SECURING JSON FILES WITH DIGITAL SIGNATURES: MYINVOIS USER GUIDE

Transform the document

ELEMENT	FIELD TYPE	DESCRIPTION	VALUE EXAMPLE	UBL SCHEMA MAPPING
Transform	Structure	The document will be required to be transformed by performing the following: • Removing the sections “UBLExtensions”, and “Signature” if they do exist. • Minify the file by removing new lines and not needed spaces. The output of this step would be a transformed document. The input document shown above once transformed it should be as per the below JSON file.	{"_D":"urn:oasis:names:specification:ubl:schema:xsd:Invoice-2","_A":"urn:oasis:names:specification:ubl:schema:xsd:CommonAggregateComponents-2","_B":"urn:oasis:names:specification:ubl:schema:xsd:CommonBasicComponents-2","Invoice":[{"ID":[{"_":"INV100003"}],"IssueDate":[{"_":"2024-07-11"}],"IssueTime":[{"_":"00:30:00Z"}],"InvoiceType……. PriceAmount":[{"_":10000,"currencyID":"MYR"}]}],"TaxTotal":[{"TaxAmount":[{"_":15680,"currencyID":"MYR"}],"TaxSubtotal":[{"TaxableAmount":[{"_":784000,"currencyID":"MYR"}],"TaxAmount":[{"_":15680,"currencyID":"MYR"}],"Percent":[{"_":2}],"TaxCategory":[{"ID":[{"_":"02"}],"TaxScheme":[{"ID":[{"_":"OTH","schemeAgencyID":"6","schemeID":"UN/ECE 5153"}]}]}]}]}]}],"TaxExchangeRate":[{"SourceCurrencyCode":[{"_":"MYR"}],"TargetCurrencyCode":[{"_":"MYR"}],"CalculationRate":[{"_":0}]}]}]}	/ubl:Invoice

POWERSHELL SCRIPT

```
function Minify-Json {
	param (
		[Parameter(Mandatory=$true)]
		[string]$JsonString
	)

	# Remove all whitespace between elements, preserving whitespace within string values
	$minifiedJson = [System.Text.RegularExpressions.Regex]::Replace($JsonString, '("(?:\\.|[^"\\])*")|\s+', {
		param($match)
		if ($match.Groups[1].Success) {
			return $match.Groups[1].Value
		} else {
			return ''
		}
	})
```

```
        return $minifiedJson
}

 # Read the content of the file
 $filePath = Read-Host -Prompt " Enter the path and name of the payload file"
 $document = Get-Content -Path $filePath -Raw

 # Write the content to the file
 $outputFilePath = Read-Host -Prompt " Enter the path and name of the minified payload file"

 # Minify the JSON string
 $minifiedJson = Minify-Json -JsonString $document

 # Write the minified content to the output file
  $minifiedJson | Out-File $outputFilePath -Force

 # Display Minified the JSON string
 Write-Output "Minified e-Invoice: $minifiedJson"
```

Signature Element

ELEMENT	FIELD TYPE	DESCRIPTION	VALUE EXAMPLE	UBL SCHEMA MAPPING
Signed information	Structure	This contains information on how to transform the document to retrieve the information that are covered by the digital signature. This includes any transformations performed. The signed information would cover all the document content except specific sections as will be presented later in the signature creation section.		/ubl:Invoice /ext:UBLExtensions /ext:UBLExtension /ext:ExtensionContent /sig:UBLDocumentSignatures /sac:SignatureInformation /ds:Signature /ds:SignedInfo

ELEMENT	FIELD TYPE	DESCRIPTION	VALUE EXAMPLE	UBL SCHEMA MAPPING
Signature value	xsd:base64Binary	This contains the signature value that was generated by signing the document digest within the signed information section above. [Reference field: **Sig**]	`MEQCIGvLa1f3uMCe0AidKUWJ5ghMiDMRcC0qO78ntcTKVOYgAiAKBkX+uuFhbIcye3JznNa45qH1twlLFu/ qPzEQ9HMNLw==`	/ubl:Invoice /ext:UBLExtensions /ext:UBLExtension /ext:ExtensionContent /sig:UBLDocumentSignatures /sac:SignatureInformation /ds:Signature /ds:SignedInfo

POWERSHELL SCRIPT

```
# Extract the private key using RSACryptoServiceProvider
$privateKeyProvider = [System.Security.Cryptography.X509Certificates.RSACertificateExtensions]::GetRSAPrivateKey($cert)

# Create RSA signature formatter
$rsaFormatter = New-Object System.Security.Cryptography.RSAPKCS1SignatureFormatter $privateKeyProvider
$rsaFormatter.SetHashAlgorithm("SHA256")

# Read the content of the e-invoice document
$filePath = Read-Host -Prompt " Enter the path and name of the payload file"
$document = Get-Content -Path $filePath -Raw

# Load necessary assemblies
Add-Type -AssemblyName System.Security

# Minify the JSON string
$minifiedJson = Minify-Json -JsonString $document

# Compute the hash of the document
 $hash = $sha256.ComputeHash([System.Text.Encoding]::UTF8.GetBytes($minifiedJson))

# Sign the hash
$signature = $rsaFormatter.CreateSignature($hash)

# Convert the signature to Base64
$signatureBase64 = [Convert]::ToBase64String($signature)

Write-Output "e-Invoice Signature value (Sig): $signatureBase64"
```

ELEMENT	FIELD TYPE	DESCRIPTION	VALUE EXAMPLE	UBL SCHEMA MAPPING
Key information	Structure	This structure contains the X509 compliant certificate public information that was used to sign the document.	<ds:KeyInfo> <ds:X509Data> <ds:X509Certificate>MIID3jCCA4SgAwIBAgITE QAAOAPF90A…/rXA==</ds:X509Certificate> </ds:X509Data> </ds:KeyInfo>	/ubl:Invoice /ext:UBLExtensions /ext:UBLExtension /ext:ExtensionContent /sig:UBLDocumentSignatures /sac:SignatureInformation /ds:Signature /ds:KeyInfo
Object element	Structure	This would contain the signed properties object that would be hashed and included in the signed information object. The object element that would be required is xades:QualifyingProperties		/ubl:Invoice /ext:UBLExtensions /ext:UBLExtension /ext:ExtensionContent /sig:UBLDocumentSignatures /sac:SignatureInformation /ds:Signature /ds:Object

Signed Information

ELEMENT	FIELD TYPE	DESCRIPTION	VALUE EXAMPLE	UBL SCHEMA MAPPING
Canonicalization Method	Empty	Name of the XML canonicalization algorithm - `xml-c14n11`.	<ds:CanonicalizationMethod Algorithm="https://www.w3.org/TR/xml-c14n11/#"/>	/ubl:Invoice /ext:UBLExtensions /ext:UBLExtension /ext:ExtensionContent /sig:UBLDocumentSignatures /sac:SignatureInformation /ds:Signature /ds:SignedInfo /ds:CanonicalizationMethod [@Algorithm = 'https://www.w3.org/TR/xml-c14n11/#']
Signature Method	Empty	This is the signature creation method which should be set to rsa-sha256.	<ds:SignatureMethod Algorithm="http://www.w3.org/2001/04/xmldsig-more#rsa-sha256"/>	/ubl:Invoice /ext:UBLExtensions /ext:UBLExtension /ext:ExtensionContent /sig:UBLDocumentSignatures /sac:SignatureInformation /ds:Signature /ds:SignedInfo /ds:SignatureMethod [@Algorithm = 'http://www.w3.org/2001/04/xmldsig-more#rsa-sha256']
Document Signed Data	Structure	This would contain the information that would be signed, which would be the document digest and related details on how that digest was computed.		/ubl:Invoice /ext:UBLExtensions /ext:UBLExtension /ext:ExtensionContent /sig:UBLDocumentSignatures /sac:SignatureInformation /ds:Signature /ds:SignedInfo /ds:Reference[@Id = 'id-doc-signed-data' AND @URI =

ELEMENT	FIELD TYPE	DESCRIPTION	VALUE EXAMPLE	UBL SCHEMA MAPPING
				‘’] (The URI has to be given as empty to refer to the parent enveloping document)
XAdES Signed Properties	Structure	This contains the signed properties representing the hash of the information of the certificate that was used to sign the document.		/ubl:Invoice /ext:UBLExtensions /ext:UBLExtension /ext:ExtensionContent /sig:UBLDocumentSignatures /sac:SignatureInformation /ds:Signature /ds:SignedInfo /ds:Reference[@URI = ‘#id-xades-signed-props’]

Document Signed Data

ELEMENT	FIELD TYPE	DESCRIPTION	VALUE EXAMPLE	UBL SCHEMA MAPPING
Transforms	Structure	The value of this field should be given exactly as per the sample value provided.	<ds:Transform Algorithm=”http://www.w3.org/TR/1999/REC-xpath-19991116”> <ds:XPath>not(//ancestor-or-self::ext:UBLExtensions)</ds:XPath> </ds:Transform> <ds:Transform Algorithm=”http://www.w3.org/TR/1999/REC-xpath-19991116”> <ds:XPath>not(//ancestor-or-self::cac:Signature)</ds:XPath> </ds:Transform> <ds:Transform Algorithm=”http://www.w3.org/2006/12/xml-c14n11”/>	/ubl:Invoice /ext:UBLExtensions /ext:UBLExtension /ext:ExtensionContent /sig:UBLDocumentSignatures /sac:SignatureInformation /ds:Signature /ds:SignedInfo /ds:Reference[@Id = ‘id-doc-signed-data’ AND @URI = ‘’] / ds:Transforms
Digest Method	Empty	The value of this field should be given exactly as per the sample value provided.	<ds:DigestMethod Algorithm=”http://www.w3.org/2001/04/xmlenc#sha256”/>	/ubl:Invoice /ext:UBLExtensions /ext:UBLExtension /ext:ExtensionContent /sig:UBLDocumentSignatures /sac:SignatureInformation /ds:Signature /ds:SignedInfo /ds:Reference[@Id = ‘id-doc-signed-data’ AND @URI = ‘’] / ds:DigestMethod

Signed Properties

ELEMENT	FIELD TYPE	DESCRIPTION	VALUE EXAMPLE	UBL SCHEMA MAPPING
SigningTime	Date and time	The property specifies the UTC time at which the signer (purportedly) performed the signing process. Time	`2024-11-26T18:00:00Z`	/ubl:Invoice /ext:UBLExtensions /ext:UBLExtension /ext:ExtensionContent /sig:UBLDocumentSignatures /sac:SignatureInformation /ds:Signature /ds:Object /xades:QualifyingProperties [@Target = 'signature'] /xades:SignedProperties [@Id = 'id-xades-signed-props'] /xades:SignedSignatureProperties /xades:SigningTime

POWERSHELL SCRIPT

```
$utcTimestamp = Get-Date -Format "yyyy-MM-ddTHH:mm:ssZ"
Write-Host "Current Sign UTC Timestamp: $utcTimestamp"
```

ELEMENT	FIELD TYPE	DESCRIPTION	VALUE EXAMPLE	UBL SCHEMA MAPPING
SigningCertificate.Cert.CertDigest.DigestMethod	Empty	This is the method used to calculate the digest and it is set to SHA256. The value should be set as per the sample value provided here.	<ds:DigestMethod Algorithm="http://www.w3.org/2001/04/xmlenc#sha256"/>	/ubl:Invoice /ext:UBLExtensions /ext:UBLExtension /ext:ExtensionContent /sig:UBLDocumentSignatures /sac:SignatureInformation /ds:Signature /ds:Object /xades:QualifyingProperties [@Target = 'signature'] /xades:SignedProperties [@Id = 'id-xades-signed-props'] /xades:SignedSignatureProperties /xades:SigningCertificate /xades:Cert /xades:CertDigest /ds:DigestMethod

ELEMENT	FIELD TYPE	DESCRIPTION	VALUE EXAMPLE	UBL SCHEMA MAPPING
signingCertificate.Cert.CertDigest.DigestValue	xsd:base64binary	This is the HEX-SHA256 encoded certificate information. [Reference field: **CertDigest**]	`ZDMwMmI0MTE1NzVjOTU2NTk4YzVlOD...k1YTA2OWQ0NjY2MjQ4NQ==`	/ubl:Invoice /ext:UBLExtensions /ext:UBLExtension /ext:ExtensionContent /sig:UBLDocumentSignatures /sac:SignatureInformation /ds:Signature /ds:Object /xades:QualifyingProperties [@Target = 'signature'] /xades:SignedProperties [@Id = 'id-xades-signed-props'] /xades:SignedSignatureProperties /xades:SigningCertificate /xades:Cert /xades:CertDigest /ds:DigestValue

POWERSHELL SCRIPT

```
$certPath = Read-Host -Prompt "Enter the path of your certificate file (*.pfx/*.p12)"
$password = Read-Host -AsSecureString -Prompt "Enter the password for your certificate file"
$cert = New-Object System.Security.Cryptography.X509Certificates.X509Certificate2($certPath, $password)

# Compute the hash of the certificate using SHA-256
$sha256 = [System.Security.Cryptography.SHA256]::Create()
$certHash = $sha256.ComputeHash($cert.RawData)

# Convert the hash to a Base64 encoded string
$CertDigest = [Convert]::ToBase64String($certHash)
Write-Output "Retrieve CertDigest: $CertDigest "
```

ELEMENT	FIELD TYPE	DESCRIPTION	VALUE EXAMPLE	UBL SCHEMA MAPPING
KeyInfo.X509Data.X509Certificate	String	Cert Information in raw format.	`MIIFlDCCA3ygAwIBAgIQeomZorO+0AwmW2BRdWJMxTANBgkqhkiG9w0BAQsFADB1MQswCQYDVQQGEwJNWTEOMAwGA1UEChMFTEhETk0xNjA0BgNVBAsTLVRlcm1zIG9mIHVZSBhdCBodHRw.....`	/ubl:Invoice /ext:UBLExtensions /ext:UBLExtension /ext:ExtensionContent /sig:UBLDocumentSignatures /sac:SignatureInformation /ds:Signature /ds: KeyInfo / xades: X509Data /xades: X509Certificate

POWERSHELL SCRIPT

```
# Extract the raw data of the certificate
$rawData = [Convert]::ToBase64String($cert.RawData)
Write-Output "Retreive X509 Certificate: $rawData"
```

ELEMENT	FIELD TYPE	DESCRIPTION	VALUE EXAMPLE	UBL SCHEMA MAPPING
KeyInfo.X509Data. X509SubjectName	String	This is the CN of the certificate used to read Certificate serial number, email, TIN etc.	E=anas.a@fgvholdings.com, SERIALNUMBER=D12345678, CN=Dummy, OU=Test Unit eInvoice, OID.2.5.4.97=C29702635060, O=Dummy, C=MY	/ubl:Invoice /ext:UBLExtensions /ext:UBLExtension /ext:ExtensionContent /sig:UBLDocumentSignatures /sac:SignatureInformation /ds:Signature /ds: KeyInfo / xades: X509Data /xades: X509SubjectName

POWERSHELL SCRIPT

```
# Retrieve the subject name from the certificate
$subjectName = $cert.SubjectName.Name
Write-Host "Subject Name: $subjectName"
```

ELEMENT	FIELD TYPE	DESCRIPTION	VALUE EXAMPLE	UBL SCHEMA MAPPING
SigningCertificate.Cert. IssuerSerial.X509Issue rName	String	This is the CN of the certificate used to generate the digital signature. This should be equal to the organization name.	CN=Trial LHDNM Sub CA V1, OU=Terms of use at http://www.testcertcomp.com.my, O=LHDNM, C=MY	/ubl:Invoice /ext:UBLExtensions /ext:UBLExtension /ext:ExtensionContent /sig:UBLDocumentSignatures /sac:SignatureInformation /ds:Signature /ds:Object /xades:QualifyingProperties [@Target = 'signature'] /xades:SignedProperties [@Id = 'id-xades-signed-props'] /xades:SignedSignatureProperties /xades:SigningCertificate /xades:Cert /xades:IssuerSerial /ds:X509IssuerName

POWERSHELL SCRIPT

```
# Retrieve Issuer Name from Cert
$issuerName = [System.Security.SecurityElement]::Escape($cert.IssuerName.Name)
Write-Output "Certificate X509 Issuer Name: $issuerName"
```

ELEMENT	FIELD TYPE	DESCRIPTION	VALUE EXAMPLE	UBL SCHEMA MAPPING
SigningCertificate.Cert.IssuerSerial.X509SerialNumber	xsd:normalizedString	This is the digital certificate serial number. This is the out of the standard certificate serial number.	379112742831380471835263969587287663520637587	/ubl:Invoice /ext:UBLExtensions /ext:UBLExtension /ext:ExtensionContent /sig:UBLDocumentSignatures /sac:SignatureInformation /ds:Signature /ds:Object /xades:QualifyingProperties [@Target = 'signature'] /xades:SignedProperties [@Id = 'id-xades-signed-props'] /xades:SignedSignatureProperties /xades:SigningCertificate /xades:Cert /xades:IssuerSerial /ds:X509SerialNumber

POWERSHELL SCRIPT

```
# Retrieve the serial number
$serialNumber = $cert.SerialNumber

# Convert the hexadecimal string to a BigInteger
$CertSerialNumber = [System.Numerics.BigInteger]::Parse($serialNumber, [System.Globalization.NumberStyles]::HexNumber)
Write-Output "Retrieve Certificate serial number (CertSerialNumber): $CertSerialNumber"
```

XAdES Signed Properties

ELEMENT	FIELD TYPE	DESCRIPTION	VALUE EXAMPLE	UBL SCHEMA MAPPING
Digest Method	Empty	The value of this field should be given exactly as per the sample value provided.	<ds:DigestMethod Algorithm="http://www.w3.org/2001/04/xmlenc#sha256"/>	/ubl:Invoice /ext:UBLExtensions /ext:UBLExtension /ext:ExtensionContent /sig:UBLDocumentSignatures /sac:SignatureInformation /ds:Signature /ds:SignedInfo /ds:Reference[@URI = '#id-xades-signed-props'] / ds:DigestMethod

ELEMENT	FIELD TYPE	DESCRIPTION	VALUE EXAMPLE	UBL SCHEMA MAPPING
Digest	xsd:base64Binary	This is the certificate digest value that would be created using HEX-to Base64 Encoder applied to the XAdES object that is described later containing the certificate and signing properties. [Reference field: **PropsDigest**]	`oRtv5YelD32v/jp/tC3MzfG0PimzfZ8lLQQkPlTBGj8=`	/ubl:Invoice /ext:UBLExtensions /ext:UBLExtension /ext:ExtensionContent /sig:UBLDocumentSignatures /sac:SignatureInformation /ds:Signature /ds:SignedInfo /ds:Reference[@URI = '#id-xades-signed-props'] / ds:DigestValue

POWERSHELL SCRIPT

```
# Assign the XML string to a variable using a here-string
$jsonString = @"
{"Target":"signature","SignedProperties":[{"Id":"id-xades-signed-
props","SignedSignatureProperties":[{"SigningTime":[{"_":"$utcTimestamp"}],"SigningCertificate":[{"Cert":[{"CertDigest":[{"DigestMethod":[{"_":"","Algorithm":"http://www.w3.org/2001/04/xmlenc#sha256"}],"DigestValue":[{"_":"$certDigest"}]}],"IssuerSerial":[{"X509IssuerName":[{"_":"$issuerName"}],"X509SerialNumber":[{"_":"$CertSerialNumber"}]}]}]}]}]}]}
"@

# Minify the JSON string
$signedprops = Minify-Json -JsonString $jsonString

# Compute the hash of the signedprops' UTF-8 bytes
$signedpropshash = $sha256.ComputeHash([System.Text.Encoding]::UTF8.GetBytes($signedprops))

# Convert the hash to a Base64 encoded string
$PropsDigest = [Convert]::ToBase64String($signedpropshash)

Write-Output "signed properties digest: $PropsDigest "
```

ELEMENT	FIELD TYPE	DESCRIPTION	VALUE EXAMPLE	UBL SCHEMA MAPPING
Digest	xsd:base64Binary	This is the document digest value that would be created using HEX-to Base64 Encoder applied to the entire transformed document content. This is the value that would be signed using the private key. [Reference field: **DocDigest**]	`oRtv5YelD32v/jp/tC3MzfG0PimzfZ8lLQQkPlTBGj8=`	/ubl:Invoice /ext:UBLExtensions /ext:UBLExtension /ext:ExtensionContent /sig:UBLDocumentSignatures /sac:SignatureInformation /ds:Signature /ds:SignedInfo /ds:Reference[@Id = 'id-doc-signed-data' AND @URI = ''] / ds:DigestValue

POWERSHELL SCRIPT

```
$signString = @"
}],"UBLExtensions":[{"UBLExtension":[{"ExtensionURI":[{"_":"urn:oasis:names:specification:ubl:dsig:enveloped:xades"}],"ExtensionContent":[{"UBLDocumentSignatures":[{"SignatureInformation":[{"ID":[{"_":"urn:oasis:names:specification:ubl:signature:1"}],"ReferencedSignatureID":[{"_":"urn:oasis:names:specification:ubl:signature:Invoice"}],"Signature":[{"Id":"signature","Object":[{"QualifyingProperties":[{"Target":"signature","SignedProperties":[{"Id":"id-xades-signed-props","SignedSignatureProperties":[{"SigningTime":[{"_":"$utcTimestamp"}],"SigningCertificate":[{"Cert":[{"CertDigest":[{"DigestMethod":[{"_":"","Algorithm":"http://www.w3.org/2001/04/xmlenc#sha256"}],"DigestValue":[{"_":"$certDigest"}]}],"IssuerSerial":[{"X509IssuerName":[{"_":"$issuerName"}],"X509SerialNumber":[{"_":"$CertSerialNumber"}]}]}]}]}]}]}]}],"KeyInfo":[{"X509Data":[{"X509Certificate":[{"_":"$rawData"}],"X509SubjectName":[{"_":"$subjectName"}],"X509IssuerSerial":[{"X509IssuerName":[{"_":"$issuerName"}],"X509SerialNumber":[{"_":"$CertSerialNumber"}]}]}]}],"SignatureValue":[{"_":"$signatureBase64"}],"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":[{"_":"$signedpropsdigest"}]},{"Type":"","URI":"","DigestMethod":[{"_":"","Algorithm":"http://www.w3.org/2001/04/xmlenc#sha256"}],"DigestValue":[{"_":"$DocDigest"}]}]}]}]}]}]}],"Signature":[{"ID":[{"_":"urn:oasis:names:specification:ubl:signature:Invoice"}],"SignatureMethod":[{"_":"urn:oasis:names:specification:ubl:dsig:enveloped:xades"}]}]}]}
"@


# Read the content of JsonFile1
$jsonFile1Content = Get-Content -Path $fileOutPath -Raw

# Remove the last 3 characters from the file content
$modifiedContent = $jsonFile1Content.Substring(0, $jsonFile1Content.Length - 7)
```

```
# Combine the data from JsonFile1 and JsonFile2
$combinedData = $modifiedContent + $signString

# Minify the JSON string
$finaldata = Minify-Json -JsonString $combinedData

# Write the updated content back to the file
$finaldata | Out-File -FilePath $fileOutPath -Encoding UTF8

# Encode the content using Base64
$encodedContent = [System.Convert]::ToBase64String([System.Text.Encoding]::UTF8.GetBytes($finaldata))
# Load necessary assemblies
Add-Type -AssemblyName System.Security

# Create a new instance of SHA256
$sha256 = [System.Security.Cryptography.SHA256]::Create()

# Compute the hash of the document
$hash = $sha256.ComputeHash([System.Text.Encoding]::UTF8.GetBytes($document))

# Convert the hash to a Base64 string
$DocDigest = [System.Convert]::ToBase64String($hash)

Write-Output "Compute the DocDigest of the e-Invois Document (DocDigest): $DocDigest "
```

JSON Signing Elements

{"_D": "urn:oasis:names:specification:ubl:schema:xsd:Invoice-2","_A": "urn:oasis:names:specification:ubl:schema:xsd:CommonAggregateComponents-2","_B": "urn:oasis:names:specification:ubl:schema:xsd:CommonBasicComponents-2","Invoice": [{…..<Transformed e-invoice data here>….."<last object>":[{ ……..}] #<--- Minified version of JSON file by removing the sections "UBLExtensions", and "Signature" if they do exist.

```
,"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": [
                {
                  "_": "<utcTimestamp>"
                }
              ],
              "SigningCertificate": [
                {
                  "Cert": [
                    {
                      "CertDigest": [
                        {
                          "DigestMethod": [
                            {
                              "_": "", "Algorithm": "http://www.w3.org/2001/04/xmlenc#sha256"
                            }
                          ],
                          "DigestValue": [
                            {
                              "_": "<CertDigest>"
                            }
                          ]
                        }
                      ],
                      "IssuerSerial": [
                        {
```

```
                                                    "X509IssuerName": [
                                                        {
                                                          "_": "<issuerName>"
                                                        }
                                                    ],
                                                    "X509SerialNumber": [
                                                        {
                                                          "_": "<CertSerialNumber>"
                                                        }
                                                    ]
                                                }
                                            ]
                                        }
                                    ]
                                }
                            ]
                        }
                    ]
                }
            ]
        }
    ],
    "KeyInfo": [
        {
            "X509Data": [
                {
                  "X509Certificate": [
                        {
                          "_": "<x509Certificate>"
                        }
                    ],
                    "X509SubjectName": [
                        {
```

```
                    "_": "<subjectName>"
                }
            ],
            "X509IssuerSerial": [
                {
                    "X509IssuerName": [
                        {
                            "_": "<issuerName>"
                        }
                    ],
                    "X509SerialNumber": [
                        {
                            "_": "<CertSerialNumber>"
                        }
                    ]
                }
            ]
        }
    ]
}
],
"SignatureValue": [
                    {
                        "_": "<Sig>"
                    }
],
"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": [
                        {
                          "_": "<PropsDigest>"
                        }
                      ]
                  },
                  {
                    "Type": "",
                    "URI": "",
                    "DigestMethod": [
                       {
                          "_": "","Algorithm": "http://www.w3.org/2001/04/xmlenc#sha256"
                       }
                      ],
                     "DigestValue": [
                       {
                          "_": "<DocDigest>"
                       }
                      ]
                  }
              ]
              }
          ]
          }
        ]
      }
```

```
                                    ]
                                }
                            ]
                        }
                ]
            }
    ],
    "Signature": [
            {
             "ID": [
              {
                "_": "urn:oasis:names:specification:ubl:signature:Invoice"
              }
             ],
             "SignatureMethod": [
              {
                "_": "urn:oasis:names:specification:ubl:dsig:enveloped:xades"
              }
             ]
            }
    ] #end of “UBLExtensions” object
   } #end of “Invoice” objects key-value array
  ]  #end of “Invoice” object
} #end of json file

# Kindly replace all the <variable>  with a correct value and
# remove all the remark (#remark detail) before paste the signing script into the minified payload.
# Minify the signed payload again to minify all the signing objects/elements above.
# Encode the sign payload data with Base64 format before send it to LHDNM
```

POWERSHELL SCRIPT (COMPILATION)

```
$InformationPreference = "SilentlyContinue"
function Minify-Json {
	param (
		[Parameter(Mandatory=$true)]
		[string]$JsonString
	)
	# Validate minified JSON content
	if ($JsonString -eq $null -or $JsonString -eq '') {
	Write-Host "Invalid JSON content."
	exit
	}

	# Remove all whitespace between elements, preserving whitespace within string values
	$minifiedJson = [System.Text.RegularExpressions.Regex]::Replace($JsonString, '("(?:\\.|[^"\\])*")|\s+', {
		param($match)
		if ($match.Groups[1].Success) {
			return $match.Groups[1].Value
		} else {
			return ''
		}
	})

	return $minifiedJson
}
cls

# Load the certificate from the file store or key store
$chkPath = Read-Host -Prompt "Enter the path of your certificate file (*.pfx/*.p12)"
```

```
# Read Certificate Password
$password = Read-Host -AsSecureString -Prompt "Enter the password for your certificate file"

# Read the content of a file
$filePath = Read-Host -Prompt "Enter the path and filename of the payload file"

# Set Output file
$fileOutPath = Read-Host -Prompt "Enter the path and filename of the output file"
$outputPath = Read-Host -Prompt "Enter the path and filename of the base64 encoded file"

# Set API Code Number
$CodeNumber = Read-Host -Prompt "Enter the API Code Number"

cls
# Loop until a valid file is found or 'X' is entered
while (-not (Test-Path $filePath)) {
   # Prompt the user to enter the path of the file or 'X' to exit
   cls
   $filePath = Read-Host -Prompt "File not found. Enter the path of the payload file or 'X' to exit"

   # Check if the file exists
   if (-not (Test-Path $filePath)) {
         # Check if the user wants to exit
         if ($filePath -eq "X") {
               cls
               exit
         }
         if ($filePath -eq "x") {
               cls
         exit
```

```
        }
    cls
    Write-Host "File not found: $filePath"
  }
}

# Read the content of the file
$document = Get-Content -Path $filePath -Raw

# Load necessary assemblies
Add-Type -AssemblyName System.Security

# Minify the JSON string
$minifiedJson = Minify-Json -JsonString $document

# Write the minified json data to the file
$minifiedJson | Out-File -FilePath $fileOutPath -Encoding UTF8

# Create a new instance of SHA256
$sha256 = [System.Security.Cryptography.SHA256]::Create()

# Compute the hash of the document
$hash = $sha256.ComputeHash([System.Text.Encoding]::UTF8.GetBytes($minifiedJson))

# Convert the hash to a Base64 string
$DocDigest = [System.Convert]::ToBase64String($hash)

# Load necessary assemblies
Add-Type -AssemblyName System.Security
```

```
cls

# Loop until a valid file is found or 'X' is entered
while (-not (Test-Path $chkPath)) {
   # Prompt the user to enter the path of the file or 'X' to exit
   cls
   $chkPath = Read-Host -Prompt "File not found. Enter the path of your certificate file (*.pfx/*.p12) or 'X' to exit"

   # Check if the file exists
   if (-not (Test-Path $chkPath)) {
           # Check if the user wants to exit
           if ($chkPath -eq "X") {
                   cls
           break
           }
           if ($chkPath -eq "x") {
                   cls
           break
           }
       cls
            Write-Host "File not found: $chkPath"
    }
} #while loop

if (-not (Test-Path $chkPath)) {
           cls
           Write-Host "File not found: $chkPath"
} else {
           $certPath = $chkPath
```

```
        # Remove the variable to free up memory
        Remove-Variable -Name $chkPath

        cls

        try { $cert = New-Object System.Security.Cryptography.X509Certificates.X509Certificate2($certPath, $password)} catch { Write-Host "Failed to load
certificate: $_" exit }

        # Remove the variable to free up memory
        Remove-Variable -Name $certPath

        # Extract the private key using RSACryptoServiceProvider
        $privateKeyProvider = [System.Security.Cryptography.X509Certificates.RSACertificateExtensions]::GetRSAPrivateKey($cert)

        # Create RSA signature formatter
        $rsaFormatter = New-Object System.Security.Cryptography.RSAPKCS1SignatureFormatter $privateKeyProvider
        $rsaFormatter.SetHashAlgorithm("SHA256")

        # Sign the hash
        $signature = $rsaFormatter.CreateSignature($hash)

        # Convert the signature to Base64
        $signatureBase64 = [Convert]::ToBase64String($signature)
}
cls

# Compute the hash of the certificate using SHA-256
$sha256 = [System.Security.Cryptography.SHA256]::Create()
$certHash = $sha256.ComputeHash($cert.RawData)
```

```
# Convert the hash to a Base64 encoded string
$certDigest = [Convert]::ToBase64String($certHash)

# Ensure data sanitization
$issuerName = [System.Security.SecurityElement]::Escape($cert.IssuerName.Name)

# Retrieve the serial number
$serialNumber = $cert.SerialNumber

# Convert the hexadecimal string to a BigInteger
$CertSerialNumber = [System.Numerics.BigInteger]::Parse($serialNumber, [System.Globalization.NumberStyles]::HexNumber)

# Extract the raw data of the certificate
$rawData = [Convert]::ToBase64String($cert.RawData)

#Set SigningTime to current timestamp based on UTC format
$utcTimestamp = Get-Date -Format "yyyy-MM-ddTHH:mm:ssZ"

cls
# Assign the XML string to a variable using a here-string
$jsonString = @"
{"Target":"signature","SignedProperties":[{"Id":"id-xades-signed-
props","SignedSignatureProperties":[{"SigningTime":[{"_":"$utcTimestamp"}],"SigningCertificate":[{"Cert":[{"CertDigest":[{"DigestMethod":[{"_":"","Algorithm":"htt
p://www.w3.org/2001/04/xmlenc#sha256"}],"DigestValue":[{"_":"$certDigest"}]}],"IssuerSerial":[{"X509IssuerName":[{"_":"$issuerName"}],"X509SerialNumber":[{"
_":"$CertSerialNumber"}]}]}]}]}]}]}
"@

# Minify the JSON string
$signedprops = Minify-Json -JsonString $jsonString
```

```
# Compute the hash of the signedprops' UTF-8 bytes
$signedpropshash = $sha256.ComputeHash([System.Text.Encoding]::UTF8.GetBytes($signedprops))

# Convert the hash to a Base64 encoded string
$signedpropsdigest = [Convert]::ToBase64String($signedpropshash)

# Retrieve the subject name from the certificate
$subjectName = $cert.SubjectName.Name

$signString = @"
}],"UBLExtensions":[{"UBLExtension":[{"ExtensionURI":[{"_":"urn:oasis:names:specification:ubl:dsig:enveloped:xades"}],"ExtensionContent":[{"UBLDocumentSignatures":[{"SignatureInformation":[{"ID":[{"_":"urn:oasis:names:specification:ubl:signature:1"}],"ReferencedSignatureID":[{"_":"urn:oasis:names:specification:ubl:signature:Invoice"}],"Signature":[{"Id":"signature","Object":[{"QualifyingProperties":[{"Target":"signature","SignedProperties":[{"Id":"id-xades-signed-props","SignedSignatureProperties":[{"SigningTime":[{"_":"$utcTimestamp"}],"SigningCertificate":[{"Cert":[{"CertDigest":[{"DigestMethod":[{"_":"","Algorithm":"http://www.w3.org/2001/04/xmlenc#sha256"}],"DigestValue":[{"_":"$certDigest"}]}],"IssuerSerial":[{"X509IssuerName":[{"_":"$issuerName"}],"X509SerialNumber":[{"_":"$CertSerialNumber"}]}]}]}]}]}]}]}],"KeyInfo":[{"X509Data":[{"X509Certificate":[{"_":"$rawData"}],"X509SubjectName":[{"_":"$subjectName"}],"X509IssuerSerial":[{"X509IssuerName":[{"_":"$issuerName"}],"X509SerialNumber":[{"_":"$CertSerialNumber"}]}]}]}],"SignatureValue":[{"_":"$signatureBase64"}],"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":[{"_":"$signedpropsdigest"}]},{"Type":"","URI":"","DigestMethod":[{"_":"","Algorithm":"http://www.w3.org/2001/04/xmlenc#sha256"}],"DigestValue":[{"_":"$DocDigest"}]}]}]}]}]}]}]}],"Signature":[{"ID":[{"_":"urn:oasis:names:specification:ubl:signature:Invoice"}],"SignatureMethod":[{"_":"urn:oasis:names:specification:ubl:dsig:enveloped:xades"}]}]}]}
"@

# Remove the variable to free up memory
Remove-Variable -Name $cert

# Read the content of JsonFile1
$jsonFile1Content = Get-Content -Path $fileOutPath -Raw
```

```
# Remove the last 3 characters from the file content
$modifiedContent = $jsonFile1Content.Substring(0, $jsonFile1Content.Length - 7)

# Combine the data from JsonFile1 and JsonFile2
$combinedData = $modifiedContent + $signString

# Minify the JSON string
$finaldata = Minify-Json -JsonString $combinedData

# Write the updated content back to the file
$finaldata | Out-File -FilePath $fileOutPath -Encoding UTF8

# Encode the content using Base64
$encodedContent = [System.Convert]::ToBase64String([System.Text.Encoding]::UTF8.GetBytes($finaldata))

# Write the encoded content to a new file
$encodedContent | Out-File -FilePath $outputPath -Encoding UTF8

# Create a SHA-256 hash object
$sha256 = [System.Security.Cryptography.SHA256]::Create()

# Compute the hash of the document
$hashString = $sha256.ComputeHash([System.Text.Encoding]::UTF8.GetBytes($finaldata))

# Convert the hash bytes to a hexadecimal string
$Digest = $hashString | ForEach-Object { $_.ToString("x2") }

# Output the encoded content
$RecHash = $Digest -join ""
```

```
$JSONApi = @"
{"documents":[{"format":"JSON","documentHash":"$RecHash","codeNumber":"$CodeNumber","document":"$encodedContent"}]}
"@

cls
$JSONApi

# Remove the variable to free up memory
Remove-Variable -Name $JSONApi
```

DISCLAIMER

The sample PowerShell script provided is for illustrative purposes only. It is important to note that the script may not be suitable for all environments or specific requirements. We strongly recommend reviewing and modifying the script as per your organization's policies, security guidelines, and best practices.

Please ensure that you understand the potential risks associated with running any script, and take appropriate precautions, such as testing in a controlled environment before deploying it in a production environment.

LHDNM cannot be held responsible for any damages, loss of data, or other consequences that may arise from the use or misuse of the provided script. Use it at your own discretion and responsibility.