CONTENIDO

Netco Signer

¿Qué es?

La aplicación Netco Signer es una herramienta de software que permite a una organización firmar documentos usando Firma Digital y/o Firma electrónica. (Si desea más información de la solución consulte el manual de usuario Netco Signer).

¿A quiénes está dirigido este manual?

Este manual técnico está dirigido a cualquier usuario que desee integrarse con una aplicación externa a Netco Signer mediante webservices. El webservice aquí descrito permite integrar aplicaciones en su organización para firmar documentos automáticamente

Consideraciones Previas

La ubicación donde se encuentra publicado el servicio web (WSDL) es la siguiente: (cambiar NOMBRESERVIDOR:PUERTO/APLICACIÓN por el nombre de la instancia proporcionada por Netco):

https://**NOMBRESERVIDOR:PUERTO/APLICACIÓN**/services/Service?wsdl

Ejemplo:

https://demos.netco.la/otpSignatureDemos/services/SOAPService?wsdl

En caso de generarse un error en el llamado a cualquier método del WebService, se genera una excepción genérica.

El WebService Netco Signer contiene métodos para firmar y/o gestionar la solución de Firma.

Métodos

generateOTP

```
Integer authType, String message, String mailOrPhone, Integer otpTTL,
String userName, String userPassword
```

Este método genera un código OTP. Recibe el medio por el cual se envía el OTP (1. SMS, 2.Llamada, 3.Emai y 4. Whatsapp), el mensaje que se añade al enviar el OTP, el teléfono (con indicativo) o el email, el TTL (tiempo de vida), nombre de usuario y contraseña. Se espera un uid.

Este servicio retornará un uid el cual se debe tener en cuenta para el siguiente servicio. Se podrá observar en el formato xml

signFileOTP

```
String uid, String otp, String base64File, String fileName, String
passwordPDF, String idTemplate, String userName, String userPassword,
String operation.
```

Este método recibe el uid generado por generateOTP, el código OTP, el archivo que se desea firmar en base64, el nombre del archivo (solo es necesario si el archivo tiene contraseña), contraseña de archivo pdf, id de la plantilla, nombre de usuario y contraseña. El método retorna el archivo firmado en base64 junto con la plantilla.

Este el servicio proporcionará el archivo ya firmado en formato Base64.

Caso de uso para el WebServices Firma con evidencia electrónica

Para el método generateOTP :

- `authType:` *Este parámetro indica porque medio se va a recibir el OTP 1. SMS, 2. Llamada, 3.Email y 4. WhatsApp)*

  Ej:
  <ser:authType>3</ser:authType> <!--Para este ejemplo se está indicando que se desea recibir el OPT por email→

- *message:* Este parámetro indica como se va a recibir el OTP y el mensaje de este.
  Ej:
  <ser:message>Este es tu OTP: {otp}</ser:message>

- *mailOrPhone: En este parámetro es necesario poner el email para donde se recibirá el OTP*
  Ej:
  <ser:mailOrPhone>ejemplo@tudominio.com</ser:mailOrPhone>

- *otpTTL: Este parámetro indica la duración del OTP*
  Ej:
  <ser:otpTTL>60</ser:otpTTL> <!--Para este ejemplo se está indicando que se desea establecer un tiempo de 60 minutos de duración→

- *userName: En este parámetro es necesario poner el usuario con el cual se va a firmar o realizar las pruebas correspondientes ejemplouser*
  Ej:
  <ser:userName>ejemplouser</ser:userName>

- *userPassword: En este parámetro se le indica la contraseña en texto plano*
  Ej:
  <ser:userName>tucontraseña123</ser:userName>

Una vez se completen correctamente todos los campos y la ejecución finalice sin problemas, el OTP deberá ser enviado al método seleccionado. En este caso, se optó por utilizar el correo electrónico. Otro parámetro importante a tener en cuenta es el "uid", el cual se mostrará en el formato XML una vez que la ejecución haya culminado exitosamente.

Para el método SingFileOTP :

Al utilizar este método, es importante tener en cuenta tres parámetros:

1. El primer parámetro clave es el "uid" generado por el método "generateOTP".
2. El segundo parámetro es el OTP que se envía al medio seleccionado.
3. El tercer parámetro es el archivo al cual se desea agregar la firma con evidencia electrónica.

A continuación se explican cada uno de los parámetros

- `uid:` Este parámetro es el identificador único de la transacción
  Ej:
  <ser:uid>193fcc7a-bcf3-47fe-a3a3-fce4abe8719e</ser:uid>

- *otp:* Para este caso es necesario entregarle el OTP que llegó al metodo seleccionado
  Ej:
  <ser:otp>13</ser:otp>

- *base64File:* Para este parámetro es necesario poner el archivo en base 64, en SoapUI se encarga de asignarle un cid, con el que posteriormente va a identificar el archivo y convertirlo en base 64
  Ej:
  <ser:base64File>cid:1441761843624</ser:base64File>

- *fileName: En este parámetro se usa para identificar el nombre del PDF a firmar*
  Ej:
  <ser:fileName>tuarchivo.pdf</ser:fileName>

- *passwordPDF: En este capo se va a ingresar la contraseña de ser necesario si el archivo esta protegido.*
  Ej:
  <ser:passwordPDF>tucontraseña123</ser:passwordPDF>

- *idTemplate163: En este capo se va a ingresar la plantilla designada para este proceso, la cual tiene como id 267*
  Ej:
  <ser:idTemplate>267</ser:idTemplate>

- *userName: En este parámetro es necesario poner el usuario con el cual se va a firmar o realizar las pruebas correspondientes ejemplouser*
  Ej:
  <ser:userName>ejemplouser</ser:userName>

- *userPassword: En este parámetro se le indica la contraseña en texto plano*
  Ej:
  <ser:userName>tucontraseña123</ser:userName>

- *ip: Si es necesario poner la ip del firmante se indicará en este campo*
  Ej:
  <ser:ip>8.8.8.8</ser:ip>

- *operation: Este parámetro indica el tipo de operación de firma que se va a realizar (2:Firma electrónica, 1:Firma digital, 3:Firma digital más sello de tiempo, 4:Firma digitalizada)*
  Ej:
  <ser:operation>1</ser:operation> #Para este ejemplo se usaría firma digital

Una vez que la ejecución se haya completado correctamente, el servicio proporcionará el archivo ya firmado en formato Base64. Para poder visualizarlo es necesario decodificarlo. Para realizar pruebas se recomienda utilizar la siguiente página web que realiza la decodificación y así asegurarse de que el archivo esté correcto: https://base64.guru/converter/decode/pdf

Código de ejemplo consumo WebServices en Java

```
public static void main(String[] args) throws Exception
{
    String filePath="c:\\";
    String fileName="test.pdf";
    String signedFileName="signed.pdf";
    String userName="test";
    String password="Apple2000!";
    String webServicePassword="webpassword";
    String encoding="UTF-8";
    String policyName="clientPolicy.xml";
    String certificateAlias="privatekeyaliasinp12";
    String p12Path="pathtop12withtheprivatekey";
    String endpoint="https://localhost:8443/netcosigner/services/Service.ServiceHttpsSoap11Endpoint/";
    int timeOutInMilliSeconds=20000;
    int idOperation=1;  //1 Firma, 2 Sello de Tiempo, 3 Firma con sello de tiempo
    ConfigurationContext ctx;
    ctx = ConfigurationContextFactory.createConfigurationContextFromFileSystem(null, null);
    ServiceStub service=new ServiceStub(ctx,endpoint);
    ServiceClient client=service._getServiceClient();
    Policy policy = loadPolicy(policyName);
    Properties merlinProp = new Properties();
    merlinProp.put("org.apache.ws.security.crypto.merlin.keystore.type", "PKCS12");
    merlinProp.put("org.apache.ws.security.crypto.merlin.file",p12Path);
    merlinProp.put("org.apache.ws.security.crypto.merlin.keystore.password", password); //Clave del p12
    merlinProp.put("org.apache.ws.security.crypto.merlin.load.cacerts", "true");
    CryptoConfig sigCryptoConfig = new CryptoConfig();
    sigCryptoConfig.setProvider(Merlin.class.getName());
    sigCryptoConfig.setProp(merlinProp);
    RampartConfig rampartConfig = new RampartConfig();
    rampartConfig.setUserCertAlias(certificateAlias);
    rampartConfig.setPwCbClass(NetcoHandler.class.getName());//Lógica para obtener la clave privada del alias
    rampartConfig.setSigCryptoConfig(sigCryptoConfig);
    policy.addAssertion(rampartConfig);
    Options options = client.getOptions();
    options.setProperty(HTTPConstants.SO_TIMEOUT, new Integer(timeOutInMilliSeconds));
    options.setProperty(HTTPConstants.CONNECTION_TIMEOUT, new Integer(timeOutInMilliSeconds));
    options.setProperty(RampartMessageData.KEY_RAMPART_POLICY,  policy);
```

```
    client.setOptions(options);

    //Enable WSecurity to sign messages
    if(endpoint.toLowerCase().contains("soap12"))
        client.engageModule("rampart");

    SignFile signFile22=new SignFile();
    byte []fileBytes=Files.readAllBytes(new File(filePath.concat(fileName)).toPath());
    signFile22.setFileBytes(getDataHandler(fileBytes,encoding));
    signFile22.setFileName(fileName);
    signFile22.setOperationId(idOperation);
    signFile22.setUserName(userName);
    signFile22.setUserPass(password);
    signFile22.setWebServicePassword(webServicePassword);
    SignFileResponse signed = service.signFile(signFile22);
    byte []signedFile=getByteArray(signed.get_return());
    FileUtils.writeByteArrayToFile(new File(filePath.concat(signedFileName)), signedFile);
}
```

Con el objeto service se puede acceder a cualquiera de los métodos mencionados anteriormente como se ve en la gráfica:

```
service.
_getServiceClient() : ServiceClient - Stub
_setServiceClient(ServiceClient _serviceClient) : void - Stub
checkStrongAuthentication(CheckStrongAuthentication
cleanup() : void - Stub
createUser(CreateUser createUser19) : void - ServiceStub
deleteUser(DeleteUser deleteUser15) : void - ServiceStub
deleteUserCredentials(DeleteUserCredentials deleteUserCr
equals(Object arg0) : boolean - Object
```

Se adjuntan los archivos fuentes de las clases requeridas para el llamado:

netco.zip

Adicionalmente, si se cuenta con una herramienta de generación de código usando el WSDL se puede acceder conociendo el endpoint del webservice.

Nota: Para consumir el webservice usando el endpoint soap12 (el endpoint soap11 no lo requiere) se requiere una identidad digital (p12) válida y el siguiente certificado:

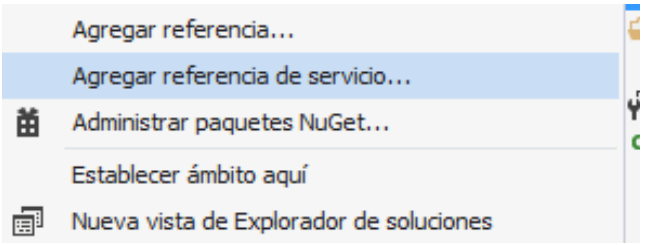
wsnetcosigner.cer

El archivo wsnetcosigner.cer debe importarse al keystore cacerts de la versión de java que use la aplicación. (Ver manual herramienta keytool de java para realizar el procedimiento).

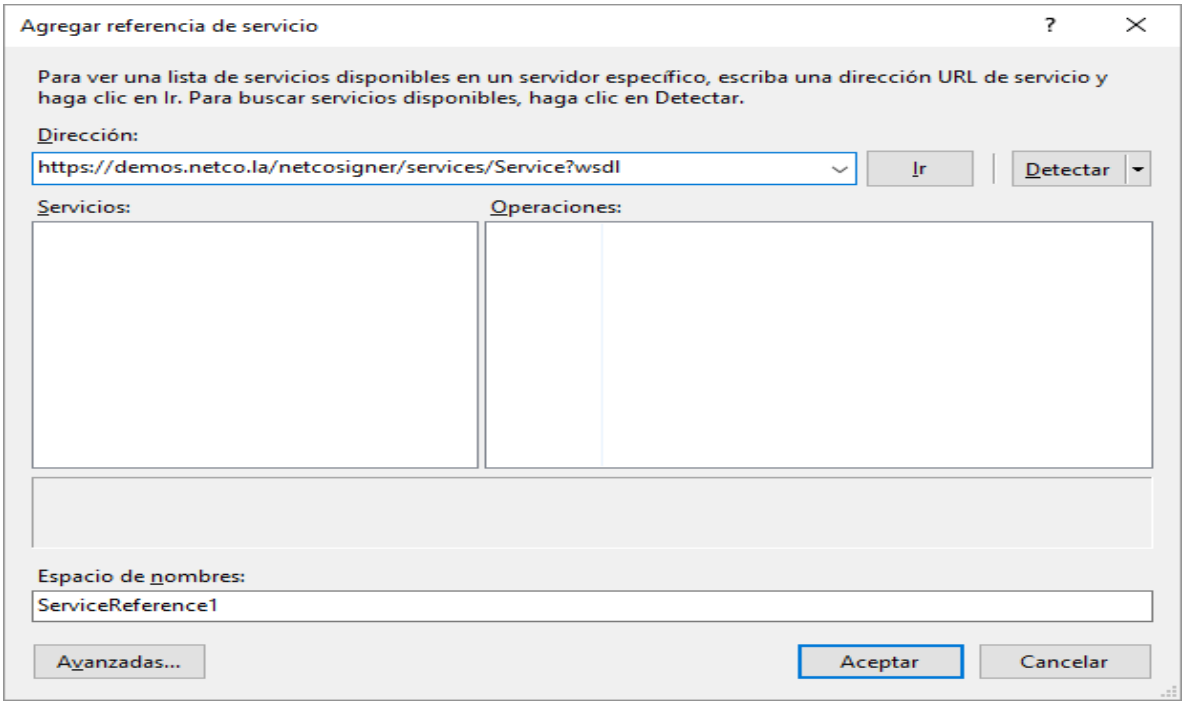
Código de ejemplo consumo WebServices en .NET

En .net se debe agregar la referencia al web service de la siguiente forma:

1. Clic derecho en el proyecto, opción Agregar referencia de servicio:



2. Copiar la ruta del wsdl proporcionado por Netco:

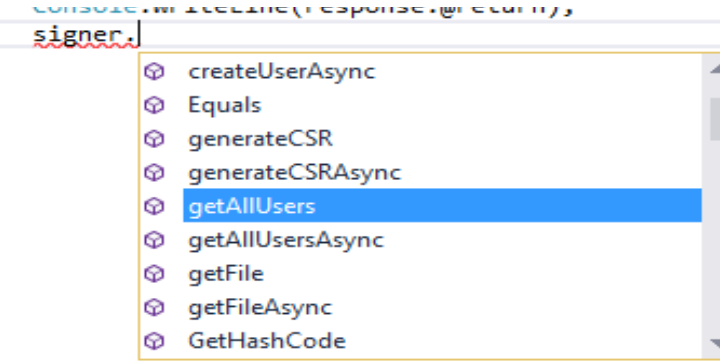


Hacer click en Ir, darle un nombre a la referencia y hace click en aceptar.

El siguiente código usa como nombre de la referencia “NetcoSigner”

```
static void Main(string[] args)
{
    String bindingName = "ServiceHttpsSoap12Endpoint";
    NetcoSigner.ServicePortTypeClient signer = new NetcoSigner.ServicePortTypeClient(bindingName);
    if (bindingName.ToLower().Contains("soap12"))
    {
        String p12Path = @"G:\DRGD\Keys\8080808080.p12";
        String p12Password = "Apple2000!";
        String serverCertPath = @"G:\DRGD\Keys\localhost.cer";
        X509Certificate2 clientCertificate = new X509Certificate2(p12Path, p12Password);
        signer.ClientCredentials.ClientCertificate.Certificate = clientCertificate;
        X509Certificate2 serviceCertificate = new X509Certificate2(serverCertPath);
        signer.ClientCredentials.ServiceCertificate.DefaultCertificate = serviceCertificate;
    }
    //Sign File
    String filePath = "c://";
    String fileName = "test.pdf";
    String signedFileName = "testfirmado.pdf";
    String userName = "diego";
    String userPassword = "Apple2000!";
    String webServicePassword = "Apple2000!";
    int idOperation = 1; //1 Firma, 2 Timestamp, 3 Firma y Timestamp
    byte[] fileBytes = File.ReadAllBytes(filePath + fileName);
    byte[] signedBytes = signer.signFile(fileBytes, fileName, idOperation, userName, userPassword, "", "", webServicePassword);
    File.WriteAllBytes(filePath + signedFileName, signedBytes);
}
```

Con el objeto signer se puede acceder a cualquiera de los métodos mencionados anteriormente como se ve en la gráfica:



Nota: Para consumir el webservice usando el endpoint SOAP12 es requerido una identidad digital (p12) y el siguiente certificado del servidor para firmar cada consumo:

wsnetcosigner.cer

Código de ejemplo consumo WebServices en .PHP

```php
<?php
class MySoapClient extends SoapClient
{
    public function __doRequest($request, $location, $action, $version, $one_way = 0)
    {
        $response = parent::__doRequest($request, $location, $action, $version, $one_way);
        $start=strpos($response,'<?xml');
        $end=strrpos($response,'>');
        $response_string=substr($response,$start,$end-$start+1);
        return($response_string);
    }
}
    $cn='servidor.dominio.com';
    $wsdl = 'https://' . $cn . ':8443/netcosigner/services/Service?wsdl';
    $wspass='WSPASS';
    $adminUser='adminuser';
    $adminPass='ADMINPASS';
    $method='getAllUsers';
    ini_set('soap.wsdl_cache_enabled', 0);
    $contextOptions = array
    (
        'ssl' => array
        (
            'verify_peer'       => true,
            'verify_peer_name'  => true,
            'allow_self_signed' => true,
            'CN_match' => $cn
        )
    );
    $sslContext = stream_context_create($contextOptions);
    $client = new MySoapClient($wsdl,array('stream_context' => $sslContext));
    $respuesta = $client->__soapCall($method, array
    (
        $method => array(
            "adminUserName"     => $adminUser,
            "adminUserPassword"     => $adminPass,
            "webServicePassword"     => $wspass
        )
    )
    );
    print_r($respuesta->return);
```

Nota: No se provee un ejemplo de autenticación usando WS-Security en PHP.