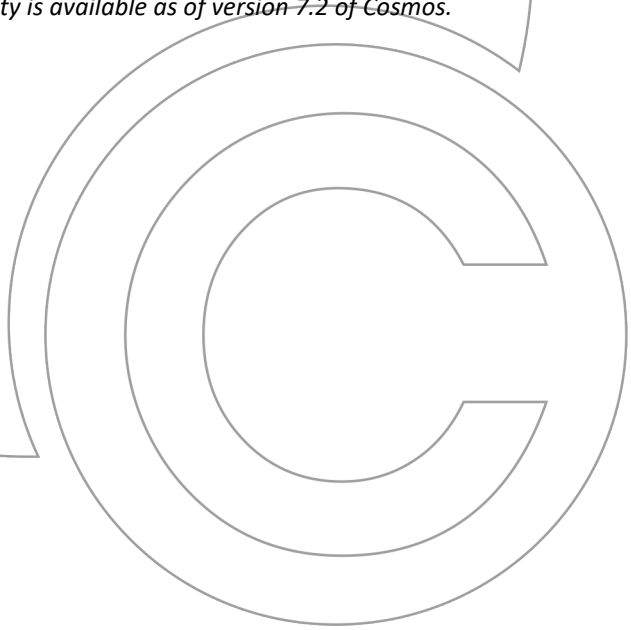
CosSignFile

The CosSignFile dll (cossigfile.dll) allows to digitally sign files in PDF, XML and binary format, as well as the signature of XML electronic invoice files in FacturaE format versions 3.2.1 and 3.2.2.

This utility is available as of version 7.2 of Cosmos.



BASE100

BASE 100, S.A.
www.base100.com

Index

1. Introduction

The **CosSignFile** dll (cossignfile.dll) allows to sign from Cosmos documents in PDF, XML and binary format, as well as XML files of electronic invoice in FacturaE format, using a digital certificate. This certificate must be installed on the system in a certificate store (file).

CosSignFile uses the JAVA libraries included in the FacturaE software, which is a free software developed in the Java language. For this reason, it is necessary that a 32-bit Java virtual machine be installed on the system and that the directory where is located the JVM.DLL file be included in the search *path*.

2. Functions exported by CosSignFile dll

The steps to follow to sign a document from Cosmos are the following:

1. Run the CosSignFileCreateSigner function to obtain a handler that allows us to execute the functions necessary to carry out the signature process.
2. Obtain the literals corresponding to the values of the properties of the document to be signed (certificate store format, signature algorithm, signature format, signature mode). The values of these properties can be obtained by invoking the functions:
    a. CosSignFileGetKeystoreFormats
    b. CosSignFileGetSignAlgorithms
    c. CosSignFileGetPAdESSignFormats
    d. CosSignFileGetCAdESSignFormats
    e. CosSignFileGetXAdESSignFormats
    f. CosSignFileGetSignModes.
3. Execution of the signature process with the call to the corresponding function (CosSignFileSignFacturaE, CosSignFileSignPDF, CosSignFileSignPAdES, CosSignFileSignXAdES, CosSignFileSignCAdES).
4. Release of the resources used during the signature process with the execution of the CosSigFileFreeSigner function.

2.1 CosSignFileCreateSigner

This function returns a unique numerical identifier that will be necessary for the execution of the signature property assignment functions and for the signature process itself.

Syntax:

```
CosSignFileCreateSigner () return Integer
```

Returns:

Unique numeric identifier needed to execute the rest of the functions of the signature process.

If the Cosmos license is not registered, it will return -1.

2.2 CosSignFileFreeSigner

This function will release the resources of the signature handler sent as a parameter.

After executing this function, the identifier passed as a parameter can't be used again to sign a document. A new identifier must be created by executing the CosSignFileCreateSigner function.

Syntax:

```
CosSignFileFreeSigner (signID as integer) return Integer
```

Parameters:

signID	Identifier of the signature returned in the call to the CosSignFileCreateSigner function.

Returns:

0	The function has been executed successfully.
-1	The signature identifier passed as a parameter does not exist.

2.3 CosSignFileSignFacturaE

This function will sign in FacturaE format the XML file passed as a parameter.

Syntax:

```
CosSignFileSignFacturaE (
    signID as integer,
    xmlPath as char,
    xsigPath as char,
    keystorePath as char,
    keystoreFormat as char,
    keyStorePasswd as char,
    alias as char,
    aliasPasswd as char) return Integer
```

Parameters:

signID	Identifier of the signature returned in the call to the CosSignFileCreateSigner function.
xmlPath	Absolute path of the XML file in FacturaE format that you want to sign.
xsigPath	Absolute path of the output XML file (XSIG) resulting from the signature of the XML file indicated in the "xmlPath" parameter.
keystorePath	Absolute path of the certificate store file where the certificate that you want to use is stored to carry out the signature process.
keystoreFormat	String of characters that indicates the format of the certificate store. The possible values of this parameter can be obtained by invoking the function CosSignFileGetKeystoreFormats function(jks,pkcs12, etc.).
keystorePasswd	Password of the certificate store specified in the "keystorePath" parameter.
alias	Alias of the electronic certificate that you want to use to make the signature of the file.
aliasPasswd	Password of the electronic certificate specified in the "alias" parameter.

Returns:

0	The signature identifier passed as a parameter does not exist or there was an error in the execution of the signature function.
1	The function has been executed successfully.

2.4 CosSignFileSignPDF

This function will sign the PDF file passed as a parameter.

Syntax:

```
CosSignFileSignPDF (
    signID as integer,
    inPDFPath as char,
    outPDFPath as char,
    keystorePath as char,
    keystoreFormat as char,
    keyStorePasswd as char,
    alias as char,
    aliasPasswd as char) return Integer
```

Parameters:

signID	Identifier of the signature returned in the call to the CosSignFileCreateSigner function.
inPDFPath	Absolute path of the PDF file that you want to sign.
outPDFPath	Absolute path of the output PDF file resulting from the signature of the PDF file indicated in the parameter "inPDFPath".
keystorePath	Absolute path of the certificate store file where the certificate that you want to use is stored to carry out the signature process.
keystoreFormat	String of characters that indicates the format of the certificate store. The possible values of this parameter can be obtained by invoking the function CosSignFileGetKeystoreFormats function (jks,pkcs12, etc.).
keystorePasswd	Password of the certificate store specified in the "keystorePath" parameter.
alias	Alias of the electronic certificate that you want to use to make the signature of the file.
aliasPasswd	Password of the electronic certificate specified in the "alias" parameter.

Returns:

0	The signature identifier passed as a parameter does not exist or there was an error in the execution of the signature function.
1	The function has been executed successfully.

2.5 CosSignFileSignPAdES

This function will sign the PDF file passed as a parameter using the PAdES format (PDF Advanced Electronic Signatures).

Syntax:

```
CosSignFileSignPAdES (
    signID as integer,
    inFilePath as char,
    outFilePath as char,
    keystorePath as char,
```

```
    keystoreFormat as char,
    keyStorePasswd as char,
    alias as char,
    aliasPasswd as char,
    algorithm as char,
    format as char,
    mode as char) return Integer
```

Parameters:

signID	Identifier of the signature returned in the call to the CosSignFileCreateSigner function.
inFilePath	Absolute path of the PDF file that you want to sign.
outFilePath	Absolute path of the output PDF file resulting from the signature of the PDF file indicated in the "inFilePath" parameter.
keystorePath	Absolute path of the certificate store file where the certificate that you want to use is stored to carry out the signature process.
keystoreFormat	String of characters that indicates the format of the certificate store. The possible values of this parameter can be obtained by invoking the CosSignFileGetKeystoreFormats function (jks,pkcs12, etc.).
keystorePasswd	Password of the certificate store specified in the "keystorePath" parameter.
alias	Alias of the electronic certificate that you want to use to make the signature of the file.
aliasPasswd	Password of the electronic certificate specified in the "alias" parameter.
algorithm	Encryption algorithm that you want to use. The list of supported algorithms can be consulted by executing the CosSignFileGetSignAlgorithms function (SHA1withRSA, SHA256withRSA, etc).
format	Signature format. The list of supported signature formats can be viewed by executing the ***CosSignFileGetPAdESSignFormats*** function.
mode	Signature mode. The list of supported signature modes can be consulted by executing the ***CosSignFileGetSignModes*** function (implicit, explicit, etc).

Returns:

0	The signature identifier passed as a parameter does not exist or there was an error in the execution of the signature function.
1	The function has been executed successfully.

2.6 CosSignFileSignXAdES

This function will sign the XML file passed as a parameter using the format XAdES (XML Advanced Electronic Signatures).

Syntax:

```
CosSignFileSignXAdES (
    signID as integer,
```

```
inFilePath as char,
outFilePath as char,
keystorePath as char,
keystoreFormat as char,
keyStorePasswd as char,
alias as char,
aliasPasswd as char,
algorithm as char,
format as char,
mode as char) return Integer
```

Parameters:

signID	Identifier of the signature returned in the call to the CosSignFileCreateSigner function.
inFilePath	Absolute path of the XML file to be signed.
outFilePath	Absolute path of the output XML file resulting from the signature of the XML file indicated in the "inFilePath" parameter.
keystorePath	Absolute path of the certificate store file where the certificate that you want to use is stored to carry out the signature process.
keystoreFormat	String of characters that indicates the format of the certificate store. The possible values of this parameter can be obtained by invoking the CosSignFileGetKeystoreFormats function (jks,pkcs12, etc.).
keystorePasswd	Password of the certificate store specified in the "keystorePath" parameter.
alias	Alias of the electronic certificate that you want to use to make the signature of the file.
aliasPasswd	Password of the electronic certificate specified in the "alias" parameter.
algorithm	Encryption algorithm that you want to use. The list of supported algorithms can be consulted by executing the CosSignFileGetSignAlgorithms function (SHA1withRSA, SHA256withRSA, etc).
format	Signature format. The list of supported signature formats can be viewed by executing the ***CosSignFileGetXAdESSignFormats*** function.
mode	Signature mode. The list of supported signature modes can be consulted by executing the ***CosSignFileGetSignModes*** function (implicit, explicit, etc).

Retorna:

0	The signature identifier passed as a parameter does not exist or there was an error in the execution of the signature function.
1	The function has been executed successfully.

2.7 CosSignFileSignCAdES

This function will sign the file passed as a parameter using the CAdES (CMS Advanced Electronic Signatures) format.

Syntax:

```
CosSignFileSignCAdES (
    signID as integer,
    inFilePath as char,
    outFilePath as char,
    keystorePath as char,
    keystoreFormat as char,
    keyStorePasswd as char,
    alias as char,
    aliasPasswd as char,
    algorithm as char,
    format as char,
    mode as char) return Integer
```

Parameters:

signID	Identifier of the signature returned in the call to the CosSignFileCreateSigner function.
inFilePath	Absolute path of the file you want to sign.
outFilePath	Absolute path of the output file resulting from the signature of the file indicated in the "inFilePath" parameter.
keystorePath	Absolute path of the certificate store file where the certificate that you want to use is stored to carry out the signature process.
keystoreFormat	String of characters that indicates the format of the certificate store. The possible values of this parameter can be obtained by invoking the CosSignFileGetKeystoreFormats function (jks,pkcs12, etc.)
keystorePasswd	Password of the certificate store specified in the "keystorePath" parameter.
alias	Alias of the electronic certificate that you want to use to make the signature of the file.
aliasPasswd	Password of the electronic certificate specified in the "alias" parameter.
algorithm	Encryption algorithm that you want to use. The list of supported algorithms can be consulted by executing the CosSignFileGetSignAlgorithms function (SHA1withRSA, SHA256withRSA, etc).
format	Signature format. The list of supported signature formats can be viewed by executing the CosSignFileGetCAdESSignFormats function.
mode	Signature mode. The list of supported signature modes can be consulted by executing the CosSignFileGetSignModes function (implicit, explicit, etc).

Returns:

0	The signature identifier passed as a parameter does not exist or there was an error in the execution of the signature function.

1	The function has been executed successfully.

2.8 CosSignFileGetKeystoreFormats

This function allows to query the list of possible types of certificate stores supported by the API.

Syntax:

```
CosSignFileGetKeystoreFormats (signID as integer) return Char
```

Parameters:

signID	Identifier of the signature returned in the call to the CosSignFileCreateSigner function.

Returns:

The list of possible types of certificate stores, separated by commas, supported by the API. If the signature identifier passed as a parameter does not exist or an error occurs in the execution of the function, an empty String will be returned.

2.9 CosSignFileGetSignAlgorithms

This function allows to query the list of possible encryption algorithms supported by the API.

Syntax:

```
CosSignFileGetSignAlgorithms (signID as integer) return Char
```

Parameters:

signID	Identifier of the signature returned in the call to the CosSignFileCreateSigner function.

Returns:

The list of possible encryption algorithms, separated by commas, supported by the API. If the signature identifier passed as a parameter does not exist or an error occurs in the execution of the function, an empty String will be returned.

2.10 CosSignFileGetPAdESSignFormats

This function allows to query the list of possible PAdES signature formats supported by the API.

Syntax:

```
CosSignFileGetPAdESSignFormats (signID as integer) return Char
```

Parameters:

signID	Identifier of the signature returned in the call to the CosSignFileCreateSigner function.

Returns:

The list of possible PAdES signature formats, separated by commas, supported by the API. If the signature identifier passed as a parameter does not exist or an error occurs in the execution of the function, an empty String will be returned.

2.11 CosSignFileGetCAdESSignFormats

This function allows to query the list of possible CAdES signature formats supported by the API.

Syntax:

```
CosSignFileGetCAdESSignFormats (signID as integer) return Char
```

Parameters:

signID	Identifier of the signature returned in the call to the CosSignFileCreateSigner function.

Returns:

The list of possible CAdES signature formats, separated by commas, supported by the API. If the signature identifier passed as a parameter does not exist or an error occurs in the execution of the function, an empty String will be returned.

2.12 CosSignFileGetXAdESSignFormats

This function allows to query the list of possible XAdES signature formats supported by the API.

Syntax:

```
CosSignFileGetXAdESSignFormats (signID as integer) return Char
```

Parameters:

signID	Identifier of the signature returned in the call to the CosSignFileCreateSigner function.

Returns:

The list of possible XAdES signature formats, separated by commas, supported by the API. If the signature identifier passed as a parameter does not exist or an error occurs in the execution of the function, an empty String will be returned.

2.13 CosSignFileGetSignModes

This function allows to query for the list of possible signature modes supported by the API.

Syntax:

```
CosSignFileGetSignModes (signID as integer) return Char
```

Parameters:

signID	Identifier of the signature returned in the call to the CosSignFileCreateSigner function.

Returns:

The list of possible signature modes, separated by commas, supported by the API. If the signature identifier passed as a parameter does not exist or an error occurs in the execution of the function, an empty String will be returned.

3. Log file

The CosSignFile.dll DLL allows the creation of an operations log in Log4Java format by defining the log4j.properties file in the directory where the CosSignFile.dll file is located.

Example:

```
# Root logger option
log4j.rootLogger=INFO, stdout, file

# Redirect log messages to console
log4j.appender.stdout=org.apache.log4j.ConsoleAppender
log4j.appender.stdout.Target=System.out
log4j.appender.stdout.layout=org.apache.log4j.PatternLayout
log4j.appender.stdout.layout.ConversionPattern=%d{yyyy-MM-dd HH:mm:ss} %-5p
%c{1}:%L - %m%n

# Redirect log messages to a log file, support file rolling.
log4j.appender.file=org.apache.log4j.RollingFileAppender
log4j.appender.file.File=c:/tmp/log4j-cosmoSigner.log
log4j.appender.file.MaxFileSize=5MB
log4j.appender.file.MaxBackupIndex=10
log4j.appender.file.layout=org.apache.log4j.PatternLayout
log4j.appender.file.layout.ConversionPattern=%d{yyyy-MM-dd HH:mm:ss} %-5p
%c{1}:%L - %m%n
```

In the attribute "log4j.rootLogger" the level of log is indicated. The possible values are ALL, DEBUG, ERROR, FATAL, INFO, OFF, TRACE and WARN.

The attribute "log4j.appeder.file.File" will indicate the path of the log file.

For more information, consult the Log4j documentation at the following address: https://logging.apache.org/log4j

4. Considerations about file paths

The CosSignFile.dll DLL makes use of a series of classes programmed in the Java language to perform the process of signing XML, PDF and binary files. For this reason, the paths of the signature files, XML files, PDF files and binary files must have the "/" character as the directory separator, instead of the Microsoft Windows "\".

5. Examples

5.1 Definition of functions

```
public dll "cossignfile.dll" CosSignFileCreateSigner() return integer
public dll "cossignfile.dll" CosSignFileFreeSigner(signer as integer)
public dll "cossignfile.dll" CosSignFileSignFacturaE(
    signer as integer,
    xmlPath as char,
    xsigPath as char,
    keystorePath as char,
    keystoreFormat as char,
    keyStorePasswd as char,
    alias as char,
    aliasPasswd as char) return integer

public dll "cossignfile.dll" CosSignFileSignPDF(
    signer as integer,
    inPDFPath as char,
    outPDFPath as char,
    keystorePath as char,
    keystoreFormat as char,
    keyStorePasswd as char,
    alias as char,
    aliasPasswd as char) return integer

public dll "cossignfile.dll" CosSignFileSignPAdES(
    signer as integer,
    inFilePath as char,
    outFilePath as char,
    keystorePath as char,
    keystoreFormat as char,
    keyStorePasswd as char,
    alias as char,
    aliasPasswd as char,
    algorithm as char,
    format as char,
    mode as char) return integer

public dll "cossignfile.dll" CosSignFileSignXAdES(
    signer as integer,
    inFilePath as char,
    outFilePath as char,
    keystorePath as char,
    keystoreFormat as char,
    keyStorePasswd as char,
    alias as char,
    aliasPasswd as char,
    algorithm as char,
    format as char,
    mode as char) return integer

public dll "cossignfile.dll" CosSignFileSignCAdES(
    signer as integer,
    inFilePath as char,
    outFilePath as char,
    keystorePath as char,
    keystoreFormat as char,
    keyStorePasswd as char,
    alias as char,
    aliasPasswd as char,
    algorithm as char,
    format as char,
    mode as char) return integer

public dll "cossignfile.dll"  CosSignFileGetKeystoreFormats(signer as integer) return
char
public dll "cossignfile.dll"  CosSignFileGetSignAlgorithms(signer as integer) return
char
```

```
public dll "cossignfile.dll"  CosSignFileGetPAdESSignFormats(signer as integer) return
char
public dll "cossignfile.dll"  CosSignFileGetCAdESSignFormats(signer as integer) return
char
public dll "cossignfile.dll"  CosSignFileGetXAdESSignFormats(signer as integer) return
char
public dll "cossignfile.dll"  CosSignFileGetSignModes(signer as integer) return char
```

5.2 Signing of an XML file in FacturaE format

```
private function firmaFacturaE_P12
objects begin
    signer as integer
    xmlPath as char
    xsigPath as char
    p12Path as char
    keyStorePasswd as char
    alias as char
    aliasPasswd as char
end
begin

    xmlPath = ProjectDir() + "\ficheros\factura_para_firmar.xml";
    xsigPath = ProjectDir() + "\ficheros\out\factura_firmada.xsig";
    p12Path = ProjectDir() + "\ficheros\p12depruebas.p12";

    xmlPath.Replace("\", "/");
    xsigPath.Replace("\", "/");
    p12Path.Replace("\", "/");

    keyStorePasswd = "contraseña";
    alias = "Pepito López";
    aliasPasswd = "contraseña";

    signer = CosSignFileCreateSigner();
    CosSignFileSignFacturaE(signer, xmlPath, xsigPath , p12Path, "pkcs12",
keyStorePasswd, alias, aliasPasswd);
    CosSignFileFreeSigner(signer);

end
```

5.3 Signing a PDF file

```
private function firmaPDF_P12
objects begin
    signer as integer
    inPDFPath as char
    outPDFPath as char
    p12Path as char
    keyStorePasswd as char
    alias as char
    aliasPasswd as char
end
begin

    inPDFPath = ProjectDir() + "\ficheros\Windows_Server_2003_Migration_Datasheet.pdf";
    outPDFPath = ProjectDir() +
"\ficheros\out\Windows_Server_2003_Migration_Datasheet.firmado.pdf";
    p12Path = ProjectDir() + "\ficheros\p12depruebas.p12";

    inPDFPath.Replace("\", "/");
    outPDFPath.Replace("\", "/");
    p12Path.Replace("\", "/");

    keyStorePasswd = "contraseña";
    alias = "Pepito López";
    aliasPasswd = "contraseña";
```

```
    signer = CosSignFileCreateSigner();
    CosSignFileSignPDF(signer, inPDFPath, outPDFPath , p12Path, "pkcs12",
keyStorePasswd, alias, aliasPasswd);
    CosSignFileFreeSigner(signer);

end
```

5.4 Signing of a PDF file in PAdES format

```
private function firmaPAdES_P12
objects begin
    signer as integer
    inFilePath as char
    outFilePath as char
    p12Path as char
    keyStorePasswd as char
    alias as char
    aliasPasswd as char
    algorithm as char
    format as char
    mode as char
end
begin

    inFilePath = ProjectDir() + "\ficheros\Windows_Server_2003_Migration_Datasheet.pdf";
    outFilePath = ProjectDir() +
"\ficheros\out\Windows_Server_2003_Migration_Datasheet_PAdES.pdf";
    p12Path = ProjectDir() + "\ficheros\p12depruebas.p12";

    inFilePath.Replace("\", "/");
    outFilePath.Replace("\", "/");
    p12Path.Replace("\", "/");

    keyStorePasswd = "contraseña";
    alias = "Pepito López";
    aliasPasswd = "contraseña";

    format = "PAdES";
    mode = "implicit";
    algorithm = "SHA512withRSA";

    signer = CosSignFileCreateSigner();
    CosSignFileSignPAdES(signer, inFilePath, outFilePath , p12Path, "pkcs12",
keyStorePasswd, alias, aliasPasswd, algorithm, format, mode);
    CosSignFileFreeSigner(signer);

end
```

5.5 Signing of an XML file in XAdES format

```
private function firmaXAdES_P12
objects begin
    signer as integer
    inFilePath as char
    outFilePath as char
    p12Path as char
    keyStorePasswd as char
    alias as char
    aliasPasswd as char
    algorithm as char
    format as char
    mode as char
end
begin

    inFilePath = ProjectDir() + "\ficheros\factura.xml";
    outFilePath = ProjectDir() + "\ficheros\out\factura_salida_XAdES.xsig";
```

```
    p12Path = ProjectDir() + "\ficheros\p12depruebas.p12";

    inFilePath.Replace("\", "/");
    outFilePath.Replace("\", "/");
    p12Path.Replace("\", "/");

    keyStorePasswd = "contraseña";
    alias = "Pepito López";
    aliasPasswd = "contraseña";

    format = "XAdES Detached";
    mode = "implicit";
    algorithm = "SHA512withRSA";

    signer = CosSignFileCreateSigner();
    CosSignFileSignXAdES(signer, inFilePath, outFilePath , p12Path, "pkcs12",
keyStorePasswd, alias, aliasPasswd, algorithm, format, mode);
    CosSignFileFreeSigner(signer);

end
```

5.6 Signing of an XML file in CAdES format

```
private function firmaCAdES_P12
objects begin
    signer as integer
    inFilePath as char
    outFilePath as char
    p12Path as char
    keyStorePasswd as char
    alias as char
    aliasPasswd as char
    algorithm as char
    format as char
    mode as char
end
begin

    inFilePath = ProjectDir() + "\ficheros\factura.xml";
    outFilePath = ProjectDir() + "\ficheros\out\factura_CAdES.xml";
    p12Path = ProjectDir() + "\ficheros\p12depruebas.p12";

    inFilePath.Replace("\", "/");
    outFilePath.Replace("\", "/");
    p12Path.Replace("\", "/");

    keyStorePasswd = "contraseña";
    alias = "Pepito López";
    aliasPasswd = "contraseña";

    format = "XAdES Detached";
    mode = "implicit";
    algorithm = "SHA512withRSA";

    signer = CosSignFileCreateSigner();
    CosSignFileSignCAdES(signer, inFilePath, outFilePath , p12Path, "pkcs12",
keyStorePasswd, alias, aliasPasswd, algorithm, format, mode);
    CosSignFileFreeSigner(signer);

end
```