

საქართველოს იუსტიციის სამინისტრო

მონაცემთა გაცვლის  
სააგენტო



# სერვისების გამოყენების მოკლე ინსტრუქცია

მინი სახელმძღვანელო – „მონაცემთა გაცვლის სააგენტოს“  
ინფრასტრუქტურის თვარგლებში არსებული სერვისების გამოყენების მოკლე  
ინსტრუქცია

იუსტიციის სამინისტროს მონაცემთა გაცვლის სააგენტო

2013

# სერვისების გამოყენების მოკლე ინსტრუქცია

მინი სახელმძღვანელო – „მონაცემთა გაცვლის სააგენტოს“ ინფრასტრუქტურის ფარგლებში არსებული სერვისების გამოყენების მოკლე ინსტრუქცია

- სინქრონული სერვისების მოთხოვნის ფორმირება
- ასინქრონული სერვისების მოთხოვნის ფორმირება
- სერთიფიკატით აუთენტიფიცირებული სერვისების გამოყენება
- ხელმოწერის კოდი .NET გარემოში
- მეთოდი რითაც დაიპოსტება გენერირებული XML-ი და დაბრუნდება პასუხი
- ხელმოწერის კოდი Java გარემოში

## სინქრონული სერვისების მოთხოვნის ფორმირება

სინქრონული სერვისების ფორმირების დროს აუცილებელია ვიცოდეთ:

- 1) რომელ სააგენტოს მივმართავთ
- 2) კლასის სახელი
- 3) <Message> ტეგის შიგნით სააგენტოს მიერ ჩამოყალიბებული XML ველების მნიშვნელობა და შევსების თავისებურებები

➤ მაგალითში წარმოდგენილია CRA -ს სერვისის გამოძახების გოვტოქი

```
<?xml version="1.0" encoding="utf-8"?>
<GovTalkMessage xmlns="http://www.govtalk.gov.uk/CM/envelope">
  <EnvelopeVersion>2.0</EnvelopeVersion>
  <Header>
    <MessageDetails>
      <Class>CRA_GetPersonInfo</Class>
      <Qualifier>request</Qualifier>
      <Function>submit</Function>
      <CorrelationID></CorrelationID>
    </MessageDetails>
```

```

<SenderDetails>
  <IDAuthentication>
    <SenderID>DEA</SenderID>
    <Authentication>
      <Method>clear</Method>
      <Value>*****</Value>
    </Authentication>
  </IDAuthentication>
</SenderDetails>
</Header>
<Body>
  <Message xmlns="urn:g3.ge:envelope:request:v1">
    <Header>
      <Signature />
      <Vendor version="700" productName="SAP R/3" />
    </Header>
    <Data gzip="no" encrypted="no">
      <Request xmlns="urn:g3.ge:cra:call:GetDataUsingPrivateNumber:v1">
        <Parameters>
          <PrivateNumber>01010101011</PrivateNumber>
        </Parameters>
      </Request>
    </Data>
  </Message>
</Body>
</GovTalkMessage>

```

#### ავტორიზაცია:

მიითითებულ ტეგებში შეავსეთ უშუალოდ თქვენი ორგანიზაციისთვის გამოყოფილი ექსენთის მონაცემები:

```

<IDAuthentication>
  <SenderID>>მომხმარებლის სახელი</SenderID>
  <Authentication>
    <Method>clear</Method>
    <Value>პაროლი</Value>
  </Authentication>

```

- 1 : <Class>CRA\_GetPersonInfo</Class> – მოცემულ ტეგში მიითითებულია იმ სერვისის სახელი რასაც ვიძახებთ, ამ შემთხვევაში სამაგალითოდ გამოყენებულია სერვისების განვითარების სააგენტოს ერთ-ერთი სერვისის სახელი.
- 2 : <Message> – ტეგის შიგნით არის სააგენტოს მიერ განსაზღვრული გადასაცემი პარამეტრები, რომელიც წარმოდგენილია სტრუქტურირებული სახით, სწორედ ამ მონაცემების საფუძველზე ხდება პასუხის დაბრუნება.
- 3 : <PrivateNumber>01010101011</PrivateNumber> – მოცემულ ველში ვუთითებთ იმ პირად ნომერს, რაზეც გვსურს ინფორმაციის მიღება.

## ასინქრონული სერვისების მოთხოვნის ფორმირება

ასინქრონული სერვისების გამოძახება ანუ Request XML-ის ფორმირება არაფრით არ განსხვავდება სინქრონულისგან (მაგალითი იხილეთ 1-2 გვერძე). განსხვავება მოლოდ პასუხის მიღების დროს და მის ტექნიკაშია. სინქრონული სერვისის შემთხვევაში გამოძახების შემდგომ მყისიერად ხდება პასუხის დაბრუნება(response), ხოლო ასინქრონულის სერვისის დროს მოთხოვნაზე რეალური პასუხის მაგივრად ბრუნდება ე.წ. “acknowledgement”, პასუხის მისაღებად აუცილებელია დაბრუნებული XML-იდან მიღებული „CorrelationID“-ით ვცადოთ პასუხის მიღება, ამისათვის უკვე საჭიროა მოცემულ რომელიმე URL-ზე ე.წ. „poll“ მოთხოვნის გაგზავნა.

- 1) <https://test.submission.e-government.ge/poll> - სატესტო გარემო
- 2) <https://submission.e-government.ge/poll> - რეალური გარემო

„poll“ მოთხოვნის გაგზავნის შემდგომ თუ დაბრუნდა ისევე ე.წ. “acknowledgement”, ე.ი. პასუხი ჯერ არ არის მზად, ხოლო თუ დაბრუნდა „response“ ეს ნიშნავს რომ რეალური პასუხი სერვისის მომწოდებლისგან მიღებულია.

- ქვემოთ ნაჩვენებია ასინქრონულ სერვისზე მიმართვის შემდგომ დაბრუნებული ე.წ. “acknowledgement”

```
<?xml version="1.0" encoding="utf-8" ?>
<GovTalkMessage xmlns="http://www.govtalk.gov.uk/CM/envelope">
  <EnvelopeVersion>2.0</EnvelopeVersion>
  <Header>
    <MessageDetails>
      <Class>RequestAsync</Class>
      <Qualifier>acknowledgement</Qualifier>
      <Function>submit</Function>
      <TransactionID />
      <CorrelationID>7AEE60FF723247D487B013100F0C6163</CorrelationID>
      <ResponseEndPoint PollInterval="60">https://test.submission.e-
government.ge/poll</ResponseEndPoint>
      <GatewayTimestamp>2015-05-14T13:23:12.869</GatewayTimestamp>
    </MessageDetails>
    <SenderDetails />
  </Header>
  <GovTalkDetails>
    <Keys />
```

```

</GovTalkDetails>
<Body>
  <Signature xmlns="http://www.w3.org/2000/09/xmldsig#" Id="Acknowledgement">
    <SignedInfo>
      <CanonicalizationMethod Algorithm="http://www.w3.org/TR/2001/REC-xml-
c14n-20010315" />
      <SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-
sha256" />
      <Reference URI="#gg.properties"
Type="http://www.w3.org/2000/02/xmldsig#SignatureProperty">
        <Transforms>
          <Transform Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-
signature" />
        </Transforms>
        <DigestMethod Algorithm="http://www.w3.org/2001/04/xmldsig#sha256" />
<DigestValue>46UtWvB3hLr6obY7jc1NEdbA4rtPhwdUQGKG1+mAUSQ=</DigestValue>
      </Reference>
    </SignedInfo>

<SignatureValue>NfioNdeZr8GTtPlG5s75RSiZ7f018Z1PJPlY7cbN2Fw==</SignatureValue>
  <Object>
    <SignatureProperties Id="gg.properties">
      <SignatureProperty Target="#Acknowledgement">
        <TimeStamp xmlns="urn:gg:ts.v2">2015-05-14 01:23:12</TimeStamp>
      </SignatureProperty>
      <SignatureProperty Target="#Acknowledgement">
        <CorrelationID
xmlns="urn:gg:ts.v2">7AEE60FF723247D487B013600F0C6163</CorrelationID>
      </SignatureProperty>
    </SignatureProperties>
  </Object>
<Object>
  <SignatureProperties Id="gg.x509">
    <SignatureProperty Target="#Acknowledgement">
      <SignerCertificate
xmlns="urn:gg:ts.v2">MIIIFmDCCBICgAwIBAgqsUASd/KqhQb89Cg=</SignerCertificate>
    </SignatureProperty>
  </SignatureProperties>
</Object>
</Signature>
</Body>
</GovTalkMessage>

```

- 1: RequestAsync - სერვისის უნიკალური სახელი
- 2: 7AEE60FF723247D487B013100F0C6163- დაბრუნებული "CorrelationID" (უნიკალური ნომერი) რისი გამოყენებითაც შემდგომ ვითხოვთ სერვისის მომწოდებლისგან პასუხს.
- 3: PollInterval="60"- მოცემული მნიშვნელობა- „60“ განსაზღვრავს თუ რამდენი წამის მერე უნდა მოვახდინოთ ხელახალი მიმართვა (გაკეთდეს „poll“ მოთხოვნა) თუ პასუხი არ დაბრუნდა.
- 4: https://test.submission.e-government.ge/poll - URL-ი რომელზეც უნდა ვაიგზავნოს „poll“ მესიჯი პასუხის დასაბრუნებლად.

შენიშვნა: ასინქრონულ/სინქრონულ სერვისზე მიმართვისთვის ( მოთხოვნის დასაპოსტად ) იხილეთ 9 გვერდი

- ქვემოთ ნაჩვენებია ასინქრონული სერვისის გამოყენებისას, ე.წ. „poll“ მესიჯის ფორმირების მაგალითი

```
<?xml version="1.0" encoding="utf-8"?>
<GovTalkMessage xmlns="http://www.govtalk.gov.uk/CM/envelope">
  <EnvelopeVersion>2.0</EnvelopeVersion>
  <Header>
    <MessageDetails>
      <Class>RequestAsync</Class>
      <Qualifier>poll</Qualifier>
      <Function>submit</Function>
      <CorrelationID>7AEE60FF723247D487B013100F0C6163</CorrelationID>
    </MessageDetails>
    <SenderDetails>
      <IDAuthentication>
        <SenderID/>
        <Authentication>
          <Method>clear</Method>
          <Value/>
        </Authentication>
      </IDAuthentication>
    </SenderDetails>
  </Header>
  <Body/>
</GovTalkMessage>
```

- 1: RequestAsync - სერვისის უნიკალური სახელი
- 2: 7AEE60FF723247D487B013100F0C6163 – ე.წ. “acknowledgement”-დან დაბრუნებული “CorrelationID” (უნიკალური ნომერი) რის მიხედვითაც ვითხოვთ სერვისის მომწოდებლისგან პასუხს.

## სერთიფიკატით აუთენტიფიცირებული სერვისების გამოყენება

მოცემული სერთიფიკატის გამოყენება არ არის სავალდებულო თუ სააგენტო არ ითხოვს სერთიფიკატით ხელმოწერილ "Request"-ს, სერთიფიკატის გამოყენებით სააგენტო აიდენტიფიცირებს მომხმარებელს უშუალოდ სააგენტოსთან მიმართვის დროს. გოვთოქში არსებული <Message> ტეგის შიგნით არსებულ მონაცემებზე სერთიფიკატით ხელმოწერის დროს აუცილებელია ვიცოდეთ რომ:

- 1) სერთიფიკატს გასცემს უშუალოდ სერვისის მომწოდებელი
- 2) ხელმოწერამდე Data ტაგის შემდეგ უნდა ჩაამატოთ Thumbprint ელემენტი რომლის მნიშვნელობაც იქნება იმ სერთიფიკატის Thumbprint რომლითაც უნდა განიხორციელდეს ხელმოწერა.
- 3) გაგზავნოთ (დაპოსტოთ) მთლიანი მიღებული ( სერთიფიკატით ხელმოწერილი) XML-ი

### გაითვალისწინეთ:

არსებობს სირთულეები, როდესაც სერვისის მომხმარებელი მოთხოვნის ფორმირების დროს ხელმოწერას PHP გარემოში გადაწყვეტს, ამიტომ გთავაზობთ ალტერნატივას, შექმნათ აპლიკაცია ან სერვისი (C# ან Java გარემოში) და ისე მოახდინოთ მოთხოვნაზე ხელმოწერა. ხელმოწერის კოდი იხ. (გვ.7 - გვ.10)

- ქვემოთ ნაჩვენებია სერთიფიკატით ხელმოწერილი Request გოვთოქის-ის <Message> -ტეგის შიგთავსი.

```
<Message xmlns="urn:g3.ge:envelope:request:v1">
  <Header>
    <Signature/>
  </Header>
  <Data gzip="no" encrypted="no">
    <Request xmlns="urn:g3.ge:cra:call:GetDataUsingPrivateNumber:v1">
```

```

        <Parameters>
          <PrivateNumber>01010101011</PrivateNumber>
        </Parameters>
      </Request>
    </Data>
    <Thumbprint>0123456789012345678901234567890123456789</Thumbprint>
  </Message>
  <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
    <SignedInfo>
      <CanonicalizationMethod Algorithm="http://www.w3.org/TR/2001/REC-xm1-c14n-20010315" />
      <SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1" />
      <Reference URI="">
        <Transforms>
          <Transform Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature" />
        </Transforms>
        <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
        <DigestValue>OIdYau4WSUMrhTJwZmzQ8IhxAU=</DigestValue>
      </Reference>
    </SignedInfo>

    <SignatureValue>o/JR657t5TpoqG0ZJ/wum7S3nBxqE8M1Wzcp+B+lha3wuQV5d1MmMbCKn63C5mgO
    08MS0Ru5ZX1/hTPQOglwxc0qrhpM/Hi3ikphQ2aS7whvknQAHy1D/Rbgw1NbewAsLcP5rGfEK/gZVC1h/
    on7GAI080n9foe/qpB4E0W15x4JMG8G6gRk8N7+f1mopC7sPLCc7En8Xb42PnCYOPPzsuqI8cMGD1oc+
    iPZVULtheTxRiUqoJkWkPwvx0QXP7YCNRoOMmyrZYd7KETD7/O2ik14PU/cFe40Kv/aBlakx0tWyQdsw
    /mj9dgK6jBuwDNYB1dWvP4FJY93aGout39aeCw==</SignatureValue>
  </Signature>

```

ხელი უნდა მოენეროს Request გოვთოქის Body ნაწილს და ისე დაიპოსტოს სისტემაში, ქვემოთ მოცემულია კოდი რომელიც მეშვეობით ხდება Request XML-ის ხელმოწერა, კოდის გამოყენებამდე აუცილებელია დაინსტალირებული იყოს სერთიფიკატი - „TrustedPeople“ განყოფილებაში

➤ ხელმოწერის კოდი C# ვარემოში

ხელმოწერისთვის აუცილებელი ბიბლიოთეკები:

```

using System.Security.Cryptography.X509Certificates;
using System.Security.Cryptography;
using System.Xml;
using System.Security.Cryptography.Xml;

```



კოდი:

```
static void Main(string[] args)
{
    X509Certificate2 certificate = new X509Certificate2();
    X509Store store = new X509Store(StoreName.TrustedPeople,
StoreLocation.CurrentUser);
    store.Open(OpenFlags.OpenExistingOnly);
    var c = store.Certificates.Find(X509FindType.FindByThumbprint,
"c54f666d707001e7w4dbd9c2daa56ac729d89820", false)[0];

    XmlDocument xmlDoc = new XmlDocument();

    xmlDoc.Load(@"input.xml");

    var XmlToSign = new XmlDocument();
    XmlToSign.LoadXml(xmlDoc.DocumentElement["Body"].OuterXml);

    SignedXml signedXml = new SignedXml(XmlToSign);
    signedXml.SigningKey = c.PrivateKey;

    Reference reference = new Reference();
    reference.Uri = "";

    XmlDsigEnvelopedSignatureTransform env = new
XmlDsigEnvelopedSignatureTransform();
    reference.AddTransform(env);

    signedXml.AddReference(reference);
    signedXml.ComputeSignature();
    XmlElement xmlDigitalSignature = signedXml.GetXml();

    xmlDoc.DocumentElement["Body"].AppendChild(xmlDoc.ImportNode(xmlDigitalSignature
, true));
    xmlDoc.Save(@"signed.xml");
    Console.WriteLine("Finish");
    Console.Read();
}
```

- 1: "c54f666d707001e7w4dbd9c2daa56ac729d89820" – პარამეტრად გადავცემთ ჩვენი სერთიფიკატის Thumbprint მნიშვნელობას
- 2: xmlDoc.Load(@"input.xml"); – პარამეტრად გადავცემთ Request XML-ს რაზეც გვსურს ხელმოწერა
- 3: xmlDoc.Save(@"signed.xml"); – გამომავალი ხელმოწერილი XML დოკუმენტი

## მეთოდი რითაც დაიპოსტება გენერირებული XML-ი და დაბრუნდება პასუხი

გოვთოქის შიგთავსის ფრომირების შემგდომ აუცილებელია მიღებული XML-ი გავაგზავნოთ (დავპოსტოთ) ადრესატთან პასუხის მიღების ან უბრალოდ საავტონტოსთვის ინფორმაციის მიწოდების მიზნით. ქვემოთ მოცემულია ის მეთოდი რითაც ხდება უკვე გამზადებული XML-ის გაგზავნა (მეთოდი დაწერილია C# პროგრამულ ენაზე)

### ➤ მეთოდი

```
public static string Submit(XmlDocument GovTalkMessage, string URL)
{
    HttpWebRequest Req;
    HttpResponse Resp;
    Req = (HttpWebRequest)WebRequest.Create(URL);
    Req.Method = "POST";
    Req.ContentType = "text/xml";
    StreamWriter streamW = new StreamWriter(Req.GetRequestStream());
    GovTalkMessage.Save(streamW);
    streamW.Close();
    Resp = (HttpResponse)Req.GetResponse();
    Stream respStream = Resp.GetResponseStream();
    StreamReader streamR = new StreamReader(respStream,
    System.Text.Encoding.GetEncoding("utf-8"));
    string ResponseText = streamR.ReadToEnd();
    streamR.Close();
    Resp.Close();
    return ResponseText;
}
```

1: GovTalkMessage – პარამეტრად გადავცემთ უშუალოდ იმ XML-ს რაც დავაფორმირეთ

2: URL – პარამეტრად გადავცემთ ლინკს სადაც უნდა მოხდეს დაპოსტვა.

3: ResponseText - გვიბრუნებს პასუხს XML სახით

თუ ორგანიზაციას არ გადაეცა IP მისამართები სადაც Request XML-ს დაპოსტავს, ამ შემთხვევაში მონაცემთა გაცვლის სააგენტოს ინფრასტრუქტურის ფარგლებში შესაძლებელია გამოყენებულ იქნეს :

1) <https://submission.e-government.ge/submission> – ფროდაქშენი (ლაივ) სერვერზე დაპოსტვა

2) <https://test.submission.e-government.ge/submission> – სატესტო გარემოზე დაპოსტვა

➤ ხელმონერის კოდი Java გარემოში

ხელმონერისთვის აუცილებელი ბიბლიოთეკები:

```
import java.io.BufferedReader;
import java.io.ByteArrayOutputStream;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileReader;
import java.io.OutputStreamWriter;
import java.io.StringReader;
import java.net.URL;
import java.security.KeyStore;
import java.security.PrivateKey;
import java.security.cert.X509Certificate;
import java.util.Collections;
import java.util.Enumeration;
import java.util.Scanner;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.net.ssl.HttpURLConnection;
import javax.xml.crypto.dsig.CanonicalizationMethod;
import javax.xml.crypto.dsig.DigestMethod;
import javax.xml.crypto.dsig.SignatureMethod;
import javax.xml.crypto.dsig.SignedInfo;
import javax.xml.crypto.dsig.Transform;
import javax.xml.crypto.dsig.XMLSignature;
import javax.xml.crypto.dsig.XMLSignatureFactory;
import javax.xml.crypto.dsig.dom.DOMSignContext;
import javax.xml.crypto.dsig.spec.C14NMethodParameterSpec;
import javax.xml.crypto.dsig.spec.TransformParameterSpec;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.transform.OutputKeys;
import javax.xml.transform.Transformer;
import javax.xml.transform.TransformerFactory;
import javax.xml.transform.dom.DOMSource;
import javax.xml.transform.stream.StreamResult;
import org.apache.commons.codec.digest.DigestUtils;
import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.Node;
import org.xml.sax.InputSource;
```

კოდი:

```
public class App {

    private static final Logger logger = Logger.getLogger(App.class.getName());

    // DEA endpoint url
```

```

private static final String ENDPOINT_URL = "https://test.submission.e-
government.ge/gg/submission";
// path to govtalk message xml to sign
private static final String TEMPLATE_2_SIGN = "/path/to/template/file";
// path to SDA provided PKCS12 keystore file
private static final String KEYSTORE_FILE = "/path/to/keystore/file";
// SDA provided PKCS12 keystore file password
private static final String KEYSTORE_PASS = "*****";

public static void main(String[] args) {
    try {
        // init
        File templateFile = new File(TEMPLATE_2_SIGN);
        File keystoreFile = new File(KEYSTORE_FILE);

        CertManager certManager = new CertManager(keystoreFile,
KEYSTORE_PASS);

        DocumentBuilderFactory docBuildFac =
DocumentBuilderFactory.newInstance();
        docBuildFac.setNamespaceAware(true);

        TransformerFactory transFac = TransformerFactory.newInstance();

        // get govtalk message and add Thumbprint element to Message element
        String xmlString = "";
        try (BufferedReader reader = new BufferedReader(new
FileReader(templateFile))) {
            String line;
            while ((line = reader.readLine()) != null) {
                xmlString = xmlString + line;
            }
        }

        Document xmlToSign = docBuildFac.newDocumentBuilder().parse(new
InputStream(new StringReader(xmlString)));

        Element thumbprintElem = xmlToSign.createElement("Thumbprint");
thumbprintElem.appendChild(xmlToSign.createTextNode(certManager.getThumbPrint()));
;

xmlToSign.getElementsByTagName("Message").item(0).appendChild(thumbprintElem);

        // get govtalk message Body element, convert to string,
// replace all spaces and newlines and convert to document
        String xmlBodyString;
        try (ByteArrayOutputStream os = new ByteArrayOutputStream()) {
            Transformer trans = transFac.newTransformer();
            trans.setOutputProperty(OutputKeys.OMIT_XML_DECLARATION, "yes");
            trans.transform(new
DOMSource(xmlToSign.getElementsByTagName("Body").item(0)), new StreamResult(os));
            xmlBodyString = new String(os.toByteArray());
            xmlBodyString = xmlBodyString.replaceAll("\n", "");
            xmlBodyString = xmlBodyString.replaceAll("\\s<", "<");
            xmlBodyString = xmlBodyString.replaceAll(">\\s+", ">");
        }
    }
}

```

```

        Document xmlBodyToSign = docBuildFac.newDocumentBuilder().parse(new
        InputSource(new StringReader(xmlBodyString)));

        // sign Body document and copy it's Signature element to govtalk
        message document
        signXml(certManager, xmlBodyToSign.getDocumentElement());

xmlToSign.getElementsByTagName("Body").item(0).appendChild(xmlToSign.importNode(x
mlBodyToSign.getElementsByTagName("Signature").item(0), true));

        // convert signed govtalk message document to string
        String request;
        try (ByteArrayOutputStream os = new ByteArrayOutputStream()) {
            (transFac.newTransformer()).transform(new DOMSource(xmlToSign),
new StreamResult(os));
            request = new String(os.toByteArray());
        }

        // send request to dea
        logger.log(Level.INFO, "\n\n\n\n\n{0}\n\n\n\n\n", request);
        String response = sendRequest(ENDPOINT_URL, request);
        logger.log(Level.INFO, "\n\n\n\n\n{0}\n\n\n\n\n", response);
    } catch (Exception ex) {
        logger.log(Level.SEVERE, "main", ex);
    }
}

private static void signXml(CertManager certManager, Node node) throws
Exception {
    // create xml signature factory
    XMLSignatureFactory fac = XMLSignatureFactory.getInstance("DOM");

    // create reference
    javax.xml.crypto.dsig.Reference reference
        = fac.newReference(
            "",
            fac.newDigestMethod(DigestMethod.SHA1, null),

Collections.singletonList(fac.newTransform(Transform.ENVELOPED,
(TransformParameterSpec) null)),
            null, null
        );

    // create signed info
    SignedInfo signedInfo
        = fac.newSignedInfo(

fac.newCanonicalizationMethod(CanonicalizationMethod.INCLUSIVE,
(C14NMethodParameterSpec) null),
            fac.newSignatureMethod(SignatureMethod.RSA_SHA1, null),
            Collections.singletonList(reference)
        );

    // sign xml document
    DOMSignContext domSignContext = new
    DOMSignContext(certManager.getPrivateKey(), node);
    XMLSignature xmlSignature = fac.newXMLSignature(signedInfo, null);
    xmlSignature.sign(domSignContext);
}

```

```

    }

    private static String sendRequest(String urlStr, String xmlMessage) throws
    Exception {
        // Construct data
        String data = xmlMessage;

        // Send data
        URL url = new URL(urlStr);
        HttpURLConnection conn = (HttpURLConnection) url.openConnection();
        conn.setDoOutput(true);
        conn.setRequestMethod("POST");
        conn.setRequestProperty("Content-Type", "application/x-www-form-
        urlencoded; charset=utf-8");
        conn.setRequestProperty("Content-Length", "" + data.getBytes().length);
        try (OutputStreamWriter wr = new
        OutputStreamWriter(conn.getOutputStream())) {
            wr.write(data);
            wr.flush();
        }

        // Get the response
        StringBuilder responseBuilder = new StringBuilder();
        try (Scanner scanner = new Scanner(conn.getInputStream(), "UTF-8")) {
            String line;
            while (scanner.hasNextLine()) {
                line = scanner.nextLine();
                responseBuilder.append(line);
            }
        }

        return responseBuilder.toString();
    }
}

class CertManager {

    private static final String KEYSTORE_TYPE = "PKCS12";
    private final PrivateKey privateKey;
    private final X509Certificate cert;
    private final String thumbPrint;

    public CertManager(File keyStoreFile, String keyStorePass) throws Exception {
        // init keystore
        KeyStore keyStore = KeyStore.getInstance(KEYSTORE_TYPE);
        try (FileInputStream fis = new FileInputStream(keyStoreFile)) {
            keyStore.load(fis, keyStorePass.toCharArray());
        }

        // get private key
        Enumeration<String> aliases = keyStore.aliases();
        String alias = aliases.nextElement();
        KeyStore.PrivateKeyEntry keyEntry = (KeyStore.PrivateKeyEntry)
        keyStore.getEntry(alias, new
        KeyStore.PasswordProtection(keyStorePass.toCharArray()));

        privateKey = keyEntry.getPrivateKey();
        cert = (X509Certificate) keyEntry.getCertificate();
    }
}

```

```
        thumbPrint = DigestUtils.shaHex(cert.getEncoded());
    }

    public PrivateKey getPrivateKey() {
        return privateKey;
    }

    public X509Certificate getCert() {
        return cert;
    }

    public String getThumbPrint() {
        return thumbPrint;
    }
}
```