working with OpenPGP Key Servers

You can use Key Maker to send a key to a public key server for others to use, or search for a public key on a specified key server.

keyserver-send

Allow others to access your public OpenPGP key by placing the key on a public key server.

Options

Option	Description
-keyServer	URL to OpenPGP key server
-inPublicPGP <file>	Specify the public keyring location for the key(s) you want to edit.
-keyID	Used to identify a particular OpenPGP key by its unique key ID. This option can be specified multiple times on the command line. Either <i>-keyID</i> or <i>-userid</i> is required when selecting specific keys. You can use the complete short or long version of the key ID with this option.
-multi	Allow multiple keys to be selected.
-userid <regex>	OpenPGP userid to be used to locate an OpenPGP key in a keyring. This value can contain a name, email address and comment; for example: Tom <tom@example.com>. The <regex> parameter is a Java Regular Expression. Java regular expressions are virtually identical to Perl. Either <i>-keyID</i> or <i>-userid</i> is required when selecting specific keys.

Sample command

Send an OpenPGP key to a key server.

```
java -jar <path> keyserver-send -keyServer http://sks.example.com -userid "Bob Smith"
```

keyserver-search

Locate an OpenPGP key on a key server with this command.

Options

Option	Description
-keyServer	URL to OpenPGP key server
-keyID	Used to identify a particular OpenPGP key by its unique key ID. This option can be specified multiple times on the command line. Either <i>-keyID</i> or <i>-userid</i> is required when selecting specific keys. You can use the complete short or long version of the key ID with this option.
-userid <regex>	OpenPGP userid to be used to locate an OpenPGP key in a keyring. This value can contain a name, email address and comment; for example: Tom <tom@example.com>. The <regex> parameter is a Java Regular Expression. Java regular expressions are virtually identical to Perl. Either <i>-keyID</i> or <i>-userid</i> is required when selecting specific keys.
-verbose	Enable verbose output

Sample commands

Search for an OpenPGP key that matches the specified email address.

```
java -jar <path> keyserver-search -keyServer http://sks.example.com -userid bob@example.com
```

Search for an OpenPGP key that matches the specified key ids.

```
java -jar <path> keyserver-search -keyServer http://sks.example.com -keyid 0x12345678
```

Making Changes to OpenPGP Keys

edit

This command allows you to perform several types of actions on OpenPGP keys, including:

- **Disable**: Temporarily stop using a particular OpenPGP key to encrypt an archive
- **Enable**: Restore a previously disabled key
- **addUserID**: Attach an additional userID to a key to make it easier to identify in your scripts
- **removeUserID**: Remove a userID label from a key
- **removeSig**: Withdraw your signature from a specified key
- **trust**: Define a level of trust when you sign a key

Options

Option	Description
-inPublicPGP <file>	Specify the public keyring location for the key(s) you want to edit.
-inSecretPGP <file>	(Optional) Specify the private keyring location for the key(s) you want to edit.
-keyID	Used to identify a particular OpenPGP key by its unique key ID. This option can be specified multiple times on the command line. Either <i>-keyID</i> or <i>-userid</i> is required when selecting specific keys. You can use the complete short or long version of the key ID with this option.

Option	Description
-multi	Allow multiple keys to be selected.
-userid <regex>	OpenPGP userid to be used to locate an OpenPGP key in a keyring. This value can contain a name, email address and comment; for example: Tom <tom@example.com>. The <regex> parameter is a Java Regular Expression. Java regular expressions are virtually identical to Perl. Either -keyID or -userid is required when selecting specific keys.
-addUserID	Specify an additional userID for an OpenPGP key. This value can contain a name, email address and comment; for example: Tom <tom@example.com>.
-disable	Temporarily stop using a specific key or keys for encryption operations.
-enable	Resume using disabled key(s) for encryption.
-removeSig <keyID >	Remove a signature from an OpenPGP key.
-removeUserID	Remove a specific UserID from an OpenPGP key
-trust <level>	Set the level of trust you have in a specified (using <i>inPublicPGP</i> or <i>inSecretPGP</i>) public or private key. Values: none, marginal, complete, implicit

Sample commands

Disable OpenPGP key id 0x12345678 in the default OpenPGP key rings.

```
java -jar <path> edit -keyid 0x12345678 -disable
```

Add a new userid to an OpenPGP key in the default key ring

```
java -jar <path> edit -userid "John Doe" -addUserID "John Doe <john.doe2@example.com>"
```

Remove all signatures signed by OpenPGP key id 0x12345678 from all keys containing example.com

```
java -jar <path> edit -removeSig 0x12345678 -userid example.com
```

Set the trust level for a private and public key pair to Complete

```
java -jar <path> edit -trust -complete -userID "John Doe" -inPublicPGP target.pkr
-inSecretPGP target.skr
```

When performing operations such as setting Trust on public and private key pairs, make sure that the keyring files match, as in the above sample.

Deleting Keys From A Ring

delete

Removes one or more OpenPGP keys from a keyring.

Options

Option	Description
-in <file>	Input file name
-keyID	Used to identify a particular OpenPGP key by its unique key ID. This option can be specified multiple times on the command line. You can use the complete short or long version of the key ID with this option. Either -keyID or -userid is required.
-multi	Allow multiple keys to be identified and selected for deletion
-userid <regex>	OpenPGP userid to be used to locate an OpenPGP key in a keyring. This value can contain a name, email address and comment; for example: Tom tom@example.com . The <regex> parameter is a Java Regular Expression. Java regular expressions are virtually identical to Perl. Either -keyID or -userid is required.

CAUTION: The delete command is not interactive, you will not be asked to confirm the keys to delete. Take care not to remove wanted keys with regular expressions.

Sample commands

Delete the specified OpenPGP key from the OpenPGP public and secret keyrings.

```
java -jar <path> delete -inType pgp -in source.skr -keyid 085F4A9D5AA93E4D
```

Delete all OpenPGP keys that contain 'example.com' in the userid from the keyring.

```
java -jar <path> delete -in pubring.pkr -userid example.com -multi
```

Displaying Help Information

help

Display program help screen showing information on commands and options available.

```
java -jar <path> help
```

To see appropriate syntax, options and examples for a command, add **-help** to that command.

```
java -jar <path> <command> -help
```

Displaying Key Maker Version Information

version

Display program version information.

```
java -jar <path> version
```

Return Codes

A completion code dependent on the results of the processing that was carried out will be issued. The completion code can take the following values:

0	Processing has completed without errors being detected.
4	A warning message has been output but processing has continued.
8	An error has occurred during processing; refer to the error messages for more details.
12	A syntax error or configuration setup error was encountered. The command and/or combination of commands should be reviewed. The error can include inappropriate processing when attempting to locate digital keys for encryption or authentication functions.

The final completion code issued is the maximum value of the conditions found during the sum. A return code greater than zero indicates that there are warning or error messages in the job output.

User Help and Contact Information

For licensing, please contact Sales at 937-847-2374 (888-4PKWARE / 888-475-9273) or email pksales@pkware.com.

For technical assistance, contact Technical Support at 937-847-2687 or visit the support web site: <http://www.pkware.com/support>.