



encrypted data gateway engine

User Manual

command line version



Authora Inc. 2011

support@authora.com
<http://www.authora.com>

COPYRIGHT

© 2002-2011 AUTHORA INC. ALL RIGHTS RESERVED. NO PART OF THIS PUBLICATION MAY BE REPRODUCED, TRANSMITTED, TRANSCRIBED, STORED IN A RETRIEVAL SYSTEM, OR TRANSLATED INTO ANY LANGUAGE IN ANY FORM OR BY ANY MEANS WITHOUT THE WRITTEN PERMISSION OF AUTHORA, INC., OR IT'S SUPPLIERS OR AFFILIATE COMPANIES. TO OBTAIN THIS PERMISSION, WRITE TO THE ATTENTION OF THE AUTHORA INC. 1405 E JOHN ST STE 2, SEATTLE WASHINGTON 98112 OR CALL +1-206-783-8000.

COMPLIANCE WITH APPLICABLE LAWS; EXPORT CONTROL LAWS.

USER ACCESS TO THIS DOCUMENT IS GOVERNED BY ALL APPLICABLE FEDERAL, STATE AND LOCAL LAWS. ALL INFORMATION AVAILABLE IN THIS DOCUMENT IS SUBJECT TO U.S. EXPORT CONTROL LAWS AND MAY ALSO BE SUBJECT TO THE LAWS OF THE COUNTRY WHERE YOU RESIDE. ALL AUTHORA PRODUCTS AND PUBLICATIONS ARE COMMERCIAL IN NATURE. USE DUPLICATION, OR DISCLOSURE BY THE UNITED STATES GOVERNMENT IS SUBJECT TO THE RESTRICTIONS SET FORTH IN DFARS 252.227-7015 AND FAR 52.227-19.

LICENSE AGREEMENT

NOTICE TO ALL USERS: CAREFULLY READ THE APPROPRIATE LEGAL AGREEMENT CORRESPONDING TO THE LICENSE YOU PURCHASED, WHICH SETS FORTH THE GENERAL TERMS AND CONDITIONS FOR THE USE OF THE LICENSED SOFTWARE. IF YOU DO NOT KNOW WHICH TYPE OF LICENSE YOU HAVE ACQUIRED, PLEASE CONSULT THE SALES AND OTHER RELATED LICENSE GRANT OR PURCHASE ORDER DOCUMENTS THAT ACCOMPANY YOUR SOFTWARE PACKAGING OR THAT YOU HAVE RECEIVED SEPARATELY AS PART OF THE PURCHASE (AS A BOOKLET, A FILE ON THE PRODUCT CD, OR A FILE AVAILABLE ON THE WEB SITE FROM WHICH YOU DOWNLOADED THE SOFTWARE PACKAGE). IF YOU DO NOT AGREE TO ALL OF THE TERMS SET FORTH IN THE AGREEMENT, DO NOT INSTALL THE SOFTWARE. IF APPLICABLE, YOU MAY RETURN THE PRODUCT TO AUTHORA INC. OR THE PLACE OF PURCHASE FOR A FULL REFUND.

Table of Contents

1 PREFACE	7
1.1 CONTACTING AUTHORA INC.	7
2 GETTING STARTED	8
2.1 CONVENTIONS USED IN THIS MANUAL	8
2.2 INTRODUCTION.....	8
2.3 BASIC PROCEDURES	9
2.4 ABOUT OPENPGP.....	11
2.5 INSTALLATION.....	14
2.5.1 WINDOWS.....	14
2.5.2 UNIX.....	14
2.5.3 z/OS	15
2.6 LICENSE FILE.....	16
3 STARTING EDGE	17
3.1 WINDOWS	17
3.2 UNIX.....	18
3.3 LOCATION OF FILES	18
3.4 SPECIFYING COMMANDS AND OPTIONS.....	20
3.5 LONG OPERATIONS	22
3.6 GETTING HELP	22
4 USING EDGE	23
4.1 RUNNING SELF-TESTS	23
4.2 BASIC OPERATIONS	24
4.3 GENERATING YOUR KEY PAIR	24
4.3.1 SPECIFYING KEY TYPE.....	26
4.3.2 SPECIFYING KEY SIZE.....	27
4.3.3 SPECIFYING A VALIDITY PERIOD	27
4.3.4 AUTOMATING KEY PAIR CREATION.....	27
4.4 GENERATING A SIGNATURE-ONLY KEY	28
4.5 ADDING A SUBKEY TO YOUR KEY PAIR	28
4.6 SENDING YOUR PUBLIC KEY.....	29
4.7 ADDING PUBLIC KEYS OF OTHER PERSONS	30
5 MANAGING KEYS	31
5.1 ADDING KEYS TO YOUR KEYRINGS.....	31
5.2 DISPLAYING YOUR KEYRING	32
5.3 STATUS OF KEYS.....	36
5.4 REMOVING KEYS	37
5.5 CERTIFYING KEYS.....	38
5.6 EXPORTING KEYS	39
5.7 CHANGING THE PASSPHRASE OF A PRIVATE KEY.....	41
5.8 ADDING A USER ID TO YOUR KEY	41
5.9 DISABLING A KEY.....	42
5.10 ENABLING A KEY	43
5.11 REVOKING A KEY.....	43
5.12 REMOVING A USER ID.....	44
5.13 REMOVING A SUB-KEY	44
5.14 REMOVING A SIGNATURE.....	45
5.15 REVOKING A SIGNATURE	45
6 ENCRYPTING AND SIGNING	46
6.2 ENCRYPTING DATA WITH A PUBLIC KEY	46

6.2 ENCRYPTING DATA WITH A PASSWORD	47
6.3 SIGNING DATA.....	48
6.4 DETACHED SIGNATURE.....	50
6.5 CLEAR-SIGNED DATA	51
6.6 ENCRYPTING AND SIGNING DATA	52
6.7 DECRYPTING DATA	53
7 ADVANCED OPTIONS	55
7.1 SPECIFYING INPUT FILE TYPES	55
7.2 SPECIFYING OUTPUT FILE TYPES.....	56
7.3 SPECIFYING OUTPUT FILE OR DIRECTORY	56
7.4 FILTER MODE	57
7.5 REDIRECTING OUTPUT TO THE SCREEN	57
7.6 REDIRECTING OUTPUT AND ERROR MESSAGES (UNIX).....	58
7.7 REMOVING USER INTERVENTION	58
7.7.1 BATCHMODE	58
7.7.2 FORCE.....	59
7.7.3 INTERACTIVE	59
7.8 SPECIFYING A PASSPHRASE.....	59
7.8.1 --PASSPHRASE OPTION	59
7.8.2 PGPPASS OPTION	60
7.8.3 PASSPHRASE FILE	60
7.8.4 ENCRYPTED PASSPHRASE FILE	61
7.9 ENCRYPTING "FOR YOUR EYES ONLY"	61
7.10 GENERATING A SELF-DECRYPTING ARCHIVE (SDA).....	62
7.11 MANAGING TEMPORARY FILES	63
8 LOG INFORMATION	64
8.1 UNIX.....	64
8.2 WINDOWS	64
9 WORKING WITH SESSION KEYS	65
9.1 EXTRACTING THE SESSION KEY TO A FILE	65
9.2 DISPLAYING THE SESSION KEY ON THE SCREEN.....	66
10 WORKING WITH KEY SERVERS (WINDOWS ONLY).....	67
10.1 DISPLAYING KEYS AVAILABLE ON THE SERVER.....	67
10.2 IMPORTING KEYS FROM THE SERVER.....	68
11 WORKING WITH X.509 CERTIFICATES (WINDOWS ONLY)	69
11.1 DISPLAYING AN X.509 CERTIFICATE	69
11.2 IMPORTING AN X.509 CERTIFICATE	70
11.3 ENCRYPTING AND SIGNING DATA	70
12 COMPATIBILITY	71
13 CONFIGURATION FILE.....	72
13.1 LOCATION OF THE CONFIGURATION FILE	72
13.1.1 WINDOWS	73
13.1.2 UNIX	73
13.2 WORKING IN A SHARED ENVIRONMENT.....	73
13.3 SUPPORTED SETTINGS.....	75
13.3.1 ADDPUBLICKEYS.....	76
13.3.2 ADDSECRETKEYS	77
13.3.3 ARMOR.....	77
13.3.4 BACKUPPUBRING	77
13.3.5 BACKUPSECRING	78
13.3.6 BATCHMODE	78

13.3.7 CHECK-SIGNED	78
13.3.8 CIPHERNUM	79
13.3.9 CLEARSIG	79
13.3.10 CMDLINE-FORMAT	80
13.3.11 COLORS	80
13.3.12 COMMENT	80
13.3.13 COMPRESS	81
13.3.14 COMPRESSLEVEL	81
13.3.15 CONFIG-FILE	81
13.3.16 COMPAT-ERRORS	81
13.3.18 DECRYPTONLY	82
13.3.19 DEFAULT-KEY	82
13.3.20 ENCRYPT-TO-SELF	82
13.3.21 ERRORFD	82
13.3.22 EXPIRES-AFTER	83
13.3.23 EXPORTPUBLIC	83
13.3.24 EXPORTSECRET	83
13.3.25 FINGERPRINT-VIEW	83
13.3.26 FORCE	84
13.3.27 GETSESSIONKEY	84
13.3.28 HASHNUM	84
13.3.29 HELP FILES	85
13.3.30 INTERACTIVE	85
13.3.31 KEY-SIZE	85
13.3.32 KEY-TYPE	85
13.3.33 LICENSE-FILE	85
13.3.34 LOGFD	86
13.3.35 LOGFILE	86
13.3.36 LOGIN	86
13.3.37 LOGINPASS	86
13.3.38 LOGLEVEL	86
13.3.39 LOGSESSION	87
13.3.40 MERGEONLY	87
13.3.41 NO-CONFIG-FILE	87
13.3.42 NOCOPYRIGHT	87
13.3.43 NOLICENSEINFO	87
13.3.44 NOLOGFILE	88
13.3.45 NOOUT	88
13.3.46 NOOUTPUT	88
13.3.47 NOPROGRESS	89
13.3.48 NOSYSLOG	89
13.3.49 PASSTRY	89
13.3.50 PRESERVE-NAME	89
13.3.51 PUBRING	89
13.3.52 PRINTSESSIONKEY	90
13.3.53 REVERSE	90
13.3.54 RSAVER	90
13.3.55 SDA	90
13.3.56 SECRING	91
13.3.57 SECURE-VIEWER	91
13.3.58 SIGN-ONLY	91
13.3.59 SIG-TYPE	91
13.3.60 SORT	92
13.3.61 STATUSFD	92
13.3.62 TEXTMODE	93
13.3.63 TMP	93
13.3.64 VERBOSE	93
13.3.65 VERSION	94
13.3.66 WIPE-PASSES	94

14 LEGACY MODE COMMANDS	95
14.1 ALLOWED COMMANDS	95
15 APPENDIX A – BIOMETRIC WORD LISTS	99
TWO SYLLABLE WORD LIST	99
THREE SYLLABLE WORD LIST	100
16 APPENDIX B - ERROR CODES	102
17 APPENDIX C – COMPATIBLE ERROR CODES	110
18 APPENDIX D - EDGE ON Z/OS	112
18.1 INTRODUCTION	112
18.2 CUSTOMIZATION	112
18.2.1 VERIFY PROPER INSTALLATION	113
18.3 USAGE	114
18.4 DISPLAYING EDGE CONFIGURATION INFORMATION	115
18.4.1 DISPLAYING EDGE INFORMATION	115
18.4.2 DISPLAYING EDGE CONFIGURATION FILE	115
18.5 IMPORTING FILES	116
18.5.1 IMPORTING A FILE FROM Z/OS	116
18.5.2 IMPORTING AN UNTAGGED USS FILE	117
18.5.3 IMPORTING AN EBCDIC USS FILE	117
18.5.4 IMPORTING AN ASCII USS FILE	118
18.5.5 IMPORTING AN UNTAGGED/EBCDIC FILE AND CONVERTING TO ASCII	118
18.6 ENCRYPTING FILES	120
18.6.1 -C	120
18.6.4 --ENCRYPT	122
18.6.5 --ENCRYPT ARMOR TEXT	123
18.7 TRANSFERRING FILES	124
18.7.1 PGP TRANSFERS	124
18.7.2 ASC TRANSFERS	124
18.8 DECRYPTING FILES	125
18.8.1 PGP FILES	125
18.8.2 ASC FILES	126
18.8.3 VIEWING FILES WITH SECURE-VIEWER	126
18.9 EXPORTING FILES	127

1 Preface

Authora's Encrypted Data Gateway Engine (EDGE) Command Line Version (CLV) enables enterprise customers to easily secure automated enterprise-class e-commerce applications and batch processes using strong encryption and authentication technology to ensure complete end-to-end security. As a result, customers can easily incorporate strong encryption and authentication technology directly into mission-critical e-commerce processes across a wide range of database servers, web servers, ftp servers, business applications and client programs.

In the real world, trust, confidentiality and authenticity of sensitive data is preserved through simple acts like putting mail in envelopes, locking doors and filing cabinets, signature cards at banks, signing transactions, and notarizing documents. To achieve these same protections on the Internet, we must first become cryptographically enabled to protect our data. Cryptography lies at the heart of confidentiality, trust, and security on the Internet. Cryptography is the foundation on which Internet trust is being built. You and/or your organization lay a secure foundation for your data and network when you take control and are enabled with the tools necessary to digitally protect your interests with strong cryptography.

This guide has been created for administrators and users implementing EDGE to cryptographically protect an organizations digital data. EDGE integrates into your organization's overall risk management and security solutions and is used to protect the security and integrity of your organization's data. Administrators use EDGE for encrypting, digitally signing, and verifying signatures. These cryptographic functions protect the integrity of digital data, can provide authentication of data, and can protect the confidentiality of digital data. In addition, if an organization has created policies for accepting digital signatures, EDGE can provide the cryptographic foundation for non-repudiation of data allowing for legally binding contracts.

This guide describes how to install and use EDGE. It is divided in two parts, Part I is an introductory manual suitable for common user who needs to encrypt and digitally sign data. Part II is intended for system administrators and developers who will use EDGE to add security to their existing process.

1.1 Contacting Authora Inc.

Authora Inc.
1405 E John St, Ste 2
Seattle, WA 98112
Tel: 206-783-8000

Sales Information: sales@authora.com
Technical Support: support@authora.com

<http://www.authora.com>

2 Getting Started

2.1 Conventions Used In This Manual

The following describes the conventions used in this guide:

Angle brackets (<>) indicate a variable. You supply a value of the type indicated.

Square brackets ([]) indicate an option. The value indicated is not required.

2.2 Introduction

Overview: Authora's Encrypted Data Gateway Engine (**EDGE**) is a command-line application used for encryption, decryption, digital signing of digital data and cryptographic key management. EDGE Administrators can perform these operations manually or they can configure the EDGE to perform these operations automatically. Since EDGE is a command-line application, all operations can be executed in a completely automated way and can be used on a server without user intervention. EDGE integrates public and private key encryption and provides a complete and easy-to-use cryptographic system. EDGE enables an enterprise to perform the following cryptographic functions:

- Encrypt digital data using public and/or private-key encryption;
- Decrypt digital data;
- Generate digital signatures;
- Verify digital signatures;
- Generate encryption keys;
- Certify encryption keys;

OpenPGP Standard: EDGE uses and produces files compatible with the OpenPGP standard (RFC1991 and RFC 2440). Files produced by EDGE are compatible with other OpenPGP applications such as Zentit, PGP, and McAfee E-Business Server.

Language: EDGE can be called from high-level languages (such as C/C++, Java, Visual Basic, etc.) and from scripting languages (such Windows batch file, Perl, PHP, csh, etc.). EDGE supports input and output redirection for easy and efficient integration with other command-line tools.

Platforms: EDGE is available for Windows and for various UNIX systems.

Shared Environment: EDGE can run in a shared environment, enabling multiple users to have a unique set of cryptographic keys and their own settings. Each user can have a different configuration and can manage EDGE individually. The administrator of the system can also apply security and risk management policies preventing inappropriate use of the service by users.

FIPS PUB140-2 Compliant: EDGE is compliant with FIPS PUB140-2 and provides several mechanisms to check the cryptographic library. Some verification is performed automatically and continuously alongside program execution. Other verification can be performed upon request. Runtime verification includes continuous random number generator testing. Power-up tests can be turned on or off and include:

- Cryptographic algorithm test;
- Statistical random number generator tests
- Pair-wise consistency test (during key generation)

All tests follow the recommendations of FIPS PUB140-2 from the National Institute of Standards and Technology (NIST).

2.3 Basic Procedures

The following bullet points consist of basic procedures a user would normally follow in the course of using EDGE. For details concerning any of these procedures, refer to the appropriate section in this user manual.

Install EDGE on your Server

Details on installing edge for different platforms are described in the Installation chapter of this manual.

Create or Import your Cryptographic Keys/Key Pair

To use EDGE you need a cryptographic key pair. If you already have an OpenPGP compliant key pair, then you can import it into EDGE. If you do not have a key pair you will need to generate one. You can use EDGE to create a new key pair at any time after you have finished installing it. Details for generating or importing cryptographic keys are described in the Managing Keys chapter of this user guide.

Exchange Public Keys with Others

After you have created your key pair you can begin corresponding with other EDGE users or users of OpenPGP compliant applications. In order for others to encrypt data to you, they will need a copy of your public key. In order for you to encrypt data to others, you will need a copy of their public key. Public keys can be converted as a block of text, so it's easy to trade public keys with someone. You can include your public key in an email message, copy it to a file, or post it on a public key server where anyone can get a copy when they need it.

Validate Public Keys

If you want to make sure the public key you want to use to encrypt data is really the public key of the recipient, you can compare the unique fingerprint on your copy of the public key to the fingerprint on the owner's original key. There are many trust models for doing this, including verifying directly with the owner of the public key in person or on the phone. When you are confident that the fingerprint is the same, then you can digitally sign your copy of the key using your private key. This action tells EDGE that you have gone through your steps to verify that it is a good copy of the recipient's public key and you therefore consider it valid to use for encrypting data or verifying digital signatures.

Encrypt and/or Digitally Sign Files

After you have generated your key pair and have exchanged public keys, you can begin encrypting and digitally signing files. Details on encrypting and decrypting files using EDGE can be found in this guide.

Decrypt and Verify Digital Signatures

When someone has encrypted data using your public key and has sent the encrypted data to you, you can decrypt it using your private key. You can also verify the sender's digital signature if you have the sender's public key. Details on decrypting files and verifying signatures using EDGE can be found in this guide.

Configure Policies in a Shared Environment

The computer administrator can pre-configure options for all users and can even restrict users from modifying options. Details on configure policies can be found in this guide.


Configure EDGE to Meet User Needs


The EDGE configuration file is created when you install it. Administrators of EDGE can re-configure EDGE to meet their specific needs.

2.4 About OpenPGP

EDGE follows the OpenPGP standard, a widely used cryptographic standard used to encrypt, sign and decrypt digital data. A complete definition of the OpenPGP standard can be found in RFC 2440 from IETF (<http://www.ietf.org>).

The OpenPGP standard is based on a highly trusted *public key encryption* system in which two complementary keys, called a *key pair*, are used to maintain secure communications. One of the keys is designated as a **private key**, to which only the owner of the private key should have access, and the other is a **public key** which one can freely exchange with other users so they can encrypt data to you and verify your digital signatures. Both the private and public keys are stored in *keyring files*.

<p>Public Key You can give your Public Key to anyone</p>		<p>Your Public Key is what others will use to encrypt data meant only for you</p>
---	---	---

<p>Private Key Do not share your Private key with anyone</p>		<p>Use your Private Key to decrypt data that has been encrypted to you. Your private key is also what you use to digitally sign data.</p>
---	---	---

PRIVATE KEY: Your private key is what you use to decrypt data encrypted to the associated public key. You also use your private key to digitally sign data you are sending so the recipient(s) can verify that the data really came from you and that the data was not compromised while in transit.

IT IS IMPORTANT TO PROTECT YOUR PRIVATE KEY – As the name implies, only you or those you or your organization authorize should have access to your private key.

PUBLIC KEY: Your public key is what others use to encrypt data to you. You can give your public key to anyone. Public keys can be sent by email or can be uploaded into a public key server. You can also use a public key server to retrieve public keys for people or organizations you want to encrypt or digitally sign data to. Recipients also use your public keys to verify digital signatures. Public keys can be sent by email or can be uploaded into a public key server. This public key server can also be used to retrieve public keys from persons you want to communicate with.

Attributes of Keys: An OpenPGP key can be identified using the following attributes:

- **Creation Date:** An OpenPGP key has a creation date indicating when the key was created by its owner. Digital signatures made using this key can never have a creation date prior to that date.
- **Expiration Date:** A key can be generated to only be valid for a certain amount of time. The expiration date of a key specifies the date until that key can be used to encrypt/decrypt data and/or sign/verify digital signatures.
- **Key ID:** The key ID is an 8 or 16 digit alphanumeric value and contains numbers from 0 to 9 and letters from A to F. It's represented by the prefix "0x" followed by 8 or 16 digits. Key IDs are the same for the private and the corresponding public key.
- **Fingerprint:** The fingerprint is a longer value of either 32 or 40 digits in length, depending on the type and version of the key. Like the key ID, both values are the same for a private and its corresponding public key.
- **Names (*a.k.a. User IDs and/or email address*):** A key can hold more than one name. By convention, a name is formed by a real name followed by an email address:

Robert J. Smith <rsmith@company.com>

EDGE uses names or key IDs to identify keys. If more than one key has the same name, you can specify the key by using the key ID, just prefix the key ID value by "0x".

Additional Attributes of Keys: OpenPGP keys also contain attributes like preferred algorithms and revocation signatures. Some of these attributes are automatically checked by EDGE before using the key. For example, an expired or revoked key cannot be used to encrypt data.

Passphrase for Private Key: A private key is encrypted using the passphrase you entered when you generated your key pair. This passphrase is requested to unlock the key before decrypting data or before signing data with that private key.

Primary Keys and Subkeys: Some keys contain a primary key and one or more subkeys. The primary key can generate and verify digital signatures, while subkeys can only encrypt/decrypt. A subkey is valid only if its primary key signs it.

About Keyrings: OpenPGP keys are stored in files called keyrings. For security reasons, a keyring contains either only public keys or only private keys, but never both. This allows you to store your private key on a removable media.

A private key should never be distributed and must be kept in a secure place. The passphrase needed to unlock a private key should never be distributed and must be kept secret. **Never send a private key to a public key server.**

A key can be revoked. By revoking a key, you inform other users that your private key has been compromised.

A revoked key cannot be used to encrypt or digitally sign data.

If your private key or passphrase has been compromised, revoke your public key immediately and post it to a public key server directly. All digital signatures created by the compromised private key after its revocation date will be invalid. All signatures created with the private key before the revocation date remain valid.

2.5 Installation

2.5.1 Windows

To install EDGE on Windows, simply launch the installer application and follow the instructions on the screen.

The installer installs the EDGE application, the EDGE User Guide, the End-UserLicense Agreement and additional files at the same location.

2.5.2 UNIX

EDGE comes as a tar-gzipped archive. To decompress this archive you need a tar utility.

To install EDGE on UNIX, open a terminal window, move to the directory containing the EDGE archive and type:

```
tar -xzf ./edge_xxxx.tar.gz
```

If your tar utility doesn't support the `-z` option, type the following commands:

```
gzip -d ./edge_xxxx.tar.gz  
tar -xf ./edge_xxxx.tar
```

EDGE can be installed anywhere on your disk. To make it available for all users, it's recommended to copy it in `/usr/bin`. This directory is one of the default directories for program files.

Before using EDGE, be sure the "edge" file has been set as "executable". To set the "executable" flag on, from the EDGE directory and type:

```
chmod u+x ./edge
```

To copy EDGE in `/usr/bin`, move to the EDGE directory and type:

```
cp ./edge /usr/bin
```

Note: To copy EDGE into this directory, you need to have write-access to the specified location.

The EDGE installation directory will contain additional files such as the User Guide, man page and help files.

The man page should be copied in the directory containing man pages of other tools. Depending on the system, it can be one of the following locations:

```
/usr/man/man.1/
/usr/local/man/man.1/
```

To copy the man page of EDGE, use the following command:

```
cp edge.1 <man pages locations>
```

To make the Self-Decrypting Archive (SDA) feature available on your system, the "SDA.bin" file must be located in the same directory as the application.

In order to get help directly available from EDGE, you can copy the edge_cmd.txt and edge_help.txt files into the same directory as the application.

EDGE is now installed.

2.5.3 z/OS

The following process describes the steps necessary to install EDGE for z/OS:

Download & unzip the `edge-zos.zip` file from Authora's website.

Upload the `EDGE.XMIT` file to the mainframe. You may want to pre-allocate an `AUTHORA.EDGE.V1R0.XMIT` file as `FB - LRECL=80 - BLKSIZE=27920`.

Receive the XMIT file by issuing the following command from TSO Command Prompt (typically TSO option 6) :

```
RECEIVE DA ('AUTHORA.EDGE.V1R0.XMIT')
```

When prompted for restore parameters, reply:

```
DA ('AUTHORA.EDGE.V1R0.INSTLIB')
```

There are 27 members that will be restored to INSTLIB.

Copy `AUTHORA.EDGE.V1R0.INSTLIB(EDGEPROC)` to `SYS1.PROCLIB(EDGE)` or other equivalent `PROCLIB`.

Copy `AUTHORA.EDGE.V1R0.INSTLIB(EDGEEXEC)` to `SYS1.SYSPROC (EDGE)` or other equivalent `SYSPROC` library that is in the TSO `SYSPROC` concatenation.

Upload the `EDGETAR` file to the Unix Systems Services directory on the mainframe where you plan to install EDGE for z/OS.

Extract the `EDGETAR` file using the following command - `tar -xzf ./EDGETAR`

2.6 License File

EDGE requires a license file to work. This license file contains details about the license including company or organization name, allowed functionality, and expiration date (if the license is time-limited). The license file also contains information about the owner of the license. This information is displayed on the screen each time EDGE is launched. EDGE must find and verify the license file before EDGE can run.

If EDGE cannot find the license file, it displays the default locations for the license file and returns an error.

To display information about the license you have purchased, type:

```
edge --version
```

EDGE displays two locations where the license file can be placed. The recommended location is in the default EDGE directory.

```
edge - Encrypted Data Gateway Engine  
Version 3.7  
Copyright (C) 2002–2007 Authora Inc. & Veridis SA  
All rights reserved.
```

```
EDGE directory:  
C:\Documents and Settings\Alice\My Documents\edge data\
```

```
Configuration file:  
C:\Documents and Settings\Alice\My Documents\edge data\edge.cfg
```

```
Signature allowed  
Encryption allowed  
Decryption allowed
```

NOTE: If you want to purchase a license for EDGE, contact Authora, Inc. or visit our website: <http://www.authora.com>

NOTE ON TIME-LIMITED LICENSES: from 30 days before the expiration date of your license, EDGE displays the number of days before its expiration.

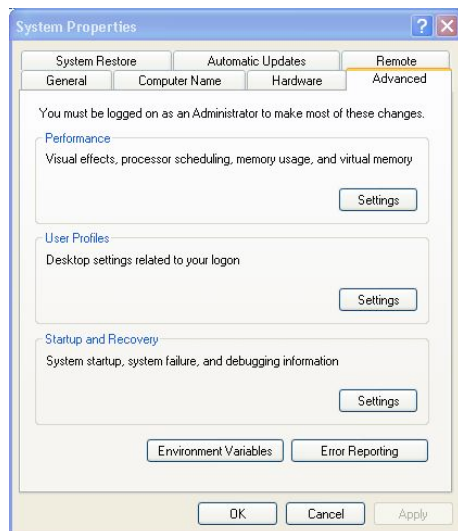
3 Starting EDGE

3.1 Windows

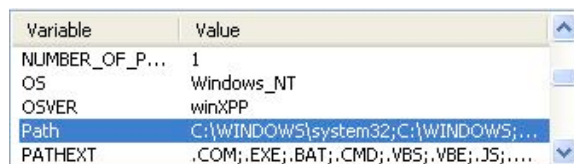
Open the "Start.bat" file located in the EDGE application directory. This directory has been created by the EDGE installer. The default location for EDGE application directory is "C:\Program Files\Authora\EDGE\". This batch file opens a command window by setting the default directory in the same directory as the location of the batch file.

You can make EDGE available from any location by adding the EDGE directory path to the default paths.

1. Click on the "Start" menu.
2. Choose "Control Panel"
3. Open the "System" item
4. Click on the "Advanced" tab.



5. Click on "Environment Variables" at the bottom of the windows.
6. Select "Path" from the list



7. Click on "Edit" and add the EDGE directory path.

3.2 UNIX

1. Open a terminal window.
2. Type "edge" at the command line.

EDGE displays copyright information and version number.

If EDGE has not been installed in /usr/bin and if the EDGE application directory is not present in the PATH environment variable, you need to move to the EDGE application directory and type:

```
./edge
```

The EDGE application located in the current working directory will be used.

To know which EDGE application will be used type:

```
which edge
```

3.3 Location of Files

EDGE needs to know where the following files are located:

Configuration File: EDGE uses a configuration file to store a number of user-defined parameters. This file is named "edge.cfg". A configuration file enables the user to define flags and parameters for EDGE, eliminating the need to define these parameters at the command line. A default configuration file is created when you run EDGE for the first time. This configuration file can be edited using a text editor or by using EDGE directly. Refer to the Configuration File section for information about changing the configuration file.

To display the location of the configuration file used by EDGE, type:

```
edge --version
```

EDGE displays the location of the configuration currently used.

```
edge - Encrypted Data Gateway Engine
Version 3.7
Copyright (C) 2002–2007 Authora Inc. & Veridis SA
All rights reserved.

EDGE directory:
C:\Documents and Settings\Alice\My Documents\edge data\

Configuration file:
C:\Documents and Settings\Alice\My Documents\edge data\edge.cfg
```

You can specify the path to the configuration file using the environment variable `PGPPATH` or by specifying it directly at the command-line:

```
SET PGPPATH=<pathName>
```

Or

```
edge --pgppath <pathName>
```

If `PGPPATH` is not defined, EDGE uses the default location depending on the operating system you are using.

Default File Locations in Windows:

EDGE first checks if `PGPPATH` is defined.

If `PGPPATH` is defined, EDGE uses the configuration file located in that the configuration file doesn't exist, EDGE creates it. If the configuration file cannot be read or created, an error is returned by EDGE and the operation is cancelled.

If `PGPPATH` is not defined, EDGE uses the "My Documents\edge data" directory of the current user account to locate the configuration file. If the configuration file cannot be found, EDGE creates it.

Default File Locations in UNIX:

EDGE first checks if `PGPPATH` is defined.

If `PGPPATH` is defined, EDGE uses the configuration file located in that directory. If a configuration file doesn't exist, EDGE creates one.

If `PGPPATH` is not defined, EDGE uses the "~/.edge" directory. If the configuration cannot be found, EDGE creates it.

Your Keyring Files: EDGE stores your key pair in two files: Your public keys are stored in `pubring.pgp` and your private keys are stored in `secring.pgp`.

Keyrings store the keys used to encrypt, sign and decrypt messages. If you add another person's public key to your keyring, it is stored in the `pubring.pgp` file. Your private keys are always stored in the `secring.pgp` file.

Locations of keyrings are specified in the configuration file. This configuration file can be edited to change keyring location using a text editor or by using EDGE directly. Refer to the Configuration File section for information on making these changes.

License File: The recommended location is in the default EDGE directory.

3.4 Specifying Commands and Options

EDGE accepts a large range of commands and options and supports two different ways to specify options. The first one is the one used by the version of EDGE prior to 2.0 and is compatible with PGP 2.6.x and other OpenPGP command-line applications. This mode is called the legacy mode. The second way to specify options is the default one and is supported only by newer OpenPGP command-line applications

With the legacy mode, a command always begins with the character '-'. Commands can be combined. For example, the three following commands have the same effect:

P

```
edge -a -t -c <file.txt> -z <mypassword>
edge -ac <file.txt> -z <myPassword> -t
edge -atc <file.txt> -z <mypassword>
```

The second mode uses more explicit names for options. In that mode, a command always begins with the characters '--'. Commands are followed by values if required:

```
edge --conventional-encrypt <file.txt>
edge --armor --text <file.txt>
```

EDGE uses the configuration file to store user options. These options can also be specified at the command-line.

With the legacy mode, an option begins by the character '+'. An option is followed by its value:

```
edge +ARMOR=ON
edge +ARMOR=OFF
edge +ARMOR=1
```

If you omit the value for a Boolean option, its value defaults to "ON". For instance, the two following lines have the same effect:

```
edge +ARMOR=ON
edge +ARMOR
```

When using long arguments, options are specified the same way as the commands, by specifying the option preceded by the characters '--'.

```
edge --armor on
edge --armor off
```

If only options are specified at the command-line then the specified options replace the same options defined in the configuration file. If other options exist in the configuration file, they are neither altered nor deleted. This mechanism is useful for changing options without editing the configuration file with a text editor.

Values for specified options remain for future operations and are saved in the configuration file. If commands and options are specified together, the given options override options from the configuration file for the current operation only and leave the configuration file unchanged.

Examples:

```
edge +ARMOR
edge --armor on
```

The above lines contain only options and no commands. EDGE changes the configuration file and the value for the ARMOR option will be set to ON.

```
edge -e <file.txt> <userID> +ARMOR
edge --encrypt <file.txt> --user <userID> --armor
```

In the above lines, EDGE encrypts the specified file with the specified public key and produces an armored file. The value for the ARMOR option remains unchanged in the configuration file.

All options can also be specified by using environment variables. If a setting is defined by an environment variable, its variable value is used instead of the value found in the configuration file.

The default mode for specifying options is the long arguments mode. The legacy mode exists to assure the compatibility of scripts written for old versions of OpenPGP command-line products.

You don't need to specify that you are using the legacy or the long arguments mode, EDGE automatically detects it and checks the syntax of your command accordingly.

The long arguments mode allows for full automation of all EDGE commands. For example, the creation of key pairs cannot be fully automated with the legacy mode but can be with the long arguments mode.

Because the long arguments mode is the default mode to specify options, this manual presents examples using that mode.

Refer to the **Legacy Mode Commands** section for the syntax of all commands and options for the legacy mode.

3.5 Long Operations

For long cryptographically intensive operations, like encrypting large files, EDGE displays progress information and an estimated remaining time.

EDGE displays a progress bar informing the user of the progress of the current task.

```
12% |****|
```

After a while, EDGE displays an estimation of the time remaining to complete the current task.

```
25% |*****| ERT 01:37
```

When the operation is complete, EDGE displays the following line:

```
100% |*****|
```

During decryption, EDGE just displays a spinning cursor.

3.6 Getting Help

EDGE comes with a complete help mechanism allowing the user to get help on most commands supported by EDGE. The `--help` (`-h` in legacy mode) command displays a summary of available commands and options. To display help on supported commands, type:

```
edge --help
```

Specific help is also available for all supported commands and group of commands. For example, if you want to get help on key operations, type:

```
edge --help --key
```

EDGE displays help on available key operations. To get help on a specific command, just type `--help` followed by the name of the command you would like to get help for.

Examples:

```
edge --help --key-list
edge --help --encrypt
edge --help --decrypt
```

On UNIX systems, man pages are also available. To display it, type:

```
man edge
```

4 Using EDGE

Authora's Encrypted Data Gateway Engine (EDGE) has been designed to seamlessly integrate into existing e-Business processes and enable new secure business processes to protect your corporate information while in storage or in transit. The flexible command line interface of EDGE allows you to quickly integrate EDGE with automated processes and web-based applications.

Before using EDGE, you need your own key pair (your public key and its corresponding private key, encrypted with a password). If you do not have a key pair you will need to generate one, see the **Generating Key Pair** section of this guide. If you already have a key pair, EDGE allows you to import your existing key pair and continue to use it with the same password.

In order to communicate securely with other persons, you have to import their public keys in your default keyring.

4.1 Running Self-Tests

EDGE is compliant with the recommendations of the National Institute of Standards and Technology (NIST) and follows the recommendations published in FIPS-140-2. EDGE performs automatic tests during its execution. In addition to those tests, EDGE allows you to run tests to check the integrity of the cryptographic engine. Those tests are performed when the application starts up. To activate those tests, use the FIPSPowerUpTests option.

To activate the startup tests:

```
edge --fipspoweruptests on
```

To deactivate the startup tests:

```
edge --fipspoweruptests off
```

4.2 Basic Operations

To encrypt data and to verify digital signatures made by another person or organization, you need to have a copy of their public key. You will learn later in this guide how to get public keys from a public key server.

To digitally sign and to decrypt data, you need to have your own key pair (public key and corresponding private key).

EDGE stores keys using keyrings. Locations of keyrings are specified in the configuration file. This configuration file can be edited using a text editor or by using EDGE directly. Refer to the **Configuration File** section for information about changing the configuration file.

EDGE allows you to display the content of your keyrings, import keys, export keys, generate new keys, sign keys, and manage key names and passwords.

To communicate securely with other parties, you need to add their public keys to your public keyring and to have your own key pair.

To send your public keys to other persons, you need to extract your public key from your public keyring.

Never share or distribute your private key. It must be kept secret and encrypted with a password.

If you already have your own key pair, skip the next section.

4.3 Generating Your Key Pair

To digitally sign data and to allow other persons to encrypt data readable only by you, you need to have your own key pair. EDGE generates key pairs compatible with other OpenPGP clients.

To generate a new key pair based on some predefined constants such as the key size and the key type, type:

```
edge --key-gen
```

By default, EDGE generates a DSS/DH key type composed of a primary key of 1024 bits length and a subkey of 2048 bits length and with no expiration date.

EDGE allows you to generate DSS/DH, RS or RSA Legacy keys.

The type of key you want to generate can depend upon what kind of OpenPGP client the person or organization you want to communicate securely with is using. Keep in mind that older versions of OpenPGP clients handle only specific formats of RSA keys and some newer OpenPGP clients only handle DSS/DH keys, the ElGamal variant of Diffie-Hellman technology. EDGE supports both RSA and DSS/DH keys. Also keep in mind that there are two different formats of RSA keys- **RSA** and **RSA Legacy**:

Legacy:

- **RSA:** If you select RSA, EDGE generates the new standard RSA key pair format by default. This means keys compatible with newer OpenPGP clients. The new RSA key format supports features previously available only to DSS/DH keys. The new RSA key format enables you to have a primary key for signing and a subkey to encrypt data. In addition the encryption key (the subkey) can be revoked or have a different expiration date than its primary key. A new subkey can always be added to a primary key and can be used for encrypting data. New RSA keys are compatible with newer versions of OpenPGP. These keys are not compatible with older PGP clients not compliant with RFC 2440 such as PGP 2.6.x. Old OpenPGP clients are compliant with RFC 1991 only, not RFC 2440.
- **RSA Legacy Format:** EDGE gives you the option to generate RSA Legacy keys (see next chapter) which are compatible with older versions of OpenPGP. Old OpenPGP clients are compliant with RFC 1991 only, not RFC 2440. ***To generate an RSA Legacy key pair go to the next chapter of this manual.***

If no key type is specified, EDGE directly asks you to select the main name (User ID) of your key. By convention, a key name is formed by your real name and by your email address.

For instance, if your name is "Robert J. Smith" and if your email address is rsmith@company.com, your User ID could be:

```
"Robert J. Smith <rsmith@company.com>".
```

```
edge - Encrypted Data Gateway Engine  
Version 3.7  
Copyright (C) 2002-2007 Authora Inc. & Veridis SA  
All rights reserved.
```

```
Enter the name of the key:
```

EDGE asks you to enter the passphrase of your key. This passphrase is used to encrypt the secret components of your private key. This passphrase will be required each time you digitally sign data and each time you decrypt data encrypted for you.

It is very important to choose a passphrase that you will remember. Remember that this passphrase protects your private key, so it is very important to carefully choose it. A stronger passphrase contains letters (lower AND upper caps) and digits or punctuation marks. The longer the passphrase is, the better and more secure the passphrase.

```
edge - Encrypted Data Gateway Engine
Version 3.7
Copyright (C) 2002-2007 Authora Inc. & Veridis SA
All rights reserved.
```

```
Enter the name of the key: Alice <alice@authora.com>
```

```
Enter the passphrase (type ^D to cancel):
```

EDGE asks you to enter the same passphrase again.

```
edge - Encrypted Data Gateway Engine
Version 3.7
Copyright (C) 2002-2007 Authora Inc. & Veridis SA
All rights reserved.
```

```
Enter the name of the key: Alice <alice@authora.com>
```

```
Enter the passphrase (type ^D to cancel): *****
Enter same passphrase again (type ^D to cancel):
```

NOTE: EDGE asks you to enter the same passphrase twice to be sure you haven't misspelled it the first time you typed it. EDGE accepts the passphrase of the key when the passphrase and the confirmation are identical.

EDGE generates your new key pair and stores it automatically in your public and private keyring.

4.3.1 Specifying Key Type

EDGE allows you to generate different kinds of keys. To generate a DSS/DH key (default value), type :

```
edge --key-gen --key-type dss
```

To generate an RSA key type composed of a primary key and a subkey of the same size, type;

```
edge --key-gen --key-type rsa
```

If you plan to transact securely with people or organizations still using RSA Legacy keys you will need to generate an RSA Legacy key pair which is compatible with

older versions of PGP. An older OpenPGP client is only compliant with RFC 1991 and not with RFC 2440.

About RSA Legacy Keys: RSA Legacy keys allow only a primary key with no subkeys. This means that the same key is used to encrypt and to digitally sign.

To generate an RSA Legacy key, type:

```
edge --key-gen --key-type rsa-legacy
```

4.3.2 Specifying Key Size

EDGE allows you to generate keys of different sizes. A bigger size of key is a more secure key. Depending on the type of key you want to generate, EDGE allows you to specify sizes up to 4096 bits.

To specify the size of the key you want to generate, use the `--key-size` option:

```
edge --key-gen --key-size 4096
```

In the above example, EDGE generates a key pair of 4096 bits length.

For DSS/DH keys, the size of the primary key is always 1024 bits even if you specify a different size. The specified size is used only for the generation of the subkey.

4.3.3 Specifying a Validity Period

By default EDGE generates a key pair with an unlimited validity period. You can tell EDGE to generate a key pair for a certain number of days since the current date. To specify a validity period, use the `--expires-after` option when generating a key pair:

```
edge --key-gen --expires-after 365
```

In the above example, EDGE generates a new key pair valid for one year from the creation date.

4.3.4 Automating Key Pair Creation

EDGE allows you to fully automate the key pair creation process by providing more options for specifying the main User ID of the key and the passphrase of the new generated key pair:

```
edge --key-gen --userid <myNewKeyPair> --passphrase  
<myPassphrase>
```

The `--userid` option can be used to specify the main User ID of the key pair. If this option is used when generating a new key pair, EDGE doesn't ask you to enter the primary User ID of the key, and uses the value provided by the user.

The `--passphrase` option can be used to avoid the passphrase request. If this option is specified, EDGE doesn't ask the user to confirm the entered passphrase and uses the passphrase provided by the user through the `--passphrase` option.

The private key is encrypted using a passphrase. This passphrase is needed to decrypt a message encrypted with the public portion of a key pair and to sign a message using the private key. It is very important to choose a passphrase you can remember. Even if your private key is encrypted with a passphrase, never share or distribute your private key.

4.4 Generating a Signature-Only Key

EDGE allows you to generate a key pair that can only be used to digitally sign messages and to certify other keys. You will learn later in this section how to add a subkey to that kind of key.

To generate a key which is only able to digitally sign messages or to certify other keys, type:

```
edge --key-gen --sign-only
```

You cannot generate RSA Legacy key pairs that are only allowed to digitally sign message or to certify other keys.

4.5 Adding a Subkey to Your Key Pair

A subkey can be added to both RSA keys and DSS/DH keys (you cannot add a subkey to a RSA Legacy key).. A subkey is used to encrypt/decrypt data. The primary key is used to digitally sign data.

If a primary key contains more than one subkey, the most recently created subkey is used to encrypt data.

To generate a new subkey, type:

```
edge --key-gen --subkey
```

By default, EDGE uses the default key size. If you want to specify the size of the new subkey, use the `--key-size` option:

```
edge --key-gen --subkey --key-size <1024>
```

When the `--subkey` option is used, the `--userid` option can be used to specify the User ID of the primary key of which you want to add a subkey to:

```
edge --key-gen --subkey --key-size <1024> --userid <userID>
```

4.6 Sending Your Public Key

To be able to communicate securely with other people, you have to make your public key available.

You must first extract your public key from your public keyring. To do this, use:

```
edge --key-export <userID>
```

EDGE creates a file having the same name as the primary User ID of the key.

HP Tandem/Guardian: The name of the file is always truncated to 8 characters.

If more than one key matches the specified User ID, EDGE returns an error and invites you to use the `--multi` option if you want to export all keys matching the specified User ID.

```
edge --key-export <userID> --multi
```

EDGE creates a file having the same name as the primary User ID of the first key matching the specified User ID. All keys are stored in the same file.

HP Tandem/Guardian: The name of the file is always truncated to 8 characters.

If the specified destination denotes a directory, EDGE generates a single file for every exported key. Files are created in the specified directory.

EDGE allows you to use the `--output` option in order to specify the name of the file containing the keys you want to export:

```
edge --key-export <userID> --output <fileName>
```

Example:

```
edge --key-export Authora --output /usr/authora/keys/
```

The above example forces EDGE to export all keys having a User ID matching Authora. One file is created for each exported key. Files containing exported keys are created in `/usr/authora/keys/`.

You can now send this file to persons with whom you want to communicate or post it to a public keyserver.

Note: Never distribute your private key. Your private key must be used only by you and should be exported only for backup purposes.

If you want to extract your key as a text file, use the `--armor` option. Extracting a key as a text file allows you to copy the text block directly in an email or in any other text file.

By default, EDGE exports only the public portion of your key pair. If you also want to export your private key, use the `--with-private` option. By using this option, EDGE automatically generates an ASCII-armored file containing the public and the corresponding private key if it exists.

```
edge --key-export <userID> --with-private
```

Examples:

```
edge --key-export <myKey> --output <myKeyFile>
edge --key-export <myKey> --output <myKeyFile.txt> --armor
edge --key-export <myKey> --output <myKeyPair.txt>
--with-private
```

4.7 Adding Public Keys of Other Persons

To be able to encrypt data for other people you need to add their public keys to your public keyring.

Public keys can be found on public key servers or can be sent directly to you by the key's owner.

To add keys stored in a file to your public keyring, type:

```
edge --key-add <keyFile>
```

KeyFile is the name of the file containing the keys to add to your keyring.

EDGE displays information about the keys added to your keyring. If a key is already in your keyring, EDGE merges both keys - missing User IDs, subkeys, and signatures are added to the key already stored in your keyring.

When the operation is completed, EDGE displays the number of keys added and merged.

The key file can be a binary file or an ASCII-armored file; EDGE automatically recognizes it. If the key file contains more than one ASCII-armored block, EDGE displays keys found in each ASCII-armored block separately.

5 Managing keys

5.1 Adding Keys to Your Keyrings

EDGE allows you to add keys stored in a keyring file into your default public and/or private keyrings depending on the type of the key to add. Public keys are always added to the default public keyring.

```
edge --key-add <keyFile>
```

By default, EDGE only adds public keys to your default public keyring, ignoring private keys. To also add private keys, use the `--with-private` option:

```
edge --key-add <keyFile> --with-private
```

Private keys are added to the default private keyring and public keys to your default public keyring.

EDGE asks for confirmation before adding or merging a key to your keyrings.

To automate the key adding process by accepting all keys, use the `--force` option:

```
edge --key-add <keyFile> --force
```

Examples:

```
edge --key-add keys.asc
```

In the above example, EDGE adds public keys contained in the file named "keys.asc" to your default public keyring. Before adding a key to your keyring, EDGE prompts for a confirmation.

```
edge --key-add keys.asc --with-private
```

In the above example, EDGE adds both public and private keys contained in the file named "keys.asc". Public keys are added to your default public keyring and private keys to your default private keyring.

NOTE: If a key is already present in your keyring, EDGE merges both keys together. Missing User IDs, subkeys and signatures are added to the key already stored in your keyring.

You can also control the keys you want to add to your keyrings by using the Boolean options: `--addpublickeys` and `--addsecretkeys`.

To import only public keys, use the `--addpublickeys` option:

```
edge --key-add <keyFile> --addpublickeys on
```

When using `--addpublickeys`, EDGE adds only public keys contained in the specified file. Private keys are ignored and not added to your private keyring. This is the default.

To import only private keys, use `--addsecretkeys` option:

```
edge --key-add <keyFile> --add-secretkeys on --addpublickeys
off
```

When using `--addsecretkeys`, EDGE adds only private keys contained in the specified file. If `--addpublickeys` is turned off, public keys are ignored and not added to your public keyring.

NOTE: Using `--addpublickeys on --addsecretkeys on` is equivalent to the `--with-private` option.

To update your keyring without adding any new keys, use the `--mergeonly` option.

```
edge --key-add <keyFile> --mergeonly
```

When using `--mergeonly`, EDGE merges any keys contained in the specified file with keys already existing in your keyrings (both public and private).

By default, EDGE imports public keys and automatically merges public keys already stored in your default public keyring.

5.2 Displaying Your Keyring

EDGE provides multiple ways to display your keyring, from a simple list to a detailed view.

To display a list of your keys, use the `--key-list` option:

```
edge --key-list
```

If no User ID is specified, EDGE displays all keys contained in your default public keyring.

Example:

```
edge - Encrypted Data Gateway Engine
Version 3.7
Copyright (C) 2002–2007 Authora Inc. & Veridis SA
All rights reserved.
```

Alg	Type	Size	Flags	Key ID	User ID
RSA	pair	1024/1024	[--]	0xF9E31687	my rsa key
DSS	pair	2048/1024	[--]	0x34E7132A	my dss key

```
2 key(s) found
```


Column Name	Meaning
Alg	Indicates the public-key algorithm of the key. It can be RSA or DSS.
Type	Indicates the type of the key. Type can be "pub" if only the public part of the key is present in the keyring, "sec" if only the private part of the key is present in the keyring or "pair" if both the public and private part of the key are present in the keyring.
Size	Indicates the size of the key. It can have a value up to 4096 bits. Larger keys are more secure than smaller keys.
Flags	Indicates the status of the key. This field is divided in two columns. The first column indicates whether the key is revoked (R) or disabled (D). The value "-" indicates that the key is neither revoked nor disabled. The second column indicates whether the key is expired or not. The character "E" in that column indicates an expired key.
Key ID	Identifier of the key. This value is based on public components of the key. A public key has the same key ID as its corresponding private key.
User ID	Names of the key. A key can be identified by its User IDs or by its key ID. A key can have more than one name. The first name displayed is always the most recently created name.

To display only keys containing a particular text, type:

```
edge --key-list [userID1] [userID2] ...
```

EDGE displays only keys having a User ID matching one of the specified User ID.

Example:

```
edge --key-list rsa
```

```
edge - Encrypted Data Gateway Engine
Version 3.7
Copyright (C) 2002-2007 Authora Inc. & Veridis SA
All rights reserved.
```

```
Alg Type Size      Flags Key ID      User ID
```

```
RSA pair 1024/1024 [--] 0xF9E31687 my rsa key
1 key(s) found
```

To display all keys in a particular keyring, use the `--pubring` or `--secring` option:

```
edge --key-list --pubring <keyFile>
edge --key-list --secring <keyFile>
```

To display keys containing a particular text in a keyring other than the default public keyring, type:

```
edge --key-list [userID1] [userID2] ... --pubring <keyFile>
```

By default, EDGE sorts the keys by User ID. You can tell EDGE to sort keys using their User IDs, creation date, expiration date, key size, subkey size or key ID. To sort displayed keys using a different attribute, use the `--sort` option followed by the field you want to use to sort keys:

```
edge --key-list --sort [field]
```

You can specify one of the following values for the field you want to use to sort keys:

Variable	Definition
keysize	Keys are displayed and sorted by the size of the key.
subkeysize	Keys are displayed and sorted by the size of the subkey.
keyid	Keys are displayed and sorted by the key ID.
userid	Keys are displayed and sorted using the User ID of the key (default value).
creation	Keys are displayed and sorted by the creation date of the key.
expiration	Keys are displayed and sorted by the expiration date of the key, if any exist.

By default, EDGE displays key in ascending order. If you want to display keys in descending order, use the `--reverse` option:

```
edge --key-list --sort [field] --reverse
```

EDGE also allows you to display more details about the keys in your keyring. To display the fingerprint, expiration date and subkey information, use the

--key-detail option:

```
edge --key-detail [userID]
```

If no User ID is specified, EDGE displays information on the first key in your default public keyring.

Example:

```
edge --key-detail rsa

edge - Encrypted Data Gateway Engine
Version 3.7
Copyright (C) 2002-2007 Authora Inc. & Veridis SA
All rights reserved.

Primary User ID: my rsa key

      Long Key ID: Ox7AE829C1F9E31687
      Short Key ID: OxF9E31687
              Type: RSA Key Pair
              Size: 1024/1024
      Created: 2004/03/13
      Expires: Never
      Status:
```

Fingerprint:

```
319B FE32 D1EA 72D7 0A19 9A6B 7AE8 29C1 F9E3 1687
```

Subkeys:

Key ID	Valid From	Expires	Size	Status
0x713D2AD7	2004/03/13	Never	1024	

By default, EDGE displays the key fingerprint as a hexadecimal number. For easier reading and verification of the fingerprint, EDGE also allows you to choose to display the key fingerprint as a word list representation.

The --fingerprint-view option allows you to choose between different representations of the fingerprint. If you want to display the fingerprint of the key as a hexadecimal number, type:

```
edge --key-detail [userID] --fingerprint-view hex
```

To choose to display the finger print as a list of words, type:

```
edge --key-detail [userID] --fingerprint-view words
```

The --key-detail command displays only the first key matching a specified User ID or, if no User ID has been specified, the first key of the keyring. If you want to display detailed information for all matching keys, use the --multi option:

```
edge --key-detail [userID] --multi
```

When using the `--multi` option, more than one User ID can be specified. In that case, all keys matching any of the specified User IDs will be displayed.

```
edge --key-detail [userID1] [userID2] ... --multi
```

5.3 Status of Keys

A key can be revoked, disabled or expired. EDGE displays this information. When displaying keys using the `--key-list` command, the "Flags" column indicates the status of the key.

Flags are displayed using two characters. The first character in the column is used for revoked and disabled keys and the second character is used for expired keys.

A key is revoked when the first character of the Flags column is set to 'R'. For disabled keys, this character is set to 'D'. When a key is neither revoked or disabled, this character is set to '-'.

When a key is expired, the second character of the Flags column is set to 'E'. If not, it's set to '-'.

For example, the following key is neither revoked nor disabled nor expired:

```
Alg Type Size      Flags Key ID      User ID
RSA pair 1024/1024 [--]  0xF9E31687 my rsa key
```

The following key is revoked:

```
Alg Type Size      Flags Key ID      User ID
RSA pair 1024/1024 [R-]  0xF9E31687 my rsa key
```

The following one is disabled:

```
Alg Type Size      Flags Key ID      User ID
RSA pair 1024/1024 [D-]  0xF9E31687 my rsa key
```

And the following one is expired:

```
Alg Type Size      Flags Key ID      User ID
RSA pair 1024/1024 [-E]  0xF9E31687 my rsa key
```

When using the `--key-detail` command, EDGE also displays the status of keys:

For revoked keys: EDGE displays in red `*** KEY REVOKED ***`

For disabled keys: EDGE displays in red `*** KEY DISABLED ***`

For expired keys: EDGE displays in red `*** KEY EXPIRED ***`

Example:

```
edge --key-detail rsa
```

```
edge - Encrypted Data Gateway Engine
Version 3.7
Copyright (C) 2002-2007 Authora Inc. & Veridis SA
All rights reserved.
```

```
Primary User ID: my rsa key
```

```
    Long Key ID: Ox7AE829C1F9E31687
    Short Key ID: OxF9E31687
        Type: RSA Key Pair
        Size: 1024/1024
    Created: 2004/03/13
    Expires: Never
    Status: *** KEY REVOKED ***
```

```
Fingerprint:
```

```
    319B FE32 D1EA 72D7 0A19 9A6B 7AE8 29C1 F9E3 1687
```

```
Subkeys:
```

Key ID	Valid From	Expires	Size	Status
0x713D2AD7	2004/03/13	Never	1024	*** KEY REVOKED

5.4 Removing Keys

The `--key-remove` command allows you to remove keys from a keyring file.

```
edge --key-remove <userID>
```

EDGE looks in the default public and private keyrings for a key that matches the specified User ID. If no key can be found, an error is returned. If a key is found, EDGE asks you to confirm the removal.

If the associated private key must be deleted too, use the `--with-private` option:

```
edge --key-remove <userID> --with-private
```

Once a key has been found, EDGE exits.

To remove all keys matching a specified User ID, add the `--multi` option:

```
edge --key-remove <userID> --multi
```

To remove user interaction, add the `--force` option:

```
edge --key-remove <userID> --multi --force
```

In that case, all keys matching the specified User ID are removed without asking the user to confirm the removal.

WARNING: If the `--with-private` option is also used, both public and private keys will be removed without confirmation.

When using the `--multi` option, EDGE asks the user to confirm the removal for each key matching the specified User ID. If the user decides to not delete a particular key, EDGE exists.

NOTE: EDGE returns an error when the operation has been completed without removing any keys.

5.5 Certifying Keys

EDGE allows you to certify keys using the `--key-sign` command. Before certifying a public key, be sure that the key you want to certify belongs to the right person.

```
edge --key-sign <userID> --sign-with <signerID>
```

EDGE signs the specified User ID with the specified signer ID. The first key matching the specified User ID is searched in the default public keyring. To be able to certify a public key, you need to have access to the private key matching the specified signer ID. That private key is searched in the default private keyring.

NOTE: If no signer ID is specified, EDGE uses the key specified by the `DEFAULT-KEY` option. If that option has not been specified, EDGE uses the latest created private key.

EDGE exits with an error if no key matching the specified User ID can be found. By default, EDGE certifies only the first key matching the specified User ID. If you want to certify all keys matching the specified User ID, add the `--multi` option:

```
edge --key-sign <userID> --sign-with <signerID> --multi
```

The passphrase request can be removed by using the `--passphrase` option:

```
edge --key-sign <userID> --sign-with <signerID> --multi
--passphrase <passPhrase>
```

By default, EDGE generates a signature with no expiration. If you want to add an expiration date to the signature, use the `--expires-after` option:

```
edge --key-sign <userID> --sign-with <signerID> --multi
--passphrase <passPhrase> --expires-after <numberOfDays>
```

EDGE also allows you to specify the signature type. It can be "local" or "exportable". A local signature is not exported by EDGE when using the `--key-export` command. If you want to export the signature you are adding, set the signature type to "exportable".

```
edge --key-sign <userID> --sign-with <signerID>
```

```
--sig-type <exportable | local>
```

NOTE: By default, EDGE generates exportable signatures.

Examples:

```
edge --key-sign <hisKey> --sign-with <myKey>
edge --key-sign <hisKey> --sign-with <myKey> --passphrase
<myPassphrase>
edge --key-sign <herKey> --sign-with <myKey> --expires-after
[365]
edge --key-sign <herKey> --sign-with <myKey> --sig-type
[local]
edge --key-sign <herKey> --sign-with <myKey> --sig-type
[exportable]
```

5.6 Exporting Keys

To distribute a key or keep a copy of a key, that key needs to be exported. To export a key, use the `--key-export` (legacy mode `-kx`) command:

```
edge --key-export <user ID>
```

EDGE scans the default public keyring and copies the key matching the specified User ID to it. If you want to export all keys matching the specified User ID, add the `--multi` option:

```
edge --key-export <user ID> --multi
```

NOTE: If the `--multi` option is not specified, EDGE exits with an error if more than one key matching the specified User ID has been found in the default public keyring. In that case, no file is created.

EDGE creates a file using the primary name of the first key matching the specified User ID.

If you want to export keys as an ASCII-armored file, use the `--armor` option. In that case EDGE creates a file with the extension `".asc"`. Otherwise, the file is created with an extension `".pgp"`.

```
edge --key-export <user ID> --armor on
```

By default, only the public keys are exported. If you also want to export private keys, add the `--with-private` option:

```
edge --key-export <user ID> --with-private
```

NOTE: When `--with-private` option is used, EDGE generates an ASCII-Armored file automatically. This file contains two different sections: one containing the private keys and the second one containing the public keys.

WARNING: Never share or distribute your private key.

You can use the `--output` option if you prefer to specify the name and the location of the destination file:

```
edge --key-export <user ID> --output [pathname]
```

If the `--output` option is used to denote an output file, EDGE generates a single file containing all keys. If the specified destination denotes a directory, EDGE generates a file for each exported key. These files are created in the specified destination directory.

EDGE also provides two additional options allowing you to choose the keys you want to export.

The `--exportpublic` option allows you to choose if the public keys must be exported or not. By default, EDGE exports public keys.

The `--exportsecret` option allows you to choose if the private keys must be exported or not. By default, EDGE doesn't export private keys.

The following example exports the public and the private key in a same destination file:

```
edge --key-export <aKey> --exportpublic on --exportsecret on
```

The above example is equivalent to:

```
edge --key-export <aKey> --with-private
```

The following example exports only the private key matching the specified User ID:

```
edge --key-export <aKey> --exportpublic off --exportsecret on
```

NOTE 1: The `--exportpublic` and `--exportsecret` options can be defined directly in the configuration file allowing you to specify the portions of your key pairs you want to export for every `--key-export` operation.

NOTE 2: When `--exportpublic` and `--exportsecret` options are both ON, an armored file is automatically generated.

NOTE 3: If `--exportpublic` and `--exportsecret` are both OFF, EDGE returns an error.

Examples:

```
edge --key-export <myKey>
```



```

edge --key-export <myKey> --armor
edge --key-export <myKey> --with-private
edge --key-export <myKey> --multi
edge --key-export <myKey> --with-private -multi
edge --key-export <myKey> --with-private -exportpublic off
edge --key-export <myKey> --exportpublic off --exportsecret on
edge --key-export <myKey> --exportsecret off
edge --key-export <myKey> --output [myKeyFile.pgp]
edge --key-export <myKey> --output [myKeyFile.txt] --armor
edge --key-export <myKey> --multi --output [/home/keys/] --armor
edge --key-export <myKey> --multi --output [/home/keys/]
--with-private

```

5.7 Changing the Passphrase of a Private Key

The `--key-edit` command lets you change the passphrase used to encrypt your private keys.

```

edge --key-edit <userID> --change-passphrase [oldPassphrase]
--new-passphrase [new passphrase]

```

EDGE searches your private and public keyrings for the specified User ID. Both private and public key are required in order to change the passphrase of the private key. Once a passphrase has been changed, the old passphrase becomes unusable for that private key.

If the old passphrase is incorrect, EDGE prompts you to enter the correct passphrase in order to decrypt the private key and to encrypt it using the new specified passphrase.

If the specified User ID cannot be found in your private keyring, EDGE exists with an error.

Example:

```

edge --key-edit <myKey> --change-passphrase
[myOldPassphrase] --new-passphrase [myNewPassphrase]

```

5.8 Adding a User ID to Your Key

OpenPGP keys can have more than one User ID. When you generate a new key pair, you specify the primary User ID of the key. Later you can add new User IDs to your key. Adding a new User ID to your key can be useful if you are known by different email addresses.

To add a new User ID to your key pair, use the `--add-userid` option:

```

edge --key-edit <userID> --add-userid [newUserID]

```

EDGE searches your default public and private keyrings for the specified User ID. Both the private and the public keys are needed.

When adding a new User ID to your public key, EDGE automatically certifies this new User ID using the corresponding private key.

If you want to automate the passphrase request, use the `--passphrase` option:

```
edge --key-edit <user ID> --add-userid [newUserID]
--passphrase [passphrase]
```

Examples:

```
edge --key-edit <myKey> --add-userid [myOtherUserID]
```

```
edge --key-edit <myKey> --add-userid [myOtherUserID]
--passphrase [myPassphrase]
```

NOTE: If the new passphrase is empty, the private key will not be encrypted and will be saved un-encrypted into your default private keyring.

It is highly recommended to always protect your private key with a passphrase.

5.9 Disabling a Key

EDGE allows you to disable a key by using the `--disable` option. **A disabled key cannot be used to encrypt data but can still be used to verify a signature.** Disabling a key is a useful method for keeping an outdated key that will be used to verify signatures only.

```
edge --key-edit <userID> --disable
```

Before disabling the key matching the specified User ID, EDGE prompts you for confirmation. To fully automate the key disabling process, use the `--force` option:

```
edge --key-edit <userID> --disable --force
```

Examples:

```
edge --key-edit <myKey> --disable
edge --key-edit <myKey> --disable --force
```

5.10 Enabling a Key

EDGE allows you to enable a previously disabled key. **An enabled key can be used to encrypt data and verify signatures.** To enable a previously disabled key, use the `--enable` option.

```
edge --key-edit <userID> --enable
```

Before enabling the key matching the specified User ID, EDGE prompts you for confirmation. To fully automate the key enabling process, use the `--force` option:

```
edge --key-edit <userID> --enable --force
```

Examples:

```
edge --key-edit <myKey> --enable
edge --key-edit <myKey> --enable --force
```

5.11 Revoking a Key

If your private key has been compromised, you must revoke it and make the new revoked public key available to everyone with whom you communicate.

A revoked key cannot be used to encrypt data. A revoked key can still be used to verify signatures made before the revocation date.

NOTE: Once a key has been revoked, it cannot be “un-revoked”. Revoke a key only if the private key has been compromised.

To revoke a key, use the `--revoke` option:

```
edge --key-edit <userID> --revoke
```

To revoke a key, the specified User ID must denote a keypair. When EDGE asks you for confirmation to revoke the key, Type “y” for “YES”. EDGE revokes the key and updates your public and private keyrings.

To fully automate the key revocation process, you need to use the `--passphrase` and `--force` options. The passphrase is used to decrypt the private key and add a revocation signature to the public key. This revocation signature is made using the corresponding private key.

```
edge --key-edit <userID> --revoke --passphrase [myPassphrase]
--force
```

Examples:

```
edge --key-edit <myKey> --revoke
edge --key-edit <myKey> --revoke --passphrase <myPassphrase>
edge --key-edit <myKey> --revoke --passphrase <myPassphrase>
```

```
--force
```

5.12 Removing a User ID

EDGE allows you to remove a User ID from an existing public and/or private key. To do that, use the `--remove-userid` option:

```
edge --key-edit <userID> --remove-userid [userIDforRemoval]
```

EDGE searches the private and public keyring for a key matching the specified User ID. If no key can be found, EDGE exits with an error. If the key contains only one User ID, EDGE also exits with an error.

If a key pair is found, the specified User ID is removed from both private and public key.

Example:

```
edge --key-edit <myKey> --remove-userid [userIDforRemoval]
```

5.13 Removing a Sub-Key

You can decide to remove outdated subkeys from keys. Using the `--remove-subkey` option provides a way to remove subkeys from keys stored in your keyrings.

```
edge --key-edit <user ID> --remove-subkey [subkeyID]
```

EDGE searches the private and public keyring for a key matching the specified User ID. If no key can be found, EDGE exits with an error.

If a key pair is found, the specified subkey is removed from both private and public key.

Example:

```
edge --key-edit <myKey> --remove-subkey [0xAB45D71F]
```

5.14 Removing a Signature

EDGE allows you to remove signature made on your key pair by using the `--remove-sig` option.

```
edge --key-edit <userID> --remove-sig <signature>
```

EDGE searches the private and public keyring for a key matching the specified User ID. If no key can be found, EDGE exits with an error.

If a key pair is found, the specified signature is removed from both private and public key. EDGE searches the User IDs for a signature granted by the specified signer. If no signature can be found, an error is returned.

Examples:

```
edge --key-edit <myKey> --remove-sig [aSignature]
edge --key-edit <myKey> --remove-sig [0xAEF302E8]
```

5.15 Revoking a Signature

EDGE also allows you to revoke a signature that you made on someone else's User ID by using the `--revoke-sig` option.

```
edge --key-edit <userID> --revoke-sig [signature]
```

EDGE searches the private and public keyring for a key matching the specified User ID. If no key can be found, EDGE exits with an error.

If a key pair is found, the specified signature is revoked for both private and public key. EDGE searches User IDs for a signature granted by the specified signer. If no signature can be found, an error is returned.

NOTE: To revoke a signature, you must have access to the private key that originally granted the signature to that User ID. Revoking a signature is adding a revocation signature on an existing signature.

Examples:

```
edge --key-edit <myKey> --revoke-sig [aSignature]
edge --key-edit <myKey> --revoke-sig [0xAEF302E8]
```

6 Encrypting and Signing

6.2 Encrypting Data with a Public Key

Data can be encrypted with one or more public keys. The recipient of the encrypted data must have the private key corresponding to the public key used to encrypt the data. Without the private key, the encrypted data cannot be decrypted.

To encrypt to a single public key:

```
edge --encrypt <clearFile> --user <userID1>
```

To encrypt to multiple public keys:

```
edge --encrypt <clearFile> --user <userID1> [--user <userID2>
...
--user <userID3>]
```

The file <clearFile> will be encrypted with the keys corresponding to the specified User IDs. The keys corresponding to the User IDs are searched for in the default public keyring and EDGE will create an encrypted file named "<clearFile>.pgp".

To produce an ASCII-armored file, add the --armor option:

```
edge --encrypt <clearFile> --user <userID1> [--user <userID2>
...
--user <userID3>] --armor
```

If the --armor option is used, EDGE produces a file named "<clearFile>.asc"

NOTE: At least one User ID must be specified.

EDGE returns an error if it cannot find a key containing a User ID matching one of the specified User IDs,

If no input is specified, EDGE uses the standard input and writes encrypted data to the standard output.

EDGE also allows you to specify the name of the destination file or the directory where EDGE must create the destination file.

To specify the output directory or the output file, use the --output option:

```
edge --encrypt <clearFile> --user <userID1> [--user <userID2>
...
--user <userID3>] --output [output directory or file]
```

NOTE: If a file name is specified, EDGE doesn't add any extension to the file name.

Examples:

```
edge --encrypt <file.txt> --user <john>

edge --encrypt <file.txt> --user <john> --user <robert>
--armor

edge --encrypt <file.txt> --user <john> --user <robert>
--armor --text

edge --encrypt <file.txt> --user <john> --output
[/home/user/]

edge --encrypt --user <john>

edge --encrypt <file.txt> --user <john> --output
[/home/user/encrypted.txt] --armor
```

6.1.1 Allow Missing Signature

In some cases, a public key you receive may have a bad signature or no signature at all. By default, EDGE will not let you encrypt with these kinds of keys. In order to bypass this protection and use a key with a bad signature or no signature at all, you must tell EDGE to allow it by using the `--allow-missing-sign` option:

```
edge --encrypt <file.txt> --user <john> --allow-missing-sign
```

6.2 Encrypting Data with a Password

Data can also be encrypted with a password instead of a public key. The recipient of the encrypted data must know the password to be able to decrypt this encrypted data.

To encrypt data with a password, type:

```
edge --encrypt --conventional <clearFile>
```

EDGE will prompt you to create a password and the file `<clearFile>` will be encrypted using the password you specify. The encrypted file will be named "`<clearFile>.pgp`".

If no input file is specified, EDGE uses the standard input as the source file and writes encrypted data to the standard output.

The passphrase can also be specified directly at the command-line by using the

`--conventional-passphrase` option:

```
edge --encrypt --conventional <clearFile>
--conventional-passphrase [passphrase]
```

To produce an ASCII-armored file, add the `--armor` option:

```
edge --encrypt --conventional <clearFile> --armor
```

EDGE will create a file named "`<clearFile>.asc`" encrypted with the specified password.

Examples:

```
edge --encrypt --conventional <file.txt>
--conventional-passphrase [passphrase]
```

```
edge --encrypt --conventional <file.txt> --conventional-
passphrase [passphrase] --armor
```

```
edge --encrypt --conventional <file.txt> --conventional-
passphrase [passphrase] --armor --text
```

```
edge --encrypt --conventional --conventional-passphrase
[passphrase] --armor
```

If the password is not specified at the command line, EDGE asks you to enter it. To cancel the operation, press `^D` ([CTRL] + D).

NOTE: If the BATCHMODE is enabled, EDGE will not ask you to enter a password; you need to specify it at the command line. If no password has been specified at the command-line, the operation will be cancelled.

6.3 Signing Data

By digitally signing data you allow the recipient to verify the integrity of the data. By verifying the signature on this data, the recipient can be sure that it has not been altered during its transport. The recipient of the signed data must have the public key corresponding to the private key used to sign the data. Without the public key, the signed data cannot be verified. Because only one person holds the private key, the recipient of the signed data can be sure that the received data originated from the owner of the private key. The owner of the private key is responsible for sending the public key to the recipient.

```
edge --sign <clearFile> --sign-with <userID>
```

The file `<clearFile>` will be signed with the private key corresponding to the specified User ID.

If the private key to use to sign data is not specified, EDGE tries to locate it using the following scheme:

- EDGE tries to find the key specified by the DEFAULT-KEY option.
- If DEFAULT-KEY has not been defined, EDGE searches your default private keyring for the latest private key you have created.

The DEFAULT-KEY option can be specified directly in the configuration file or at the command-line. The two following commands have the same effect:

```
edge --sign <clearFile> --sign-with <userID>
edge --sign <clearFile> --default-key <userID>
```

Because the private key is used to digitally sign data, EDGE prompts for the passphrase of the private key to unlock it and to sign the data. The passphrase of the private key can also be specified at the command line by using the `--passphrase` option:

```
edge --sign <clearFile> --sign-with <userID>
--passphrase [passphrase]
```

By default, EDGE creates a signed file named "`<clearFile>.pgp`".

To produce an ASCII-armored file, add the `--armor` option:

```
edge --sign <clearFile> --sign-with <userID> --armor
```

In that case, EDGE creates a signed file named "`<clearFile>.asc`".

By digitally signing data, EDGE produces a file containing the data and the signature on that data. By verifying the signature, the recipient can verify the original data.

See the Detached Signature section to learn how to produce detached signatures, which allow you to separate the signature from the signed data.

If the input file is not specified, EDGE uses the standard input and writes signed data to the standard output.

The `--output` option can be used to specify the destination file or the directory where EDGE must create the destination file.

If the specified output is a file, no additional extension is added to the specified name.

```
edge --sign <clearFile> --sign-with <userID>
--output [directory or filename]
```

Examples:

```
edge --sign <file.txt> --sign-with <john> --passphrase
[passphrase]
```

```
edge --sign <file.txt> --sign-with <john> --passphrase
[passphrase] --armor
```

```
edge --sign <file.txt> --sign-with <john> --passphrase
[passphrase] --armor --text
```

```
edge --sign <file.txt> --sign-with <john> --passphrase
[passphrase] --armor --text --output
[/home/user/signedFile.txt]
```

```
edge --sign <file.txt> --sign-with <john> --passphrase
[passphrase] --armor --text --output [/home/user/]
```

6.3.1 Signature Version

By default RSA-Legacy keys create Version 3 signatures and RSA and DSS/DH keys create Version 4 signatures. To force EDGE to create a Version 3 signature on a RSA or DSS/DH key, use the `--force-v3` option:

```
edge --sign file.txt --sign-with john --force-v3
```

6.4 Detached Signature

EDGE can produce detached signatures. This means that the signed data is stored separately from the signature. This allows you to keep the original document (the signed data) in the same format and to store the signature separately, even in another location or on a server. This option can be used to sign all outgoing emails while storing the signature on a server or to use two separate channels for sending the data and the signature on it. The integrity of the original data can always be verified without changing the format of the original data and without storing the data twice.

This option is useful when the signature and the data need to be stored separately or when the signature and the signed data follow different paths to the recipient.

Using a separate signature allows the recipient to open the signed data even if he has no OpenPGP client to verify the signature.

To create a detached signature, add the `--detached` option:

```
edge --sign <clearFile> --sign-with <userID> --detached
```

The file `<clearFile>` will be signed with the private key corresponding to the specified User ID.

EDGE will create a signed file named "`<clearFile>.sig`" containing only the signature and not the signed data.

To produce an ASCII-armored signature, add the `--armor` option:

```
edge --sign <clearFile> --sign-with <userID> --detached --armor
```

Examples:

```
edge --sign <file.txt> --sign-with <john> --passphrase
[my passphrase] --detached
```

```
edge --sign <file.txt> --sign-with <john> --passphrase
[my passphrase] --armor --detached
```

```
edge --sign <file.txt> --sign-with <john> --passphrase
[my passphrase] --armor --text --detached
```

```
edge --sign <file.txt> --sign-with <john> --passphrase
[my passphrase] --armor --text --output
[/home/user/signature.txt] --detached
```

```
edge --sign <file.txt> --sign-with <john> --passphrase
[my passphrase] --armor --text --output
[/home/user/] --detached
```

6.5 Clear-Signed Data

EDGE allows you to create clear-signed data. Clear-signed data is data where the signed text is still readable. This is useful if you want to post a message to a mailing list. The signed text is preceded and followed by a special header and footer. This kind of message looks like:

```
-----BEGIN PGP SIGNED MESSAGE-----

The Signed Text

-----BEGIN PGP SIGNATURE-----
iQCVAwUBPtvgWegBJJ2jQbYlAQHPwQP+MJzCnBS0FFlYoM+ilix2DnGbtrt38lO
i
WytWSsLI8Bi65SAx2phy+XvYmNgClzr6Cmp0660+v1uULgwtzrGKPMva2x9X4GT
D
EFits0V059WQ+zP3M51URWtQUq7aBfJSbYOCVpKwUQ3VGgr3qG7v0eLLPBCDiB1
A
v9wiGxyqfSo=
=1xfu
-----END PGP SIGNATURE-----
```

To create this kind of message, use the `--sign` command with the `--clearsig`, `--armor` and `--text` options.

```
edge --sign <clearFile> --sign-with <userID> --clearsig --text
--armor
```

NOTE 1: Only text data can be clear-signed.

NOTE 2: the `--clear-sig` option and the `--detached` option cannot be used together.

You can also set the `CLEARSIG`, `TEXTMODE` and `ARMOR` options on directly in your configuration file. In that case, EDGE will always produce clear-signed messages.

Examples:

```
edge --sign <file.txt> --sign-with <john> --passphrase
[my passphrase] --armor --text --output
[/home/user/signedFile.txt] --clearsig
```

```
edge --sign <file.txt> --sign-with <john> --passphrase
[my passphrase] --armor --text --output
[/home/user/] --clearsig
```

6.6 Encrypting and Signing Data

To sign data and then encrypt the signed data, you can combine the encryption option and the signature option:

```
edge --encrypt --sign <clearFile> --user <userID> --sign-with
<userIDforSig>
```

NOTE: When encrypting and signing a file, the `--clearsig` and `--detached` options cannot be used.

To generate an ASCII-armored file, add the `--armor` option:

```
edge --encrypt --sign <clearFile> --user <userID1>
[--user <userID2> [... --user <userID3>]] [--sign-with
<userIDforSig>] --armor
```

To tell EDGE the input data must be considered as text, use the `--text` option:

```
edge --encrypt --sign <clearFile> --user <userID1>
[--user <userID2> [... --user <userID3>]] [--sign-with
<userIDforSig>] --armor --text
```

Examples:

```
edge --encrypt --sign <file.txt> --user <robert> --user
<john> --passphrase [my passphrase]
```

```
edge --sign <file.txt> --encrypt --user <robert> --user
<john> --passphrase [my passphrase] --armor
```

```
edge --sign --encrypt <file.txt> --user <john> --sign-with
[myKey] --passphrase [my passphrase] --armor --text
```

```

--output [/home/user/output.txt]

edge --sign --encrypt <file.txt> --user <john> --user
<robert> --sign-with [myKey] --passphrase [mypassphrase]
--armor --text --output [/home/user/output.txt]

edge --sign --encrypt <file.txt> --user <john> --sign-with
[myKey] --passphrase [mypassphrase] --armor --text--output
[/home/user/]

```

6.7 Decrypting Data

To decrypt encrypted and/or signed data, just type:

```
edge --decrypt <cipherFile>
```

EDGE automatically decrypts the data and/or verifies the signature and produces a clear text file.

If a passphrase is needed to decrypt the data, it asks you to enter the passphrase for the required private key or to decrypt a message encrypted with a password.

To avoid the passphrase request, you can use the `--passphrase` option.

```
edge <cipherFile> --passphrase [passphrase]
```

EDGE produces a file named as the input file without its last extension. If that file already exists, EDGE asks you to confirm the file deletion. If you choose file deletion, the same-named file will be overwritten with the decrypted file.

You can specify the destination file or the destination directory directly at the command-line by using the `--output` option:

```
edge <cipherFile> --output [clearfile or output directory]
```

If the specified file is a separate signature file, EDGE tries to locate a signature file using the name of the input file without its last extension. If this file doesn't exist, EDGE asks you to specify the file to use to verify the signature.

NOTE: EDGE does not produce an output file if the input file is a separate signature or if the input file is a clear-text signature data.

To display the clear text on the screen, use the `--secure-viewer` option:

```
edge <cipherFile> --secure-viewer
```

When encrypting or signing a file, EDGE stores the name of the file in the signed and/or encrypted file. To restore the original file name, use the `--preserve-name` option:

```
edge <cipherFile> --preserve-name
```

EDGE tries to use the name saved inside the cipher file. If a file with that name already exists, EDGE asks you to confirm its deletion.

To decrypt a file without verifying the signature and without decompressing the file, use the `--decryptonly` option.

```
edge <cipherFile> --decryptonly
```

If the file is encrypted, EDGE only decrypts the file. If the file is not encrypted, the file is decompressed and the signature (if any) is verified.

If the file is encrypted, EDGE produces a compressed file holding a signature (if any). This file can be decrypted by EDGE.

7 Advanced Options

7.1 Specifying Input File Types

By default, EDGE treats an input file as a binary file. You can inform EDGE that the input file is a text file. In that case, it will be converted into canonical form. During decryption, the file will be converted into a text file according to the platform specification. Depending on the destination platform, line endings change and EDGE converts line endings from the canonical form to the line endings of the destination platform.

To specify the input file as a text file, use the `--text` option.

HP Tandem/Guardian: The `--text` option must be used for edit files (code 101).

Examples:

```
edge --encrypt <file.txt> --user <john> --text
edge --sign <file.txt> --sign-with [myKey] --text
```

With the above option, EDGE treats the input file as a text file.

NOTE: A text file is an ASCII text file. A Microsoft Word document is NOT an ASCII text file and must be considered a binary file. An ASCII text file is a file generated by NotePad on Windows or vi on UNIX, for instance.

7.2 Specifying Output File Types

EDGE can work with binary and text files. In certain cases, you will prefer to generate files containing only printable ASCII characters.

EDGE supports an ASCII-armored format. This format is similar to Base64 format, but it also contains a checksum on data. This checksum allows you to check the correct transfer of the data. An ASCII-armored file contains only 7 bits characters.

To enable this feature, use the `--armor` option.

Examples:

```
edge --encrypt <file.txt> --user <john> --armor
edge --sign <file.txt> --sign-with [myKey] --armor
```

Files generated with the ASCII-armored option are bigger than their binary representation. Because EDGE also compresses data before encryption, the size of the output data is, most of the time, smaller than the original data.

7.3 Specifying Output File or Directory

EDGE creates the destination file name automatically and places it in the same directory as the input file. EDGE also allows you to specify the destination file location and/or its name.

To specify only the destination directory, use:

```
edge ... --output [destinationDir]
```

The same syntax can be used to specify the complete path of the file including the destination file name:

```
edge ... --output [destinationPath]
```

- If only the destination directory is specified, EDGE creates the destination file name automatically based on the input file name and creates the destination file in the specified directory.
- If the specified destination path includes the file name, EDGE uses it.
- If only a file name is specified, the destination file is created in the current working directory.

Examples:

```
edge --encrypt --armor <clearfile> --user <userID> --output
[/home/encrypted/]
```


In the above example, EDGE is forced to use a particular destination directory. EDGE automatically generates the destination file name and places it in the specified directory.

The following example forces EDGE to use a particular location and name for the destination file:

```
edge --encrypt --armor <clearfile> --user <userID>
--output [/home/encrypted/anEncryptedFile.asc]
```

Specifying the output directory can be useful when writing scripts that ignore the input file name. In that case, only the destination directory is relevant, not the file name.

The following Windows script decrypts all ".asc" files from a directory and places the resulting file in a particular directory:

```
for %%b in (*.asc) do edge --decrypt %%b
--output "C:\DecryptedFiles" --passphrase "passphrase" --force
--batchmode
```

The following example also securely deletes the encrypted file if the decryption process ends with no error:

```
for %%b in (*.asc) do edge --decrypt %%b
--output "C:\DecryptedFiles" --wipe --passphrase "passphrase"
--force --batchmode
```

7.4 Filter Mode

In most commands, EDGE can use the standard input as the source file and write the data to the standard output. On a UNIX machine, this option is useful to use EDGE as a pipe.

For encryption/signature and decryption commands, the standard input is used as source file if no file is specified.

Example:

```
tar /home/aDirectory/ |
edge --encrypt --user <userID> --noout > encrypted.asc
```

The above example encrypts the content of a directory into a unique named file.

7.5 Redirecting Output to the Screen

By using the `--secure-viewer` option when decrypting data, EDGE prints the result to the screen instead of creating a disk file.

The output is displayed by page. EDGE waits until you press a key to show the next page.

By pressing “q” the display is aborted;
By pressing “[ENTER]” the next line is displayed;
By pressing “[SPACE]” the next page is displayed.

Example:

```
edge --decrypt <file.txt.pgp> --secure-viewer
```

NOTE: If BATCHMODE setting is on, the output is displayed without asking you to press a key to continue.

7.6 Redirecting Output and Error Messages (UNIX)

EDGE supports the standard UNIX output redirection. EDGE displays all error messages to `stderr` (2) and displays other output to `stdout` (1).

To redirect error messages to a file and discard other output, use:

```
edge ... 2> [filename] > /dev/null
```

Where `filename` is the name of the file to write error messages to.

The default error stream can also be specified by using the `ERRORFD` setting.

```
edge ... --errorfd 1
```

The above example redirects error messages to `stdout`.

7.7 Removing User Intervention

A command-line application is used most of the time to automate processes. EDGE offers additional options enabling you to completely automate a process, without any user intervention.

7.7.1 BATCHMODE

This option allows you to remove unnecessary questions. This allows you to fully automate your process.

By using this option, EDGE automatically answers “**NO**” to all questions. To answer “**YES**” to all questions, use the `FORCE` setting (see below).

Example:

```
edge --encrypt <file.txt> --user <userID> --batchmode on
```

In this example, EDGE will fail if a file named "file.txt.pgp" already exists.

NOTE: If this option is specified during decryption, EDGE returns 1 if the file is not signed and 0 if the file is signed. If an error occurred during the decryption process, EDGE returns an error. Errors have a value less than 0.

7.7.2 FORCE

This option allows you to remove prompts for more information and to therefore fully automate your process by answering "YES" to all questions.

Example:

```
edge --encrypt <file.txt> --user <userID> --force on
```

In this example, if a file named "file.txt.pgp" already exists, EDGE will **NOT** fail it will automatically replace the existing file named "file.txt.pgp".

7.7.3 INTERACTIVE

If this setting is ON, EDGE requests a confirmation for every key added to your keyring.

If you want to be able to add keys to your keyrings without user intervention, you must turn this setting off by changing the configuration file or by specifying this setting every time you add a key to your keyrings.

Example:

```
edge --key-add <keyringfile.asc> --interactive off
```

7.8 Specifying a Passphrase

EDGE allows you to specify one or more passphrases directly at the command-line allowing you to fully automate your process. If a passphrase is requested and if no passphrase has been specified at the command-line, EDGE fails if BATCHMODE option is set to ON. If BATCHMODE option is set to OFF, EDGE asks you to enter a passphrase.

7.8.1 --passphrase Option

Operations like signing data or decrypting data require unlocking your private key. A private key is encrypted by a passphrase. This passphrase can be specified at the command line by using the `--passphrase` option followed by the passphrase.

Using this option doesn't unlock your private key permanently. Your private key is only unlocked during the signature/decryption process.

The private keyring is not changed when the private key needs to be unlocked.

Example:

```
edge --decrypt <file.txt.pgp> --passphrase [myPassphrase]
```

EDGE decrypts the specified file and uses the specified passphrase if needed to unlock a private key or to decrypt symmetrically encrypted data.

Multiple passphrases can be specified by using the `--passphrase` option for every passphrase you want to specify. EDGE tries to use the first passphrase specified. If this passphrase cannot be used to unlock a key, the second one is used.

Once a passphrase has been used, it is automatically removed from the list of available passphrases.

Example:

```
edge --decrypt <file.txt.pgp> --passphrase [myPassphrase]  
--passphrase [passphrase2] --passphrase [passphrase3]
```

7.8.2 PGPPASS Option

Operations like signing data or decrypting data require unlocking your private key. A private key is encrypted by a passphrase. This passphrase can be specified at the command line by using the PGPPASS environment variable.

Using this option does not unlock your private key permanently. Your private key is only unlocked during the signature/decryption process. The private keyring is not changed when the private key is unlocked.

If this environment variable is defined, EDGE uses the value of this variable as the first passphrase to try to unlock a key.

7.8.3 Passphrase File

The passphrase can also be stored in a text file. To use this feature, simply create a text file containing your passphrase and use the `--passphrase-file` option to specify the passphrase file.

Example:

```
edge --decrypt <file.txt.pgp> --passphrase-file  
[myPassphrase.txt]
```

In the above example, EDGE reads the passphrase from the specified passphrase file. If the passphrase is wrong, EDGE simply prompts for the correct passphrase.

7.8.4 Encrypted Passphrase File

The passphrase file can also be encrypted using data stored in the license file. This means that the passphrase file can only be used with that particular license file. If the license file changes, the passphrase file becomes invalid.

This option allows you to hide the passphrase from the people supporting the system. The encrypted passphrase file is created by an Administrator.

To create an encrypted passphrase file, use the `--create-enc-passfile` option:

```
edge --create-enc-passfile [myEncryptedPassfile] --passphrase
<myPassphrase>
```

In the above example, EDGE creates an encrypted file containing the passphrase `<myPassphrase>`. This encrypted passphrase file is linked to the license file used by EDGE.

To use this encrypted passphrase file in a script, use the `--enc-passphrase-file` option:

Example:

```
edge --decrypt <file.txt.pgp> --enc-passphrase-file
[myEncryptedPassFile]
```

7.9 Encrypting “For Your Eyes Only”

By using the `--secure-viewer` option when encrypting a file, additional information is added to the destination file indicating to the recipient that the file must be decrypted and displayed only using a secure viewer and the file should not be saved to disk.

Please note that when using this option, you cannot be sure that the decrypted file will never be saved on disk.

Example:

```
edge --encrypt <file.txt> --user <userID> --secure-viewer
```

When decrypting the file, EDGE asks you if you want to view the file now. If you choose to view the file now, EDGE displays it on the screen and clears the screen when the file has been completely displayed.

7.10 Generating a Self-Decrypting Archive (SDA)

EDGE allows the generation of Self-Decrypting Archives (SDA). An SDA is a Windows application containing encrypted files.

EDGE can generate an SDA for a file or for a complete hierarchy of directories.

When the SDA is launched, EDGE asks the user to select a destination directory where the decrypted files will be restored and to enter a password. If this password is correct, EDGE decrypts all embedded files and, if directories are included, restores the complete hierarchy of directories.

If a file already exists at the same location, the SDA asks the user to select another location or the existing file must be overwritten.

No additional application is needed to decrypt files embedded in an SDA.

Files are encrypted using a password. To generate an SDA, you have to use the `--sda` option in addition to the `--encrypt --conventional` command.

Examples:

```
edge --encrypt --conventional <file.txt> --sda on

edge --encrypt --conventional <file.txt> --passphrase
[myPassword] --sda on

edge --encrypt --conventional <aDirectory> --sda on
```

The input can either be a file or a directory. If a directory is specified, EDGE creates an SDA file containing all files and sub-directories contained in the specified directory. When the SDA is used to restore the encrypted files, the whole hierarchy is restored.

By default EDGE uses a file named "SDA.bin". This file is a stub application. You can specify a different stub by using the `--sda-stub` option.

EDGE comes with two different stubs. The default stub creates a Windows application with a graphical user interface and the second stub creates a Windows command-line application allowing the recipient to automate the decryption of the encrypted archive.

To generate a Windows command-line application, type:

```
edge --encrypt -conventional <aDirectory> --sda on --sda-stub  
[sda_cmdline.bin]
```

7.11 Managing Temporary Files

The `--compatible` mode requires the creation of temporary files. These files are automatically wiped and deleted by EDGE.

EDGE allows you to specify a directory where temporary files must be created. If the `TMP` setting is defined, EDGE uses the directory specified by this setting to create temporary files.

Examples:

```
TMP=C:\Temporary\  
TMP=/usr/tmp/
```

```
edge --encrypt --conventional --text --armor <file.txt>  
--compatible on --tmp [/usr/tmp/]
```

8 Log Information

EDGE stores information in a log file. Depending on the action performed by the application, different information is stored in this log file.

EDGE uses the standard logging mechanism for the operating system it is running on.

In addition, EDGE allows you to also store all logged information in a text file. This text file is located in the EDGE directory inside the current user account.

8.1 UNIX

On UNIX systems, EDGE uses syslog to log information. It allows you to use relay mechanism, log analyzer tools and more.

A typical syslog event is composed of date, time, name of process, process ID and is followed by a message.

Here is a typical syslog event:

```
2003/08/02 12:03:17: EDGE[1317]: Decrypting file "file.txt.pgp"
```

EDGE can also add a session ID to a recorded event. This session ID helps you to link events together.

To specify a session ID, use the `--logsession` option:

```
edge ... --logsession [7]
```

By adding this session ID, EDGE adds the specified session ID to all recorded event. The above example becomes:

```
2003/08/02 12:03:17: EDGE[1317]: (7) Decrypting file  
"file.txt.pgp"
```

If LOGSESSION equals 0, no session ID is added to the recorded events.

8.2 Windows

On Windows systems, the Windows event mechanism is used. Log information can be viewed by the standard Event Viewer application.

NOTE: If NOLOG setting is set to OFF (or 0), no log file will be created and no information will be logged.

9 Working with Session Keys

When data is encrypted, a session key is used to initialize a cipher. This session key is derived from a password or is composed of random data and encrypted using a public key.

When decrypting data, the session key is computed using the same password or by decrypting it using a private key.

EDGE can extract this session key while decrypting the file and place it in a separate file. If the same encrypted file is decrypted again, the session key file can be used to decrypt the data instead of using the same password or the private key.

This option can also be used to allow somebody to decrypt specified encrypted data without compromising files that have been encrypted with that private key.

The session key can be stored in a separate file with the same name as the encrypted file with a ".sky" extension, or it can be printed on the screen.

9.1 Extracting the Session Key to a File

To extract the session key from an encrypted file, use the `--getsessionkey` option.

Example:

```
edge --decrypt <file.txt.pgp> --getsessionkey
```

This option can also be used with `NOOUTPUT` and `DECRYPTONLY` settings to only extract the session key used to decrypt the data without decompressing or verifying signatures. By using the `NOOUTPUT` option, no file other than the session key file is produced.

```
edge --decrypt <file.txt.pgp> --getsessionkey --nooutput  
--decryptonly
```

9.2 Displaying the Session Key on the Screen

To display the session key from an encrypted file on the screen, use the `--printsessionkey` option.

Example:

```
edge --decrypt <file.txt.pgp> --printsessionkey
```

This option can also be used with `--nooutput` and `--decryptonly` settings to only extract the session key used to decrypt the data without decompressing or verifying signatures. By using the `--nooutput` option, no file is produced.

```
edge --decrypt <file.txt.pgp> --printsessionkey --nooutput  
--decryptonly
```

10 Working with Key Servers (Windows Only)

EDGE is able to get keys from your company's or another organization's Sovereign Server. Authora's Sovereign Server is a risk manageable public key server that enforces enrollment and authentication of public keys based upon each company's unique trust models. Sovereign Server offers elegant public key management to its members. Sovereign Servers and Sovereign Trust Zones are currently accessible from both EDGE and Zendit (the GUI OpenPGP client from Authora). For more information see: <http://www.authora.com>.

To communicate with a Sovereign Server, you need to be a registered user. As a registered user, you own a login and a password. To become a registered user ask your company's Sovereign Server administrator or your partner company's Sovereign Server administrator. Authora offers a Public Sovereign Server located at <http://www.authora.com>.

EDGE uses the `--login` and `--loginpass` options to log into a Sovereign Server. If these options are not defined, EDGE asks you to enter them when necessary.

10.1 Displaying Keys Available on the Server

To display keys available on the server, use the `-kv` option:

```
edge -kv <userID> <serverLocation>
```

EDGE tries to log in into the specified server and requests information for the specified key.

Examples:

```
edge -kv alice@mycompany.com
http://www.mycompany.com/sovereign
edge -kv alice@mycompany.com
http://www.mycompany.com/sovereign
+LOGIN=myLogin
```

10.2 Importing Keys From the Server

To import keys from the server, use the `-kx` option:

```
edge -kx <userID> <keyFile> <serverLocation>
```

EDGE tries to log into the specified server and get information for the specified key.

Examples:

```
edge -ka <alice@mycompany.com> <alice.pgp>  
<http://www.mycompany.com/sovereign>
```

```
edge -ka <alice@mycompany.com> <alice.pgp>  
<http://www.mycompany.com/sovereign +LOGIN=myLogin>
```

11 Working with X.509 Certificates (Windows Only)

EDGE is able to convert an X.509 certificate to an OpenPGP key. This feature allows you to conserve your X.509 certificate and its public components and to use it as a regular OpenPGP key. Depending on the recipient's Public Key Infrastructure (PKI) you can choose your X.509 certificate or your PGP key.

The OpenPGP key is built using the public key contained in the X.509 certificate. Public key components for both keys are the same. The name of the OpenPGP key is built using the SubjectName sequence of the certificate. The format of the name follows LDAP recommendations.

A special kind of signature is added to the OpenPGP key. This signature has the same validity period as the certificate and contains the whole certificate.

The converted X.509 certificate can then be used as a regular OpenPGP key.

11.1 Displaying an X.509 Certificate

To display an X.509 certificate, use the same syntax as you use for a regular OpenPGP key:

```
edge --key-list --pubring <myCertificate.pem>
```

EDGE converts the certificate contained in the specified file and displays it as a regular OpenPGP key.

NOTE: The file must contain a PEM encoded certificate.

To display a particular certificate, use:

```
edge --key-list <myCertName> --pubring <myCertificate.pem>
```

EDGE converts the certificate contained in the specified file and displays the X.509 certificate matching the specified User ID as a regular OpenPGP key.

If the specified file contains more than one PEM certificate part, EDGE displays X.509 certificates contained in each part separately.

11.2 Importing an X.509 Certificate

To import an X.509 certificate, use the same syntax you use for a regular OpenPGP key:

```
edge --key-add <myCertificate.pem>
```

EDGE converts the certificate contained in the specified file and imports it as a regular OpenPGP key. The new created OpenPGP key can now be used to encrypt data and to produce OpenPGP files. The OpenPGP key created from the X.509 certificate can be distributed as any ordinary OpenPGP key. Please refer to the Key Management section in this manual to learn how to extract a key.

NOTE: The file must contain a PEM encoded certificate.

If the specified file contains more than one PEM certificate part, EDGE imports the X.509 certificates contained in each part.

11.3 Encrypting and Signing Data

Converted X.509 certificates can be used as regular OpenPGP keys and can therefore be used to encrypt data using EDGE. The encrypted data will be an OpenPGP message even if the X.509 certificate is used to encrypt the data.

EDGE supports Cryptographic Tokens for signing and decrypting data.

To use keys contained on Tokens, use the `--usetoken` option:

```
edge --sign <file.txt> --sign-with <myCertificate>  
--usetoken on
```

EDGE tries to find the cryptographic device (Token) containing the specified certificate.

12 Compatibility

EDGE is able to generate files compatible with all versions of the OpenPGP standards. Some OpenPGP implementations, such as PGP 2.6.2, work only with files as described in RFC 1991.

To force EDGE to produce these kinds of files you need to turn the `--compatible` option on.

The default value for this option is OFF. Files generated by EDGE are therefore not compatible with old OpenPGP clients.

Examples:

```
edge --encrypt <file.txt> --user <userID> --compatible on
edge --encrypt <file.txt> --user <userID> --compatible off
```

Files encrypted and signed using the `--compatible` option can be decrypted by all OpenPGP clients.

Some of the oldest OpenPGP clients don't support the new OpenPGP signature mechanism. To be sure that signed data will be decrypted by all OpenPGP clients, turn the compatible mode on.

EDGE works with the same kind of keyrings as PGP. A PGP keyring can be used directly by EDGE without any conversion. If you already have a public and a private keyring, you can change your configuration file to use it directly or you can import keys contained in your existing keyrings to the default EDGE keyrings.

To generate keys compatible with PGP 2.6.2, use the `--key-type rsa-legacy` option. When generating a new RSA key pair, EDGE generates keys compatible with PGP 2.6.2. This kind of key has no subkeys. Please refer to the Key Management section in this manual to learn more about key types and how to generate a new key pair.

Example:

```
edge --key-gen --key-type <rsa-legacy>
```

13 Configuration File

EDGE uses a configuration file to store the default user options such as the path of the default public keyring and the default private keyring.

EDGE creates this configuration file automatically at the first launch of the application.

The configuration file is a text file and can therefore be modified with a text editor (such as NotePad on Windows or vi on UNIX).

You can also modify the configuration file by specifying the options you want to modify directly at the command-line. EDGE automatically updates the configuration by replacing only the specified options:

```
edge --[option] [value]
```

Examples:

```
edge --armor on
edge --armor 1
edge --armor on --textmode off
```

For Boolean options, you can omit the value 1 or ON.

Examples:

```
edge --armor
edge --armor --textmode
edge --armor --textmode off
```

The configuration file is a text file composed of lines. Each line can be an empty line which is ignored by EDGE, a comment which is also ignored by EDGE or an option which is composed of a tag and a value.

```
[setting]=[value]
```

A line beginning with `#` is considered a comment and is ignored.

13.1 Location of the Configuration File

The configuration file is named "edge.cfg".

To display the location of the configuration file used by EDGE, type:

```
edge --version

edge - Encrypted Data Gateway Engine
```


Version 3.7
Copyright (C) 2002-2007 Authora Inc. & Veridis SA
All rights reserved.

License : 10000-0000-0000
Licensed to: Alice

EDGE directory:
C:\Documents and Settings\Laurent\My Documents\edge data\

Configuration file:
C:\Documents and Settings\Laurent\My Documents\edge
data\edge.cfg

License file:
C:\Documents and Settings\Laurent\My
Documents\ACL\Bin\edge.lic

13.1.1 Windows

EDGE checks if PGPPATH is defined.

- If PGPPATH is defined, EDGE uses the configuration file located in that directory. If the configuration file doesn't exist, EDGE creates it. If the configuration file cannot be read or created, an error is returned by EDGE and the operation is cancelled.
- If PGPPATH is not defined, EDGE uses the "My Documents\edge data" directory of the current account to locate the configuration file. If the configuration file cannot be found, EDGE creates it.

13.1.2 UNIX

EDGE checks if PGPPATH is defined.

- If PGPPATH is defined, EDGE uses the configuration file located in that directory. If a configuration file doesn't exist, EDGE creates one.
- If PGPPATH is not defined, EDGE uses the "~/.edge" directory. If the configuration cannot be found, EDGE creates it.

13.2 Working in a Shared Environment

EDGE can be installed in a computer shared by different users. Every user has their own configuration file and keyrings.

The computer administrator can pre-configure options for all users and can even restrict some options modification.

When a configuration file is missing, EDGE creates a new one based on a template file named "edge_template.cfg". On Windows, this file is located in "/Documents and Settings/All Users/Application Data/edge data/". On UNIX, this file is located in "/etc/.edge".

This file can contain any options you want.

When EDGE is started, it looks for a default configuration file named "edge_default.cfg". On Windows, this file is located in "/Documents and Settings/All Users/Application Data/edge data/". On UNIX, this file is located in "/etc/.edge".

This file is read before the user's configuration file and contains default options for the user. Options from the default configuration are discarded by the user's configuration. If the same option is defined in the default configuration and in the user's configuration, the value of this option will be the value defined in the user's configuration.

After the user's configuration has been read, EDGE tries to read the policy configuration file named "edge_policy.cfg". On Windows, this file is located in "/Documents and Settings/All Users/Application Data/edge data/". On UNIX, this file is located in "/etc/.edge".

Options found in that policy configuration file discard options from the user's configuration file. If the same option is defined in the user's configuration and in the policy configuration, the value of this option will be the value defined in the policy configuration.

This policy configuration file can be useful to force some options that could be deactivated manually by the user and allows an administrator to enforce the use of internal policies.

Examples:

Here is the content of a default configuration file:

```
ARMOR=ON
TEXTMODE=OFF
CLEARSIG=OFF
```

Here is the content of a user's configuration file:

```
ARMOR=OFF
TEXTMODE=ON
CLEARSIG=OFF
```

Here is the content of a policy configuration file:

```
ARMOR=ON
```

When EDGE is started, the default configuration is read. Value for ARMOR is set to ON, value for TEXTMODE is set to OFF and value for CLEARSIG is set to OFF.

EDGE continues by reading the user's configuration file. Value for ARMOR is replaced and set to OFF, value for TEXTMODE is replaced and set to ON, value for CLEARSIG stays unchanged.

Now EDGE reads the policy configuration. Value for ARMOR is replaced and set to ON, values for TEXTMODE and CLEARSIG stay unchanged.

If the user now types:

```
edge --encrypt <clearfile> --user <userID>
```

The destination file will be armored because it is requested by the policy configuration file. Same with the following command:

```
edge --encrypt <clearfile> --user <userID> --armor off
```

Environment variables can be used inside the configuration file. For example, the value for the following value will be replaced dynamically by EDGE when the value for this option is required. EDGE gets the value for the environment variable enclosed between the characters "<" and ">".

```
PUBRING=<HOME>/pubring.pgp
```

For example, if the home directory for the current user is "/usr/user", the final value for the PUBRING option will be "/usr/user/pubring.pgp".

13.3 Supported Settings

EDGE recognizes the following settings which can be implemented in a number of ways.

Specifying Options in the Configuration File:

Options can be edited directly within your configuration file, which will affect all operations.

Example:

```
ARMOR=ON
```

Specifying Options within a Command String:

Options can also be specified at the command line within a string of commands, and are therefore turned on or off only for the current operation.

Example:

```
edge --encrypt <fileName> --user <userName> --armor on
```

In the above example, armor will be turned on only while encrypting this specific file.

Editing the Configuration File Options from the Command Line:

If only an option is specified at the command line and it is not part of a command string, the option setting will be edited in the configuration file. This is convenient for editing the configuration file without using a text editor.

Example:

```
edge --armor on
```

In the above example, the option is not specified within a command string and will therefore edit and save the configuration file with the option setting: ARMOR=ON

Value Settings

For Boolean options, the ON value can also be specified as 1, while the OFF can also be specified as 0.

For Boolean options, you can omit the value 1 or ON. If you omit a value for a Boolean option, its value will automatically default to "ON".

Example:

```
edge --armor
edge --armor on
```

In the above example, both of these option commands do the same thing: turn armor on.

Note: If operating EDGE in legacy mode, remember that an option at the command line always begins with a "+":

```
edge +ARMOR=ON
```

13.3.1 ADDPUBLICKEYS

This is a Boolean value indicating if the public keys must be added to the default keyring. This setting can be used when adding keys (`--key-add` option).

Default value is ON.

Examples:

```
edge --key-add --addpublickeys
```

```
ADDPUBLICKEYS=ON
ADDPUBLICKEYS=OFF
ADDPUBLICKEYS=0
ADDPUBLICKEYS=1
```

13.3.2 ADDSECRETKEYS

This is a Boolean value indicating if the private keys must be added to the default keyring. This setting can be used when adding keys (`--key-add` option).

Default value is ON.

Examples:

```
edge --key-add --addsecretkeys
```

```
ADDSECRETKEYS=ON
ADDSECRETKEYS=OFF
ADDSECRETKEYS=0
ADDSECRETKEYS=1
```

13.3.3 ARMOR

This is a Boolean value indicating if the destination file should be armored. This setting is equivalent to the `"-a"` option.

If this value is set to ON, all files created by EDGE are automatically armored.

Default value is OFF.

Examples:

```
edge --encrypt <fileName> --user <userID> --armor on
```

```
ARMOR=ON
ARMOR=OFF
ARMOR=0
ARMOR=1
```

13.3.4 BACKUPPUBRING

If this setting is ON, EDGE keeps a backup copy of your public keyring. The copy is located in the same directory as the default public keyring.

Default value is ON.

Examples:

```
BACKUPPUBRING=ON
```

```
BACKUPPUBRING=OFF  
BACKUPPUBRING=0  
BACKUPPUBRING=1
```

13.3.5 BACKUPSECRING

If this setting is ON, EDGE keeps a backup copy of your private keyring. The copy is located in the same directory as the default private keyring.

Default value is ON.

Examples:

```
BACKUPSECRING=ON  
BACKUPSECRING=OFF  
BACKUPSECRING=0  
BACKUPSECRING=1
```

13.3.6 BATCHMODE

If this option is set to ON, EDGE suppresses unnecessary questions by automatically answering NO.

Default value is OFF.

Examples:

```
BATCHMODE=ON  
BATCHMODE=OFF  
BATCHMODE=0  
BATCHMODE=1
```

13.3.7 CHECK-SIGNED

When this option is set to ON, EDGE returns the value 1 when decrypting a file which is not signed and when the BATCHMODE option is also enabled.

Default value is ON.

Example:

```
edge --decrypt <fileName> --check-signed on
```

13.3.8 CIPHERNUM

This option indicates the symmetrical algorithm to use to encrypt data. The following values are allowed:

Default	or	0
IDEA	or	1
3DES	or	2
CAST5	or	3
AES128	or	7
AES192	or	8
AES256	or	9
TWOFISH	or	10

You can use either the algorithm name or the number as the value.

Examples:

```
edge --encrypt <filename> --user <username> --ciphernum
cast5
```

```
edge --encrypt <filename> --user <username> --ciphernum 3
```

or

```
CIPHERNUM=3
```

The default value depends on the type and version of the key used. For RSA keys and for old key versions, IDEA is the default algorithm used. For other keys, 3DES is the default value. If there is a conflict between keys and algorithm usage, 3DES is used as the default algorithm.

13.3.9 CLEARSIG

This option indicates if a clear-text signature must be generated.

To generate this kind of signature, the input file must be a text file and the output file must be armored. This setting must be used with `TEXTMODE` (or `--text`) and `ARMOR` (or `--armor`) options.

Default value is OFF.

Examples:

To generate a clear-text signature on file "input.txt", type:

```
edge --sign --armor --text <input.txt> --sign-with <myKey>
--clearsig
```

```
CLEARSIG=ON
CLEARSIG=OFF
CLEARSIG=0
CLEARSIG=1
```

13.3.10 CMDLINE-FORMAT

This sets the preferred format for options. Value for this option can be "long" for long options format or legacy for compatible options.

EDGE automatically guesses the format you are using, so in most circumstances this option should not be set.

13.3.11 COLORS

This is a Boolean value indicating if colors can be used when displaying information on the screen.

Default value is ON.

Examples:

```

COLORS=ON
COLORS=OFF
COLORS=0
COLORS=1

```

13.3.12 COMMENT

The value of this setting appears on all armored files. This allows you to add a short comment to an armored file. If the comment is bigger than 64 characters, EDGE splits the comment into several lines.

Examples:

```

COMMENT=This file has been generated by EDGE

edge --comment "This file has been generated by EDGE"

```

Your comment will appear in an armored file as follows:

```

-----BEGIN PGP SIGNATURE-----
Comment: This file has been generated by EDGE

iQCVAwUBPtvgWegBJJ2jQbYlAQHPwQP+MJzCnBS0FFlYoM+ilix2DnGbtrt38lO
i
WytWSsLI8Bi65SAx2phy+XvYmNgClzr6Cmp0660+v1uULgwtzrGKPMva2x9X4GT
D
EFits0V059WQ+zP3M51URWtQUq7aBfJSbYOCVpKwUQ3VGgr3qG7v0eLLPBCDiB1
A
v9wiGxyqfSo=
=1xfu

```


-----END PGP SIGNATURE-----

13.3.13 COMPRESS

If this setting is set to ON, data is compressed before encryption. Compressing data before encryption reduces some types of attacks. It is recommended to leave this option turned on.

Default value is ON.

Example:

```
COMPRESS=ON  
COMPRESS=OFF  
COMPRESS=0  
COMPRESS=1
```

13.3.14 COMPRESSLEVEL

This option lets you control the compression of the ZIP algorithm used by the EDGE. This can be a value from 0 to 9. Value 0 indicates to use the default compression level.

Default value is 6.

Example:

```
COMPRESSLEVEL=2
```

13.3.15 CONFIG-FILE

This option tells EDGE where to find the configuration file. If this option is specified, EDGE doesn't try to locate the configuration file and uses the specified file location.

13.3.16 COMPAT-ERRORS

If this option is set to ON, EDGE returns the same error numbers as McAfee eBusiness Server and PGP command-line. Errors returned are less detailed than the EDGE errors.

Default value is OFF.

13.3.17 CREATE-PUB

If this option is ON, EDGE automatically creates a corresponding public key when a private key is added to the private keyring. If this option is OFF, EDGE simply adds the private key to the private keyring and no signature can be verified for that signer until the corresponding public key is added to the public keyring.

Default value is ON.

13.3.18 DECRYPTONLY

If this option is ON, EDGE stops the decryption/verification process after the decryption phase and creates no output file.

Default value is OFF.

13.3.19 DEFAULT-KEY

This option sets the default signing key. The default key will be used if no signing key is specified when signing a message.

13.3.20 ENCRYPT-TO-SELF

This option indicates if data must also be encrypted with the key specified by `DEFAULT-KEY`.

An error occurs if this setting is ON and if no value is specified for `DEFAULT-KEY`.

13.3.21 ERRORFD

This setting allows the redirection of errors to a particular file descriptor. By default, EDGE displays error data to `stderr`. Both of the following examples do the same thing

Examples:

```
edge 2> &1
edge +ERRORFD=1
```

The above examples redirect the error data to be displayed on the file descriptor 1 which is `stdout`.

Default value is 2 (`stderr`).

13.3.22 EXPIRES-AFTER

This option defines the number of days after which a signature expires. This option is used when certifying a User ID or when a new key pair is generated.

Default value is 0 (never).

13.3.23 EXPORTPUBLIC

If this option is ON, EDGE exports public keys (see also `-kx` command).

Default value is ON.

13.3.24 EXPORTSECRET

If this option is ON, EDGE exports private keys (see also `-kx` command).

By default, private keys are not exported. If you want to export both public and private keys, you need to turn on both EXPORTPUBLIC and EXPORTSECRET options.

Default value is OFF.

13.3.25 FINGERPRINT-VIEW

This option tells EDGE how to display fingerprints when using the `--key-detail` command. Values for this option can be "HEX" for a hexadecimal display or "WORDS" for a display using biometric words. Appendix A contains the list of all words used by EDGE.

Default value is "hex".

13.3.26 FORCE

This option allows you to remove user interaction and to fully automate the encryption/decryption process. If this option is set to ON, actions performed by EDGE will never be interrupted by a question because a "YES" answer will be given as default.

For example, if the destination file of a decryption operation already exists and the `FORCE` option is on, EDGE automatically overwrites the redundant same-named file without asking you for a overwrite confirmation or an alternate destination file.

Default value is OFF.

Examples:

```
FORCE=ON
FORCE=OFF
FORCE=0
FORCE=1
```

13.3.27 GETSESSIONKEY

If this option is ON when decrypting data, the session key used to decrypt this data is saved on a file named like the encrypted file with a ".sky" extension.

Default value is OFF.

13.3.28 HASHNUM

Indicates the data digest algorithm to use. Following values are allowed:

Default	or	0
MD5	or	1
SHA1	or	2
RIPMD160	or	3
SHA256	or	8
SHA384	or	9
SHA512	or	10

Examples:

```
HASHNUM=2
HASHNUM=SHA1
```

13.3.29 HELP FILES

This option tells EDGE where to find the help files. If this option is specified, EDGE tries to use the specified directory to find the help files.

13.3.30 INTERACTIVE

If this setting is set to ON, EDGE prompts for confirmation for every key added to your keyring.

If you want to fully automate the import of new keys to your keyring, you should turn this setting off.

Default value is OFF.

Examples:

```
INTERACTIVE=ON
INTERACTIVE=OFF
INTERACTIVE=0
INTERACTIVE=1
```

13.3.31 KEY-SIZE

This option tells EDGE the size of the key to generate when using the `--key-gen` command.

Default value is 2048.

13.3.32 KEY-TYPE

This option tells EDGE the type of the key to generate when using the `--key-gen` command. Values for this option can be "RSA" for RSA keys, "RSA-Legacy" for RSA-Legacy keys (keys compatible with PGP 2.6.x) or "DSS" for DSS/DH keys.

Default value is DSS.

13.3.33 LICENSE-FILE

This option tells EDGE where to locate the license file to use. If this option is specified, EDGE tries to use the license file located at the specified location.

13.3.34 LOGFD

This defines a file descriptor where log information is written to. To write log information on the standard output, use a value of 1. A value of 0 indicates no file descriptor.

Default value is 0.

13.3.35 LOGFILE

This defines the path to the log file used by EDGE to store information about the performed actions. If this option is not defined, the default log file name and location is used.

By default the log file is located in the same directory as the configuration file and named "log.txt".

Example:

```
LOGFILE=C:\Logs\log20030616.txt
```

13.3.36 LOGIN

This defines the username to use when accessing the Authora Sovereign Server. If this setting is not specified, EDGE asks you to enter your user name when necessary.

13.3.37 LOGINPASS

This defines the password to use when accessing the Authora Sovereign Server. If this setting is not specified, EDGE asks you to enter the password for the specified user name when necessary.

13.3.38 LOGLEVEL

This defines the amount of information stored in the log file. Can be a value between 0 and 2. A value of 0 means less information.

Default value is 1.

13.3.39 LOGSESSION

Specifies the session number added to all logged event. If `LOGSESSION` equals 0, no session number is added to the logged event.

Default value is 0.

13.3.40 MERGEONLY

If this option is ON, no new keys are added when importing keys from a file. This option allows you to update your keyring without adding any new keys.

Default value is OFF.

13.3.41 NO-CONFIG-FILE

If this option is ON, EDGE doesn't try to locate a configuration file and uses only options specified at the command-line.

Default value is OFF.

NOTE: If this option is set to ON, the default public and private keyrings must be specified at the command-line.

13.3.42 NOCOPYRIGHT

If this option is ON, no copyright information is displayed on the screen.

Default value is OFF.

13.3.43 NOLICENSEINFO

If this option is OFF, no licensing information is displayed on the screen.

Default value is ON.

13.3.44 NOLOGFILE

Is a Boolean value indicating if the operation should be logged into a file or not. If this setting is OFF, no information about encryption/decryption will be logged to the log file.

Default value is OFF.

Examples:

```
NOLOG=ON
NOLOG=OFF
NOLOG=0
NOLOG=1
```

13.3.45 NOOUT

This option is equivalent to:

```
--nocopyright --nolicenseinfo --verbose off
```

Examples:

```
NOOUT=ON
NOOUT=OFF
NOOUT=0
NOOUT=1
```

Default value is OFF.

13.3.46 NOOUTPUT

If this option is ON, no output file is produced when decrypting a file.

Examples:

```
NOOUTPUT=ON
NOOUTPUT=OFF
NOOUTPUT=0
NOOUTPUT=1
```

Default value is OFF.

13.3.47 NOPROGRESS

If this option is ON, no progress bar is displayed when encrypting/signing files.

Default value is OFF.

Examples:

```
NOPROGRESS=ON
NOPROGRESS=OFF
NOPROGRESS=0
NOPROGRESS=1
```

13.3.48 NOSYSLOG

If this option is ON, EDGE doesn't use the UNIX syslog mechanism to log events.

Default value is OFF.

Examples:

```
NOSYSLOG=ON
NOSYSLOG=OFF
NOSYSLOG=0
NOSYSLOG=1
```

13.3.49 PASSTRY

Defines the number of re-tries allowed for entering a passphrase before aborting the current operation.

Default value is 3.

13.3.50 PRESERVE-NAME

This option tells EDGE to recover the file name stored in the encrypted/signed file and use it as destination file name.

Default value is OFF.

13.3.51 PUBRING

Defines the path of the default public keyring.

Examples:

```
PUBRING=C:\Keyrings\pubring.pgp
PUBRING=/home/Keyrings/pubring.pgp
```

13.3.52 PRINTSESSIONKEY

This setting is equivalent to the GETSESSIONKEY setting except that the session key is displayed on the screen and is not saved on disk.

Default value is OFF.

13.3.53 REVERSE

If this option is set to ON, EDGE displays keys in descending order instead of ascending order. This option is used when displaying keys with the --key-list and --key-detail options.

Default value is OFF.

13.3.54 RSAVER

This option tells EDGE which version of RSA keys to generate. Allowed values are 3 for RSA-Legacy keys and 4 for new RSA keys.

Default value is 4.

NOTE: This option is supported only when using the legacy options. Generating a RSA-Legacy key by using the --key-gen command can be done by using the --key-type option.

13.3.55 SDA

This is a Boolean value indicating if the source file must be encrypted as a Self-Decrypting Archive (SDA). If on, EDGE creates a Windows application containing the encrypted file(s).

This option can only be used with the -c command.

Examples:

```
SDA=ON  
SDA=OFF  
SDA=0  
SDA=1
```

13.3.56 SECRING

Defines the path of the default private keyring.

Examples:

```
PUBRING=C:\Keyrings\secring.pgp  
PUBRING=/home/Keyrings/secring.pgp
```

13.3.57 SECURE-VIEWER

This option tells EDGE to display the decrypted data on the screen and not to store it on the disk.

Default value is OFF.

13.3.58 SIGN-ONLY

This option tells EDGE to generate signature-only keys. If this option is set to ON, EDGE generates a signature-only key.

Default value is OFF.

Examples:

```
edge --key-gen --sign-only on  
edge --key-gen --sign-only off
```

13.3.59 SIG-TYPE

This option tells EDGE the type of signature to generate when certifying a User ID. Values for this option can be "local" or "exportable". A local signature is not exported by EDGE.

Default value is "exportable"

13.3.60 SORT

This option defines the default sorting order when displaying a list of keys, such as in

`--key-list` or `--key-detail`.

This sorting order is also used when displaying a list of multiple keys.

Allowed values are:

keysize	Keys are displayed sorted by using the size of the key.
subkeysize	Keys are displayed sorted by using the size of the subkey.
keyid	Keys are displayed sorted by using the key ID of the key.
userid	Keys are displayed sorted by using the User IDs of the key (default value).
creation	Keys are displayed sorted by using the creation date of the key.
expiration	Keys are displayed sorted by using the expiration date of the key, if any.

13.3.61 STATUSFD

This setting allows the redirection of the output to a particular file descriptor.

Default value is 1 (`stdout`).

13.3.62 TEXTMODE

This value indicates whether or not the source file should be considered a text file. If `TEXTMODE` is on, the source file is converted into canonical form. During decryption, the line endings will be converted according to the specification of the platform.

This setting is equivalent to the `--text` option.

Default value is OFF.

Examples:

```
TEXTMODE=ON
TEXTMODE=OFF
TEXTMODE=0
TEXTMODE=1
```

NOTE: This option is automatically disabled for the current file if the specified input file is not an ASCII text file. If this option is disabled, EDGE displays a warning (if `VERBOSE > 0`) to the screen and adds a warning to the log file.

13.3.63 TMP

This option lets you specify the directory for storing temporary files. Temporary files are used when encrypting/signing data using the `COMPATIBLE` option.

Examples:

```
TMP=C:\Temp\
TMP=/home/Temp/
```

13.3.64 VERBOSE

This preference specifies the amount of information displayed by the application. It can be one of the followed values:

- 0 - No information
- 1 - A reasonable amount of information.
- 2 - A large amount of information

Examples:

```
VERBOSE=0
VERBOSE=1
VERBOSE=2
```

Default value is 1.

13.3.65 VERSION

The value of this setting appears on all armored files. This allows you to add short version information to an armored file.

Example:

```
VERSION=EDGE 3.7
```

Your version information will appear in an armored file as follows:

```
-----BEGIN PGP SIGNATURE-----
Version: EDGE 3.7

iQCVAwUBPtvGWegBJJ2jQby1AQHPwQP+MJzCnBS0FFlYoM+ilix2DnGbtrt3
810i
4GTD
iB1A
WyTWSsLI8Bi65SAx2phy+XvYmNgClzr6Cmp0660+v1uULgwtzrGKPMva2x9X
EFits0V059WQ+zP3M51URWtQUq7aBfJSbYOCVpKwUQ3VGgr3qG7v0eLLPBCD
v9wiGxyqfSo=
=1xfu
-----END PGP SIGNATURE-----
```

13.3.66 WIPE-PASSES

This option tells EDGE the number of passes used when a file is wiped.

Default value is 7.

14 Legacy Mode Commands

EDGE allows you to specify commands and options using two different modes:

- Long Arguments
- Legacy Mode

Versions of EDGE prior to 2.0 support only the legacy mode. This mode is compatible with OpenPGP applications such as PGP 2.6.x or PGP 6.5.8 command-line version.

Versions of EDGE 2.0 or higher, also support long arguments as used in McAfee E-Business Server version 7.1.1.

EDGE automatically detects the mode you are using allowing you to use both modes indifferently.

For compatibility reasons, EDGE still provides a way to specify commands and options using the legacy mode.

14.1 Allowed Commands

The following table describes the EDGE commands supported in the legacy mode. Further sections of this user guide tell you how to use these commands.

Command	Description
-a	Converts the destination file to ASCII-armored format. This command is used with other commands such as encryption and signing. Files in ASCII-armored format are text files and can be pasted in an email or concatenated to other text or ASCII-armored text files.
-b	When used with the signing option (-s), this command tells EDGE to generate a detached signature. A detached signature can be useful if the signed data must be transmitted unchanged (in the same format). It is also useful if the signed data and the signature must travel two separate ways to their destination.
-c	Encrypts data with a password. This command is also called conventional encryption. Data is encrypted using a password and the same password is requested to decrypt the data.

- e Encrypts data using public-key encryption. Public keys are needed to encrypt the data. More than one public key can encrypt the same data, but only one private key corresponding to one of the public keys used to encrypt is needed to decrypt the data.
- f Filter mode. When this command is used, EDGE reads data from the standard input and writes data to the standard output. This command allows you to use EDGE as a UNIX pipe.
- h Displays a summary of available commands and options.
- k Key operations. When this command is used, some other commands have different meaning. See the table below for more information about key operations.
- s Digitally signs data. A private key is needed to sign the data. The corresponding public key is needed to verify the signature. Signed data can be read by everybody, it just authenticates the author of the data. Verifying the signature allows you to verify whether or not the data has been altered during its transport.
- t Text mode. When this command is used, EDGE treats the input file as a text file and converts it into canonical form. You should use this command only for ASCII text files. Using it on a binary file can damage it. A Word document is not an ASCII text file. To prevent this kind of mistake, EDGE automatically checks whether the input file is an ASCII text file or not. If EDGE considers it a binary file, the text mode is disabled and the file is automatically treated as a binary file.
- u Identifies the key to use to digitally sign data. This command is followed by the User ID or the key ID of the key you want to use to digitally sign the data.
- v Displays version and license information. EDGE also displays the default EDGE directory path and the path of the configuration file used.
- vv Displays the content of the configuration file.
- w Wipe input file after the current operation has been completed without error. If this command is used while encrypting a file, EDGE securely deletes the input file (the file to encrypt) after the encryption has been done and only if no error occurred during encryption.

- z Identifies the passphrase to use to encrypt data with a password, to digitally sign data or to decrypt data.
 This command is a way to specify a passphrase directly at the command-line allowing you to remove user intervention. This command is followed by the passphrase. If more than one passphrase is required, use the -z command as the number of passphrases requested.
 If more than one passphrase is specified for decrypting a file, EDGE tries all passphrases until it finds a matching one. In that case, only the matching passphrase is removed from the passphrase list and all others remain available if another passphrase is requested.

The following table describes the key operations in legacy mode. These commands must be used together with the -k option.

Command	Description
-ka	Adds keys to default keyrings. Depending of the type of the key to add, it will be added to the public or private keyring.
-kd	Enables/disables or revokes a key. The revoke operation is only allowed if the private key can be found in the default private keyring. In that case, the passphrase to unlock that private key is requested.
-ke	Key edit command. This command allows you to add new User IDs to your keys or to change the passphrase of a private key. In both cases, this command can only be used if a private key matching the specified User ID can be found in the default private keyring.
-kr	Removes keys from keyrings.
-ks	Key signature command. This command allows you to sign a User ID. By signing it, you certify that the key belongs to the right person.
-kv	Displays the content of a keyring. This command allows you to display all keys contained in a particular keyring or from the default public keyring. It also allows you to display keys matching a particular User ID.
-kvc	Displays the content of a keyring and each key. This command also displays the fingerprint.
-kvv	Displays the content of a keyring including subkeys and signatures.

- kvvc Displays the content of a keyring including subkeys, signatures and fingerprint.
- kx Key extract command. This command allows you to get a copy of the keys contained in your keyrings. This command doesn't remove the key once it has been extracted. It can be useful to extract your public key from your default keyring or to extract your key pairs for backup purposes.
- kxa Key extract command. Same as the previous command except that keys are extracted as ASCII-armored files.
Armored output is automatically enabled if both public and private keys are extracted in one operation.

15 Appendix A – Biometric Word Lists

Two Syllable Word List

aardvark	absurd	accrue	acme	adrift
adult	afflict	ahead	aimless	Algol
allow	Alone	ammo	ancient	apple
artist	assume	Athens	atlas	Aztec
baboon	backfield	backward	banjo	beaming
bedlamp	beehive	beeswax	befriend	Belfast
berserk	billiard	bison	blackjack	blockade
blowtorch	bluebird	bombast	bookshelf	brackish
headline	breakup	brickyard	briefcase	Burbank
button	buzzard	cement	chairlift	chatter
checkup	chisel	choking	chopper	Christmas
clamshell	classic	classroom	cleanup	clockwork
cobra	commence	concert	cowbell	crackdown
cranky	crowfoot	crucial	crumpled	crusade
cubic	dashboard	deadbolt	deckhand	dogsled
dragnet	drainage	dreadful	drifter	dropper
drumbeat	drunken	Dupont	dwelling	eating
edict	egghead	eightball	endorse	endow
enlist	erase	escape	exceed	eyeglass
eyetooth	facial	fallout	flagpole	flatfoot
flytrap	fracture	framework	freedom	frighten
gazelle	Geiger	glitter	glucose	goggles
goldfish	gremlin	guidance	hamlet	highchair
hockey	indoors	indulge	inverse	involve
island	jawbone	keyboard	kickoff	kiwi
klaxon	locale	lockup	merit	minnow
miser	Mohawk	mural	music	necklace
Neptune	newborn	nightbird	Oakland	obtuse
offload	optic	orca	payday	peachy
pheasant	physique	playhouse	Pluto	preclude
prefer	preshrunk	printer	prowler	pupil
puppy	python	quadrant	quiver	quota
ragtime	ratchet	rebirth	reform	regain
reindeer	rematch	repay	retouch	revenge
reward	rhythm	ribcage	ringbolt	robust
rocker	ruffled	sailboat	sawdust	scallion
scenic	scorecard	Scotland	seabird	select
sentence	shadow	shamrock	showgirl	skullcap
skydive	slingshot	slowdown	snapline	snapshot
snowcap	snowslide	solo	southward	soybean
spaniel	spearhead	spellbind	spheroid	spigot
spindle	spyglass	stagehand	stagnate	stairway
standard	stapler	steamship	sterling	stockman
stopwatch	stormy	sugar	surmount	suspense
sweatband	swelter	tactics	talon	tapeworm
tempest	tiger	tissue	tonic	topmost
tracker	transit	trauma	treadmill	Trojan
trouble	tumor	tunnel	tycoon	uncut
unearth	unwind	uproot	upset	upshot
vapor	village	virus	Vulcan	waffle

wallet
Zulu

watchword

wayside

willow

woodlark

Three Syllable Word List

adroitness	adviser	aftermath	aggregate	alkali
almighty	amulet	amusement	antenna	applicant
Apollo	armistice	article	asteroid	Atlantic
atmosphere	autopsy	Babylon	backwater	barbecue
belowground	bifocals	bodyguard	bookseller	borderline
bottomless	Bradbury	bravado	Brazilian	breakaway
Burlington	businessman	butterfat	Camelot	candidate
cannonball	Capricorn	caravan	caretaker	celebrate
cellulose	certify	chambermaid	Cherokee	Chicago
clergyman	coherence	combustion	commando	company
component	concurrent	confidence	conformist	congregate
consensus	consulting	corporate	corrosion	councilman
crossover	crucifix	cumbersome	customer	Dakota
decadence	December	decimal	designing	detector
detergent	determine	dictator	dinosaur	direction
disable	disbelief	disruptive	distortion	document
embezzle	enchancing	enrollment	enterprise	equation
equipment	escapade	Eskimo	everyday	examine
existence	exodus	fascinate	filament	finicky
forever	fortitude	frequency	gadgetry	Galveston
getaway	glossary	gossamer	graduate	gravity
guitarist	hamburger	Hamilton	handiwork	hazardous
headwaters	hemisphere	hesitate	hideaway	holiness
hurricane	hydraulic	impartial	impetus	inception
indigo	inertia	infancy	inferno	informant
insincere	insurgent	integrate	intention	inventive
Istanbul	Jamaica	Jupiter	leprosy	letterhead
liberty	maritime	matchmaker	maverick	Medusa
megaton	microscope	microwave	midsummer	millionaire
miracle	misnomer	molasses	molecule	Montana
monument	mosquito	narrative	nebula	newsletter
Norwegian	October	Ohio	onlooker	opulent
Orlando	outfielder	Pacific	pandemic	Pandora
paperweight	paragon	paragraph	paramount	passenger
pedigree	Pegasus	penetrate	perceptive	performance
pharmacy	phonetic	photograph	pioneer	pocketful
politeness	positive	potato	processor	provincial
proximate	puberty	publisher	pyramid	quantity
racketeer	rebellion	recipe	recover	repellent
replica	reproduce	resistor	responsive	retraction
retrieval	retrospect	revenue	revival	revolver
sandalwood	sardonic	Saturday	savagery	scavenger
sensation	sociable	souvenir	specialist	speculate
stethoscope	stupendous	supportive	surrender	suspicious
sympathy	tambourine	therapeutic	therapist	tobacco
tolerance	tomorrow	torpedo	tradition	travesty
trombonist	truncated	typewriter	ultimate	undaunted
underfoot	unicorn	unify	universe	unravel
upcoming	vacancy	vagabond	vertigo	Virginia
visitor	vocalist	voyager	warranty	Waterloo
whimsical	Wichita	Wilmington	Wyoming	yesteryear

Yucatan

16 Appendix B - Error Codes

0	No error	The operation has been completed without error.
-1	End of file reached	The end of file has been reached unexpectedly.
-2	No more memory available	No more memory is available to allocate more objects.
-3	No key found	The specified or required key has not been found.
-5	Checksum error	The checksum of an ASCII-armored file is wrong. Generally this occurs when the file has been badly transferred.
-6	Bad packet found	An unexpected packet has been found while reading a key or decrypting a message.
-7	Compression error	The compression library has encountered an error while decoding a compressed file.
-8	FIPS Test error	Power-up tests have encountered an error.
-9	Data is too long	This error occurs when the size of a key is bigger than the maximum supported key size or when a path is bigger than the maximum path length supported by the system.

- 10 Unknown algorithm
- The specified algorithm or the algorithm information stored in a key or in an encrypted file is not supported.
- 11 Random number generation error
- This error occurs when the Pseudo-Random Number Generator has encountered an error.
- 12 Write error
- An I/O error has occurred while trying to write data to disk.
- 14 Encryption error
- An error has occurred while trying to use a key to encrypt data.
- 15 Unknown version
- The specified version or the version found in an encrypted file or key is not supported.
- 17 Access error
- The current user does not have the proper access rights to access the specified file or directory.
- 18 Invalid path
- The specified path is invalid.
- 19 Read error
- An I/O error has occurred while trying to read data from the disk.
- 20 Specified path is not a file
- The specified path denotes a directory and not a file.

- 21 File error
- An error has occurred while trying to use a disk file. The file can be a keyring, the configuration file or the license file.
- 22 Destination file already exists
- The destination file already exists. This error occurs when the specified output file or the file generated by EDGE already exists. To avoid this error, use the --force option.
- 23 Unable to create input file
- The input file cannot be created. This error can occur when using the standard input (stdin) as source file.
- 24 Unable to create output file
- The destination file cannot be created.
- 26 User cancelled the operation
- The user has cancelled the current operation or the operation has been cancelled because of the BATCHMODE option.
- 27 No public keyring has been found
- The public keyring has not been found at the specified location or has not been specified.
- 28 Public key has not been found
- The specified or required public key has not been found or has not been specified.
- 29 Specified key cannot be used to encrypt
- The specified key cannot be use to encrypt the message. This error occurs when a primary key doesn't contain a valid encryption subkey.

- 30 Key size is too small
- The specified key size is too small. This error can occur while generating a key if the requested key size is less than 512 bits.
- 31 Key size is too long
- The specified key size is too long. This error can occur while generating a key if the requested key size is bigger than 4096 bits.
- 32 Key cannot be used to sign
- The specified key cannot be use to sign the message. This error occurs if the specified key has been marked as an encryption-only key.
- 33 Private key is still encrypted by a passphrase
- The specified private key has not been unlocked and cannot be used.
- 34 Signature error
- This error occurs when a signature contains badly formed data.
- 35 No private keyring has been found
- The specified private keyring has not been found or has not been specified.
- 36 Private key has not been found
- The specified private key has not been found or has not been specified.
- 37 Specified passphrase is bad
- The specified passphrase is wrong. This error can occur while decrypting a message or when a private key needs to be unlocked using a passphrase.
- 38 Private key is already unlocked
- The specified private key has already been unlocked.

- 39 File not found

 The specified file cannot be found.
- 40 Bad format

 The key or message is badly formed.
- 41 File is empty

 The specified file contains no data.
- 42 File is not encrypted

 The specified file is neither encrypted nor signed.
- 43 Unknown encrypted session key version

 The version of the session key stored inside an encrypted file is not supported.
- 44 Unknown conventionally encrypted session key version

 The version of the session key stored inside a conventionally encrypted file is not supported.
- 45 Unknown string-to-key algorithm

 The algorithm used to convert the passphrase to the session key is unknown.
- 46 Bad key specified

 The specified key is an unexpected one or is badly formed.
- 47 Signature is bad

 The signature being verified is bad.
- 48 Key cannot be used to verify a signature

 The specified key is an encryption-only key or has been marked as not able to verify signatures.

- 49 Key cannot be used to decrypt data
- The specified key is a signature-only key or has been marked as not able to decrypt data.
- 50 Error while trying to create a temporary file
- A temporary file cannot be created. This error can occur while encrypting a message using the COMPATIBLE option or when saving a keyring. See the TMP option to learn how to change the location used by EDGE to create temporary files.
- 52 Specified name has not been found
- The specified User ID has not been found.
- 53 Already signed
- The specified User ID or key has already been signed by the specified signer's key.
- 54 Error while generating a new key
- An error has occurred while generating a new key pair.
- 55 Unsupported algorithm
- The specified algorithm is not supported or is unknown.
- 56 No passphrase
- The user has entered an empty passphrase. This error can occur when conventionally encrypting a file with an empty passphrase.
- 57 X.509 Certificate error
- An error has occurred while using or reading a X.509 certificate.
- 58 Bad session key used
- The specified session key is badly formed or is not the expected one.

- 1000 Invalid option specified
The specified option is invalid.
- 1001 No passphrase specified
No passphrase has been specified.
- 1002 No input file specified
EDGE is not able to find the input file to use.
- 1003 Unknown option
The specified option is unknown.
- 1005 Preference not found
The required preference cannot be found in the configuration file.
- 1006 Error while opening the log file
The log file cannot be opened.
- 1007 Error while writing the log file
An I/O error has occurred while trying to add data to the log file.
- 1008 Conflicts between options
This error occurs when incompatible options are used together. For example, this error could occur if a signed and encrypted message is requested with a detached signature.
- 1009 Error while trying to add keys to the default keyrings
An I/O error has occurred while trying to add keys to the default public or private keyring.
- 1010 Error while trying to remove a key
The specified key cannot be removed from the keyring or the specified key cannot be found in the keyring.

- 1012 Unknown argument

The specified argument is unknown.
- 1013 Missing parameter

The specified option or command needs an additional value.
- 1014 Key generation error

An error has occurred while generating a new key pair.
- 1015 Option is too long

The required or specified option is too long.
- 1016 Unknown option

The specified option is unknown.
- 1017 Operation not allowed

The license file you have purchased doesn't allow this operation. For example, if you have purchased a license file allowing you only to sign a message, you will not be able to encrypt a message.
- 1018 Error while trying to log into the Authora Sovereign Server

The specified server cannot be accessed or the specified username or password are bad.
- 1019 Operation stopped because of the BATCHMODE option

Additional information is requested from the user. The operation has been cancelled because of the use of the BATCHMODE option.

17 Appendix C – Compatible Error Codes

The following list contains error codes returned by EDGE when the COMPAT-ERRORS option is turned on.

The following errors are compatible with McAfee E-Business Server and PGP Command Line.

To turn this feature on, use the following command:

```
edge --compat-errors on
```

EDGE updates the configuration file to return these errors instead of the regular errors.

0	No error
1	Invalid file
2	File not found
3	Unknown file
4	Batchmode error
5	Bad argument
6	Process interrupted
7	Out of memory
8	Environment error
10	Key generation error
11	Non-existing key error
12	Keyring add error
13	Keyring extract error
14	Keyring edit error
15	Keyring view error
16	Keyring removal error
18	Key signature error or key signature revoke error
19	Key signature removal error

20	Signature error
21	Public-key encryption error
22	Encryption error
23	Compression error
30	Signature check error
31	Public-key decryption error
32	Decryption error
33	Decompression error
100	File wiping error
101	File parsing error

18 Appendix D - EDGE on z/OS

18.1 Introduction

EDGE is an OpenPGP compliant application allowing you to encrypt and/or digitally sign messages and to decrypt and verify digital signatures. It also allows you to manage your public and private encryption keys.

EDGE for z/OS brings these OpenPGP capabilities to the mainframe allowing users to exchange OpenPGP encrypted files with users on other platforms. For instance, mainframe-based EBCDIC files can be encrypted and then decrypted using OpenPGP compliant software on ASCII-based UNIX/PC platforms.

EDGE for z/OS requires the following operating environment:

- IBM mainframe capable of supporting z/OS 1.2 or later operating system
- z/OS 1.2 or later operating system
- Unix Systems Services (USS) with Enhanced ASCII support

EDGE for z/OS is implemented as a Unix System Services application. Once EDGE has been configured, most EDGE commands can be invoked via batch jobs or TSO commands. Users may also combine the USS, batch, and TSO capabilities with USS scripts to provide advanced usage scenarios as found in most UNIX systems.

One of the major differences in EDGE for z/OS is the EBCDIC / ASCII data formats. EDGE uses the Enhanced ASCII Support capabilities of z/OS to recognize the file type being processed and convert it to ASCII as needed. In most cases, the end user will simply have to “tag” the file as either ASCII (ISO8859-1) format or EBCDIC (IBM-1047) format. See the Usage section of this document for additional information.

18.2 Customization

The following process describes the steps necessary to customize EDGE for z/OS:

EDGE for z/OS requires that Unix Systems Services have the AUTOCVT enabled either globally or individually for selected EDGE users. The OMVS AUTOCVT option can be enabled either globally via BPXPRMxx by specifying:

```
AUTOCVT(ON)
```

This option may also be set temporarily via the operator command:

```
SETOMVS AUTOCVT=ON
```


If the global setting of this option is not practical at your site, `AUTOCVT` may also be enabled in the logon profile using the following command:

```
export _BPXX_AUTOCVT=on
```

EDGE requires the setting of the `PGPPATH` environment variable. If all users will share a global set of configuration files, the following command should be inserted into `/etc/profile` to tell EDGE to look in `/u/ibmuser` directory for EDGE configuration files:

```
export PGPPATH=/u/ibmuser
```

If individual users require separate EDGE configuration files, the following command should be inserted into `.profile` in the users home directory:

```
export PGPPATH=/u/ibmuser
```

EDGE for z/OS does not support the `--colors` option of EDGE. You should disable this functionality by entering the following command upon creation of a new EDGE configuration file:

```
edge --colors off
```

18.2.1 Verify Proper Installation

To verify proper installation of EDGE for z/OS, execute the following commands in sequence:

```
edge -v
Displays the basic license and configuration information
```

```
edge -vv
Displays the contents of the EDGE configuration file
```

```
cp edge.cfg test
Makes a copy of the configuration file for testing
```

```
edge -c test -z password
Encrypts the test file
```

```
edge test.pgp --secure-viewer -z password
Displays encrypted file using secure-viewer
```

```
edge test.pgp -z password
Decrypts the file
```

```
cmp edge.cfg test
Compares to the original file
```

The same IVP process should be completed from TSO by executing the commands in sequence from TSO Command Prompt (typically TSO option 6).

Note that EDGE TSO interface will append the `--batchmode` & `--force` options to eliminate the need for terminal intervention when communicating with TSO. With this said, some EDGE commands that require responses other than Y or N, must be completed via Unix Systems Services.

The same IVP process should be completed via batch by executing the `EDGEIVP` job that was sent in the `INSTLIB`.

Note that EDGE batch interface will append the `--batchmode` & `--force` options to eliminate the need for terminal intervention when communicating with TSO. With this said, some EDGE commands that require responses other than Y or N, must be completed via Unix Systems Services.

18.3 Usage

Although EDGE for z/OS can be invoked via USS, TSO, or batch, the following usage examples use the batch method. Most mainframe users will find this method the easiest to way to learn EDGE for z/OS.

All of the example jobs listed below have been included in the installation file:

```
AUTHORA.EDGE.V1R0.INSTLIB.
```

The examples listed below will guide you through the typical usage scenario of:

Importing:

Importing z/OS-based files into USS and tagging the file either EBCDIC or ASCII

Encrypting:

Transforming a text file into an encrypted PGP or ASC file

Transferring:

Methods of securely transmitting files to other users

Decrypting:

Transforming an encrypted file back into a text file

Exporting:

Exporting USS-based files back into z/OS & USS

Once the user has a thorough understanding of these basic processes, additional information is available in the EDGE User Manual regarding advanced procedures.

18.4 Displaying EDGE Configuration Information

The following examples will display basic EDGE configuration information and the contents of the EDGE configuration files:

18.4.1 Displaying EDGE Information

```
//EDGEMIS1 JOB CLASS=A,MSGCLASS=H
//*****
*****
//*          EDGEMIS1:          EDGE          -v          command
*
//*****
*****
//EDGE      EXEC EDGE,CMD='edge -v'
```

Output from EDGEMIS1 job:

```
edge - Encrypted Data Gateway Engine
Version 3.7
Copyright (C) 2002-2007 Authora Inc. & Veridis SA
All rights reserved.
```

```
Application directory:
/u/ibmuser/
```

```
Configuration file:
/u/ibmuser/edge.cfg
```

```
License file:
/u/ibmuser/edge.lic
```

```
Signature allowed
Encryption allowed
Decryption allowed
```

18.4.2 Displaying EDGE Configuration File

```
//EDGEMIS2 JOB CLASS=A,MSGCLASS=H
//*****
*****
//*          EDGEMIS2:          EDGE          -vv          command
*
//*****
*****
```

```
//EDGE      EXEC EDGE,CMD='edge -vv'
```

Output from EDGEMIS2 job:

```
edge - Encrypted Data Gateway Engine
Version 3.7
Copyright (C) 2002-2007 Authora Inc. & Veridis SA
All rights reserved.
```

```
PUBRING=/u/ibmuser/pubring.pgp
SECRING=/u/ibmuser/secring.pgp
COLORS=off
BATCHMODE=off
FORCE=off
```

18.5 Importing Files

The objective of the Importing process is to get the file into USS and tagged as either an EBCDIC or ASCII codeset. The following examples illustrate various scenarios for importing data into USS and getting the files tagged properly for use by EDGE for z/OS:

18.5.1 Importing a file from z/OS

```
//EDGEIMP1 JOB CLASS=A,MSGCLASS=H
//*****
*****
//* EDGEIMP1: Import MVS file to USS for EDGE processing
*
//*
*
//*      1)      Copy      file      to      USS      using      oput
*
//*      2)      Change   tag   to   EBCDIC   using   chtag
*
//*****
*****
//S1      EXEC PGM=IKJEFT01,REGION=0M
//SYSPROC DD DSN=SYS1.SBPXEXEC,DISP=SHR
//SYSTSPRT DD SYSOUT=*
//SYSTSIN DD *
oput 'ibmuser.sysout' '/u/ibmuser/edgeimp1' convert(no)
oshell chmod 600 /u/ibmuser/edgeimp1
oshell chtag -tc IBM-1047 /u/ibmuser/edgeimp1
```

18.5.2 Importing an Untagged USS File

```
//EDGEIMP2 JOB CLASS=A,MSGCLASS=H
//*****
*****
//* EDGEIMP2: Import untagged USS file for EDGE processing
*
//*          (although untagged, EBCDIC data is the default
codeset) *
//*
*
//*          1)          Copy          file          using          cp
*
//*          2)          Change          read/write          permissions          using          chmod
*
//*          3)          Change          tag          to          EBCDIC          using          chtag
*
//*****
*****
//S1          EXEC PGM=IKJEFT01,REGION=0M
//SYSPROC DD DSN=SYS1.SBPXEXEC,DISP=SHR
//SYSTSPRT DD SYSOUT=*
//SYSTSIN DD *
oshell cp /etc/profile /u/ibmuser/edgeimp2
oshell chmod 600 /u/ibmuser/edgeimp2
oshell chtag -tc IBM-1047 /u/ibmuser/edgeimp2
```

18.5.3 Importing an EBCDIC USS File

```
//EDGEIMP3 JOB CLASS=A,MSGCLASS=H
//*****
*****
//* EDGEIMP3: Import EBCDIC USS file for EDGE processing
*
//*
*
//*          1)          Copy          file          using          cp
*
//*          2)          Change          read/write          permissions          using          chmod
*
//*          3)          Change          tag          to          EBCDIC          using          chtag
*
//*****
*****
//S1          EXEC PGM=IKJEFT01,REGION=0M
//SYSPROC DD DSN=SYS1.SBPXEXEC,DISP=SHR
```

```
//SYSTSPRT DD SYSOUT=*
//SYSTSIN DD *
oshell cp /u/ibmuser/ebcdic.txt /u/ibmuser/edgeimp3
oshell chmod 600 /u/ibmuser/edgeimp3
oshell chtag -tc IBM-1047 /u/ibmuser/edgeimp3
```

18.5.4 Importing an ASCII USS File

```
//EDGEIMP4 JOB CLASS=A,MSGCLASS=H
//*****
//* EDGEIMP4: Import ASCII USS file for EDGE processing
*
//*
*
//* 1) Copy file using cp
*
//* 2) Change read/write permissions using chmod
*
//* 3) Change tag to EBCDIC using chtag
*
//*****
//S1 EXEC PGM=IKJEFT01,REGION=0M
//SYSPROC DD DSN=SYS1.SBPXEXEC,DISP=SHR
//SYSTSPRT DD SYSOUT=*
//SYSTSIN DD *
oshell cp /u/ibmuser/ascii.txt /u/ibmuser/edgeimp4
oshell chmod 600 /u/ibmuser/edgeimp4
oshell chtag -tc ISO8859-1 /u/ibmuser/edgeimp4
```

18.5.5 Importing an Untagged/EBCDIC file and Converting to ASCII

```
//EDGEIMP5 JOB CLASS=A,MSGCLASS=H
```

```
//*****  
*****  
/* EDGEIMP5: Import untagged/EBCDIC file and convert to ASCII  
for      *  
/*                               EDGE    processing  
*  
/*  
*  
/*      1)      Copy      and      convert      file      using      iconv  
*  
/*      2)      Change      read/write      permissions      using      chmod  
*  
//*****  
*****  
//S1      EXEC PGM=IKJEFT01,REGION=0M  
//SYSPROC DD DSN=SYS1.SBPXEXEC,DISP=SHR  
//SYSTSPRT DD SYSOUT=*  
//SYSTSIN DD *  
oshell iconv -f IBM-1047 -t ISO8859-1 -T ebcdic.txt > edgeimp5  
oshell chmod 600 /u/ibmuser/edgeimp5
```

18.6 Encrypting Files

The following examples illustrate various scenarios for encrypting files with EDGE for z/OS:

18.6.1 -c

```
//EDGEENC1 JOB CLASS=A,MSGCLASS=H
//*****
*****
//* Cleanup: Remove old PGP file (if present)
*
//*****
*****
//CLEANUP EXEC PGM=IKJEFT01,REGION=0M
//SYSPROC DD DSN=SYS1.SBPXEXEC,DISP=SHR
//SYSTSPRT DD SYSOUT=*
//SYSTSIN DD *
oshell rm edgeimpl.pgp
//*****
*****
//* EDGEENC1: EDGE -c command
*
//*****
*****
//EDGE EXEC EDGE,CMD='edge -c edgeimpl -z password'
```

Output from EDGEENC1 job:

```
edge - Encrypted Data Gateway Engine
Version 3.7
Copyright (C) 2002-2007 Authora Inc. & Veridis SA
All rights reserved.
```

```
Encrypting file:
Clear file: edgeimpl
Cipher file: edgeimpl.pgp
```

```
100% |*****|
```

18.6.2 -ct

```
//EDGEENC2 JOB CLASS=A,MSGCLASS=H
//*****
*****
```



```

/* Cleanup: Remove old PGP file (if present)
*
/*****
//CLEANUP EXEC PGM=IKJEFT01,REGION=0M
//SYSPROC DD DSN=SYS1.SBPXEXEC,DISP=SHR
//SYSTSPRT DD SYSOUT=*
//SYSTSIN DD *
oshell rm edgeimp2.pgp
/*****
/* EDGEENC2: EDGE -ct command
*
/*****
//EDGE EXEC EDGE,CMD='edge -ct edgeimp2 -z password'

```

Output from EDGEENC2 job:

```

edge - Encrypted Data Gateway Engine
Version 3.7
Copyright (C) 2002-2007 Authora Inc. & Veridis SA
All rights reserved.

```

```

Encrypting file:
Clear file: edgeimp2
Cipher file: edgeimp2.pgp

```

```
100% |*****|
```

18.6.3 -cta

```

//EDGEENC3 JOB CLASS=A,MSGCLASS=H
/*****
/* Cleanup: Remove old PGP file (if present)
*
/*****
//CLEANUP EXEC PGM=IKJEFT01,REGION=0M
//SYSPROC DD DSN=SYS1.SBPXEXEC,DISP=SHR
//SYSTSPRT DD SYSOUT=*
//SYSTSIN DD *
oshell rm edgeimp3.asc
/*****
/* EDGEENC3: EDGE -cta command
*
/*****
//EDGE EXEC EDGE,CMD='edge -cta edgeimp3 -z password'

```

Output from EDGEENC3 job:

```
edge - Encrypted Data Gateway Engine
Version 3.7
Copyright (C) 2002-2007 Authora Inc. & Veridis SA
All rights reserved.
```

```
Encrypting file:
  Clear file: edgeimp3
  Cipher file: edgeimp3.asc
```

```
100% |*****|
```

18.6.4 --encrypt

```
//EDGEENC4 JOB CLASS=A,MSGCLASS=H
//*****
//*      Cleanup:      Remove      old      PGP      file      (if      present)
*
//*****
//CLEANUP EXEC PGM=IKJEFT01,REGION=0M
//SYSPROC DD DSN=SYS1.SBPXEXEC,DISP=SHR
//SYSTSPRT DD SYSOUT=*
//SYSTSIN DD *
oshell rm edgeimp4.pgp
//*****
//*      EDGEENC4:      EDGE      -c      command
*
//*****
//EDGE EXEC EDGE,CMD='edge --encrypt edgeimp4 --user
mvsbatch'
```

Output from EDGEENC4 Job:

```
edge - Encrypted Data Gateway Engine
Version 3.7
Copyright (C) 2002-2007 Authora Inc. & Veridis SA
All rights reserved.
```

```
Encrypting file:
  Clear file: edgeimp4
  Cipher file: edgeimp4.pgp
```

```
Encrypting file with the following public keys:
  0x968D9CDD mvsbatch
```

```
Checking ADK...
```

```
100% |*****|
```

18.6.5 --encrypt armor text

```
//EDGEENC5 JOB CLASS=A,MSGCLASS=H
//*****
*****
//* Cleanup: Remove old ASC file (if present)
*
//*****
*****
//CLEANUP EXEC PGM=IKJEFT01,REGION=0M
//SYSPROC DD DSN=SYS1.SBPXEXEC,DISP=SHR
//SYSTSPRT DD SYSOUT=*
//SYSTSIN DD *
oshell rm edgeimp5.asc
//*****
*****
//* EDGEENC5: EDGE --encrypt command w/--armor --text
*
//*****
*****
//EDGE EXEC EDGE,
// CMD='edge --encrypt edgeimp5 --user mvbatch --armor
--text'
```

Output from EDGEENC5 job:

```
edge - Encrypted Data Gateway Engine
Version 3.7
Copyright (C) 2002-2007 Authora Inc. & Veridis SA
All rights reserved.
```

```
Encrypting file:
Clear file: edgeimp5
Cipher file: edgeimp5.asc
```

```
Encrypting file with the following public keys:
0x968D9CDD mvbatch
```

```
Checking ADK...
```

```
100% |*****|
```

18.7 Transferring Files

The following examples illustrate how to transfer PGP & ASC files to other platforms:

18.7.1 PGP transfers

```
//EDGEFTP1 JOB CLASS=A,MSGCLASS=H
//*****
*****
//*  EDGEFTP1:  FTP  encrypted  PGP  file  to  other  platforms
*
//*              (PGP files MUST be sent using BINARY transfer!)
*
//*****
*****
//FTP          EXEC PGM=FTP,REGION=0M
//OUTPUT      DD  SYSOUT=*
//INPUT       DD  *
208.234.5.106
userid
password
bin
lcd /u/ibmuser
put edgeimp1.pgp
quit
```

18.7.2 ASC Transfers

```
//EDGEFTP2 JOB CLASS=A,MSGCLASS=H
//*****
*****
//*  EDGEFTP2:  FTP  encrypted  ASC  file  to  other  platforms
*
//*              (ASC files must be converted to EBCDIC before
transfer) *
//*****
*****
//ICONV       EXEC PGM=IKJEFT01,REGION=0M
//SYSPROC    DD  DSN=SYS1.SBPXEXEC,DISP=SHR
//SYSTSPT    DD  SYSOUT=*
//SYSTSIN    DD  *
oshell iconv -t IBM-1047 -f ISO8859-1 -T edgeimp3.asc >
edgeimp3.asc.e
oshell chmod 600 /u/ibmuser/edgeimp3
//FTP        EXEC PGM=FTP,REGION=0M
//OUTPUT     DD  SYSOUT=*
//INPUT      DD  *
```

```

208.234.5.106
userid
password
lcd /u/ibmuser
put edgeimp3.asc.e
quit

```

18.8 Decrypting Files

The following examples illustrate how to decrypt PGP & ASC files using EDGE for z/OS:

18.8.1 PGP Files

```

//EDGEDEC1 JOB CLASS=A,MSGCLASS=H
//*****
*****
//* Cleanup: Remove old PGP file (if present)
*
//*****
*****
//CLEANUP EXEC PGM=IKJEFT01,REGION=0M
//SYSPROC DD DSN=SYS1.SBPXEXEC,DISP=SHR
//SYSTSPRT DD SYSOUT=*
//SYSTSIN DD *
oshell rm edgeimp1
//*****
*****
//* EDGEDEC1: EDGE decrypt command
*
//*****
*****
//EDGE EXEC EDGE,CMD='edge edgeimp1.pgp -z password'

```

Output from EDGEDEC1 job:

```

edge - Encrypted Data Gateway Engine
Version 3.7
Copyright (C) 2002-2007 Authora Inc. & Veridis SA
All rights reserved.

```

```

Decrypting file:
Cipher file: edgeimp1.pgp

```

18.8.2 ASC Files

```
//EDGEDEC3 JOB CLASS=A,MSGCLASS=H
//*****
*****
//* Cleanup: Remove old ASC file (if present)
*
//*****
*****
//CLEANUP EXEC PGM=IKJEFT01,REGION=0M
//SYSPROC DD DSN=SYS1.SBPXEXEC,DISP=SHR
//SYSTSPRT DD SYSOUT=*
//SYSTSIN DD *
oshell rm edgeimp3
//*****
*****
//* EDGEDEC3: EDGE decrypt command
*
//*****
*****
//EDGE EXEC EDGE,CMD='edge edgeimp3.asc -z password'
```

Output from EDGEDEC3 job:

```
edge - Encrypted Data Gateway Engine
Version 3.7
Copyright (C) 2002-2007 Authora Inc. & Veridis SA
All rights reserved.
```

```
Decrypting file:
Cipher file: edgeimp3.asc
```

18.8.3 Viewing Files with secure-viewer

```
//EDGEDEC6 JOB CLASS=A,MSGCLASS=H
//*****
*****
//* EDGEDEC6: EDGE decrypt and view with secure-viewer
*
//*****
*****
//EDGE EXEC EDGE,
// CMD='edge edgeimp3.asc --passphrase password --secure-
viewer'
```

Output from EDGEDEC6 job:

```

edge - Encrypted Data Gateway Engine
Version 3.7
Copyright (C) 2002-2007 Authora Inc. & Veridis SA
All rights reserved.

```

```

Decrypting file:
  Cipher file: edgeimp3.asc

```

```

This message is marked "for your eyes only". Display it now
(y/N)? Y (force)

```

```

(data from file)

```

18.9 Exporting Files

The following example illustrates how to transfer the decrypted files back to z/OS:

```

//EDGEEXP1 JOB CLASS=A,MSGCLASS=H
//*****
*****
//*   EDGEEXP1:   Export   USS   file   to   MVS   using   oget
*
//*****
*****
//S1          EXEC PGM=IKJEFT01,REGION=0M
//SYSPROC    DD   DSN=SYS1.SBPXEXEC,DISP=SHR
//SYSTSPRT   DD   SYSOUT=*
//SYSTSIN    DD   *
oshell iconv -t IBM-1047 -f ISO8859-1 -T edgeimp1 > edgeimp1.e
oget '/u/ibmuser/edgeimp1.e' 'ibmuser.sysout' convert(no)

```