Olá pessoal,

Com o intuito de acabar com a dependência da CAPICOM, nos fontes do Projeto ACBr, apliquei um amplo refactoring, nas Units de **ACBrDFeSSL** e suas derivadas...

## O que é CAPICOM ?

https://en.wikipedia.org/wiki/CAPICOM

## Porque usávamos a CAPICOM ?

Usar diretamente as APIs do Windows não é uma tarefa simples.... A CAPICOM, facilita um pouco, as tarefas que podem ser feitas com a WinCrypt (ou MS Crypto), para acesso a certificados digitais instalados no Windows

## Quais as desvantagens da CAPICOM ?

- A Microsoft condenou a mesma como obsoleta. (esse é o principal motivo)
- Ela precisa ser registrada no Windows para funcionar
- Não suporta 64 bits

## O que será usado no lugar da CAPICOM ?

Usaremos diretamente as APIs do Windows, ou seja, a  WinCrypt (também conhecida como "MS Crypto" ou "CAPI").

Ou seja, encaramos o desafio e agora usamos apenas métodos da WinCrypt para acessos a Certificados Digitais no Windows. Para facilitar o acesso a API WinCrypt, estamos usando as Units do diretório: "Fontes\Terceiros\CodeGear\", mas especificamente a Unit "ACBr_WinCrypt.pas".

## Quais as vantagens da WinCrypt ?

- Ela está presente de forma nativa, em todas as versões do Windows (desde o Windows XP), ou seja, não requer instalação.
- Possui versões 32 e 64 bits
- Não requer registro da DLL
- Não requer a instalação de pacotes .NET ou Java

## Onde posso encontrar a WinCrypt ?

Ela já está instalada, de forma nativa, no seu Windows... com o nome: **"crypt32.dll"**

- Se o seu Windows é 64 bits, você encontrará a mesma em:
  - 32 bits: "C:\Windows\SysWOW64"
  - 64 bits "C:\Windows\System32"
- Se o seu Windows é 32 bits, você encontrará a mesma em:
  - "C:\Windows\System32"

## O suporte a Delphi7 será mantido ?

SIM. Apesar de já anunciarmos o fim do Suporte a D7, tivemos o cuidado de testar as alterações no D7. Para isso, adaptamos as units da pasta "Fontes\Terceiros\CodeGear\" para o suporte a D7...

## Como configurar para usar a WinCrypt e não a CAPICOM ?

A maneira mais simples é configurar a seguinte propriedade:

```
  ACBrNFe1.Configuracoes.Geral.SSLLib := libWinCrypt;
```

Na verdade, a propriedade *ACBrDFe.Configuracoes.Geral.SSLLib* passou a ser virtual... ou seja, ela configurará de forma indireta, as 3 novas bibliotecas de TDFeSSL... Se você ler os fontes, quando rodamos o código acima, o seguinte código será executado.

```
procedure TGeralConf.SetSSLLib(AValue: TSSLLib);
    case AValue of
.....
      libWinCrypt:
      begin
        SSLCryptLib := cryWinCrypt;
        SSLHttpLib := httpWinHttp;
        SSLXmlSignLib := xsMsXml;
      end;
    end;
```

Se você deseja uma configuração diferenciada, poderá configurar as bibliotecas individualmente...Exemplo:

```
 ACBrNFe1.Configuracoes.Geral.SSLCryptLib := cryWinCrypt;
 ACBrNFe1.Configuracoes.Geral.SSLHttpLib := httpWinINet;
 ACBrNFe1.Configuracoes.Geral.SSLXmlSignLib := xsXmlSec;
```

## Como remover completamente, as Units da CAPICOM dos meus fontes ?

Abra o arquivo **\ACBr\Fontes\ACBrComum\ACBr.inc** e altere a seguinte linha:

```
{.$DEFINE DFE_SEM_CAPICOM}
```

para:

```
{$DEFINE DFE_SEM_CAPICOM}
```

Ou seja, remova o "." do inicio

## O que mudou em ACBrDFeSSL ?

Muita coisa.... (veja abaixo o trecho do "Change-Log").. Estudar os fontes do projeto Demo *"\ACBr\Exemplos\ACBrDFe\ACBrNFe\Delphi"*, é a melhor maneira de conhecer as modificações. Veja abaixo, um resumo ilustrado:

**1 - Agora você pode criar a sua própria janela de escolha de Certificado**


Veja esse exemplo de código, extraído de ACBrNFe_Demo. onde usamos o método *"ACBrNFe1.SSL.LerCertificadosStore"*, para carregar todos os certificados da Store, definida

em *"ACBrNFe1.SSL.StoreName"*, após isso, as informações dos certificados podem ser obtidas em *"ACBrNFe1.SSL.ListaCertificados"*

```
ACBrNFe1.SSL.LerCertificadosStore;

For I := 0 to ACBrNFe1.SSL.ListaCertificados.Count-1 do
begin
  with ACBrNFe1.SSL.ListaCertificados[I] do
  begin
```

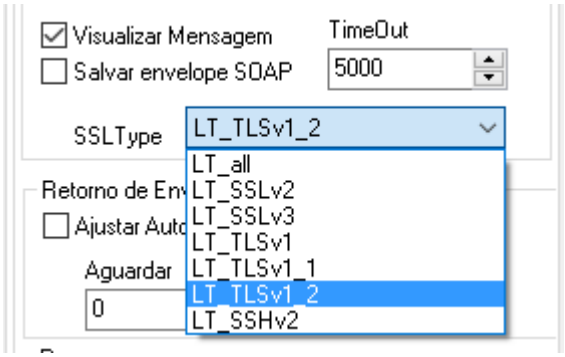
**2 - Agora você pode selecionar as bibliotecas de TDFeSSL, individualmente**


**CryptLib:** Permite definir qual será a biblioteca de Criptografia. Ela possui métodos como:"SelecionarCertificado", "CarregarCertificado", "CalcHash". além de propriedades como "DadosCertificado" e "ListaCertificados".
TSSLCryptLib = (cryNone, cryOpenSSL, cryCapicom, cryWinCrypt)

**HttpLib:** Usada para acesso HTTP e HTTPs, permitindo informar o Certificado na conexão. Possui métodos como: "Enviar" e propriedades como: "HTTPResultCode" e "InternalErrorCode"
TSSLHttpLib = (httpNone, httpWinINet, httpWinHttp, httpOpenSSL, httpIndy);

**XMLSignLib**: Usada para validar XMLs (contra um Schema), assinar um XML, Validar a assinatura existente em um XML. Possui métodos como: "Assinar", "Validar" e "VerificarAssinatura"
TSSLXmlSignLib = (xsNone, xsXmlSec, xsMsXml, xsMsXmlCapicom);

**3 - Independência das configurações de segurança do I.E.**



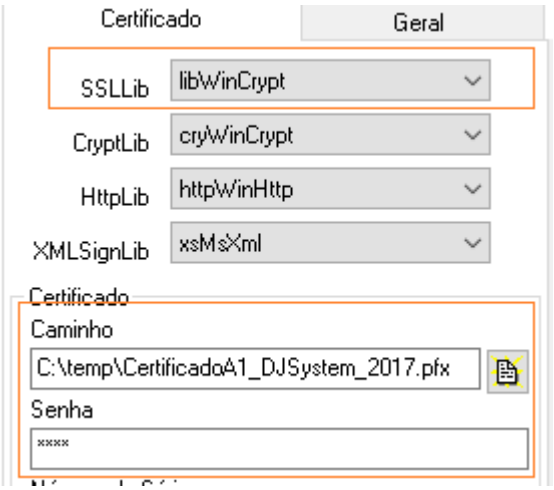

Isso pode ser obtido, se você utilizar *SSLHttpLib = "httpWinHttp" ou "httpOpenSSL"*

Você poderá definir nos seus fontes, independente das configurações do Internet Explorer, configurações como o Tipo de segurança e TimeOut da tentativa de conexão.

Essa funcionalidade já estava presente nas Units de acesso que utilizavam o OpenSSL a algum tempo. e agora com a nova Unit que faz acesso a HTTPS, usando a API do Windows chamada "WinHTTP", isso também será possível.

O modelo: "httpWinINet" irá usar a API do Windows, chamada "WinINet", a qual já utilizávamos, e ela depende de configurações do I.E.

**4 - Carregar o certificado por ArquivoPFX ou DadosPFX, com a WinCrypt ou CAPICOM**




Essa funcionalidade já estava presente, quando *SSLCryptLib = cryOpenSSL.* e não estava disponível para CAPICOM. Mas agora isso é possível, com a *SSLCryptLib* = cryCapicom ou cryWinCrypt.

Ou seja, Se você tem um **certificado A1, você não precisa instalar o certificado no Windows.** Isso pode parecer pouco importante em uma primeira impressão... Mas veja as possibilidades: O certificado A1 poderia estar em um Banco de dados, ou em um Servidor Web, e ser carregado de forma dinâmica pela sua aplicação, independente de ser instalado manualmente no Windows.

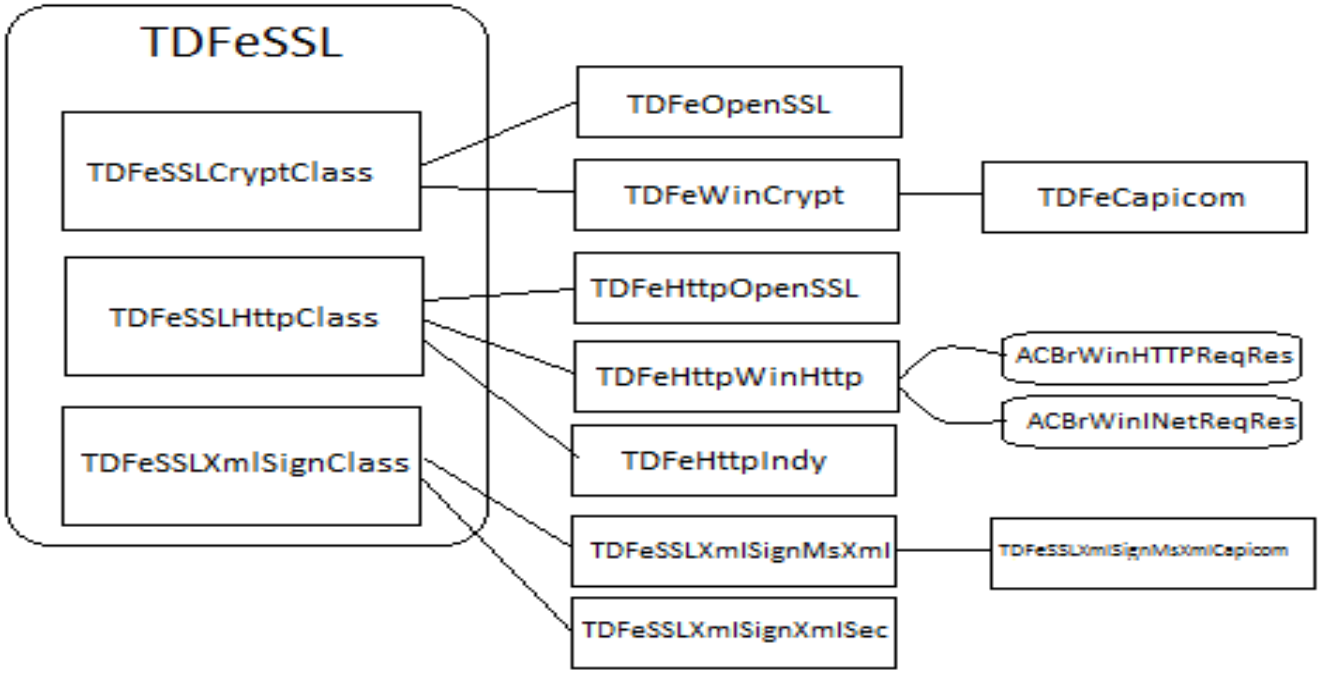
**5 - Compilar seu Executável em 64 bits**

Lembre-se que quando você compila o seu programa em 64 bits, todas as DLLs externas de qual ele necessitar, também devem ser de 64 bits. Portanto para isso, você não poderá usar a *XMLSignLib = xsMsXml*, pois a biblioteca da Microsoft para assinatura de XMLs "MSXML" não possui versão 64 bits.

Mas observe que agora você pode usar a biblioteca WinCrypt com a XmlSec, basta configurar corretamente as bibliotecas de criptografia.

Nota: Ainda não conseguimos, fazer com que a XMLSec possa usar certificados A3, mas isso deverá ser possível no futuro, pois a XMLSec tem suporte a "MSCrypto"

## Diagrama de Classes




## Como posso ajudar ? (Tarefas a serem efetuadas)

**1 - Fazer a XmlSec funcionar usando a "mscrypto"**

Ainda não conseguimos fazer a XMLSec, usar a MSCrypto, atualmente ele apenas usa a "openssl". Porque isso é importante ? Temos vários problemas, com a **msxml**, como por exemplo:

- A Microsoft não distribui a mesma, de forma nativa, com o Windows (arquivo msxml5.dll)
- Ela não suporta 64 bits
- A licença de uso dassa biblioteca, é valida apenas para quem tem o Office instalado...

Portanto, seria ótimo se pudéssemos ficar livres da MSXML, mas para isso, precisamos fazer o ACBr conseguir usar a XMLSec com suporte a MSCrypto (hoje ele só suporta OpenSSL)...
Na verdade, já podemos usar WinCrypt + XmlSec, mas apenas para certificados A1, pois o ACBr é capaz de exportar o certificado A1 do Windows, para que o mesmo seja usado pelo OpenSSL. (ele fará isso internamente, e de forma transparente para o usuário)

Quando conseguirmos fazer a XmlSec usar a MSCrypto (ou WinCrypt), conseguiremos compilar a aplicação em 64 bits, e com suporte a certificados A3

**2 - Compilar os fontes da XMLSec no Windows, em 32 e 64 bits**

Hoje o único site que distribui a XMLSec já compilada para Windows
é https://www.zlatkovic.com/libxml.en.html (Thanks Igor).

Entretanto, podemos notar que os binários estão defasados, e não há uma versão 64 bits, com suporte a "mscrypto"

## Veja como ficou o "Change-Log" do refactoring em ACBrDFeSSL

```
-- ACBrDFeSSL --
[*] Amplo refactoring promovido, separando a classe "TDFeSSLClass" em 3 novas
    classes:
    "TDFeSSLCryptClass" - para Carregar certificados e efetuar criptografia
    "TDFeSSLHttpClass" - para comunicação HTTP/HTTPS com suporte a Certificados
    "TDFeSSLXmlSignClass" - Para Validar XMLs, validar assinaturas e Assinar XML
                            com Certificados
[+] "TSSLLib", adicionado os tipos "libWinCrypt, libCustom"
[+] Criada nova classe "TDadosCertificado", para conter os dados do
    certificado carregado
[+] Criada nova classe "TListaCertificados",para conter uma lista de Objetos
    do tipo TDadosCertificado, com todos os certificados de uma "Store", e após
    a chamada do método "TDFeSSL.LerCertificadosStore"
[+] Adicionada propriedade "TDFeSSL.StoreName: String", usada apenas no Windows.
    Nome da Store a ser aberta, padrão "MY"
[+] Adicionada propriedade "TDFeSSL.StoreLocation: TSSLStoreLocation", usada
    apenas no Windows. Default "slCurrentUser".
    TSSLStoreLocation = (slMemory, slLocalMachine, slCurrentUser, slActiveDirectory,
slSmartCard);
[+] Adicionado o método: "TDFeSSL.LerCertificadosStore", apenas Windows, para
    carregar todos os Certifcados de "TDFeSSL.StoreName" para a lista de Objetos:
    "TDFeSSL.ListaCertificados"
[+] Adicionado a propriedade "TDFeSSL.DadosCertificado", para permitir acesso
    aos dados do certificado carregado
[+] Adicionada a propriedade "TDFeSSL.SSLCryptLib: TSSLCryptLib" default cryNone;
    para definir a classe de criptografia
    TSSLCryptLib = (cryNone, cryOpenSSL, cryCapicom, cryWinCrypt);
[+] Adicionada a propriedade "TDFeSSL.SSLHttpLib: TSSLHttpLib" default httpNone;
    para definir a classe de comunicação HTTP/HTTPS
    TSSLHttpLib = (httpNone, httpWinINet, httpWinHttp, httpOpenSSL, httpIndy);
[+] Adicionada a propriedade "TDFeSSL.SSLXmlSignLib: TSSLXmlSignLib" default xsNone;
    para definir a classe de assinatura de validação de XML
    TSSLXmlSignLib = (xsNone, xsXmlSec, xsMsXml, xsMsXmlCapicom);
[+] Adicionada a propriedades "TDFeSSL"SSLType: TSSLType" default LT_all;
    para permitir definir o tipo de criptografia em HTTPS sendo:
    TSSLType = (LT_all, LT_SSLv2, LT_SSLv3, LT_TLSv1, LT_TLSv1_1, LT_TLSv1_2, LT_SSHv2)
    suportado apenas em TDFeHttpOpenSSL e TDFeHttpWinHttp

-- ACBrDFeConfiguracoes --
[+] Adicionada as propriedades:
    property SSLCryptLib: TSSLCryptLib
    property SSLHttpLib: TSSLHttpLib
    property SSLXmlSignLib: TSSLXmlSignLib
[*] Propriedade "SSLLib: TSSLLib" passou a ser virtual, e mantida por
    compatibilidade. Ajusta-la irá produzir ajustes em "SSLCryptLib",
    "SSLHttpLib" e "SSLXmlSignLib". Exemplo:
    if SSLLib = libOpenSSL then
    begin
      SSLCryptLib := cryOpenSSL;
      SSLHttpLib := httpOpenSSL;
      SSLXmlSignLib := xsXmlSec;
    end;

-- ACBrDFe --
[+] Adicionado suporte a configurações de "SSLCryptLib", "SSLHttpLib",
    "SSLXmlSignLib"

-- ACBrDFeOpenSSL --
[*] Amplo refactoring. Removido código referente a comunicação HTTP/HTTPs
    que foi migrado para "ACBrDFeHttpOpenSSL"
[*] Removido código referente a assinatura digital e Validação de XML,
    que foi migrado para "ACBrDFeXsXmlSec"

-- ACBRDFeWinCrypt --
[+] Nova Unit, para manipular Certificados do Windows e efetuar assinatura
```

```
    digital, usando a Win API WinCrypt (MSCrypto/CAPI)

-- ACBrDFeCapicom --
[*] Refactoring, para usar boa parte do código de "ACBRDFeWinCrypt"

-- ACBrDFeHttpOpenSSL --
[+] Adicionada nova Unit, derivada de ACBrDFeOpenSSL, criando implementação da
    classe de TDFeSSLHttpClass para comunicação http e https, usando a Synapse
    e OpenSSL

-- ACBrDFeHttpWinApi --
[+] Adicionada nova Unit, derivada de ACBrDFeCapicom, criando implementação da
    classe de TDFeSSLHttpClass para comunicação http e https, usando as APIs do
    Windows WinHttp ou WinINet

-- ACBrDFeHttpIndy, ACBrDFeCapicomDelphiSoap --
[*] Unit renomeada de "ACBrDFeCapicomDelphiSoap" para "ACBrDFeHttpIndy", e
    refatorada para não depender da CAPICOM

-- ACBrDFeXsXmlSec --
[+] Adicionada nova Unit, derivada de ACBrDFeOpenSSL, criando implementação da
    classe de TDFeSSLXmlSignClass usando a Lib XMLSEC

-- ACBrDFeXsMsXml --
[+] Adicionada nova Unit, derivada de ACBrDFeCapicom, criando implementação da
    classe de TDFeSSLXmlSignClass usando a Lib MSXML

-- ACBrDFeXsMsXmlCapicom --
[+] Adicionada nova Unit, derivada de ACBrDFeCapicom, criando implementação da
    classe de TDFeSSLXmlSignMsXml usando a Lib MSXML e CAPICOM

-- ACBrDFeException --
[+] Adicionado o exception "EACBrDFeExceptionNoPrivateKey"

-- ACBrDFeUtil --
[+] Adicionado o método "SignatureElement: String"
    (por DSA)
```

Obrigado... e considere nos ajudar, **contratando o SAC,** por pelo menos 1 mês
http://www.projetoacbr.com.br/forum/sacv2/sobre/
http://www.projetoacbr.com.br/forum/sacv2/questoes_importantes/
http://www.projetoacbr.com.br/forum/sacv2/cadastro/