STEP TO GENERATE SIGNATURE (JSON) (via C#)

1) Prepare JSON document "without signature chunk" (document version 1.1).

```
X509Certificate2 cert;
 cert = new X509Certificate2();
     cert.Import(File.ReadAllBytes(certPath), certPass,
                                         X509KeyStorageFlags.MachineKeySet |
                                         X509KeyStorageFlags.PersistKeySet |
                                         X509KeyStorageFlags.Exportable);
```

2) Minify the JSON > Create SHA256 Hash > Convert Hash to Base64

Use raw document in step 1, **minify and remove white space.** Then, apply **SHA 256 and base64 string encode**.

```
string jsonString = SerializeJson(invoiceJsonModel);
byte[] docHash = HashUtility.Sha256Hash(jsonString);
var DocDigest = Convert.ToBase64String(docHash); //replace to {{Replace Value 1}}

 static string SerializeJson(object doc)
 {
     var settings = new JsonSerializerSettings
     {
         DateFormatString = "yyyy-MM-ddTH:mmZ",
         DateTimeZoneHandling = DateTimeZoneHandling.Utc,
         NullValueHandling = NullValueHandling.Ignore,

     };
     var jsonString = JsonConvert.SerializeObject(doc, settings);

     return jsonString;
 }

 byte[] Sha256Hash(string text)
 {
     using (SHA256 sha256 = SHA256.Create())
     {
         byte[] byteData = Encoding.UTF8.GetBytes(text);
         var hashBytes = sha256.ComputeHash(byteData);

         return hashBytes;
     }
 }
```

*Copy the generated value and replace to {{Replace Value 1}} in the signature template.

3) Load the Private Certificate > Extract the Private Key > Create RSA Signature Formatter > Sign the Hash > Convert Hash to Base64

```
 byte[] sign = SignData(docHash,cert);
 var SignatureValue = Convert.ToBase64String(sign);   //replace to {{Replace Value 2}}

 byte[] SignData(byte[] hashdata, X509Certificate2 cert)
 {
     byte[] signedData = null;
     using (RSA rsa = cert.GetRSAPrivateKey())
     {
         signedData = rsa.SignHash(hashdata, HashAlgorithmName.SHA256, RSASignaturePadding.Pkcs1);
     }
     return signedData;
 }
```

Use hashed value (before apply base 64 encode in step 2) for raw document together with private certificate (.p12).
Use private key in the certificate to hash the document and apply base64 string encode.

*Copy the generated value and replace to {{Replace Value 2}} in the signature template.

4) Load the Private Certificate > Compute the Hash of the Certificate > Convert the Hash to Base64

Use the private certificate (.p12) apply **SHA 256 and base64 string encode** to the certificate raw data.

```
  var certHash= GetCertHash(X509Certificate2 cert)  // replace to {{Replace Value 3}}

   string GetCertHash(X509Certificate2 cert)
     {
         byte[] rawcertbytes = cert.RawData;
         byte[] certbytes = Sha256HashBytes(rawcertbytes);

         return Convert.ToBase64String(certbytes);
     }

     byte[] Sha256HashBytes(byte[] byteData)
     {
         using (SHA256 sha256 = SHA256.Create())
         {
             var hashBytes = sha256.ComputeHash(byteData);

             return hashBytes;
         }
     }
```
*Copy the generated value and replace to {{Replace Value 3}} in the signature template.

5) Load the Public Certificate > Retrieve and Convert the Serial Number

Use the **SerialNumber** of the public certificate (.cer) **OR** private certificate (.p12), **convert the hexadecimal string to a BigInteger**

```
    var serialNum = GetCertSerialNumber(cert)  //  {{Replace Value 4}}
     BigInteger GetCertSerialNumber(X509Certificate2 cert)
     {
         return BigInteger.Parse(cert.SerialNumber, NumberStyles.HexNumber);

     }
```

*Copy the generated value and replace to {{Replace Value 4}} in the signature template.

6) Load the Public Certificate > Extract the Raw Data > convert it to a Base64

Use the public certificate (.cer) **OR** private certificate (.p12) and apply **base64 string conversion** to the **rawdata**.

```
    var certData = GetX509Certificate(cert);  // replace to {{Replace Value 5}}

   string GetX509Certificate(X509Certificate2 cert)
   {
       byte[] rawcertbytes = cert.RawData;
       return Convert.ToBase64String(rawcertbytes);
   }
```
*Copy the generated value and replace to {{Replace Value 5}} in the signature template.

7) Obtain QualifyingProperties String > Minify the String > Compute the Hash of the Signed Properties > convert it to a Base64

Use the "QualifyingProperties" object of the signature template after fiilled with value from step2 to step 6, **minify and remove white space**. Then, apply **SHA 256 and base64 string encode**.

*** make sure the SigningTime in the chunk is later than document submission time.*

```
var SignedProperties = doc.UBLExtensions[0].UBLExtension[0].ExtensionContent[0].
                                UBLDocumentSignatures[0].SignatureInformation[0].
                                Signature[0].Object[0].QualifyingProperties[0];//.SignedProperties[0];
 var jsonSignedProperties = SerializeJson(SignedProperties);
```

{"Target":"signature","SignedProperties":[{"Id":"id-xades-signed-props","SignedSignatureProperties":[{"SigningTime":[{"_":"2024-07-16T05:56:06Z"}],"SigningCertificate":[{"Cert":[{"CertDigest":[{"DigestMethod":[{"_":"","Algorithm":"http://www.w3.org/2001/04/xmlenc#sha256"}],"DigestValue":[{"_":"SLFswNMf8a6muzczA+EO356bvJNDkr9LhT25+pqacdE="}]}],"IssuerSerial":[{"X509IssuerName":[{"_":"CN=Trial LHDNM Sub CA V1, OU=Terms of use at http://www.posdigicert.com.my, O=LHDNM, C=MY"}],"X509SerialNumber":[{"_":"3528254343444"}]}]}]}]}]}]}

```
byte[] propCert = Sha256Hash(jsonSignedProperties);
string PropsDigest = Convert.ToBase64String(propCert);  // replace to {{Replace Value 6}}
```

*Copy the generated value and replace to {{Replace Value 6}} in the signature template.

8) Add the signature chunk below to the JSON document (document version 1.1).

SIGNATURE TEMPLATE (JSON)

** Remember to replace issuer name and subject name to related certificate provider.

```
"UBLExtensions": [
    {
      "UBLExtension": [
        {
          "ExtensionURI": [
            {
              "_": "urn:oasis:names:specification:ubl:dsig:enveloped:xades"
            }
          ],
          "ExtensionContent": [
            {
              "UBLDocumentSignatures": [
                {
                  "SignatureInformation": [
                    {
                      "ID": [
                        {
                          "_": "urn:oasis:names:specification:ubl:signature:1"
                        }
                      ],
                      "ReferencedSignatureID": [
                        {
                          "_": "urn:oasis:names:specification:ubl:signature:Invoice"
                        }
                      ],
                      "Signature": [
                        {
                          "Id": "signature",
                          "Object": [
                            {
                              "QualifyingProperties": [
                                {
                                  "Target": "signature",
                                  "SignedProperties": [
                                    {
                                      "Id": "id-xades-signed-props",
                                      "SignedSignatureProperties": [
                                        {
                                          "SigningTime": [
                                            {
                                              "_": "2024-07-02T07:46:03Z"
                                            }
                                          ],
                                          "SigningCertificate": [
                                            {
                                              "Cert": [
                                                {
                                                  "CertDigest": [
                          {
                            "DigestMethod": [
                              {
                                "_": "",
                                "Algorithm": "http://www.w3.org/2001/04/xmlenc#sha256"
                              }
                            ],
                            "DigestValue": [
                              {
                                "_": "{{Replace Value 3}}"
                              }
                            ]
                          }
                        ],
                        "IssuerSerial": [
                          {
                            "X509IssuerName": [
                              {
                                "_": "CN=Trial LHDNM Sub CA V1, OU=Terms of use at http://www.posdigicert.com.my, O=LHDNM, C=MY"
                              }
                            ],
                            "X509SerialNumber": [
                              {
                                "_": "{{Replace Value 4}}"
                              }
                            ]
                          }
                        ]
                      }
                    ]
                  }
                ]
              }
            ]
          }
        ]
      }
    ]
  }
],
"KeyInfo": [
  {
    "X509Data": [
      {
        "X509Certificate": [
          {
            "_": "{{Replace Value 5}}"
          }
        ],
        "X509SubjectName": [
          {
            "_": "CN=Trial LHDNM Sub CA V1, OU=Terms of use at http://www.posdigicert.com.my, O=LHDNM, C=MY"
          }
        ],
        "X509IssuerSerial": [
          {
            "X509IssuerName": [
              {
                "_": "CN=Trial LHDNM Sub CA V1, OU=Terms of use at http://www.posdigicert.com.my, O=LHDNM, C=MY"
              }
            ],
            "X509SerialNumber": [
              {
                "_": "{{Replace Value 4}}"
              }
            ]
          }
        ]
      }
    ]
  }
],
"SignatureValue": [
  {
    "_": "{{Replace Value 2}}"
  }
],
            "SignedInfo": [
             {
              "SignatureMethod": [
               {
                "_": "",
                "Algorithm": "http://www.w3.org/2001/04/xmldsig-more#rsa-sha256"
               }
              ],
              "Reference": [
               {
                "Type": "http://uri.etsi.org/01903/v1.3.2#SignedProperties",
                "URI": "#id-xades-signed-props",
                "DigestMethod": [
                 {
                  "_": "",
                  "Algorithm": "http://www.w3.org/2001/04/xmlenc#sha256"
                 }
                ],
                "DigestValue": [
                 {
                  "_": "{{Replace Value 6}}"
                 }
                ]
               },
               {
                "Type": "",
                "URI": "",
                "DigestMethod": [
                 {
                  "_": "",
                  "Algorithm": "http://www.w3.org/2001/04/xmlenc#sha256"
                 }
                ],
                "DigestValue": [
                 {
                  "_": "{{Replace Value 1}}"
                 }
                ]
               }
              ]
             }
            ]
           }
          ]
         }
        ]
       }
      ]
     }
    ]
   }
  ]
 }
],
"Signature": [
 {
  "ID": [
   {
    "_": "urn:oasis:names:specification:ubl:signature:Invoice"
   }
  ],
  "SignatureMethod": [
   {
    "_": "urn:oasis:names:specification:ubl:dsig:enveloped:xades"
   }
  ]
 }
]
```