PGP SupportPac for IBM Integration Bus v10

Part-1: A User Guide for PGP SupportPac Installation, Configuration, Key Management and Messageflow Development

By

Dipak Kumar Pal
(dipakpal.opentech@gmail.com)

MyOpenTech
(PGP SupportPac)

Summary

This article is the first in a multi-part series of articles describing PGP security implementation in IBM Integration Bus v10. This series of articles introduces an industry standard solution to Data Security in IBM Integration Bus, enforcing data confidentiality and integrity by implementing PGP cryptographic solution. This solution is developed as a custom pluggable feature (or SupportPac) of IBM Integration Bus v10, attached with this article as an additional artifact. This article describes a step-by-step user guide of **PGP SupportPac** (v1.0.0.1) installation, configuration including PGP key/repository management and application development. Assuming intended readers (Architects/Designers/Developers) are familiar with basics of PGP encryption, decryption and signature processes, this article does not discuss PGP basics. However it provides a list of useful resources at reference section.

Introduction

Security facilities in IBM Integration Bus are typically based on Websphere MQ security, transport layer security (e.g. SSL/TLS) provided by underlying transport mechanism, and Access Controls (e.g. Authentication and Authorization) mechanism powered by internal (broker’s security manager) and external security providers (e.g. WS-Trust V1.3 compliant security token servers, Tivoli Federated Identity Manager [TIFM], Lightweight Directory Access Protocol [LDAP]). If the message flow implements Web Services using SOAP nodes, WS-Security standards can be implemented through appropriate Policy sets and bindings.

But in today’s enterprise integration world, Webservice technology is not considered as a preferred solution for asynchronous and one-way data communication especially while dealing with large volume of data. Apart from WS-Security standard (**which is applicable for Web services only**), IBM Integration Bus does not provide any in-built solution for application layer security enforcing data confidentiality and integrity. It requires implementing an industry standard cryptographic solution to enforce data security.

PGP (Pretty Good Privacy) is a widely used cryptographic solution for data communication. It was created by Phil Zimmermann in 1991. PGP follows the OpenPGP standard (RFC 4880) for encrypting and decrypting data. Besides data confidentiality and integrity, PGP also supports strong data compression.

PGP SupportPac (*version 1.0.0.1***) for IBM Integration Bus v10** implements PGP cryptographic solution providing encryption, decryption, and signature functionalities as an extended feature (SupportPac). It leverages Bouncy Castle PGP Java libraries for core PGP functionalities. Bouncy Castle is a Java based open source solution for PGP implementation, available under MIT License.

This **SupportPac** ships with a Java based command-line tool (**pgpkeytool**) for PGP key generation and key management. You do not need any third-party open source or commercial tool for PGP key management.

Installation and Configuration

Following set of variables are used throughout the article, because it varies from platform to platform. Make sure you set correct and suitable directory path as per your system.

Table-1: List of variables used in this article.

S/N	Variable Name	Windows	UNIX	Description
1	TOOLKIT_INSTALL_DIR	C:\IBM\IIB\10.0.0.4\tools		WMB Toolkit v10 installation directory.
2	MQSI_ROOT_DIR	C:\IBM\IIB\10.0.0.4\server		WMB v10 installation directory.
3	MQSI_JRE_HOME	C:\IBM\IIB\10.0.0.4\common\jdk\jre		MQSI Java Runtime Environment home directory.
4	MQSI_USR_LILPATH	C:\IBM\IIB\USR\LIL		Directory that contains the user-defined extension libraries. This should be customized based on your system/platform.
5	KEY_REPOSITORY	C:\PGP\KeyRepository	/var/pgp/keyrepository	Directory that contains individual private/public key files.
6	SDR_KEY_REPOSITORY	C:\PGP\KeyRepository\Sender	/var/pgp/keyrepository/sender	Directory that contains key repository files for Sender (PGP Encrypter) messageflow.
7	RCVR_KEY_REPOSITORY	C:\PGP\KeyRepository\Recipient	/var/pgp/keyrepository/recipient	Directory that contains key repository files for Recipient (PGP Decrypter) messageflow.

Download **PGP SupportPac v1.0.0.1.zip** from GitHub repository (https://github.com/dipakpal/MyOpenTech-PGP-SupportPac/tree/master/binary/IIBv10.0.0.4) and unzip it in a temporary directory. Zip file contains following directory structure and files.

```
PGP SupportPac v1.0.0.1/
                          lib/
                             bcpg-jdk16-146.jar
                             bcprov-ext-jdk16-146.jar
                             com.ibm.broker.supportpac.PGP.jar
                          plugins/
                                  PGPSupportPac_1.0.0.1.jar
```

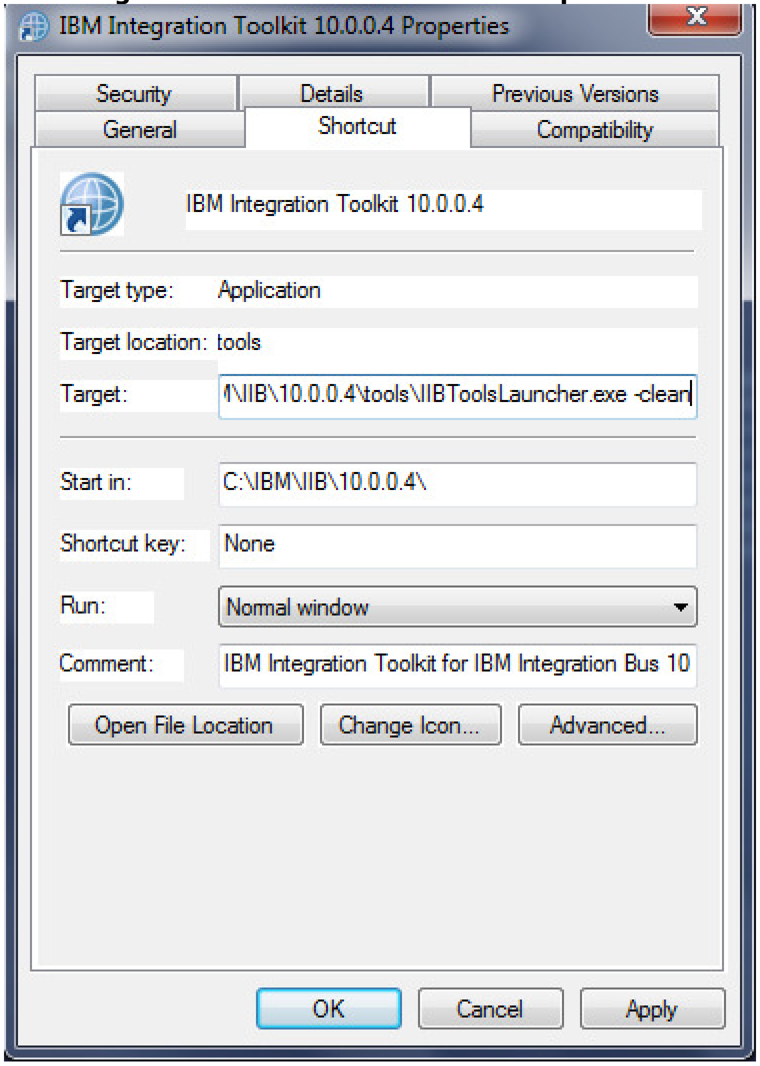
This supportPac consists of following two components.

- PGP SupportPac plugins for IBM Integration Bus toolkit.
- PGP SupportPac runtime libraries (.jar files) for IBM Integration Bus.

Install PGP SupportPac plugins for IBM Integration Bus (v10) toolkit

Copy **PGPSupportPac_1.0.0.1.jar** into IBM Integration Bus Toolkit's plugins directory (i.e. **$TOOLKIT_INSTALL_DIR**/plugins). Restart the toolkit with **–clean** option in order to make the PGP Encrypter/Decrypter nodes shown up in the palette.

Figure-1: Restart IBM Integration Bus Toolkit with –clean option.



IBM Integration Toolkit 10.0.0.4

Target type: Application

Target location: tools

Target: M\IIB\10.0.0.4\tools\IIBToolsLauncher.exe -clean

Start in: C:\IBM\IIB\10.0.0.4\

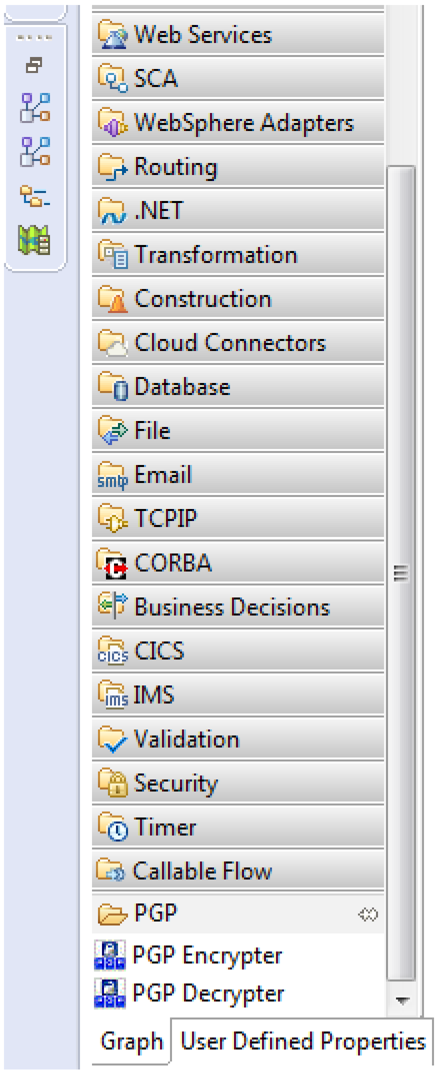
Shortcut key: None

Run: Normal window

Comment: IBM Integration Toolkit for IBM Integration Bus 10

Once PGP supportPac plugins is applied to the IBM Integration Bus Toolkit, PGP Encrypter/Decrypter nodes will be available in the PGP drawer of the message flow node palette.

Figure-2: PGP drawer of the message flow node palette.



Install PGP supportPac runtime libraries (jar files) on IBM Integration Bus

Install the supportPac runtime libraries (.jar files) on the broker on which you want to configure it. Following steps describe how to install and configure these supportPac runtime libraries.

Step 1: Create a directory (**$MQSI_USR_LILPATH**) if you do not already have one for this purpose. Add the directory to the broker's LILPATH by using the **mqsichangebroker** command. Make sure you stop the broker and then execute this command.

Sample command:

```
mqsichangebroker WMBBROKER -l C:\IBM\IIB\USR\LIL
```

Step 2: Copy following jar files into **$MQSI_USR_LILPATH** directory you created at step 1.

bcpg-jdk16-146.jar
bcprov-ext-jdk16-146.jar
com.ibm.broker.supportpac.PGP.jar

Note: Do not put these .jar files in the IBM Integration Bus installation directory, because they might be overwritten by the broker. Make sure broker has access to these jar files. For example, on Linux or UNIX, use the **chmod 755 *.jar** command on the file.

Step 3: In comply with the United States of America export restrictions, IBM's SDKs/JREs ship with strong but limited jurisdiction policy files. Unlimited jurisdiction policy files can be obtained from the IBM site (https://www14.software.ibm.com/webapp/iwm/web/preLogin.do?source=jcesdk).
To work with strong encryption and larger key size, replace following two jar files in **$MQSI_JRE_HOME/lib/security** with following unrestricted JCE policy jar files obtained from IBM site.

local_policy.jar
US_export_policy.jar

Step 4: Start the broker and it is now ready for messageflow deployment, containing PGP Encrypter/Decrypter nodes.

PGP Key pair generation and Key repository management

Examples in this article consist of a PGP Encrypter messageflow (Sender application) and a PGP Decrypter messageflow (Recipient application), use two separate pair of PGP key repositories.

PGP Private Key Repository ($SDR_KEY_REPOSITORY/private.pgp): PGP private key repository is a container (file) contains multiple private keys in binary data format. Once you create a PGP key pair, make sure you import the private key into private key repository file.

PGP Public Key Repository ($SDR_KEY_REPOSITORY/public.pgp): PGP public key repository is a container (file) contains multiple public keys in binary data format. Once you create a PGP key pair or received public keys from your partner (sender or recipient) applications, make sure you import public keys into public key repository file.

Following steps illustrate how to generate PGP Key pairs and manage key repositories. Refer to **pgpkeytool** manual for installation, environment setup and supported command details.

Step 1: Generate PGP key pairs

Following table illustrates a list of various key generation parameters for both the PGP key pairs used by Encrypter/Decrypter (Sender/Recipient) messageflows. Refer to fourth article (Part-4) of this series for installation and configuration guide of **pgpkeytool**.

Note: Make sure you use key generation parameters as per your organization standard.

Table-2: List of various key generation parameters.

S/N	Key Parameters	PGP Encrypter messageflow (Sender application)	PGP Decrypter messageflow (Recipient application)
1	**Key User Id**	Sender <sender-pgp-keys@ibm.com>	Recipient <recipient-pgp-keys@ibm.com>
2	**PGP Signature Key Algorithms**	DSA	DSA
3	**PGP Encryption Key Algorithm**	ELG (El Gamal)	RSA
4	**Private key passphrase**	sdrpassphrase	rcvrpassphrase
5	**ASCII Armored**	true	true
6	**Key size (DSA)**	1024	1024
7	**Key size (RSA)**	N/A	2048
8	**Key size (ELG)**	2048	N/A
9	**Cipher Algorithm**	AES_256	AES_256
10	**Private key file**	**$KEY_REPOSITORY**/SenderSecretKey.asc	**$KEY_REPOSITORY**/RecipientSecretKey.asc
11	**Public key file**	**$KEY_REPOSITORY**/SenderPublicKey.asc	**$KEY_REPOSITORY**/RecipientPublicKey.asc
12	**Private key repository file**	**$SDR_KEY_REPOSITORY**/private.pgp	**$RCVR_KEY_REPOSITORY**/private.pgp
13	**Public key repository file**	**$SDR_KEY_REPOSITORY**/public.pgp	**$RCVR_KEY_REPOSITORY**/public.pgp

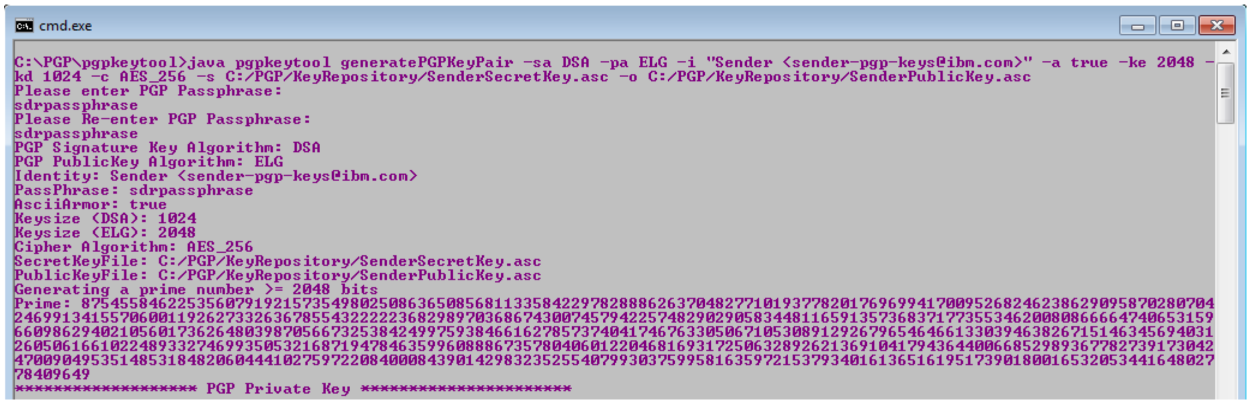
PGP key generation command for Sender's PGP key pair.

```
java pgpkeytool generatePGPKeyPair -sa DSA -pa ELG -i "Sender <sender-pgp-
keys@ibm.com>" -a true -ke 2048 -kd 1024 -c AES_256 -s
C:/PGP/KeyRepository/SenderSecretKey.asc -o
C:/PGP/KeyRepository/SenderPublicKey.asc
```

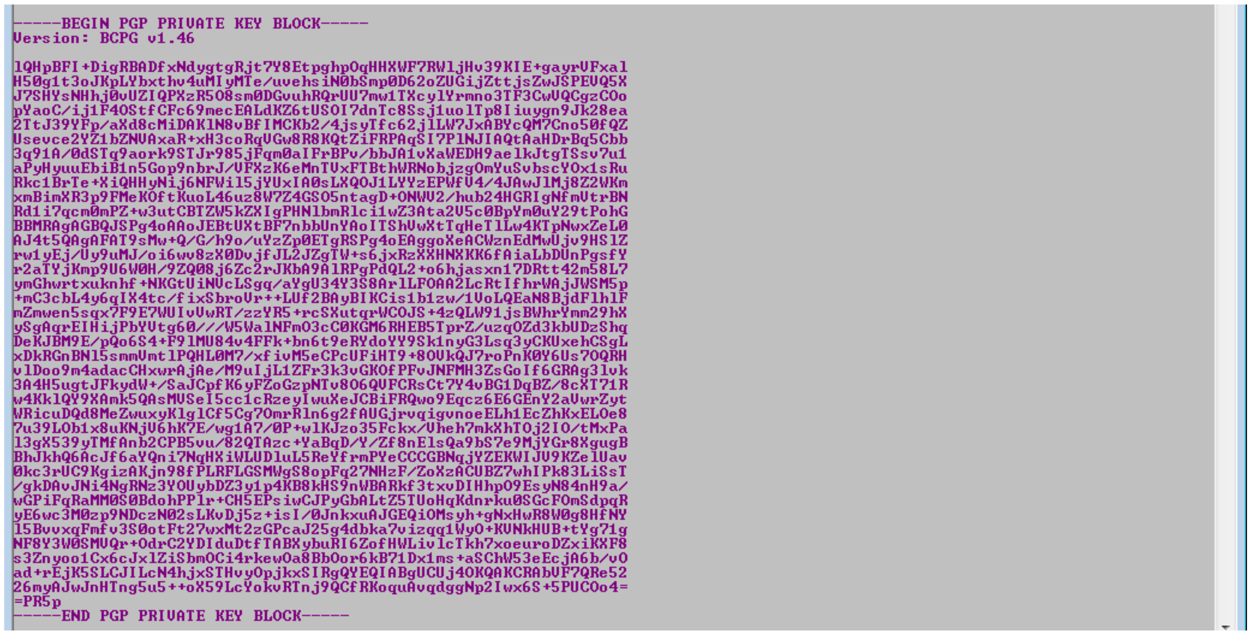
PGP key generation command for Recipient's PGP key pair.

```
java pgpkeytool generatePGPKeyPair -sa DSA -pa RSA -i "Recipient <recipient-pgp-
keys@ibm.com>" -a true -kr 2048 -kd 1024 -c AES_256 -s
C:/PGP/KeyRepository/RecipientSecretKey.asc -o
C:/PGP/KeyRepository/RecipientPublicKey.asc
```

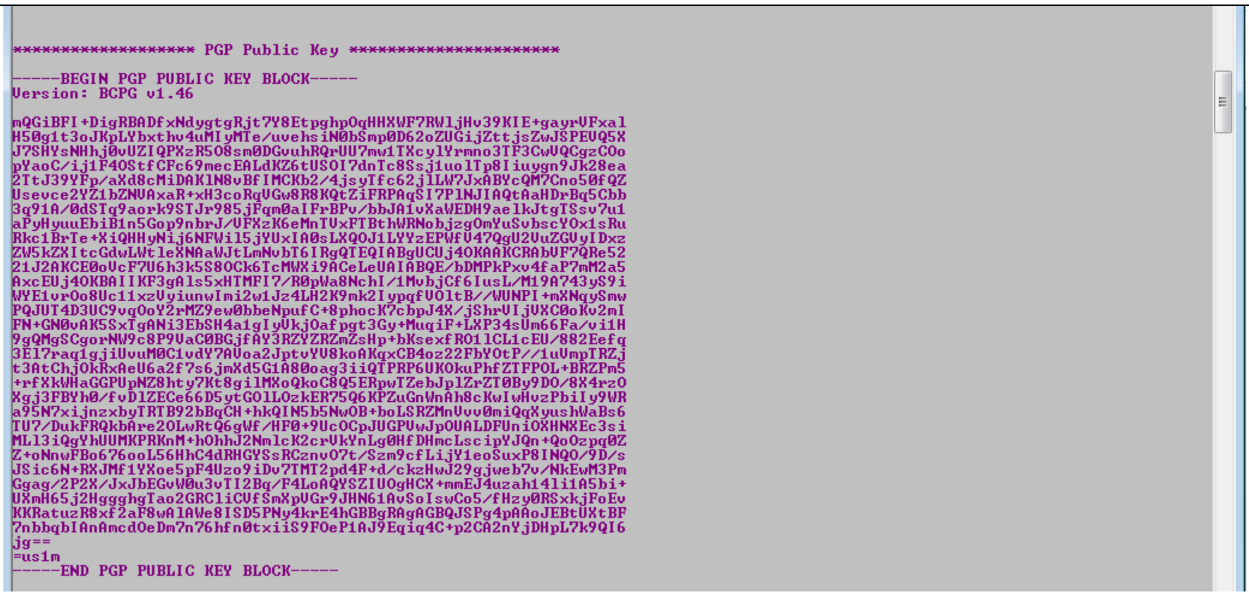
Figure-3: pgpkeytool screen-shot of PGP key pair generation in Windows system.



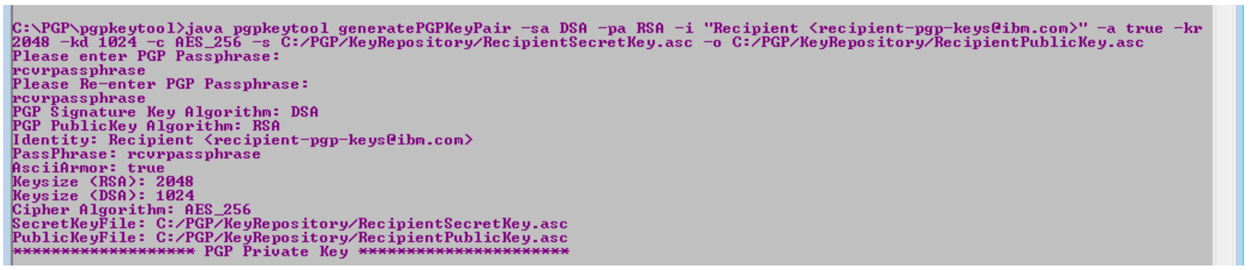
```
C:\PGP\pgpkeytool>java pgpkeytool generatePGPKeyPair -sa DSA -pa ELG -i "Sender <sender-pgp-keys@ibm.com>" -a true -ke 2048 -
kd 1024 -c AES_256 -s C:/PGP/KeyRepository/SenderSecretKey.asc -o C:/PGP/KeyRepository/SenderPublicKey.asc
Please enter PGP Passphrase:
sdrpassphrase
Please Re-enter PGP Passphrase:
sdrpassphrase
PGP Signature Key Algorithm: DSA
PGP PublicKey Algorithm: ELG
Identity: Sender <sender-pgp-keys@ibm.com>
PassPhrase: sdrpassphrase
AsciiArmor: true
Keysize (DSA): 1024
Keysize (ELG): 2048
Cipher Algorithm: AES_256
SecretKeyFile: C:/PGP/KeyRepository/SenderSecretKey.asc
PublicKeyFile: C:/PGP/KeyRepository/SenderPublicKey.asc
Generating a prime number >= 2048 bits
Prime: 87545584622535607919215735498025086365085681133584229782888862637048277101937782017696994170095268246238629095870280704
2469913415570600119262733263678554322223682989703686743007457942257482902905834481165913573683717735534620080866664740653159
66098629402105601736264803987056673253842499759384661627857374041746763305067105308912926796546466133039463826715146345694031
26050616610224893327469935053216871947846359960888673578040601220468169317250632892621369104179436440066852989367782739173042
47009049535148531848206044410275972208400084390142983235255407993037599581635972153793401613651619517390180016532053441648027
78409649
******************** PGP Private Key ***********************
```



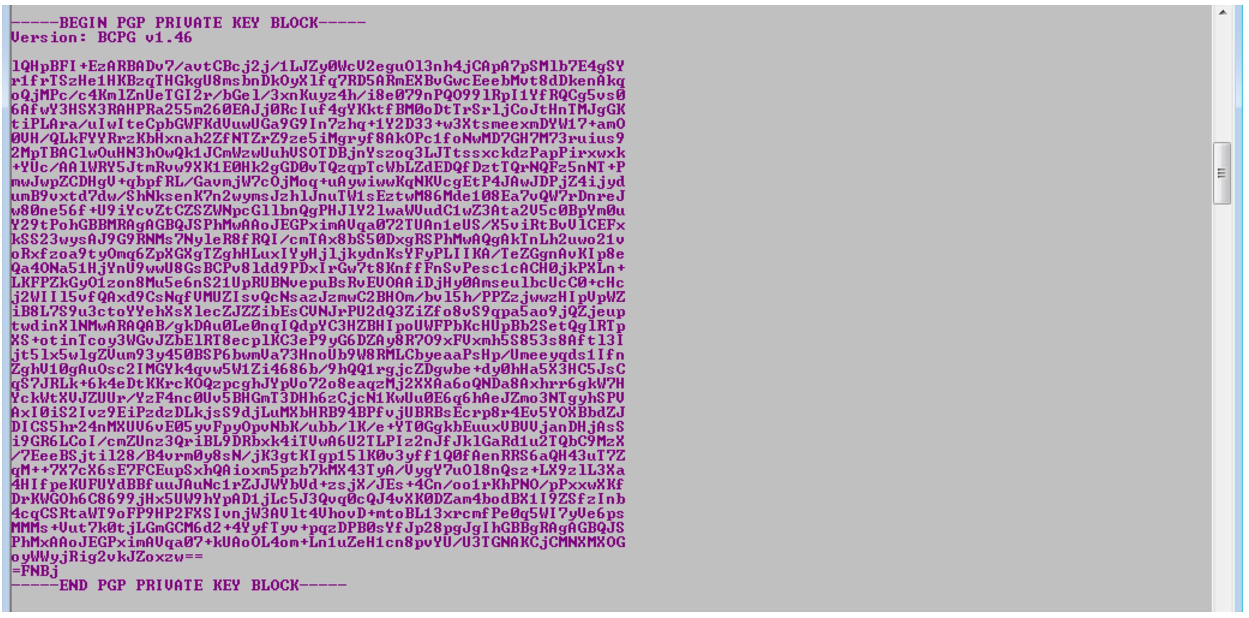
```
-----BEGIN PGP PRIVATE KEY BLOCK-----
Version: BCPG v1.46

lQHpBFI+DigRBADfxNdygtgRjt7Y8EtpghpOqHHXWF7RWljHv39KIE+gayrVFxal
H50g1t3oJKpLYbxthv4uMIyMTe/uvehsiN0bSmp0D62oZUGijZttjsZwJSPEVQ5X
J7SHYsNHhj0vUZIQPXzR5O8sm0DGvuhRQrUU7mw1TXcylYrmno3TF3CwVQCgzCOo
pYaoC/ij1F4OStfCFc69mecEALdKZ6tUSOI7dnTc8Ssj1uolTp8Iiuygn9Jk28ea
2TtJ39YFp/aXd8cMiDAKlN8vBfIMCKb2/4jsyTfc62jlLW7JxABYcQM7Cno50fQZ
Usevce2YZ1bZNVAxaR+xH3coRqVGw8R8KQtZiFRPAqSI7PlNJIAQtAaHDrBq5Cbb
3q91A/0dSTq9aork9STJr985jFqm0aIFrBPv/bbJA1vXaWEDH9aelkJtgTSsv7u1
aPyHyuuEbiB1n5Gop9nbrJ/VFXzK6eMnTVxFTBthWRNobjzgOmYuSvbscYOx1sRu
Rkc1BrTe+XiQHHyNij6NFWil5jYUxIA0sLXQOJ1LYYzEPWfV4/4JAwJlMj8Z2WKm
xmBimXR3p9FMeKOftKuoL46uz8W7Z4GSO5ntagD+ONWV2/hub24HGRIgNfmVtrBN
Rd1i7qcm0mPZ+w3utCBTZW5kZXIgPHNlbmRlci1wZ3Ata2V5c0BpYm0uY29tPohG
BBMRAgAGBQJSPg4oAAoJEBtUXtBF7nbbUnYAoITShVwXtTqHeTlLw4KTpNwxZeL0
AJ4t5QAgAFAT9sMw+Q/G/h9o/uYzZp0ETgRSPg4oEAggoXeACWznEdMwUjv9HSlZ
rw1yEj/Uy9uMJ/oi6wv8zX0DvjfJL2JZgTW+s6jxRzXXHNXKK6fAiaLbDUnPgsfY
r2aTYjKmp9U6W0H/9ZQ08j6Zc2rJKbA9AlRPgPdQL2+o6hjasxn17DRtt42m58L7
ymGhwrtxuknhf+NKGtUiNVcLSgq/aYgU34Y3S8ArlLFOAA2LcRtIfhrWAjJWSM5p
+mC3cbL4y6qIX4tc/fixSbroVr++LUf2BAyBIKCis1b1zw/1VoLQEaN8BjdFlhlF
mZmwen5sqx7F9E7WUIvVwRT/zzYR5+rcSXutqrWCOJS+4zQLW91jsBWhrYmm29hX
ySgAqrEIHijPbYVtg60///W5WalNFmO3cC0KGM6RHEB5TprZ/uzqOZd3kbUDzShq
DeKJBM9E/pQo6S4+F9lMU84v4FFk+bn6t9eRYdoYY9Sk1nyG3Lsq3yCKUxehCSgL
xDkRGnBNl5smmVmtlPQHL0M7/xfivM5eCPcUFiHT9+8OVkQJ7roPnK0Y6Us7OQRH
vlDoo9m4adacCHxwrAjAe/M9uIjL1ZFr3k3vGKOfPFvJNFMH3ZsGoIf6GRAg3lvk
3A4H5ugtJFkydW+/SaJCpfK6yFZoGzpNTv8O6QVFCRsCt7Y4vBG1DqBZ/8cXT71R
w4KklQY9XAmk5QAsMVSeI5cc1cRzeyIwuXeJCBiFRQwo9Eqcz6E6GEnY2aVwrZyt
WRicuDQd8MeZwuxyKlglCf5Cg7OmrRln6g2fAUGjrvqigvnoeELh1EcZhKxELOe8
7u39LOb1x8uKNjV6hK7E/wg1A7/0P+wlKJzo35Fckx/Vheh7mkXhTOj2IO/tMxPa
l3gX539yTMfAnb2CPB5vu/82QTAzc+YaBqD/Y/Zf8nElsQa9bS7e9MjYGr8XgugB
BhJkhQ6AcJf6aYQni7NqHXiWLUDluL5ReYfrmPYeCCCGBNqjYZEKWIJV9KZelUav
0kc3rUC9KgizAKjn98fPLRFLGSMWgS8opFq27NHzF/ZoXzACUBZ7whIPk83LiSsT
/gkDAvJNi4NgRNz3YOUybDZ3y1p4KB8kHS9nWBARkf3txvDIHhpO9EsyN84nH9a/
wGPiFqRaMM0S0BdohPPlr+CH5EPsiwCJPyGbALtZ5TUoHqKdnrku0SGcFOmSdpqR
yE6wc3M0zp9NDczN02sLKvDj5z+isI/0JnkxuAJGEQiOMsyh+gNxHwR8W0g8HfNY
l5BvvxqFmfv3S0otFt27wxMt2zGPcaJ25g4dbka7vizqq1WyO+KVNkHUB+tYg71g
NF8Y3W0SMVQr+OdrC2YDIduDtfTABXybuRI6ZofHWLivlcTkh7xoeuroDZxiKXF8
s3Znyoo1Cx6cJxlZiSbmOCi4rkewOa8BbOor6kB71Dx1ms+aSChW53eEcjA6b/vO
ad+rEjK5SLCJILcN4hjxSTHvyOpjkxSIRgQYEQIABgUCUj4OKQAKCRAbVF7QRe52
26myAJwJnHTng5u5++oX59LcYokvRTnj9QCfRKoquAvqdggNp2Iwx6S+5PUCOo4=
=PR5p
-----END PGP PRIVATE KEY BLOCK-----
```



```
******************* PGP Public Key **********************

-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: BCPG v1.46

mQGiBFI+DigRBADfxNdygtgRjt7Y8EtpghpOqHHXWF7RWljHv39KIE+gayrVFxal
H50g1t3oJKpLYbxthv4uMIyMTe/uvehsiN0bSmp0D62oZUGijZttjsZwJSPEVQ5X
J7SHYsNHhj0vUZIQPXzR5O8sm0DGvuhRQrUU7mw1TXcylYrmno3TF3CwVQCgzCOo
pYaoC/ij1F4OStfCFc69mecEALdKZ6tUSOI7dnTc8Ssj1uolTp8Iiuygn9Jk28ea
2TtJ39YFp/aXd8cMiDAKlN8vBfIMCKb2/4jsyTfc62jlLW7JxABYcQM7Cno50fQZ
Usevce2YZ1bZNVAxaR+xH3coRqVGw8R8KQtZiFRPAqSI7PlNJIAQtAaHDrBq5Cbb
3q91A/0dSTq9aork9STJr985jFqm0aIFrBPv/bbJA1vXaWEDH9aelkJtgTSsv7u1
aPyHyuuEbiB1n5Gop9nbrJ/VFXzK6eMnTVxFTBthWRNobjzgOmYuSvbscYOx1sRu
Rkc1BrTe+XiQHHyNij6NFWil5jYUxIA0sLXQOJ1LYYzEPWfV47QgU2VuZGVyIDxz
ZW5kZXItcGdwLWtleXNAaWJtLmNvbT6IRgQTEQIABgUCUj4OKAAKCRAbVF7QRe52
21J2AKCE0oVcF7U6h3k5S8OCk6TcMWXi9ACeLeUAIABQE/bDMPkPxv4faP7mM2a5
AxcEUj4OKBAIIKF3gAls5xHTMFI7/R0pWa8NchI/1MvbjCf6IusL/M19A743yS9i
WYE1vrOo8Uc11xzVyiunwImi2w1Jz4LH2K9mk2IypqfVOltB//WUNPI+mXNqySmw
PQJUT4D3UC9vqOoY2rMZ9ew0bbeNpufC+8phocK7cbpJ4X/jShrVIjVXC0oKv2mI
FN+GN0vAK5SxTgANi3EbSH4a1gIyVkjOafpgt3Gy+MuqiF+LXP34sUm66Fa/vi1H
9gQMgSCgorNW9c8P9VaC0BGjfAY3RZYZRZmZsHp+bKsexfRO1lCL1cEU/882Eefq
3El7raq1gjiUvuM0C1vdY7AVoa2JptvYV8koAKqxCB4oz22FbYOtP//1uVmpTRZj
t3AtChjOkRxAeU6a2f7s6jmXd5G1A80oag3iiQTPRP6UKOkuPhfZTFPOL+BRZPm5
+rfXkWHaGGPUpNZ8hty7Kt8gilMXoQkoC8Q5ERpwTZebJplZrZT0By9DO/8X4rzO
Xgj3FBYh0/fvDlZECe66D5ytGOlLOzkER75Q6KPZuGnWnAh8cKwIwHvzPbiIy9WR
a95N7xijnzxbyTRTB92bBqCH+hkQIN5b5NwOB+boLSRZMnVvv0miQqXyushWaBs6
TU7/DukFRQkbAre2OLwRtQ6gWf/HF0+9UcOCpJUGPVwJpOUALDFUniOXHNXEc3si
MLl3iQgYhUUMKPRKnM+hOhhJ2NmlcK2crVkYnLg0HfDHmcLscipYJQn+QoOzpq0Z
Z+oNnwFBo676ooL56HhC4dRHGYSsRCznvO7t/Szm9cfLijY1eoSuxP8INQO/9D/s
JSic6N+RXJMf1YXoe5pF4Uzo9iDv7TMT2pd4F+d/ckzHwJ29gjweb7v/NkEwM3Pm
Ggag/2P2X/JxJbEGvW0u3vTI2Bq/F4LoAQYSZIUOgHCX+mmEJ4uzah14li1A5bi+
UXmH65j2HggghgTao2GRCliCVfSmXpVGr9JHN61AvSoIswCo5/fHzy0RSxkjFoEv
KKRatuzR8xf2aF8wAlAWe8ISD5PNy4krE4hGBBgRAgAGBQJSPg4pAAoJEBtUXtBF
7nbbqbIAnAmcdOeDm7n76hfn0txiiS9FOeP1AJ9Eqiq4C+p2CA2nYjDHpL7k9QI6
jg==
=us1m
-----END PGP PUBLIC KEY BLOCK-----
```

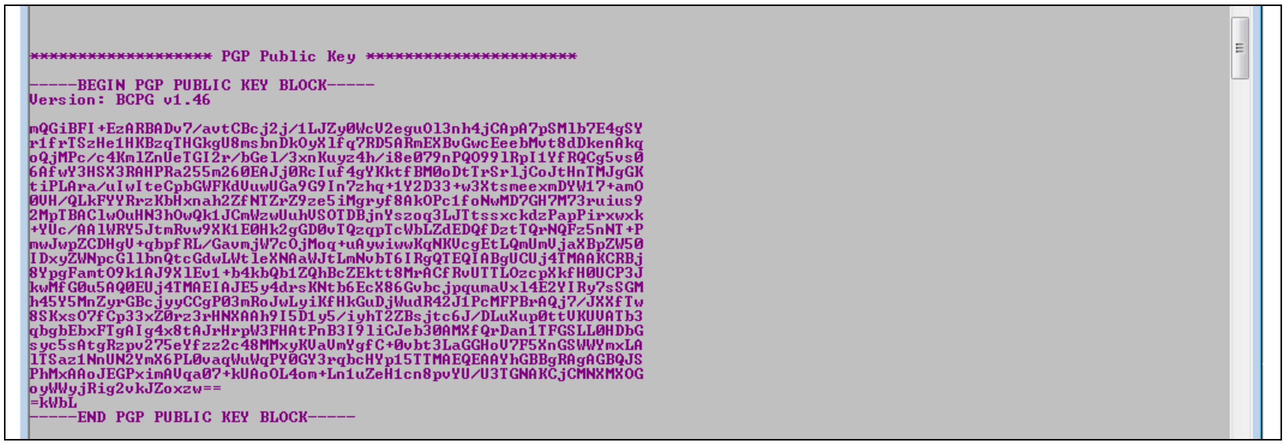


```
C:\PGP\pgpkeytool>java pgpkeytool generatePGPKeyPair -sa DSA -pa RSA -i "Recipient <recipient-pgp-keys@ibm.com>" -a true -kr
2048 -kd 1024 -c AES_256 -s C:/PGP/KeyRepository/RecipientSecretKey.asc -o C:/PGP/KeyRepository/RecipientPublicKey.asc
Please enter PGP Passphrase:
rcvrpassphrase
Please Re-enter PGP Passphrase:
rcvrpassphrase
PGP Signature Key Algorithm: DSA
PGP PublicKey Algorithm: RSA
Identity: Recipient <recipient-pgp-keys@ibm.com>
PassPhrase: rcvrpassphrase
AsciiArmor: true
Keysize (RSA): 2048
Keysize (DSA): 1024
Cipher Algorithm: AES_256
SecretKeyFile: C:/PGP/KeyRepository/RecipientSecretKey.asc
PublicKeyFile: C:/PGP/KeyRepository/RecipientPublicKey.asc
******************* PGP Private Key **********************
```



```
-----BEGIN PGP PRIVATE KEY BLOCK-----
Version: BCPG v1.46

lQHpBFI+EzARBADv7/avtCBcj2j/1LJZy0WcV2eguOl3nh4jCApA7pSMlb7E4gSY
r1frTSzHe1HKBzqTHGkgU8msbnDkOyXlfq7RD5ARmEXBvGwcEeebMvt8dDkenAkq
oQjMPc/c4KmlZnUeTGI2r/bGel/3xnKuyz4h/i8e079nPQO99lRpI1YfRQCg5vs0
6AfwY3HSX3RAHPRa255m260EAJj0RcIuf4gYKktfBM0oDtTrSrljCoJtHnTMJgGK
tiPLAra/uIwIteCpbGWFKdVuwUGa9G9In7zhq+1Y2D33+w3XtsmeexmDYW17+amO
0VH/QLkFYYRrzKbHxnah2ZfNTZrZ9ze5iMgryf8AkOPc1foNwMD7GH7M73ruius9
2MpTBAClwOuHN3hOwQk1JCmWzwUuhVSOTDBjnYszoq3LJTtssxckdzPapPirxwxk
+YUc/AAlWRY5JtmRvw9XK1E0Hk2gGD0vTQzqpTcWbLZdEDQfDztTQrNQFz5nNT+P
mwJwpZCDHgV+qbpfRL/GavmjW7cOjMoq+uAywiwwKqNKVcgEtP4JAwJDPjZ4ijyd
umB9vxtd7dw/ShNksenK7n2wymsJzhlJnuTW1sEztwM86Mde108Ea7vQW7rDnreJ
w80ne56f+U9iYcvZtCZSZWNpcGllbnQgPHJlY2lwaWVudC1wZ3Ata2V5c0BpYm0u
Y29tPohGBBMRAgAGBQJSPhMwAAoJEGPximAVqa072TUAn1eUS/X5viRtBvVlCEFx
kSS23wysAJ9G9RNMs7NyleR8fRQI/cmTAx8bS50DxgRSPhMwAQgAkTnLh2uwo21v
oRxfzoa9tyOmq6ZpXGXgTZghHLuxIYyHjljkydnKsYFyPLIIKA/TeZGgnAvKIp8e
Qa4ONa51HjYnU9wwU8GsBCPv8ldd9PDxIrGw7t8KnffFnSvPesc1cACH0jkPXLn+
LKFPZkGyO1zon8Mu5e6nS21UpRUBNvepuBsRvEVOAAiDjHy0AmseulbcUcC0+cHc
j2WIIl5vfQAxd9CsNqfVMUZIsvQcNsazJzmwC2BHOm/bvl5h/PPZzjwwzHIpVpWZ
iB8L7S9u3ctoYYehXsXlecZJZZibEsCVNJrPU2dQ3ZiZfo8vS9qpa5ao9jQZjeup
twdinXlNMwARAQAB/gkDAu0Le0nqIQdpYC3HZBHIpoUWFPbKcHUpBb2SetQglRTp
XS+otinTcoy3WGvJZbElRT8ecplKC3eP9yG6DZAy8R7O9xFVxmh5S853s8Aftl3I
jt5lx5wlgZVum93y450BSP6bwmVa73HnoUb9W8RMLCbyeaaPsHp/Umeeyqds1Ifn
ZghV10gAuOsc2IMGYk4qvw5W1Zi4686b/9hQQ1rgjcZDgwbe+dy0hHa5X3HC5JsC
qS7JRLk+6k4eDtKKrcKOQzpcghJYpVo72o8eaqzMj2XXAa6oQNDa8Axhrr6gkW7H
YckWtXVJZUUr/YzF4nc0Uv5BHGmT3DHh6zCjcN1KwUu0E6q6hAeJZmo3NTgyhSPV
AxI0iS2Ivz9EiPzdzDLkjsS9djLuMXbHRB94BPfvjUBRBsEcrp8r4Ev5YOXBbdZJ
DICS5hr24nMXUV6vE05yvFpyOpvNbK/ubb/lK/e+YT0GgkbEuuxVBVVjanDHjAsS
i9GR6LCoI/cmZUnz3QriBL9DRbxk4iTVwA6U2TLPIz2nJfJklGaRd1u2TQbC9MzX
/7EeeBSjtil28/B4vrm0y8sN/jK3gtKIgp15lK0v3yff1Q0fAenRRS6aQH43uT7Z
qM++7X7cX6sE7FCEupSxhQAioxm5pzb7kMX43TyA/VygY7uOl8nQsz+LX9zlL3Xa
4HIfpeKUFUYdBBfuuJAuNc1rZJJWYbVd+zsjX/JEs+4Cn/oo1rKhPNO/pPxxwXKf
DrKWGOh6C8699jHx5UW9hYpAD1jLc5J3Qvq0cQJ4vXK0DZam4bodBX1I9ZSfzInb
4cqCSRtaWT9oFP9HP2FXSIvnjW3AVlt4VhovD+mtoBL13xrcmfPe0q5WI7yVe6ps
MMMs+Vut7k0tjLGmGCM6d2+4YyfTyv+pqzDPB0sYfJp28pgJgIhGBBgRAgAGBQJS
PhMxAAoJEGPximAVqa07+kUAoOL4om+Ln1uZeH1cn8pvYU/U3TGNAKCjCMNXMXOG
oyWWyjRig2vkJZoxzw==
=FNBj
-----END PGP PRIVATE KEY BLOCK-----
```

MyOpenTech
(PGP SupportPac)



```
********************* PGP Public Key *************************

-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: BCPG v1.46

mQGiBFI+EzARBADv7/avtCBcj2j/1LJZy0WcV2eguO13nh4jCApA7pSMlb7E4gSY
r1frTSzHe1HKBzqTHGkgU8msbnDkOyXlfq7RD5ARmEXBvGwcEeebMvt8dDkenAkq
oQjMPc/c4KmlZnUeTGI2r/bGel/3xnKuyz4h/i8e079nPQO99lRpI1YfRQCg5vs0
6AfwY3HSX3RAHPRa255m260EAJj0RcIuf4gYKktfBM0oDtTrSrljCoJtHnTMJgGK
tiPLAra/uIwIteCpbGWFKdVuwUGa9G9In7zhq+1Y2D33+w3XtsmeexmDYW17+amO
0VH/QLkFYYRrzKbHxnah2ZfNTZrZ9ze5iMgryf8AkOPc1foNwMD7GH7M73ruius9
2MpTBAClwOuHN3hOwQk1JCmWzwUuhVSOTDBjnYszoq3LJTtssxckdzPapPirxwxk
+YUc/AAlWRY5JtmRvw9XK1E0Hk2gGD0vTQzqpTcWbLZdEDQfDztTQrNQFz5nNT+P
mwJwpZCDHgV+qbpfRL/GavmjW7cOjMoq+uAywiwwKqNKVcgEtLQmUmVjaXBpZW50
IDxyZWNpcGllbnQtcGdwLWtleXNAaWJtLmNvbT6IRgQTEQIABgUCUj4TMAAKCRBj
8YpgFamtO9k1AJ9XlEv1+b4kbQb1ZQhBcZEktt8MrACfRvUTTLOzcpXkfH0UCP3J
kwMfG0u5AQ0EUj4TMAEIAJE5y4drsKNtb6EcX86GvbcjpqumaVxl4E2YIRy7sSGM
h45Y5MnZyrGBcjyyCCgP03mRoJwLyiKfHkGuDjWudR42J1PcMFPBrAQj7/JXXfTw
8SKxsO7fCp33xZ0rz3rHNXAAh9I5D1y5/iyhT2ZBsjtc6J/DLuXup0ttVKUVATb3
qbgbEbxFTgAIg4x8tAJrHrpW3FHAtPnB3I9liCJeb30AMXfQrDan1TFGSLL0HDbG
syc5sAtgRzpv275eYfzz2c48MMxyKVaVmYgfC+0vbt3LaGGHoV7F5XnGSWWYmxLA
lTSaz1NnUN2YmX6PL0vaqWuWqPY0GY3rqbcHYp15TTMAEQEAAYhGBBgRAgAGBQJS
PhMxAAoJEGPximAVqa07+kUAoOL4om+Ln1uZeH1cn8pvYU/U3TGNAKCjCMNXMXOG
oyWWyjRig2vkJZoxzw==
=kWbL
-----END PGP PUBLIC KEY BLOCK-----
```

Step 2: Import Sender’s private key into Sender’s private key repository.

Command:

```
java pgpkeytool importPrivateKey -sr C:/PGP/KeyRepository/Sender/private.pgp -i true -sf
C:/PGP/KeyRepository/SenderSecretKey.asc
```

Step 3: Import Recipient’s private key into Recipient’s private key repository.

Command:

```
java pgpkeytool importPrivateKey -sr C:/PGP/KeyRepository/Recipient/private.pgp -i true
-sf C:/PGP/KeyRepository/RecipientSecretKey.asc
```

Step 4: Import Sender’s public key into Sender’s public key repository.

Command:

```
java pgpkeytool importPublicKey -pr C:/PGP/KeyRepository/Sender/public.pgp -i true -pf
C:/PGP/KeyRepository/SenderPublicKey.asc
```

Step 5: Import Recipient’s public key into Sender’s public key repository.

Command:

```
java pgpkeytool importPublicKey -pr C:/PGP/KeyRepository/Sender/public.pgp -i true -pf
C:/PGP/KeyRepository/RecipientPublicKey.asc
```

Step 6: Import Recipient’s public key into Recipient’s public key repository.

Command:

```
java pgpkeytool importPublicKey -pr C:/PGP/KeyRepository/Recipient/public.pgp -i true -
```

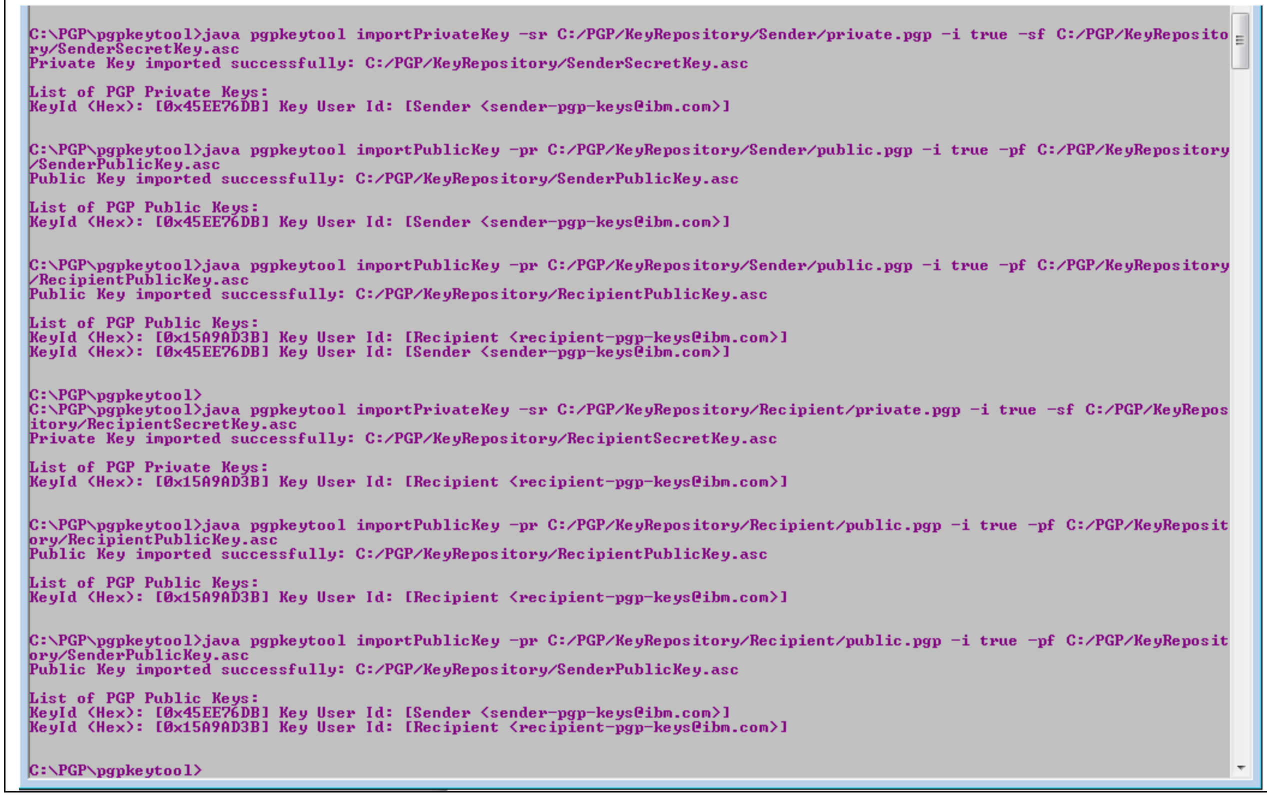
```
pf C:/PGP/KeyRepository/RecipientPublicKey.asc
```

Step 7: Import Sender's public key into Recipient's public key repository.

Command:

```
java pgpkeytool importPublicKey -pr C:/PGP/KeyRepository/Recipient/public.pgp -i true -
pf C:/PGP/KeyRepository/SenderPublicKey.asc
```

Figure-4: pgpkeytool key management screen-shots.



```
C:\PGP\pgpkeytool>java pgpkeytool importPrivateKey -sr C:/PGP/KeyRepository/Sender/private.pgp -i true -sf C:/PGP/KeyReposito
ry/SenderSecretKey.asc
Private Key imported successfully: C:/PGP/KeyRepository/SenderSecretKey.asc

List of PGP Private Keys:
KeyId (Hex): [0x45EE76DB] Key User Id: [Sender <sender-pgp-keys@ibm.com>]

C:\PGP\pgpkeytool>java pgpkeytool importPublicKey -pr C:/PGP/KeyRepository/Sender/public.pgp -i true -pf C:/PGP/KeyRepository
/SenderPublicKey.asc
Public Key imported successfully: C:/PGP/KeyRepository/SenderPublicKey.asc

List of PGP Public Keys:
KeyId (Hex): [0x45EE76DB] Key User Id: [Sender <sender-pgp-keys@ibm.com>]

C:\PGP\pgpkeytool>java pgpkeytool importPublicKey -pr C:/PGP/KeyRepository/Sender/public.pgp -i true -pf C:/PGP/KeyRepository
/RecipientPublicKey.asc
Public Key imported successfully: C:/PGP/KeyRepository/RecipientPublicKey.asc

List of PGP Public Keys:
KeyId (Hex): [0x15A9AD3B] Key User Id: [Recipient <recipient-pgp-keys@ibm.com>]
KeyId (Hex): [0x45EE76DB] Key User Id: [Sender <sender-pgp-keys@ibm.com>]

C:\PGP\pgpkeytool>
C:\PGP\pgpkeytool>java pgpkeytool importPrivateKey -sr C:/PGP/KeyRepository/Recipient/private.pgp -i true -sf C:/PGP/KeyRepos
itory/RecipientSecretKey.asc
Private Key imported successfully: C:/PGP/KeyRepository/RecipientSecretKey.asc

List of PGP Private Keys:
KeyId (Hex): [0x15A9AD3B] Key User Id: [Recipient <recipient-pgp-keys@ibm.com>]

C:\PGP\pgpkeytool>java pgpkeytool importPublicKey -pr C:/PGP/KeyRepository/Recipient/public.pgp -i true -pf C:/PGP/KeyReposit
ory/RecipientPublicKey.asc
Public Key imported successfully: C:/PGP/KeyRepository/RecipientPublicKey.asc

List of PGP Public Keys:
KeyId (Hex): [0x15A9AD3B] Key User Id: [Recipient <recipient-pgp-keys@ibm.com>]

C:\PGP\pgpkeytool>java pgpkeytool importPublicKey -pr C:/PGP/KeyRepository/Recipient/public.pgp -i true -pf C:/PGP/KeyReposit
ory/SenderPublicKey.asc
Public Key imported successfully: C:/PGP/KeyRepository/SenderPublicKey.asc

List of PGP Public Keys:
KeyId (Hex): [0x45EE76DB] Key User Id: [Sender <sender-pgp-keys@ibm.com>]
KeyId (Hex): [0x15A9AD3B] Key User Id: [Recipient <recipient-pgp-keys@ibm.com>]

C:\PGP\pgpkeytool>
```

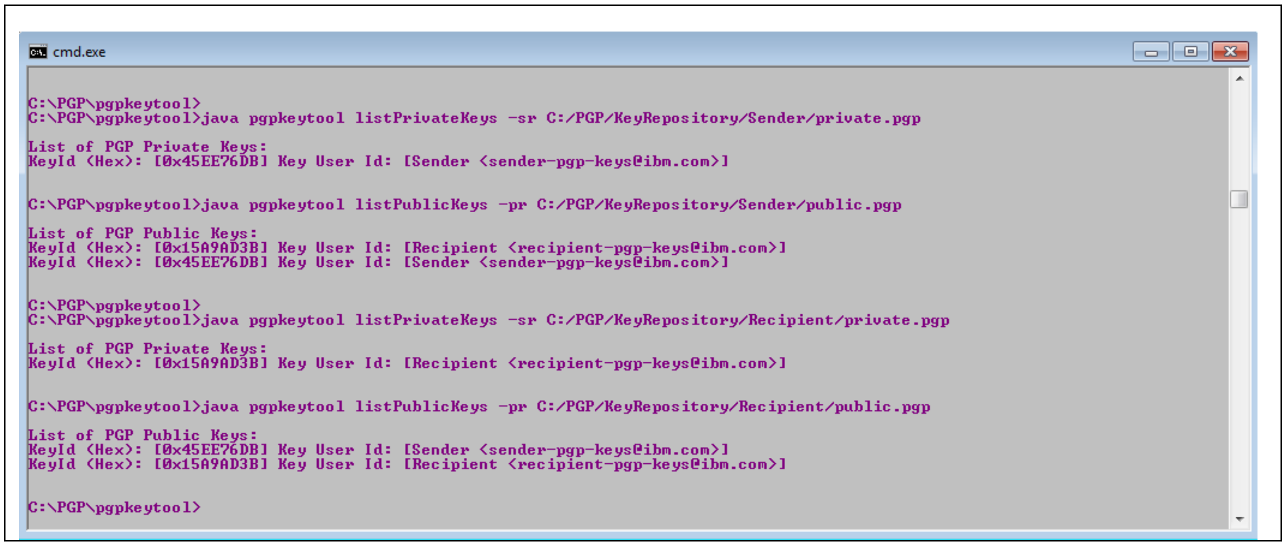
Step 8: Validate PGP key repository files.
List PGP keys contained by Sender/Recipient private/public key repository files.

Commands:

```
java pgpkeytool listPrivateKeys -sr C:/PGP/KeyRepository/Sender/private.pgp

java pgpkeytool listPublicKeys -pr C:/PGP/KeyRepository/Sender/public.pgp

java pgpkeytool listPrivateKeys -sr C:/PGP/KeyRepository/Recipient/private.pgp

java pgpkeytool listPublicKeys -pr C:/PGP/KeyRepository/Recipient/public.pgp
```

Figure-5: pgpkeytool screen-shots for listing key repositories.

MyOpenTech
(PGP SupportPac)


```
C:\PGP\pgpkeytool>
C:\PGP\pgpkeytool>java pgpkeytool listPrivateKeys -sr C:/PGP/KeyRepository/Sender/private.pgp

List of PGP Private Keys:
KeyId (Hex): [0x45EE76DB] Key User Id: [Sender <sender-pgp-keys@ibm.com>]

C:\PGP\pgpkeytool>java pgpkeytool listPublicKeys -pr C:/PGP/KeyRepository/Sender/public.pgp

List of PGP Public Keys:
KeyId (Hex): [0x15A9AD3B] Key User Id: [Recipient <recipient-pgp-keys@ibm.com>]
KeyId (Hex): [0x45EE76DB] Key User Id: [Sender <sender-pgp-keys@ibm.com>]

C:\PGP\pgpkeytool>
C:\PGP\pgpkeytool>java pgpkeytool listPrivateKeys -sr C:/PGP/KeyRepository/Recipient/private.pgp

List of PGP Private Keys:
KeyId (Hex): [0x15A9AD3B] Key User Id: [Recipient <recipient-pgp-keys@ibm.com>]

C:\PGP\pgpkeytool>java pgpkeytool listPublicKeys -pr C:/PGP/KeyRepository/Recipient/public.pgp

List of PGP Public Keys:
KeyId (Hex): [0x45EE76DB] Key User Id: [Sender <sender-pgp-keys@ibm.com>]
KeyId (Hex): [0x15A9AD3B] Key User Id: [Recipient <recipient-pgp-keys@ibm.com>]

C:\PGP\pgpkeytool>
```

Step 9: Create UserDefined Configurable services
PGP Encrypter/Decrypter nodes read default signature key user Id, default decryption/sign key passphrases and private/public keys from respective key repository files specified at User Defined Configurable Service. By using a configurable service, you can change the PGP private/public key repository details, default signature key User Id, default decryption/sign key passphrases information without the need to redeploy the messageflow. You need to restart the execution group for the change of property values to take effect.

You can also use the IBM Integration Bus Explorer to view, add, modify and delete the configurable service.

Alternatively, use the following commands to create the user defined configurable service. Examples illustrated by this article use two UserDefined Configurable services consist of two separate pair of PGP key repository files. In general all the interfaces (messageflows) deployed in a Message Broker instance use a single pair of PGP key repository represented by a UserDefined Configurable Service. However you can design your interfaces if there is a need to create multiple pair of PGP key repositories and UserDefined Configurable Services as per your organization best practices/standards.

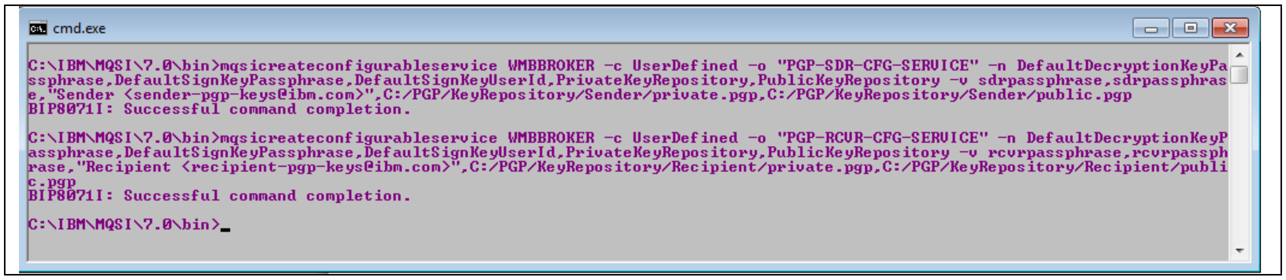
MQSI Command to create UserDefined Configurable Service.

```
mqsicreateconfigurableservice WMBBROKER -c UserDefined -o "PGP-SDR-CFG-SERVICE" -n DefaultDecryptionKeyPassphrase,DefaultSignKeyPassphrase,DefaultSignKeyUserId,PrivateKeyRepository,PublicKeyRepository -v sdrpassphrase,sdrpassphrase,"Sender <sender-pgp-keys@ibm.com>",C:/PGP/KeyRepository/Sender/private.pgp,C:/PGP/KeyRepository/Sender/public.pgp
```

```
mqsicreateconfigurableservice WMBBROKER -c UserDefined -o "PGP-RCVR-CFG-SERVICE" -n DefaultDecryptionKeyPassphrase,DefaultSignKeyPassphrase,DefaultSignKeyUserId,PrivateKeyRepository,PublicKeyRepository -v rcvrpassphrase,rcvrpassphrase,"Recipient <recipient-pgp-keys@ibm.com>",C:/PGP/KeyRepository/Recipient/private.pgp,C:/PGP/KeyRepository/Recipient/public.pgp
```

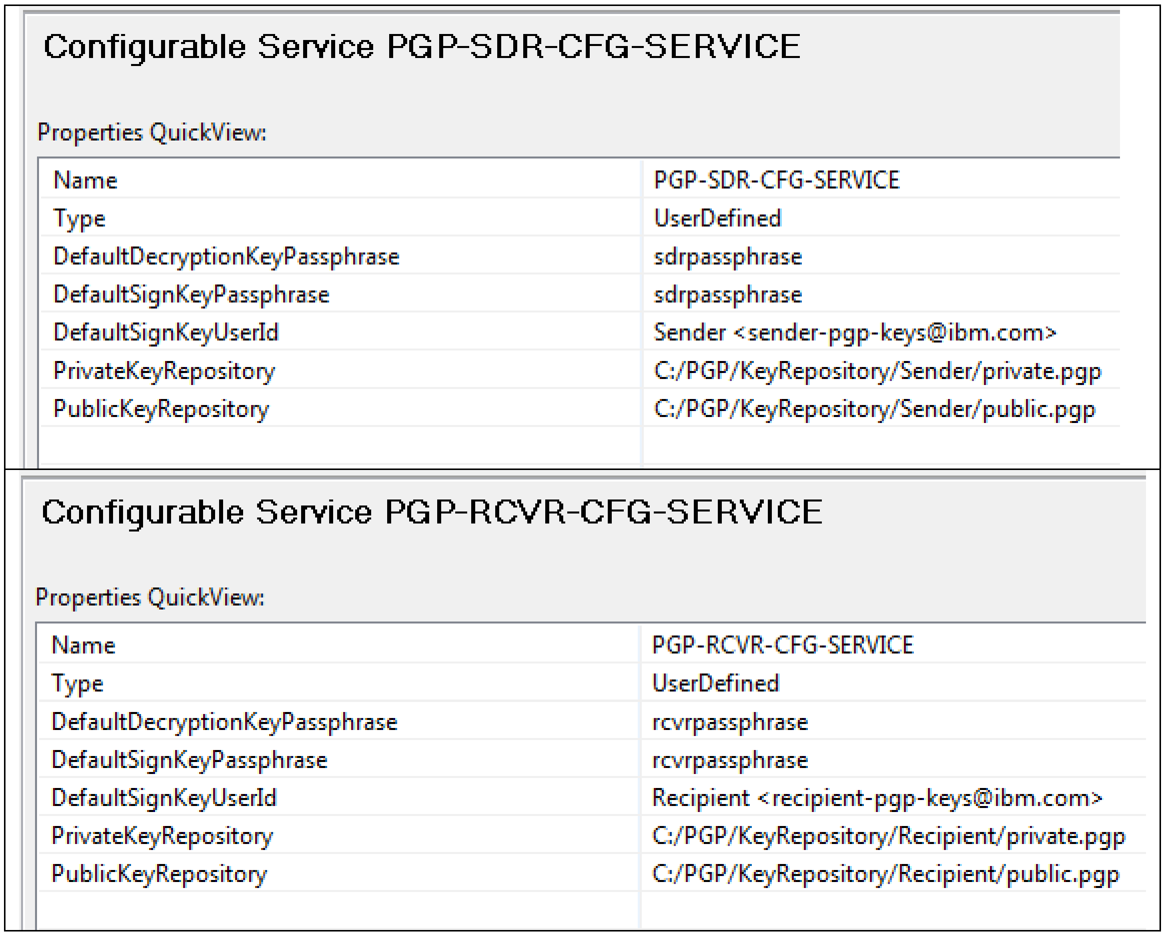
MyOpenTech
(PGP SupportPac)

Figure-6: Screen-shot of MQSI Command to create UserDefined Configurable Service



```
C:\IBM\MQSI\7.0\bin>mqsicreateconfigurableservice WMBBROKER -c UserDefined -o "PGP-SDR-CFG-SERVICE" -n DefaultDecryptionKeyPa
ssphrase,DefaultSignKeyPassphrase,DefaultSignKeyUserId,PrivateKeyRepository,PublicKeyRepository -v sdrpassphrase,sdrpassphras
e,"Sender <sender-pgp-keys@ibm.com>",C:/PGP/KeyRepository/Sender/private.pgp,C:/PGP/KeyRepository/Sender/public.pgp
BIP8071I: Successful command completion.

C:\IBM\MQSI\7.0\bin>mqsicreateconfigurableservice WMBBROKER -c UserDefined -o "PGP-RCVR-CFG-SERVICE" -n DefaultDecryptionKeyP
assphrase,DefaultSignKeyPassphrase,DefaultSignKeyUserId,PrivateKeyRepository,PublicKeyRepository -v rcvrpassphrase,rcvrpassph
rase,"Recipient <recipient-pgp-keys@ibm.com>",C:/PGP/KeyRepository/Recipient/private.pgp,C:/PGP/KeyRepository/Recipient/publi
c.pgp
BIP8071I: Successful command completion.

C:\IBM\MQSI\7.0\bin>_
```

Figure-7: UserDefined Configurable Services shown at Broker Explorer



Configurable Service PGP-SDR-CFG-SERVICE

Properties QuickView:

Name	PGP-SDR-CFG-SERVICE
Type	UserDefined
DefaultDecryptionKeyPassphrase	sdrpassphrase
DefaultSignKeyPassphrase	sdrpassphrase
DefaultSignKeyUserId	Sender <sender-pgp-keys@ibm.com>
PrivateKeyRepository	C:/PGP/KeyRepository/Sender/private.pgp
PublicKeyRepository	C:/PGP/KeyRepository/Sender/public.pgp

Configurable Service PGP-RCVR-CFG-SERVICE

Properties QuickView:

Name	PGP-RCVR-CFG-SERVICE
Type	UserDefined
DefaultDecryptionKeyPassphrase	rcvrpassphrase
DefaultSignKeyPassphrase	rcvrpassphrase
DefaultSignKeyUserId	Recipient <recipient-pgp-keys@ibm.com>
PrivateKeyRepository	C:/PGP/KeyRepository/Recipient/private.pgp
PublicKeyRepository	C:/PGP/KeyRepository/Recipient/public.pgp

Messageflow Development

Following examples illustrate how to use PGP Encrypter/Decrypter nodes in messageflows. Refer to second (Part-2) and third (Part-3) articles of this series for node properties details of PGP Encrypter/Decrypter nodes.

Example-1:

This example consists of a PGP Encrypter (**Sender: PGPEncrypterMF.msgflow**) messageflow and a PGP Decrypter (**Recipient: PGPDecrypterMF.msgflow**) messageflow. It implements MQ message encryption/decryption by using PGP Encrypter/Decrypter nodes mostly configured with default node properties.

PGPEncrypterMF.msgflow: This messageflow receives input message through MQ Input node, uses PGP Encrypter node to sign and encrypt the message and place the encrypted data into output queue. Flow uses **PGP-SDR-CFG-SERVICE** configurable service to load private/public key repositories and default sign key/passphrase details. It uses Sender's private key [Key user Id: **Sender <sender-pgp-keys@ibm.com>**] to sign the data and Recipient's public key [Key User Id: **Recipient <recipient-pgp-keys@ibm.com>**] for encrypting purpose. Note that PGP Encrypter node uses default sign key and corresponding passphrase configured at **PGP-SDR-CFG-SERVICE** configurable service.

Figure-8: Messageflow diagram

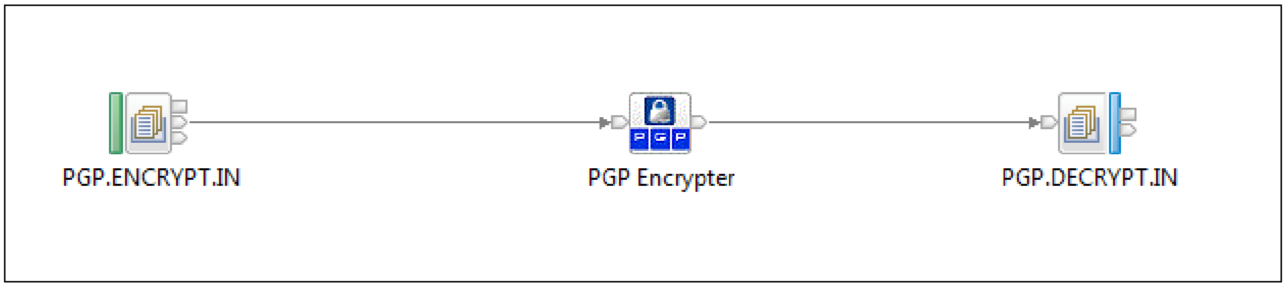
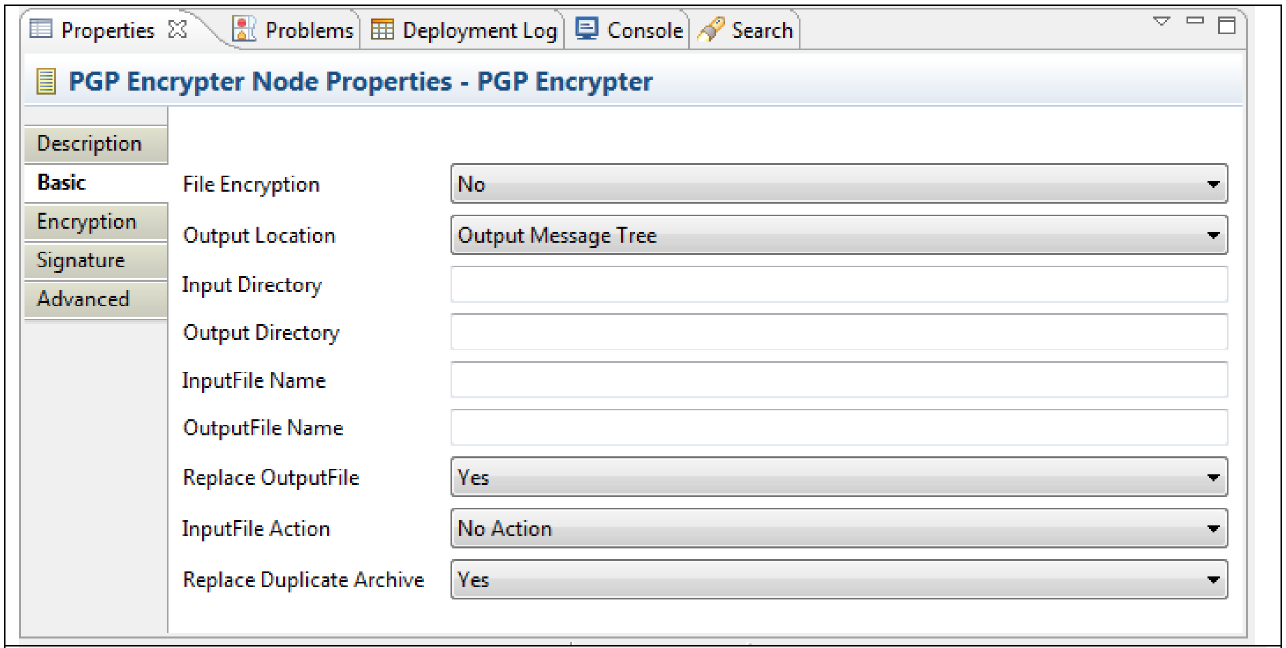
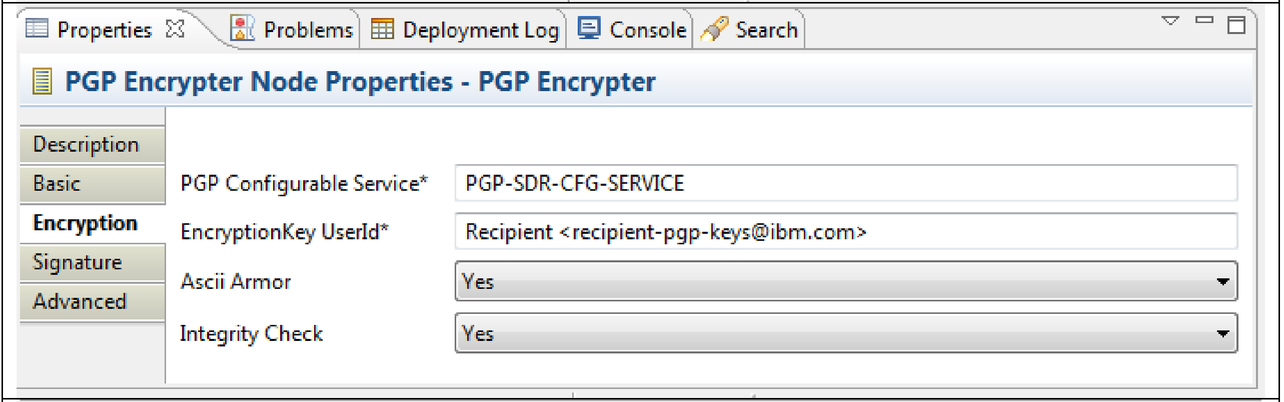


Figure-9: PGP Encrypter node properties



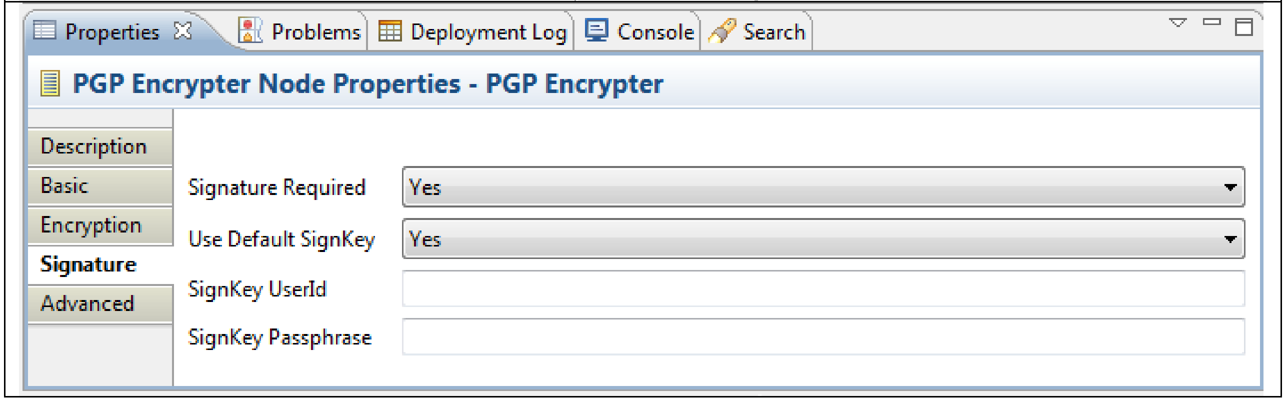
PGP Encrypter Node Properties - PGP Encrypter

File Encryption	No
Output Location	Output Message Tree
Input Directory	
Output Directory	
InputFile Name	
OutputFile Name	
Replace OutputFile	Yes
InputFile Action	No Action
Replace Duplicate Archive	Yes



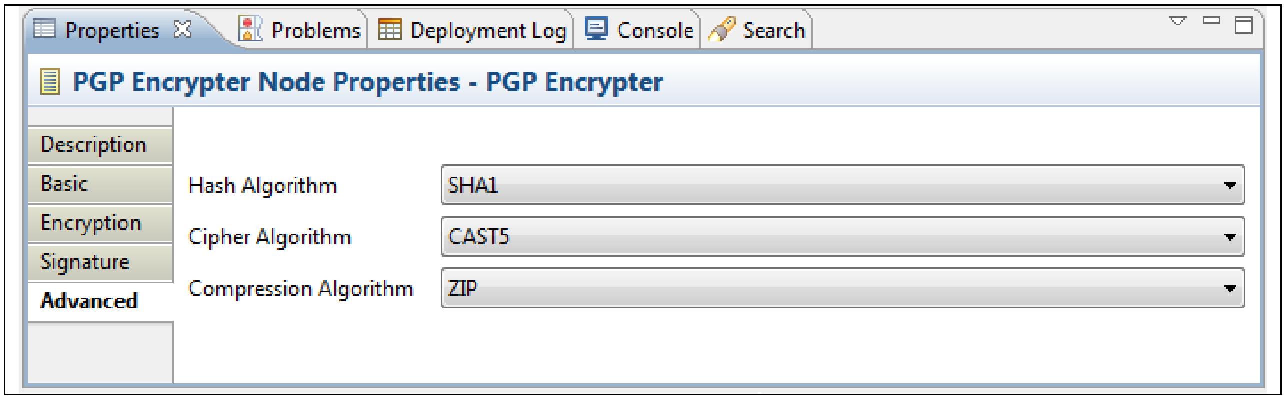
PGP Encrypter Node Properties - PGP Encrypter

PGP Configurable Service*	PGP-SDR-CFG-SERVICE
EncryptionKey UserId*	Recipient <recipient-pgp-keys@ibm.com>
Ascii Armor	Yes
Integrity Check	Yes



PGP Encrypter Node Properties - PGP Encrypter

Signature Required	Yes
Use Default SignKey	Yes
SignKey UserId	
SignKey Passphrase	



PGP Encrypter Node Properties - PGP Encrypter

Description
Basic
Encryption
Signature
Advanced

Hash Algorithm	SHA1
Cipher Algorithm	CAST5
Compression Algorithm	ZIP

PGPDecrypterMF.msgflow: This messageflow receives input message through MQ Input node, uses PGP Decrypter node to decrypt encrypted message, validates PGP signature, put the decrypted data into output queue. Flow uses **PGP-RCVR-CFG-SERVICE** configurable service to load key repositories.

Figure-10: Messageflow diagram

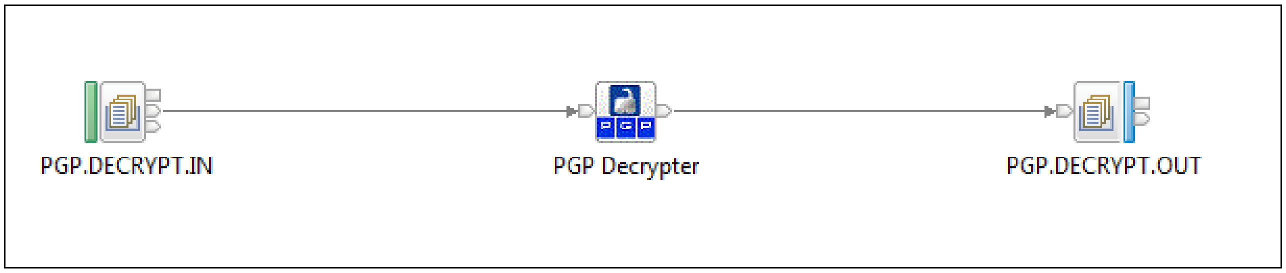
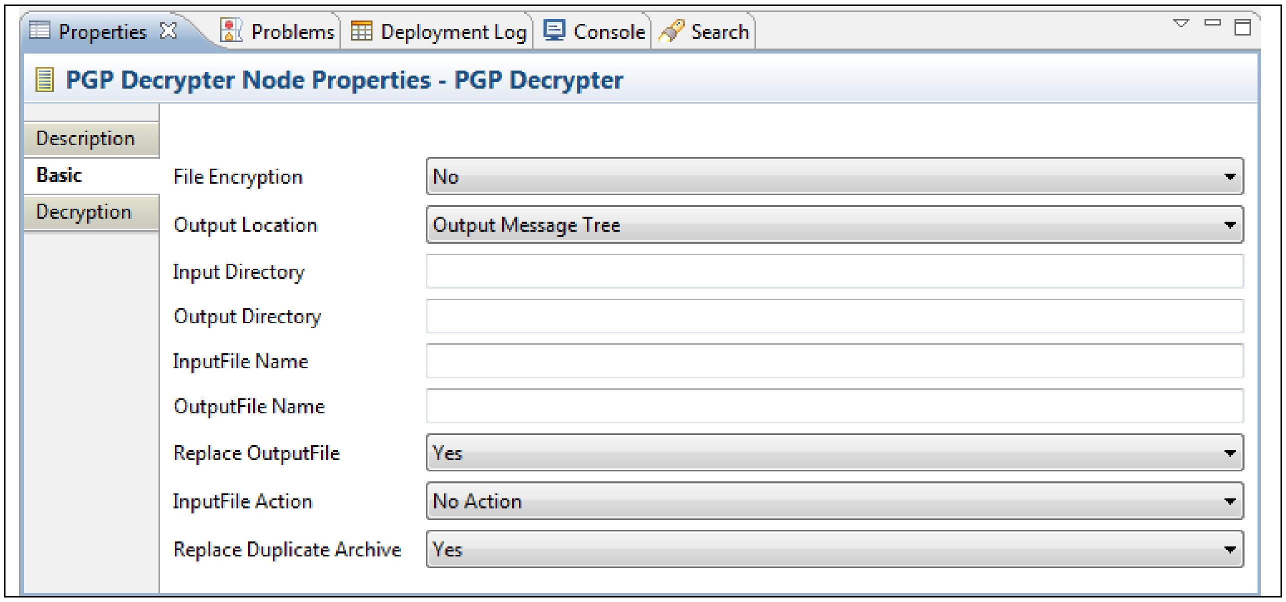


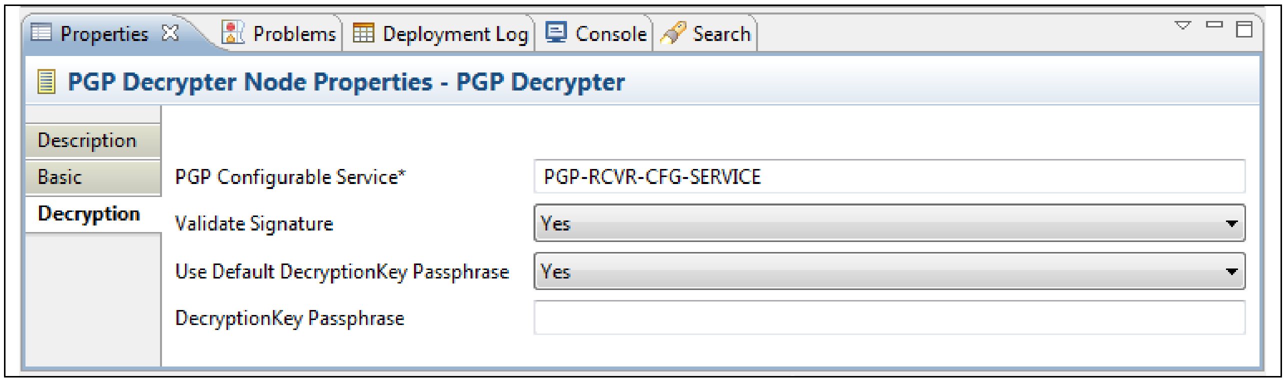
Figure-11: Node properties



PGP Decrypter Node Properties - PGP Decrypter

Description
Basic
Decryption

File Encryption	No
Output Location	Output Message Tree
Input Directory	
Output Directory	
InputFile Name	
OutputFile Name	
Replace OutputFile	Yes
InputFile Action	No Action
Replace Duplicate Archive	Yes



PGP Decrypter Node Properties - PGP Decrypter

PGP Configurable Service*	PGP-RCVR-CFG-SERVICE
Validate Signature	Yes
Use Default DecryptionKey Passphrase	Yes
DecryptionKey Passphrase	

Test: Put a sample text message into input queue of the **PGPEncrypterMF.msgflow.** Get signed and encrypted message at output queue. Use this signed and encrypted data as input message for **PGPDecrypterMF.msgflow** and get decrypted message at output queue. **PGP Decrypter** node throws exception if signature validation failed.

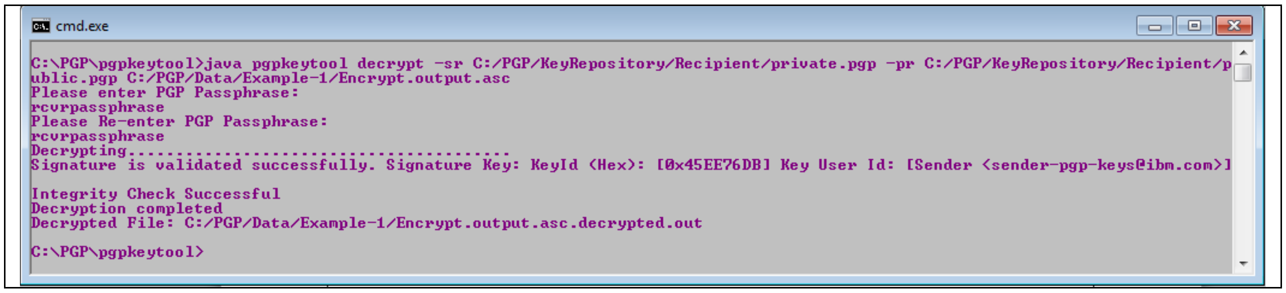
Sample Signed & Encrypted message:

```
-----BEGIN PGP MESSAGE-----
Version: BCPG v1.46

hQEMA9AYGr8LqnmoAQf9HdGn05yLZf989ncPPHN/vxhxpOqO9YdydY1KbhZ9FTIJ
MMypprcEfFfX9PCHr5glddwOZRemlKY3XsBoP3wKkdFA3BH3+KUcMO58HbaDIrnc
HYAnoAc/92rXmqEfVi4ra/sZc975YA/gFYPj0RbIYCFFbgFzmCMA+EYbKOt9gFgr
DYY/zbqq5zL1TXWXsn5flI6lQfXuQFftNPF7kErWNf33UJDB47LnZiQT2jUzjB6E
CxuUngh3uOCcCOCaLtnSkzSBC0KvZFytDJzoxLYIbW1D8bBjmG8xwyQuO6mIHUnt
VAk0pcgEMSy/t6QMCCBV3Lv+pnYzgXak4n+d1ZJoitKyATOzGaeoAdu9yhweld0X
mU8lW8mBlif/J82O/G1qyGQ0dIhYLCg8LlB/+dCrOCGFtnKU5U/McitCuDJDBbqD
B5ciM7frgbLjRDJv6wrSOgu6gtCunLog4kIsDoYP6RQ51/XMlNVGWG9HDNQ9ssF/
LLLjRPjVlYn3s/seR45VWns1EJOVsvAHmzVxwkdenbr6I7HLkrXxVl4DfabnQBdv
MqtOw1H9V87RwLWV8OAaEdXohw==
=Xj1N
-----END PGP MESSAGE-----
```

Optionally you can use **pgpkeytool** to decrypt and validate signature of the output message generated by **PGPEncrypterMF.msgflow.** Save signed and encrypted message into a file (C:/PGP/Data/Example-1/Encrypt.output.asc) and use following command to decrypt the message.

```
java pgpkeytool decrypt -sr C:/PGP/KeyRepository/Recipient/private.pgp -pr
C:/PGP/KeyRepository/Recipient/public.pgp C:/PGP/Data/Example-1/Encrypt.output.asc
```

Figure-12: pgpkeytool decryption screen-shot



```
C:\PGP\pgpkeytool>java pgpkeytool decrypt -sr C:/PGP/KeyRepository/Recipient/private.pgp -pr C:/PGP/KeyRepository/Recipient/p
ublic.pgp C:/PGP/Data/Example-1/Encrypt.output.asc
Please enter PGP Passphrase:
rcvrpassphrase
Please Re-enter PGP Passphrase:
rcvrpassphrase
Decrypting.........................................
Signature is validated successfully. Signature Key: KeyId (Hex): [0x45EE76DB] Key User Id: [Sender <sender-pgp-keys@ibm.com>]

Integrity Check Successful
Decryption completed
Decrypted File: C:/PGP/Data/Example-1/Encrypt.output.asc.decrypted.out

C:\PGP\pgpkeytool>
```

Example-2

This example consists of a PGP Encrypter (**Sender: PGPEncrypterMF.msgflow**) messageflow and a PGP Decrypter (**Recipient: PGPDecrypterMF.msgflow**) messageflow illustrating file encryption/decryption processes.

PGPEncrypterMF.msgflow: This messageflow starts with a MQ Input node just to get triggered by a dummy input message. Flow uses a PGP Encrypter node to sign and encrypt the file specified at node properties and place the encrypted data into file system. It load private/public key repositories from **PGP-SDR-CFG-SERVICE** configurable service and uses Sender's private key [Key user Id: **Sender <sender-pgp-keys@ibm.com>**] specified at node properties to sign the data and Recipient's public key [Key User Id: **Recipient <recipient-pgp-keys@ibm.com>**] for encrypting purpose.

Figure-13: Messageflow diagram

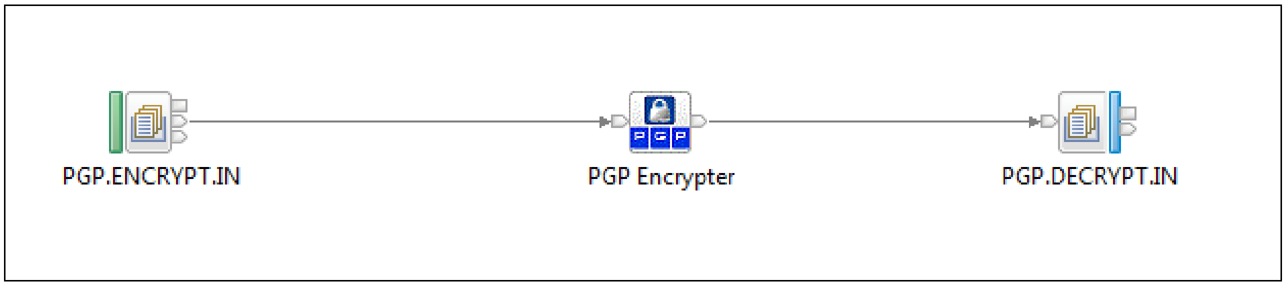
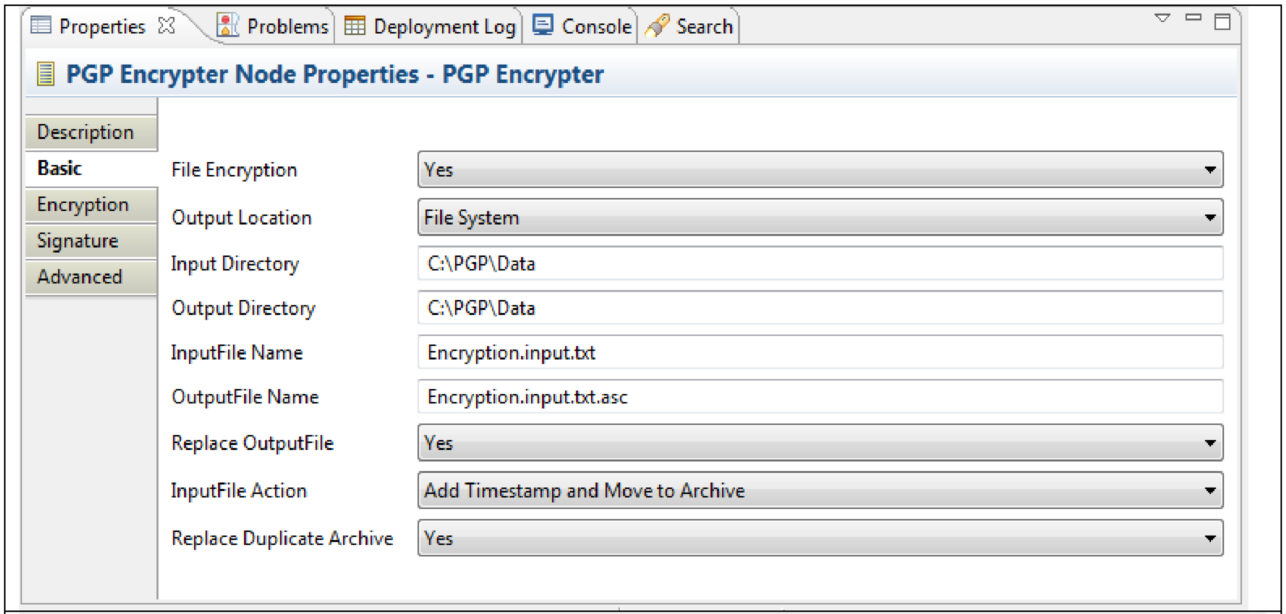
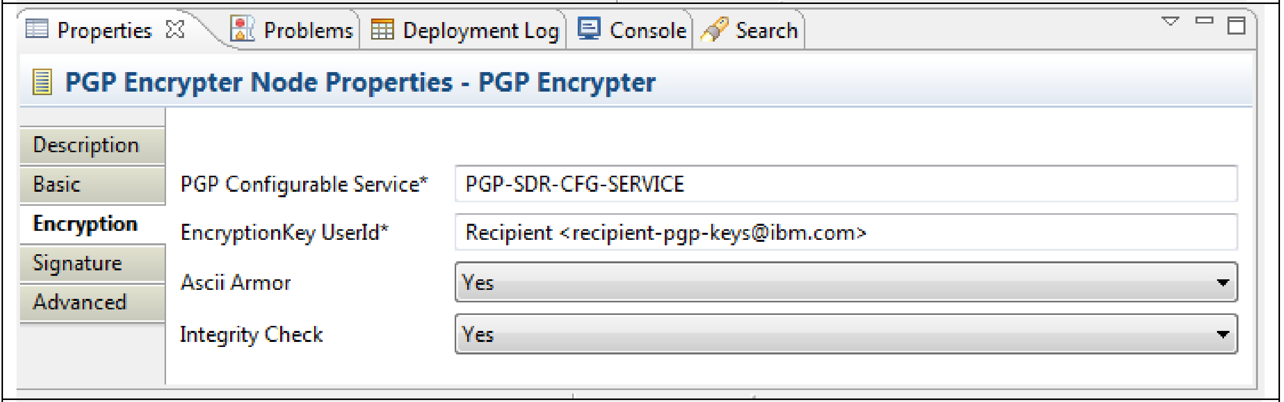


Figure-14: PGP Encrypter node properties



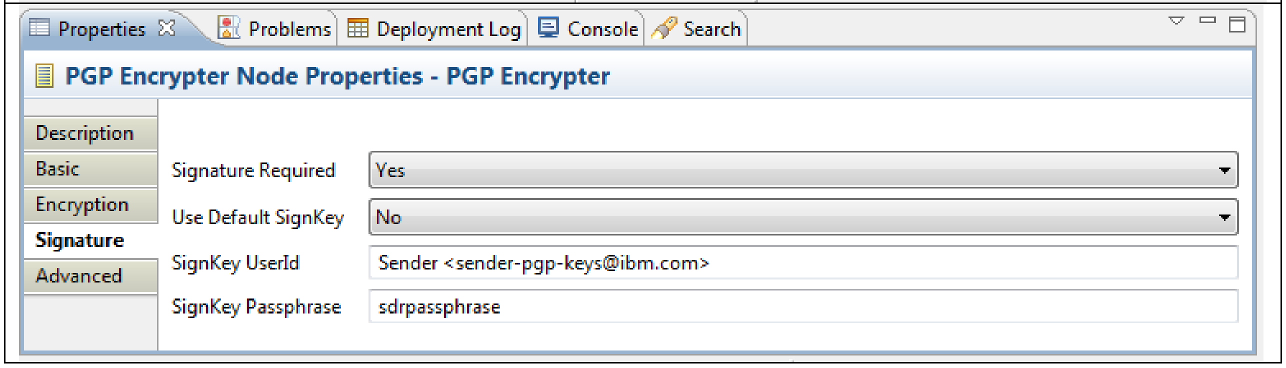
PGP Encrypter Node Properties - PGP Encrypter

File Encryption	Yes
Output Location	File System
Input Directory	C:\PGP\Data
Output Directory	C:\PGP\Data
InputFile Name	Encryption.input.txt
OutputFile Name	Encryption.input.txt.asc
Replace OutputFile	Yes
InputFile Action	Add Timestamp and Move to Archive
Replace Duplicate Archive	Yes



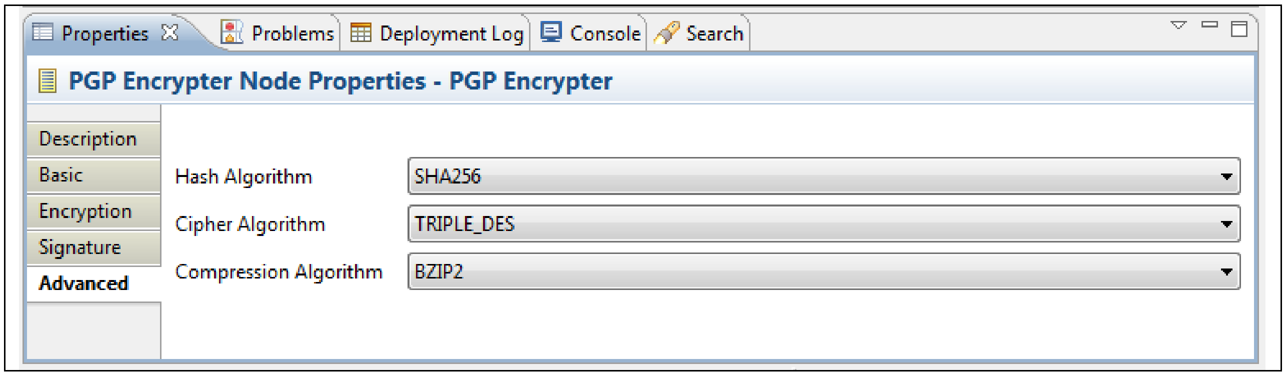
PGP Encrypter Node Properties - PGP Encrypter

PGP Configurable Service*	PGP-SDR-CFG-SERVICE
EncryptionKey UserId*	Recipient <recipient-pgp-keys@ibm.com>
Ascii Armor	Yes
Integrity Check	Yes



PGP Encrypter Node Properties - PGP Encrypter

Signature Required	Yes
Use Default SignKey	No
SignKey UserId	Sender <sender-pgp-keys@ibm.com>
SignKey Passphrase	sdrpassphrase



Properties | Problems | Deployment Log | Console | Search

PGP Encrypter Node Properties - PGP Encrypter

Description | Basic | Encryption | Signature | **Advanced**

Hash Algorithm	SHA256
Cipher Algorithm	TRIPLE_DES
Compression Algorithm	BZIP2

PGPDecrypterMF.msgflow: This messageflow starts with a MQ Input node just to get triggered by a dummy input message. Flow uses PGP Decrypter node to decrypt and validate signature of the encrypted file specified at node properties and place the decrypted data into file system. Flow load key repositories specified at **PGP-RCVR-CFG-SERVICE** configurable service.

Figure-15: Messageflow diagram

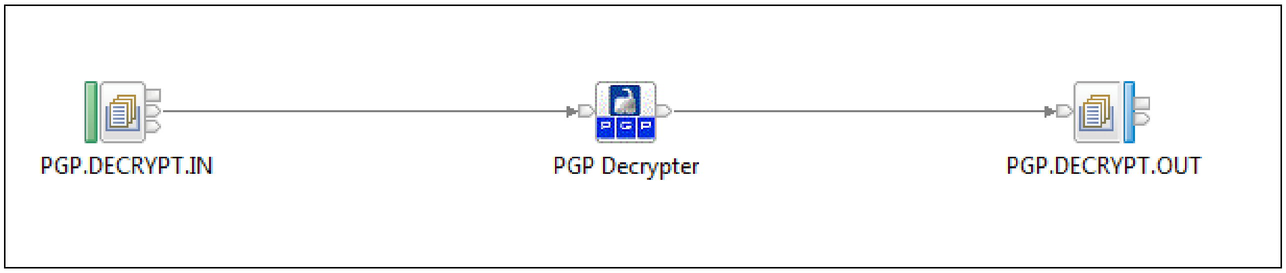
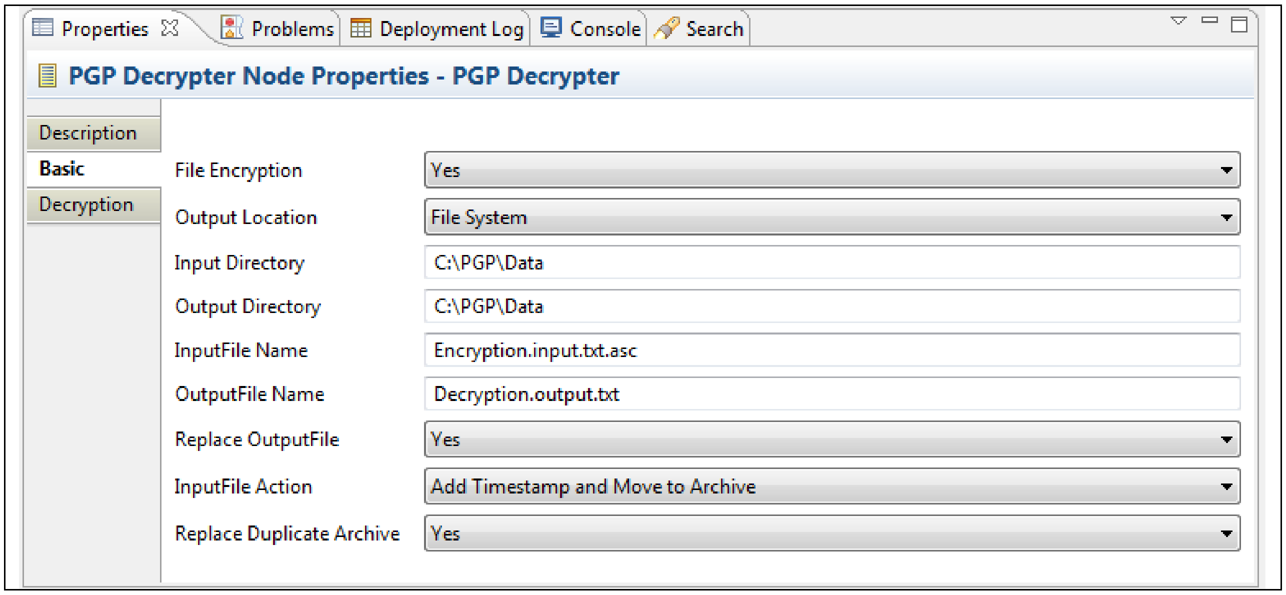


Figure-16: Node properties

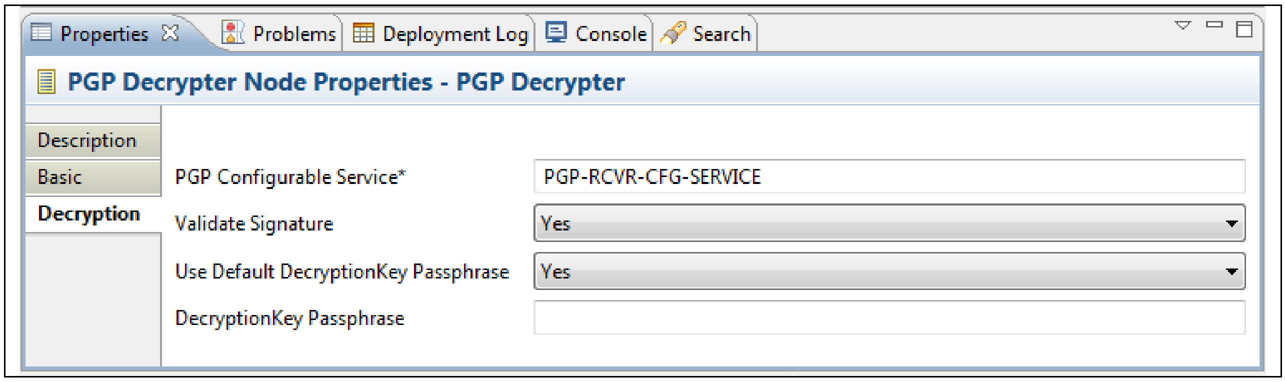


Properties | Problems | Deployment Log | Console | Search

PGP Decrypter Node Properties - PGP Decrypter

Description | **Basic** | Decryption

File Encryption	Yes
Output Location	File System
Input Directory	C:\PGP\Data
Output Directory	C:\PGP\Data
InputFile Name	Encryption.input.txt.asc
OutputFile Name	Decryption.output.txt
Replace OutputFile	Yes
InputFile Action	Add Timestamp and Move to Archive
Replace Duplicate Archive	Yes



PGP Decrypter Node Properties - PGP Decrypter

PGP Configurable Service*	PGP-RCVR-CFG-SERVICE
Validate Signature	Yes
Use Default DecryptionKey Passphrase	Yes
DecryptionKey Passphrase	

Test: Create a sample text file (Encryption.input.txt) in C:\PGP\Data directory. Put a dummy trigger message into input queue of the **PGPEncrypterMF.msgflow**. Flow read the file (C:\PGP\Data\Encryption.input.txt), signs and encrypts, writes the encrypted data into file system (C:\PGP\Data\Encryption.input.txt.asc) specified at node properties. As per **InputFile Action** property **(Add Timestamp and Move to Archive)** specified in node properties, PGP Encrypter node moves the input file renamed with current timestamp suffix into archive directory (C:\PGP\Data\pgparchive). Note that archive directory name is fixed (**pgparchive**) and cannot be altered or overridden.

Example-3

This example consists of a PGP Encrypter (Sender: PGPEncrypterMF.msgflow) messageflow and a PGP Decrypter (Recipient: PGPDecrypterMF.msgflow) messageflow. It describes file encryption/decryption processes with overriding node properties at nodes' local input environment.

PGPEncrypterMF.msgflow: This messageflow starts with a MQ Input node just to get triggered by a dummy input message. Flow contains a compute node to override required node properties at PGP Encrypter node's local input environment. It uses PGP Encrypter node to sign and encrypt the specified file and place the encrypted data into file system. Flow uses **PGP-SDR-CFG-SERVICE** configurable service to load key repositories.

Figure-17: Messageflow diagram

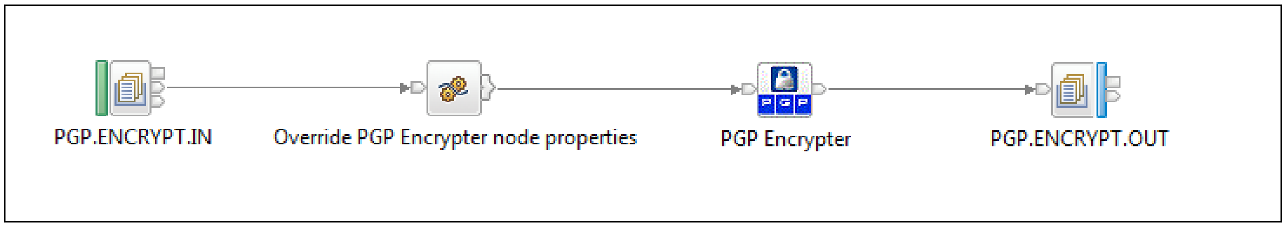
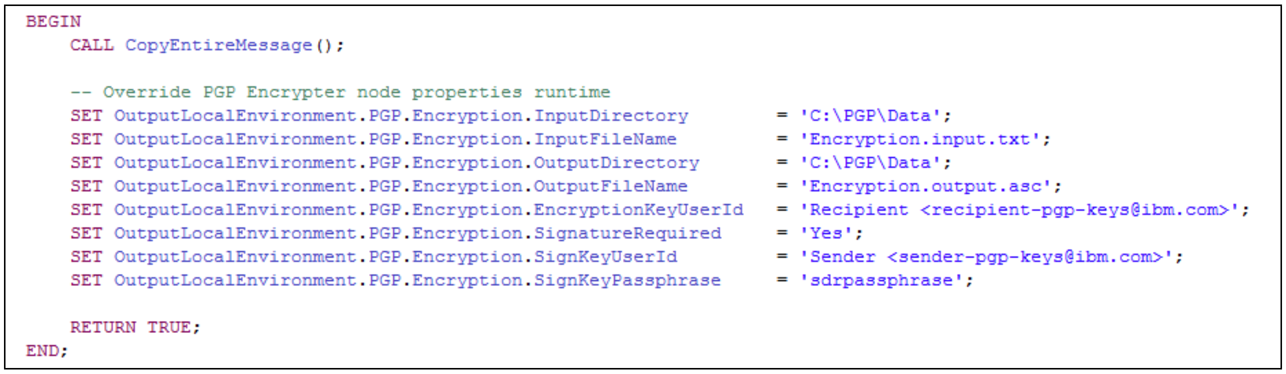
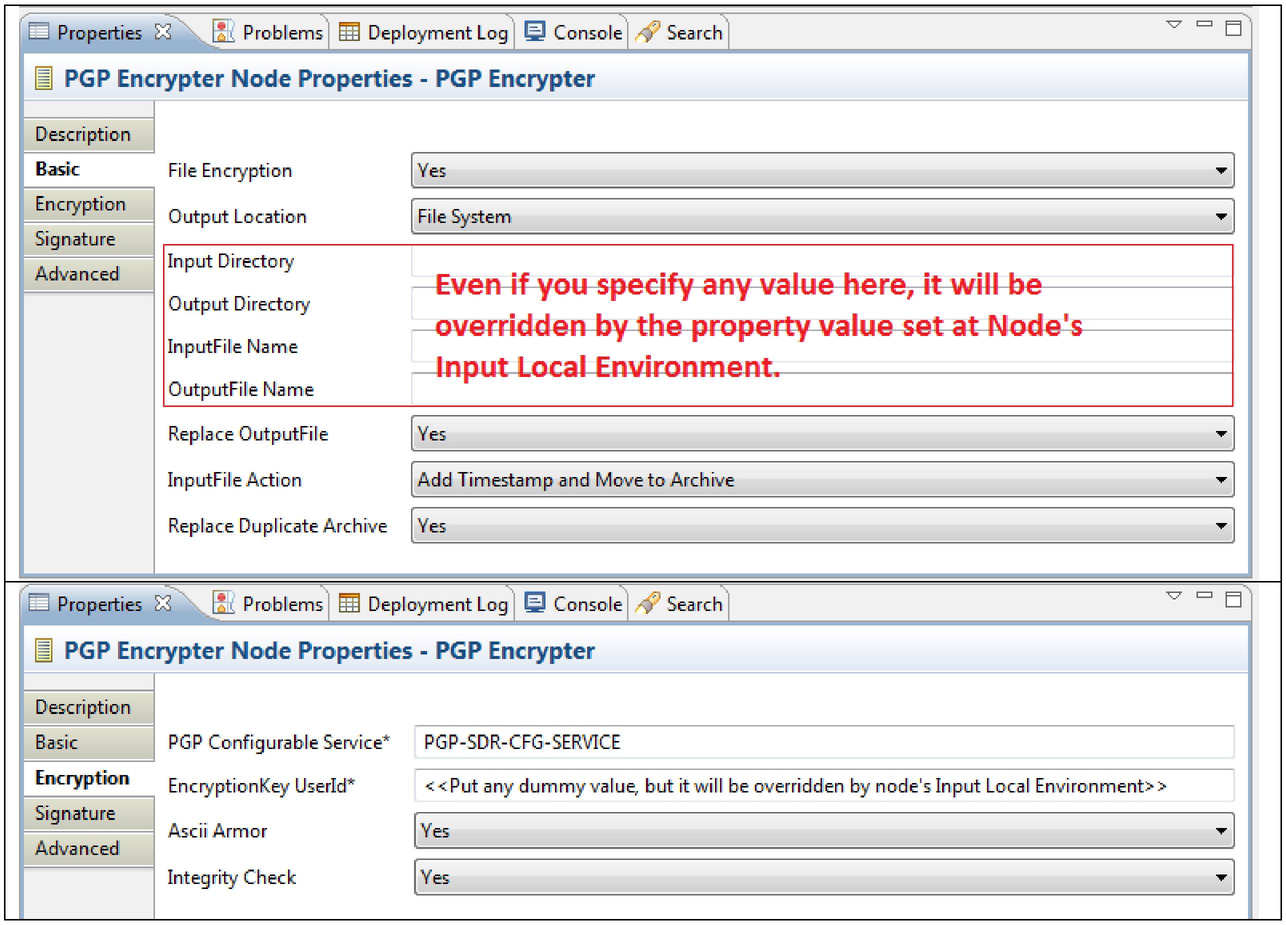


Figure-18: ESQL code overrides required node properties



```
BEGIN
    CALL CopyEntireMessage();

    -- Override PGP Encrypter node properties runtime
    SET OutputLocalEnvironment.PGP.Encryption.InputDirectory        = 'C:\PGP\Data';
    SET OutputLocalEnvironment.PGP.Encryption.InputFileName         = 'Encryption.input.txt';
    SET OutputLocalEnvironment.PGP.Encryption.OutputDirectory       = 'C:\PGP\Data';
    SET OutputLocalEnvironment.PGP.Encryption.OutputFileName        = 'Encryption.output.asc';
    SET OutputLocalEnvironment.PGP.Encryption.EncryptionKeyUserId   = 'Recipient <recipient-pgp-keys@ibm.com>';
    SET OutputLocalEnvironment.PGP.Encryption.SignatureRequired     = 'Yes';
    SET OutputLocalEnvironment.PGP.Encryption.SignKeyUserId         = 'Sender <sender-pgp-keys@ibm.com>';
    SET OutputLocalEnvironment.PGP.Encryption.SignKeyPassphrase     = 'sdrpassphrase';

    RETURN TRUE;
END;
```

Figure-19: PGP Encrypter node properties



PGP Encrypter Node Properties - PGP Encrypter

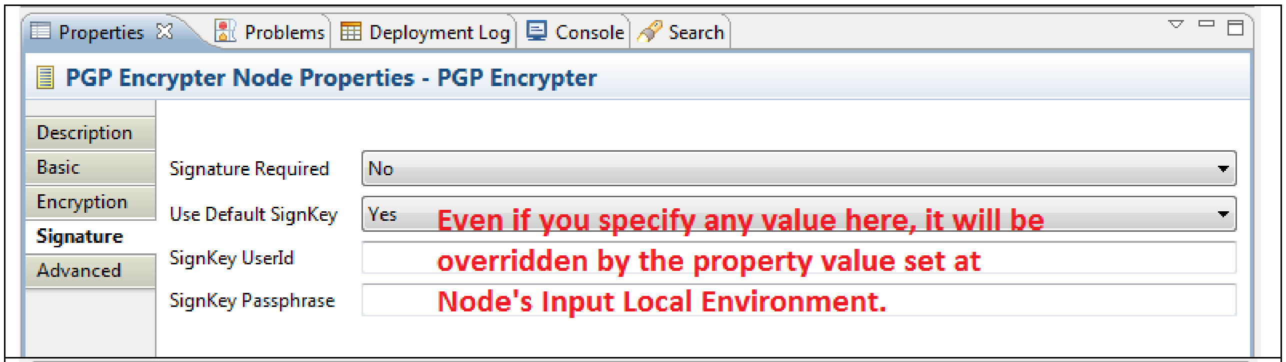
File Encryption	Yes
Output Location	File System
Input Directory	
Output Directory	
InputFile Name	
OutputFile Name	
Replace OutputFile	Yes
InputFile Action	Add Timestamp and Move to Archive
Replace Duplicate Archive	Yes

Even if you specify any value here, it will be overridden by the property value set at Node's Input Local Environment.

PGP Encrypter Node Properties - PGP Encrypter

PGP Configurable Service*	PGP-SDR-CFG-SERVICE
EncryptionKey UserId*	<<Put any dummy value, but it will be overridden by node's Input Local Environment>>
Ascii Armor	Yes
Integrity Check	Yes

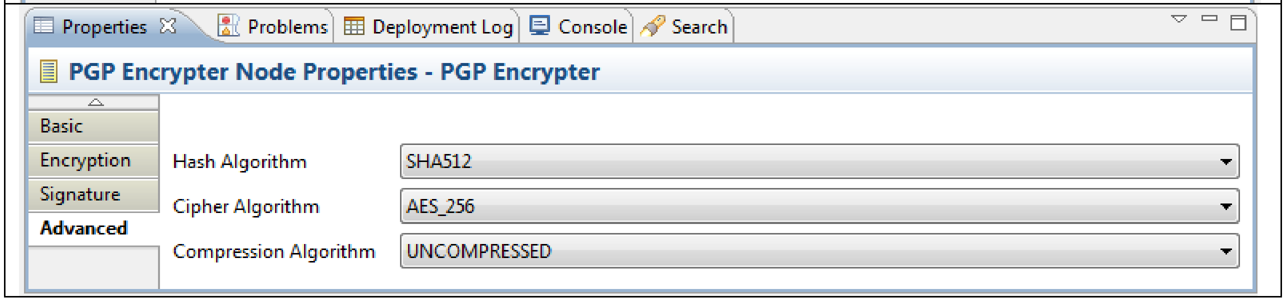
MyOpenTech
(PGP SupportPac)



PGP Encrypter Node Properties - PGP Encrypter

Signature Required: No
Use Default SignKey: Yes
SignKey UserId:
SignKey Passphrase:

Even if you specify any value here, it will be overridden by the property value set at Node's Input Local Environment.



PGP Encrypter Node Properties - PGP Encrypter

Hash Algorithm: SHA512
Cipher Algorithm: AES_256
Compression Algorithm: UNCOMPRESSED

PGPDecrypterMF.msgflow: This messageflow starts with a MQ Input node just to get triggered by a dummy input message. Flow contains a compute node to override required node properties at PGP Decrypter node's local input environment. It uses PGP Decrypter node to decrypt and validate signature of the specified encrypted file and place the decrypted data into file system. It uses **PGP-RCVR-CFG-SERVICE** configurable service to load key repositories.

Figure-20: Messageflow diagram

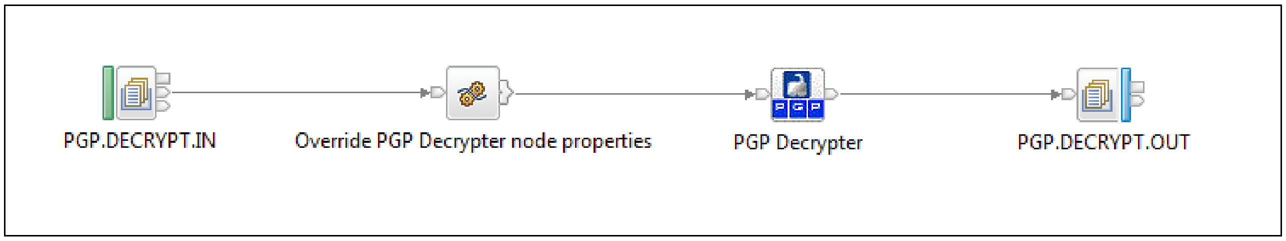
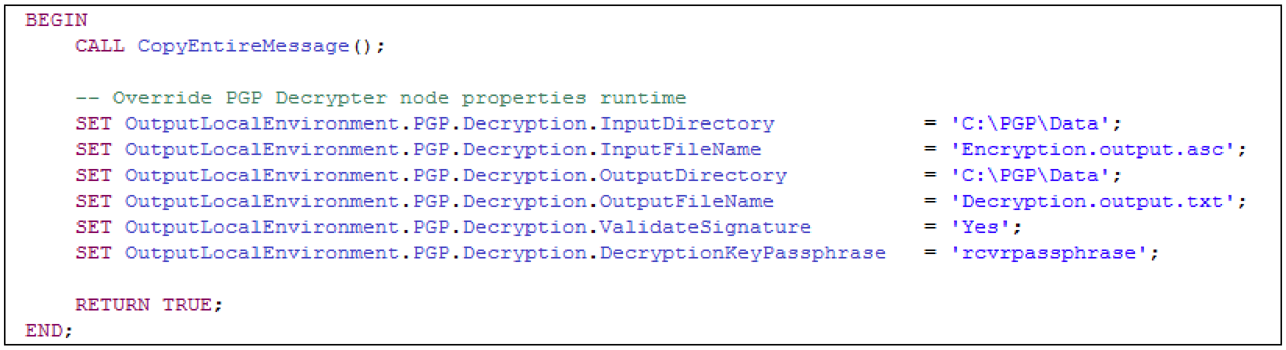
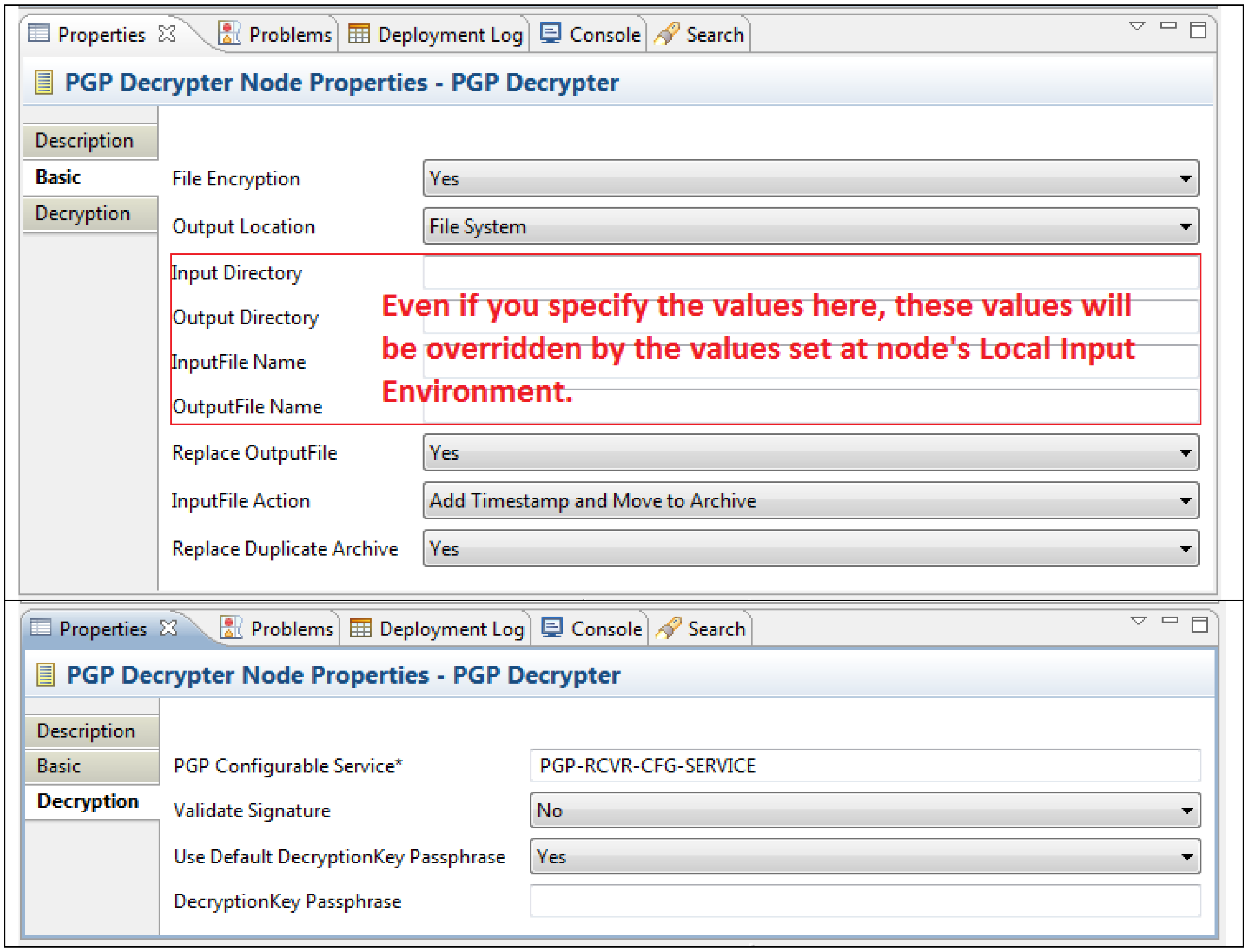


Figure-21: ESQL code overrides required node properties



```
BEGIN
    CALL CopyEntireMessage();

    -- Override PGP Decrypter node properties runtime
    SET OutputLocalEnvironment.PGP.Decryption.InputDirectory            = 'C:\PGP\Data';
    SET OutputLocalEnvironment.PGP.Decryption.InputFileName             = 'Encryption.output.asc';
    SET OutputLocalEnvironment.PGP.Decryption.OutputDirectory           = 'C:\PGP\Data';
    SET OutputLocalEnvironment.PGP.Decryption.OutputFileName            = 'Decryption.output.txt';
    SET OutputLocalEnvironment.PGP.Decryption.ValidateSignature         = 'Yes';
    SET OutputLocalEnvironment.PGP.Decryption.DecryptionKeyPassphrase   = 'rcvrpassphrase';

    RETURN TRUE;
END;
```

Figure-22: Node properties



PGP Decrypter Node Properties - PGP Decrypter

File Encryption	Yes
Output Location	File System
Input Directory	
Output Directory	
InputFile Name	
OutputFile Name	
Replace OutputFile	Yes
InputFile Action	Add Timestamp and Move to Archive
Replace Duplicate Archive	Yes

Even if you specify the values here, these values will be overridden by the values set at node's Local Input Environment.

PGP Decrypter Node Properties - PGP Decrypter

PGP Configurable Service*	PGP-RCVR-CFG-SERVICE
Validate Signature	No
Use Default DecryptionKey Passphrase	Yes
DecryptionKey Passphrase	

Test: Create a text file (Encryption.input.txt) in C:\PGP\Data directory. Put a dummy trigger message into input queue of the **PGPEncrypterMF.msgflow.** Flow read the file (C:\PGP\Data\Encryption.input.txt) from file system, signs and encrypts, writes the encrypted data into file system (C:\PGP\Data\Encryption.input.txt.asc) specified at node's input local environment. As per **InputFile Action** property **(Add Timestamp and Move to Archive)** specified in node properties, PGP Encrypter node moves the input file renamed with current timestamp suffix into archive directory (C:\PGP\Data\pgparchive). Note that archive directory name is fixed (**pgparchive**) and cannot be altered or overridden.

Troubleshooting

Following table illustrates some common errors and their troubleshooting guide.

Table-3: List of some common errors and their troubleshooting guide

S/N	Error Message	Troubleshooting Guide
1	com.ibm.broker.plugin.MbUserException class:com.ibm.broker.supportpac.pgp.impl.**PGPEncrypterNode** method:evaluate source:Message Encryption Failed! key: **Exception creating cipher message**	Make sure you updated **$MQSI_JRE_HOME/lib/security** directory with following unrestricted JCE policy jar files obtained from IBM site. • **local_policy.jar** • **US_export_policy.jar**
2	com.ibm.broker.plugin.MbUserException class:com.ibm.broker.supportpac.pgp.impl.**PGPEncrypterNode** method:evaluate source:Message Encryption Failed! key: **PGP Public Key not found: *Recipient1 <recipient1-pgp-keys@ibm.com>***	Verify whether the specified public key [Key User Id: **Recipient1 <recipient1-pgp-keys@ibm.com>**] exists in PGP public key repository specified at userdefined configurable service used by the PGP Encrypter node for encrypting the message/file.
3	com.ibm.broker.plugin.MbUserException class:com.ibm.broker.supportpac.pgp.impl.**PGPEncrypterNode** method:evaluate source:Message Encryption Failed! key: **PGP Private Key not found: *Sender1 <sender1-pgp-keys@ibm.com>***	Verify whether the specified private key [Key User Id: **Sender1 <sender1-pgp-keys@ibm.com>**] exists in PGP private key repository specified at userdefined configurable service used by the PGP Encrypter node to sign the message/file.
4	com.ibm.broker.plugin.MbUserException class:com.ibm.broker.supportpac.pgp.impl.**PGPEncrypterNode** method:evaluate source:Message Encryption Failed! key: **Private (Sign) key [0x45EE76DB] not found at Key Repository. Verify the key repository and/or passphrase.** Root cause: checksum mismatch at 0 of 20	Make sure whether passphrase of the PGP sign key (Signer's private key) is correct.
5	com.ibm.broker.plugin.MbUserException class:com.ibm.broker.supportpac.pgp.impl.**PGPDecrypterNode** method:evaluate source:Message Encryption Failed! key: **Private key [0xBAA79A8] not found at Key Repository [PGP-RCVR-CFG-SERVICE]. Verify the key repository and/or passphrase**. Root cause: checksum mismatch at 0 of 20	Possible reasons: • Message is encrypted by a public key whose conjugate private key does not exist at recipient's private key repository. • Passphrase of the PGP decryption key (Recipient's private key) is not correct.
6	com.ibm.broker.plugin.MbUserException class:com.ibm.broker.supportpac.pgp.impl.**PGPDecrypterNode** method:evaluate source:Message Encryption Failed! key: **Invalid Signature: Cannot find the public key [0x471B2AD9] in the PublicKey Repository [PGP-RCVR-CFG-SERVICE]**	Encrypted message is signed by a private key whose conjugate public key does not exist in recipient's public key repository. Get signer's public key and import into recipient's public key repository.

Conclusion

This article provides an industry standard solution that mitigates a huge gap in IBM Integration Bus Data Security zone. This solution (SupportPac) is not an IBM supplied in-built feature of IBM Integration Bus. This SupportPac is developed by the author of this article. Current version (v1.0.0.1) of this SupportPac only supports integrated signature generation/validation combined with PGP encryption/decryption processes. However future version will provide isolated signature generation/validation functionalities. Also future version of **pgpkeytool** will be enhanced with user-friendly GUI similar to IBM Key Management tool shipped with Websphere MQ.

You can post any query regarding to this PGP SupportPac at following IBM DeveloperWorks public community forum, author of this article will address those queries.

PGP SupportPac for IBM Integration Bus
(https://www.ibm.com/developerworks/community/groups/community/pgpsupportpaciib)

References

- **PGP Basics**
  - PGP Basics: PGP basic concepts (http://www.pgpi.org/doc/pgpintro/)
  - Bouncy Castle: Bouncy Castle Resources (http://www.bouncycastle.org/)
  - Gpg4Win: PGP encryption/decryption command line and GUI tool (http://www.gpg4win.org/index.html)
  - Portable PGP: Java based GUI tool for PGP (http://ppgp.sourceforge.net/)
  - GnuPG: GnuPG PGP library (http://www.gnupg.org/)
  - GitHub: Samples and other Artifacts (https://github.com/dipakpal/MyOpenTech-PGP-SupportPac)

- **Public Community at IBM DeveloperWorks**
  - PGP SupportPac for IBM Integration Bus:
https://www.ibm.com/developerworks/community/groups/community/pgpsupportpaciib

- **IBM Integration Bus resources**
  - IBM Integration Bus product page
    Product descriptions, product news, training information, support information, and more.
  - IBM Integration Bus V7 information center
    A single Web portal to all IBM Integration Bus V6 documentation, with conceptual, task, and reference information on installing, configuring, and using your IBM Integration Bus environment
  - Download free trial version of IBM Integration Bus
    IBM Integration Bus is an ESB built for universal connectivity and transformation in heterogeneous IT environments. It distributes information and data generated by business events in real time to people, applications, and devices throughout your extended enterprise and beyond.

MyOpenTech
(PGP SupportPac)

- IBM Integration Bus documentation library
  IBM Integration Bus specifications and manuals.
- IBM Integration Bus forum
  Get answers to technical questions and share your expertise with other IBM Integration Bus users.
- IBM Integration Bus support page
  A searchable database of support problems and their solutions, plus downloads, fixes, and problem tracking.

- **WebSphere resources**
  - developerWorks WebSphere
    Technical information and resources for developers who use WebSphere products. developerWorks WebSphere provides product downloads, how-to information, support resources, and a free technical library of more than 2000 technical articles, tutorials, best practices, IBM Redbooks, and online product manuals. Whether you're a beginner, an expert, or somewhere in between, you'll find what you need to build enterprise-scale solutions using the open-standards-based WebSphere software platform.
  - developerWorks WebSphere application integration developer resources
    How-to articles, downloads, tutorials, education, product info, and other resources to help you build WebSphere application integration and business integration solutions.
  - Most popular WebSphere trial downloads
    No-charge trial downloads for key WebSphere products.
  - WebSphere forums
    Product-specific forums where you can get answers to your technical questions and share your expertise with other WebSphere users.
  - WebSphere demos
    Download and watch these self-running demos, and learn how WebSphere products can provide business advantage for your company.
  - WebSphere-related articles on developerWorks
    Over 3000 edited and categorized articles on WebSphere and related technologies by top practitioners and consultants inside and outside IBM. Search for what you need.
  - developerWorks WebSphere weekly newsletter
    The developerWorks newsletter gives you the latest articles and information only on those topics that interest you. In addition to WebSphere, you can select from Java, Linux, Open source, Rational, SOA, Web services, and other topics. Subscribe now and design your custom mailing.
  - WebSphere-related books from IBM Press
    Convenient online ordering through Barnes & Noble.
  - WebSphere-related events
    Conferences, trade shows, Webcasts, and other events around the world of interest to WebSphere developers.