

OpenEdge Chiffrage avec SSL

Paul Koufalis
Président
Progresswiz Consulting

Progresswiz

Progresswiz Informatique

- **Offre de l'expertise technique Progress, UNIX, Windows et plus depuis 1999**
- **Spécialisé en matière de performance, disponibilité des systèmes et planification de la continuité d'affaires**

Agenda

Introduction

- **Pourquoi chiffrer**
- **Comment...**
 - Démarrer serveur
 - Connecter client ABL, JDBC et ODBC
- **Démo**
- **Questions**

Introduction

- **Qu'est-ce que c'est « SSL » ?**
 - *Secure Sockets Layer*
 - **Chiffage des communications**
- **Internet : HTTPS**
 - *HTTP Tunneling through SSL*
- **Intranet : SSL OpenEdge**
 - *TCP/IP tunneling through SSL*

Standards supportés

- **SSL Protocol Version 2.0/3.0**
 - **Transport Layer Security Working Group**
 - **Netscape**
- **TLS Protocol Version 1.0**
 - **Network Working Group RFC2246**
- **PKI x.509**
 - **PKIX Working Group**

- **OpenEdge utilise clé privée – clé publique**
- **Clé privé générée par OpenEdge**
 - **Envoyé à un CA**
 - ***Certificat Authority***
- **CA retourne la clé publique**
 - **Importé dans OpenEdge**

Certificats CA

- **Le CA est le « tiers de confiance* »**
 - « Trusted authentication authority »
 - Signe les demandes de clé
- **Client doit avoir certificat *root* du CA**
- **Certificats CA root connus distribués avec OE**
 - RSA, Thawte, Verisign...

* Merci au grand dictionnaire terminologique

Gestions des clés/certificats

- **OpenEdge gère ses clés dans \$DLC**
 - **\$DLC/keys pour les clés**
 - **\$DLC/certs pour les certificats CA**
- **Outils**
 - **\$DLC/bin/pkiutil**
 - **\$DLC/bin/certutil**
 - **\$DLC/java/jdk/bin/keytool**

Clé par défaut

- **OpenEdge fourni une clé test:**
 - « **default_server** »

Keystore entry: `default_server`

Certificate:

```
subject= /C=US/ST=NH/O=Progress Software  
Corporation/OU=Server Technologies/CN=Default  
Progress SSL Server
```

```
issuer= /C=US/ST=NH/O=Progress Software  
Corporation/OU=Server Technologies/CN=Progress Server  
Certificate Authority
```

```
notBefore=Feb 25 22:04:12 2004 GMT
```

```
notAfter=Feb 22 22:04:12 2014 GMT
```

JDBC/ODBC

- **Clients non-OpenEdge doivent gérer eux-mêmes leurs certificats**
 - **Java : keytool**
 - **ODBC : Fichier certificat**
 - **.Net : Microsoft Certificate Store Mgmt**

Pourquoi chiffrer?

- **Données confidentielles**
- **Mots de passe**
- ***Sniffer* réseau gratuit et facile à utiliser**
 - **WireShark**

Exemple 1

```
FOR EACH customer FIELDS(name) :  
    DISPLAY NAME.  
END.
```

Sniffer réseau

Realtek RTL8187 Wireless LAN USB NIC (Microsoft's Packet Sched...)

File Edit View Go Capture Analyze Statistics Telephony Tools Help

Filter: ip.addr == 192.168.0.103

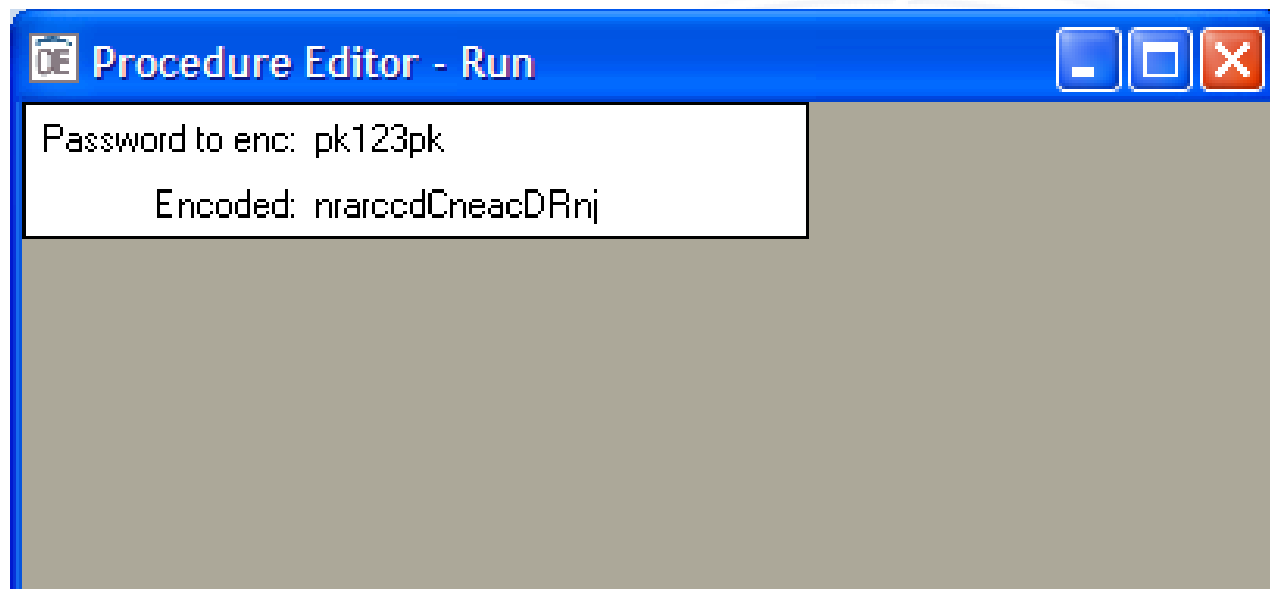
No.	Time
563	60.843476
564	60.845185
565	60.845212
566	60.847059
567	60.847084
568	60.986606

```

0000 00 14 a5 d4 f4 1e 00 15
0010 00 94 1b df 40 00 80 06
0020 00 67 0b b8 08 61 c8 eb
0030 f7 98 b3 24 00 00 00 00
0040 00 18 00 01 00 00 00 00
0050 00 00 00 00 01 b0 00 02
0060 00 00 14 08 04 00 40 01
0070 00 00 00 0c 00 00 0f 02
0080 01 81 fa 00 0b 00 01 02
0090 e8 10 4c 69 66 74 20 4c
00a0 6e 67
  
```

Realtek RTL8187 Wireless LAN USB NIC ... Packets: 797 Displayed: 411 M... Profile: Default

Exemple 2



Sniffer réseau

Realtek RTL8187 Wireless LAN USB NIC (Microsoft's Packet Scheduler) : Capturing - Wireshark

Filter: ip.addr==192.168.0.103

No. -	Time	Source	Destination	Protocol	Info
4107	49.007519	192.168.0.100	192.168.0.103	TCP	hbcj > concomp1 [PSH, ACK] Seq=58665 Ack=8758 win=16668 L
4108	49.008818	192.168.0.103	192.168.0.100	TCP	concomp1 > hbcj [PSH, ACK] Seq=8586 Ack=58665 win=16668 L
4109	*REF*	192.168.0.100	192.168.0.103	TCP	hbcj > concomp1 [PSH, ACK] Seq=8758 Ack=8586 win=16668 L
4166	49.039374	192.168.0.103	192.168.0.100	TCP	concomp1 > hbcj [PSH, ACK] Seq=8586 Ack=8758 win=16668 L
4167	49.039509	192.168.0.100	192.168.0.103	TCP	hbcj > concomp1 [PSH, ACK] Seq=8758 Ack=8586 win=16668 L
4168	49.041189	192.168.0.103	192.168.0.100	TCP	concomp1 > hbcj [PSH, ACK] Seq=8586 Ack=8758 win=16668 L
4169	49.041479	192.168.0.103	192.168.0.100	TCP	concomp1 > hbcj [PSH, ACK] Seq=8586 Ack=8758 win=16668 L
4170	49.041493	192.168.0.100	192.168.0.103	TCP	hbcj > concomp1 [PSH, ACK] Seq=8758 Ack=8586 win=16668 L
4171	49.042283	192.168.0.103	192.168.0.100	TCP	concomp1 > hbcj [PSH, ACK] Seq=8586 Ack=8758 win=16668 L
4172	49.042685	192.168.0.103	192.168.0.100	TCP	concomp1 > hbcj [PSH, ACK] Seq=8586 Ack=8758 win=16668 L
4173	49.042702	192.168.0.100	192.168.0.103	TCP	hbcj > concomp1 [PSH, ACK] Seq=8758 Ack=8586 win=16668 L
4174	49.042765	192.168.0.100	192.168.0.103	TCP	hbcj > concomp1 [PSH, ACK] Seq=8758 Ack=8586 win=16668 L
4175	49.044694	192.168.0.103	192.168.0.100	TCP	concomp1 > hbcj [PSH, ACK] Seq=8586 Ack=8758 win=16668 L
4176	49.044803	192.168.0.100	192.168.0.103	TCP	hbcj > concomp1 [PSH, ACK] Seq=8758 Ack=8586 win=16668 L
4177	49.046230	192.168.0.103	192.168.0.100	TCP	concomp1 > hbcj [PSH, ACK] Seq=8586 Ack=8758 win=16668 L
4178	49.046692	192.168.0.103	192.168.0.100	TCP	concomp1 > hbcj [PSH, ACK] Seq=8586 Ack=8758 win=16668 L
4179	49.046712	192.168.0.100	192.168.0.103	TCP	hbcj > concomp1 [PSH, ACK] Seq=8758 Ack=8586 win=16668 L

Destination port: concomp1 (1802)
[Stream index: 196]
Sequence number: 58665 (relative sequence number)
[Next sequence number: 58818 (relative sequence number)]
Acknowledgement number: 8758 (relative acknowledgement number)
Header length: 20 bytes
 Flags: 0x18 (PSH, ACK)
window size: 63688
 Checksum: 0x2455 [validation disabled]
 [SEQ/ACK analysis]
 Data (153 bytes)
 data: 0000006002b0099004b001800010000000000000000
 [Length: 153]

```

0000 00 14 a5 d4 f4 1e 00 15 af 03 94 29 08 00 45 00
0010 00 c1 74 86 40 00 80 06 03 95 c0 a8 00 64 c0 a8
0020 00 67 0b b8 07 0a ba ff 4d 7c b7 ec 8a fa 50 18
0030 f8 c8 24 55 00 00 00 00 00 6d 00 2d 00 99 00 4d
0040 00 18 00 01 00 00 00 00 00 00 00 00 00 00 00 00
0050 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0060 00 00 00 02 00 00 01 00 00 7f 68 00 4d ff fb
0070 00 00 00 00 00 0f 02 00 00 00 00 00 00 00 00 00
0080 22 b2 e7 00 02 00 3a 01 fb 02 70 6b 10 6e 72 61
0090 72 63 63 64 43 6e 65 61 63 44 52 6e 6a 02 70 6b
00a0 fa 00 09 fd fd fd fd fd fd fd fd fd fd fd fd fd
00b0 fd fd fd fd fd fd fd fd fd fd fd fd fd fd fd fd
00c0 fd fd fd fd fd fd fd fd fd fd fd fd fd fd fd fd
  
```

Data (data.data), 153 bytes | Packets: 4346 Displayed: 745 Marked: 1 | Profile: Default

Procédure

- **Créer la demande**
- **Faire signer la demande**
- **Importer le certificat retourné**
- **Démarrer BD**
- **Connecter clients**
 - **ABL**
 - **JDBC**
 - **ODBC**

Créer une demande de cert

```
C:\apps\openedge\wrk102a>pkiutil -newreq gupq
Loading 'screen' into random state - done
Generating a 1024 bit RSA private key
.....+++++++
writing new private key to 'C:\apps\openedge\oe102a/keys/requests/gupq.pk1`

Country Name (2 letter code) [US]:CA
State or Province Name (full name) []:QC
Locality Name (eg, city) []:
Organization Name (eg, company) []:Progresswiz
Organizational Unit Name (eg, section) []:
Server DNS name []:pckoup
```

You may now use the file `C:\apps\openedge\oe102a/keys/requests/gupq.pk10` to request a new Digital Certificate from a CA Certificate Authority.

After you obtain the new Digital Certificate from the CA use the `-import` command to insert the certificate into the keystore.

Faire signer la demande

- **Verisign, etc...**
 - Envoyer le .pk10
- **J'utilise un certificat CA créé avec OpenSSL**

```
C:\apps\openedge\wrk102a\sslkeys>openssl ca -config  
pk.conf -notext -out gupq.pem.cert -infile  
C:\apps\openedge\oe102a/keys/requests/gupq.pk10
```

```
<détails omis...>
```

```
Certificate is to be certified until Feb  1 02:19:29  
2011 GMT (365 days)  
Sign the certificate? [y/n]:y
```

Importer le certificat signé

- **Dans l'installation OpenEdge serveur**
 - **Attention au comportement bizarre de pkiutil (voir démo)**

```
C:\apps\openedge\wrk102a>pkiutil -import gupq  
sslkeys\gupq.pem.cert
```

```
Importing private key alias gupq:
```

```
Importing certificate file sslkeys\gupq.pem.cert
```

```
Enter keystore password to alias gupq: gupq
```

Valider

```
C:\apps\openedge\wrk102a>pkiutil -list
```

```
Keystore entry: gupq
```

```
Certificate:
```

```
subject= /C=CA/ST=QC/O=Progresswiz/CN=pckoup
```

```
issuer= /C=CA/ST=Some-State/O=Internet Widgits Pty Ltd
```

```
notBefore=Feb 1 02:19:29 2010 GMT
```

```
notAfter=Feb 1 02:19:29 2011 GMT
```

Importer le certificat du CA

- **Seulement si CA interne**
- **Certificats connus sont déjà installer avec OE**

```
C:\apps\openedge\wrk102a\ssl>certutil -import  
..\sslkeys\pkca.crt
```

```
Importing trusted certificate to alias name: 39d36856
```

- **Nouveau fichier \$DLC/certs/39d36856.0**

Distribuer certificat CA

- **Encore - seulement si demande signé par CA interne**
 - **Chaque client OpenEdge doit importer avec certutil -import**

Mot de passe

- Lors de la création de la requête un mot de passe a été spécifié
- Générer le mot de passe encrypté:

```
C:\apps\openedge\wrk102a\ssl>genpassword -password gupq  
37273f36
```

Démarrer la BD

```
C:\apps\openedge\wrk102a\ssl>_mprosrv ssl -H pckoup -S  
5000 -ssl -keyalias gupq  
-keyaliaspasswd 37273f36
```

■ Dans le .lg de la BD:

```
SSL Encryption has been enabled for ALL TCP/IP  
connections to this database  
SSL Key Alias Name (-keyalias): gupq
```


Attention!

- **Impossible d'avoir des connexions distantes chiffrées et non-chiffrées**
- **Tous les brokers démarrés utiliseront la même clé SSL**

Connecter un client ABL

```
prowin32 ssl -H pckoup -S 5000
```

- **Pas besoin de spécifié `-ssl`**
 - Le serveur le dit au client
- **Rien dans le `.lg` confirme que la connexion est SSL**
 - À part les messages de démarrage

Sans certificat CA

```
prowin32 ssl -H pckoup -S 5000
```

```
+----- Error -----+
| SSL error 12072 - SSL Client handshake failure (-54) unable to get local |
| issuer certificate: for 39d36856.0 in C:\apps\OpenEdge\oe102a\certs |
| occurred. (12168) |
| Error starting SSL handshake with the OpenEdge database server. (12167) |
|-----|
| <OK> |
+-----+
```

■ Dans le .lg de la BD

```
Usernum 1 terminated abnormally.
```

Client JDBC

- Créer un *keystore* Java avec `keytool`
 - `$DLC/jdk/bin/keytool`

```
C:\apps\openedge\wrk102a\sslkeys>keytool -import -alias  
ca -file pkca.crt -keypass ca -keystore gupqstore -  
storepass gupq123
```

```
<snip...>
```

```
Trust this certificate? [no]: y
```

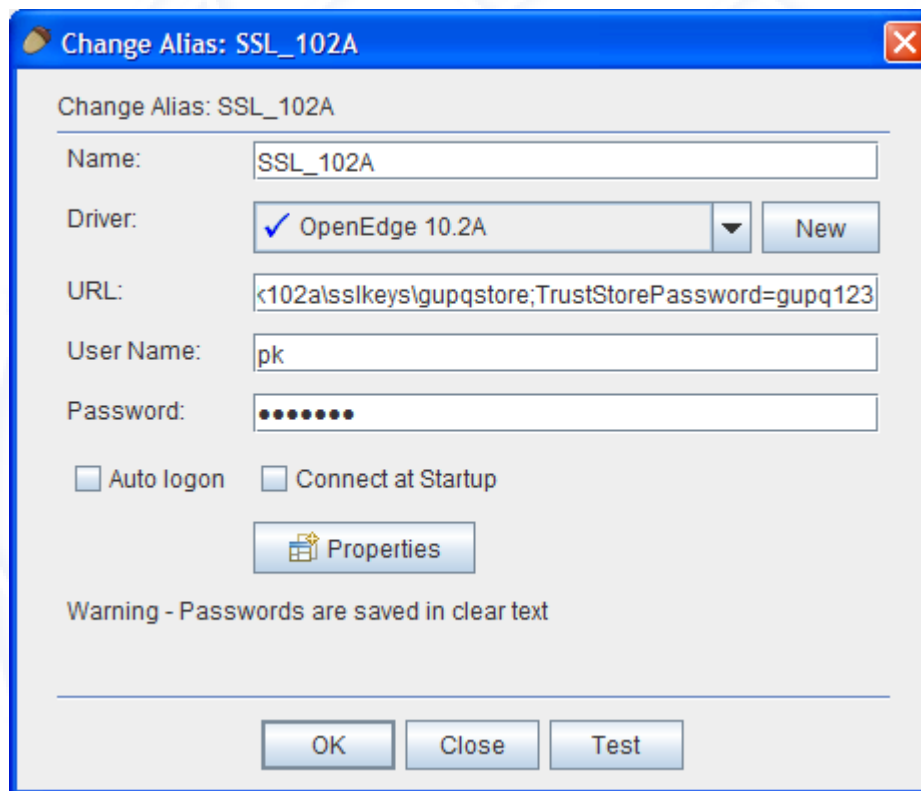
```
Certificate was added to keystore
```

- Fichier « `gupqstore` » créé

Test JDBC

■ Squirrel SQL Client (gratuit)

```
jdbc:datadirect:openedge://localhost:5000;databaseName=ssl;EncryptionMethod=ssl;Truststore=c:\apps\openedge\wrk102a\sslkeys\gupqstore;TrustStorePassword=gupq123
```



Test JDBC

■ Sans paramètres SSL:

```
SSL_102A_NOSSLPARAMS: [DataDirect][OpenEdge JDBC  
Driver]SSL Mismatch. Encryption method in client and  
server must match.
```

Client ODBC

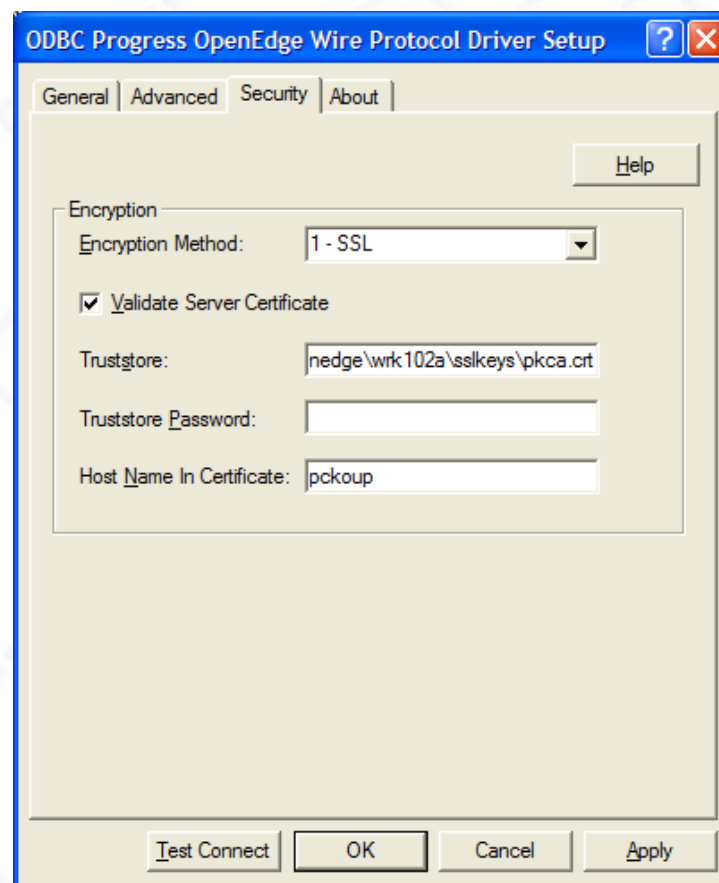
- **Un peu plus compliqué**
- **Vraiment mais vraiment pas du tout documenté**
 - **Rien dans la doc ni dans le KB**

Client ODBC

- **OE 10.1C SP 4 et plus**
 - Avant ça...pas certain
 - Il y avait un bug...
- **Copier pgcrypto.dll et pgss123.dll dans %windir%**
- **Créer DSN dans administrateur ODBC**

DSN ODBC

- Créer DSN comme d'habitude
- Spécifier encryption SSL
- Fichier certificat CA dans « TrustStore »
- Aucun mot de passe n'est nécessaire



Démo

- **Démarrage de serveur**
- **Connexion client ABL**
- **Connexion client JDBC**
- **Connexion client ODBC**



Questions ?

Informatique Progresswiz

- **Questions ou commentaires?**
pk@progresswiz.com
- **PPTs, articles et outils disponible au**
www.progresswiz.com



Merci !