



SWIFT

Electronic Bank Account Management - EBAM

EBAM and Digital Signature

This guide provides an overview of how to use a digital signature in the EBAM solution to sign the XML messages and the potential attachments.

V1.0

January 2010

Table of Contents

1	Introduction	3
1.1	Purpose and Scope	3
1.2	Digital signature - Business requirements	3
1.2.1	Signature card	3
1.2.2	Attachments under electronic format.....	4
2	Recommendations	5
2.1	Transporting the "Signature Card"	5
2.2	Signing an XML message	5
2.2.1	Signing parameters	5
2.2.2	Whitespace handling	6
2.2.3	Example.....	6
2.2.4	Verifying the Signature	8
2.3	Signing an XML message and attached documents	8
2.3.1	Example of signing attachments.....	8
2.3.2	Availability of the documents signed when verifying the Signature.....	9
2.4	Signing and transporting a signed document	10
2.4.1	Signature of a file.....	10
2.4.2	Signed file format.....	10
3	Legal notices	12

1 Introduction

1.1 Purpose and Scope

The EBAM Solution captures within ISO 20022 XML messages some key information that is today exchanged on paper between banks and corporates when they manage bank accounts.

The process steps in scope include:

- Bank account opening
- Bank account maintenance
- Bank account closing
- Reporting of bank account features (e.g. ownership, mandates)

The main benefits for both banks and corporates are a reduced total elapse time, an increased corporate customer satisfaction, reduced cost and an improved STP and traceability.

The EBAM solution only addresses bank account management activities for customers that already have relationship with a bank or for a new legal entity of this customer.

1.2 Digital signature - Business requirements

1.2.1 Signature card

Managing bank accounts involves the exchange of information on individuals authorised to perform financial transactions from the designated bank account(s). This information – commonly known as bank mandates - includes:

- Name
- Function
- Financial limits up to which the individual is authorised to perform transactions (e.g. treasury payment up to KEUR 10,000)
- The wet signature of the individual
- Currently, this information is captured on paper on a “signature card” (“carton de signature”). As the below illustration indicates, wet signatures of authorised individuals are documented as well as the signature of the main mandator. The main mandator signature is mandatory as it officially delegates authority to the authorised individuals registered on the signature card.

Signature card

In the context of the EBAM solution, the signature card is no longer needed because the related info may be captured within the XML message. Therefore, two business needs have been identified:

1. **transport signatures** of authorised individuals (e.g. CFO, Treasurer, GL Accountant)
2. **digitally sign the XML message** containing the information on authorised individuals. This signature will replace today's wet signature of the main mandator. It is to be noted that this signature is at the individual – vs the corporate – level.

1.2.2 Attachments under electronic format

Although the developed XML messages capture the core of the information needed to manage bank accounts, it is impossible to replace the entire set of paper documents that is used today. Main reasons for that are:

- Lack of harmonisation between/within banking groups
- Lack of harmonisation between countries/regions
- Legal constraints

As a result, banks and corporates may decide in the future to complement the XML messages with these paper documents under electronic format, in which case we have a third business need:

3. some of these **attached documents will have to be digitally signed** to comply with security requirements.

Electronic formats may be pdf, doc, jpg, xls,... The SWIFT Solution also addresses that need.

2 Recommendations

The below section describes the recommended options that have been identified to address the above business requirements. They are based on consultation work performed end 2008 with banks, corporates and application vendors.

SWIFT is setting up a solution to cater for multibank personal digital signature and identity which is compliant with the below recommendations.

2.1 Transporting the "Signature Card"

The "Signature Card" provides a way to authenticate who signs. For digital signatures using a Public Key Infrastructure, the X509 certificate issued by a Certificate Authority is used to authenticate what is signed by whom. That means that with digital signatures the X509 certificate is used to represent the wet signature on the "Signature Card".

The X509v3 certificate is added in the appropriate element using the built in xs:base64Binary schema datatype, where the xs namespace is defined as `xmlns:xs="http://www.w3.org/2001/XMLSchema"`.

There are no constraining facets such as `maxLength`. The size of the certificate depends on the attributes that are registered and the size of the keys used. The size is typically less than 2000 characters.

The certificate is "known" by the receiver, so that the application can check whether a given certificate identifies the person described within other fields of the message.

As an example, the acmt.016.001.01 Account Mandate Maintenance Request contains the certificate within an element.

```
<Doc:Document xmlns:Doc="urn:iso:....">
  <Doc:acmt.016.001.01>
    ...
    <Doc:Cert>MII4diDF5FE52d5...</Doc:Cert>
    ...
  </Doc:acmt.016.001.01>
</Doc:Document>
```

2.2 Signing an XML message

The recommended option for digitally signing XML messages is to use the existing standard **DSIG** and to embed the signature within the XML message.

The DSIG standard is documented in the *XML - Signature Syntax and Processing (Second Edition) - W3C Recommendation*. This document is public available on <http://www.w3.org/TR/xmlsig-core/>. The recommendation defines the syntax and processing rules of XML digital signatures. Note that the DSIG standard is in a second edition, but that the features used in EBAM were already available in the first edition that was finalised in 2002.

2.2.1 Signing parameters

Not all features of the DSIG standard are needed. DSIG specifies a number of algorithms and is extensible. The EBAM will use the following setup and algorithms for signing the XML messages:

- What is signed is the ISO 20022 Document element without the embedded Signature element.
- Since the Signature only covers data that is part of the message that includes the Signature, the Manifest construct is not required. In this case the SignedInfo includes the reference(s) to the data to be signed.
- The canonicalization algorithm is the Canonical XML 1.0 omitting comments.
- The recommended digesting algorithm is SHA256
- The recommended signing algorithm is RSA-SHA256. Allowed is RSA-SHA1
- The KeyInfo contains the certificates required to validate the SignatureValue element. These certificates can be compared with the deposited certificates in order to identify the signer of the message. By adding the certificate within the KeyInfo, the receiving application can use the certificate as such to find back in its database who actually signed it and whether this person had the entitlement to sign this message.
- It is recommended that the SignatureValue contains the CMS (Cryptographic Message Syntax) without certificates as detailed in RFC3852. This avoids to send the certificates twice. If for some reason the CMS still contains certificates, then the certificates within the CMS must match the ones in the KeyInfo.

2.2.2 Whitespace handling

The ISO 20022 account management messages do not contain mixed content that is signed. That means that whitespace between elements is meaningless. In order to exclude this whitespace from the data signed, it should first be removed. To make this explicit in the Signature, a transformation algorithm can be specified.

The algorithm uses an XSLT stylesheet Transform algorithm with as identifier for the algorithm <http://www.w3.org/TR/1999/REC-xslt-19991116>. The stylesheet element is simply:

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:output encoding="UTF-8" />
  <xsl:strip-space elements="*" />
  <xsl:template match="@*|node()">
    <xsl:copy>
      <xsl:apply-templates select="@*|node()" />
    </xsl:copy>
  </xsl:template>
</xsl:stylesheet>
```

The processing is to strip whitespace between elements. It is not required to use an XSLT processor to perform this job.

The usage of the whitespace stripping can be avoided by mandating that no whitespace is allowed between elements within the messages signed.

2.2.3 Example

We take as example the Account Opening Request message acmt.007.001.01. This message contains somewhere the Signature that covers the message signed by the creator of the account.

The ISO 20022 message looks like

```
<Doc:Document xmlns:Doc="urn:iso:....">
  <Doc:acmt.007.001.01>
```

```

...
  <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
    ...
  </ds:Signature>
...
</Doc:acmt.007.001.01>
</Doc:Document>

```

The Signature is signing the complete ISO 20022 Document except the Signature element itself.

The Signature element looks like:

```

<ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
  <ds:SignedInfo>
    <ds:CanonicalizationMethod
      ds:Algorithm="http://www.w3.org/TR/2001/12/REC-xml-c14n-20010315"/>
    <ds:SignatureMethod
      ds:Algorithm="http://www.w3.org/2001/04/xmldsig-more#rsa-sha256"/>
    <ds:Reference URI="">
      <ds:Transforms>
        <ds:Transform
          ds:Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature"/>
        <ds:Transform
          ds:Algorithm="http://www.w3.org/TR/1999/REC-xslt-19991116">
          <xsl:stylesheet version="1.0"
            xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
            <xsl:output encoding="UTF-8" />
            <xsl:strip-space elements="*" />
            <xsl:template match="@*|node()">
              <xsl:copy>
                <xsl:apply-templates select="@*|node()" />
              </xsl:copy>
            </xsl:template>
          </xsl:stylesheet>
        </ds:Transform>
        <ds:Transform
          ds:Algorithm="http://www.w3.org/TR/2001/12/REC-xml-c14n-20010315"/>
      </ds:Transforms>
      <ds:DigestMethod ds:Algorithm="http://www.w3.org/2001/04/xmenc#sha256"/>
      <ds:DigestValue>fji4f1v5dt581sd55dK...</ds:DigestValue>
    </ds:Reference>
  </ds:SignedInfo>
  <ds:SignatureValue>...CMS without certificates...</ds:SignatureValue>
  <ds:KeyInfo>
    <ds:X509Data>
      <ds:X509Certificate>
        MII4ybroe8...
      </ds:X509Certificate>
      <ds:X509Certificate>
        MIIqvnippi...
      </ds:X509Certificate>
    </ds:X509Data>
  </ds:KeyInfo>
</ds:Signature>

```

The SignedInfo element contains one Reference element. This Reference element is pointing to the XML document that contains the Signature. As specified, this is the ISO 20022 message itself. The ISO 20022 Document is canonicalized using the same canonicalization algorithm and the Signature element is removed from the digested content. The result of the SHA256 digest is put in the DigestValue as a 44 byte base64 encoded string.

2.2.4 Verifying the Signature

The verification of the Signature will check that the SignatureValue matches with the SignedInfo and that the DigestValue within the SignedInfo matches with message data after applying the canonicalization algorithm and the Signature element is removed.

2.3 Signing an XML message and attached documents

Signing attachments is done by signing a digest that corresponds to the content of the attachment when it is stored as a file on a disk.

The attached document has a file name. This file name can be used to identify the file.

The signing of the attachments is always done in the same way using the Manifest element that is part of the DSIG standard.

The Manifest is put within the Signature itself.

The attachments can be signed together with the message, when the Signature in the message also covers attachments. Another case is where the attachment and its signature are shipped together as a file. Both cases are covered in the sub-sections below.

2.3.1 Example of signing attachments

Assume that the Account Opening Request requires that the signer also signs two documents that contain contractual rules about the usage of the account. Assume these two documents are available on both sides.

The file "Terms and Conditions.pdf" and the file "ServiceDescription.pdf" are to be signed.

The application creating the message also creates the Signature. The algorithm to digest the file is identified within the Reference in the Manifest.

```
<ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
  <ds:SignedInfo>
    <ds:CanonicalizationMethod
      ds:Algorithm="http://www.w3.org/TR/2001/12/REC-xml-c14n-20010315"/>
    <ds:SignatureMethod
      ds:Algorithm="http://www.w3.org/2001/04/xmldsig-more#rsa-sha256"/>
    <ds:Reference URI="#Manifest">
      <ds:Transforms>
        <ds:Transform
          ds:Algorithm="http://www.w3.org/TR/2001/12/REC-xml-c14n-20010315"/>
        <ds:Transform
          ds:Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature"/>
      </ds:Transforms>
      <ds:DigestMethod ds:Algorithm="http://www.w3.org/2001/04/xmenc#sha256"/>
      <ds:DigestValue>vg4kjfgJKF545RTR5...</ds:DigestValue>
    </ds:Reference>
  </ds:SignedInfo>
  <ds:SignatureValue>...CMS without certificates...</ds:SignatureValue>
  <ds:KeyInfo>
    <ds:X509Data>
      <ds:X509Certificate>
        MII4ybroe8...
      </ds:X509Certificate>
      <ds:X509Certificate>
        MIIqvnippi...
      </ds:X509Certificate>
    </ds:X509Data>
  </ds:KeyInfo>
</ds:Signature>
```

```

</ds:KeyInfo>
<ds:Object>
  <ds:Manifest Id="Manifest">
    <ds:Reference URI="">
      <ds:Transforms>
        <ds:Transform
          ds:Algorithm="http://www.w3.org/TR/2001/12/REC-xml-c14n-20010315"/>
        <ds:Transform
          ds:Algorithm="http://www.w3.org/2000/09/xmlsig#enveloped-signature"/>
      </ds:Transforms>
      <ds:DigestMethod ds:Algorithm="http://www.w3.org/2001/04/xmlenc#sha256"/>
      <ds:DigestValue>fji4f1v5dt581sd55dK...</ds:DigestValue>
    </ds:Reference>
    <ds:Reference URI="Terms%20and%20Conditions.pdf">
      <ds:DigestMethod ds:Algorithm="http://www.w3.org/2001/04/xmlenc#sha256"/>
      <ds:DigestValue>fji4f1v5dt581sd55dK...</ds:DigestValue>
    </ds:Reference>
    <ds:Reference URI="ServiceDescription.pdf">
      <ds:DigestMethod ds:Algorithm="http://www.w3.org/2001/04/xmlenc#sha256"/>
      <ds:DigestValue>fg555nlylfg5h57j6UK...</ds:DigestValue>
    </ds:Reference>
  </ds:Manifest>
</ds:Object>
</ds:Signature>

```

Some remarks about the example.

The first Reference in the Manifest is the ISO 20022 message itself. Exactly the same process is executed as in the previous section. The DigestValue of this Reference must be verified against the message itself.

The Manifest contains a `Reference` element for each document that is to be signed. The identification of the document is done through a relative URI. The value of the URI can be simply the name of the document, in our example it is the file name. The digest is calculated on the file content. The name of the document in the URI and the value of the digest identify uniquely the document that is signed.

The processing of the Signature verification is done in two steps.

The first step is to verify the SignatureValue. This verification will validate that the SignedInfo is correct and that the Digest within the Reference element in the SignedInfo is correct as well. In our case it validates that the Manifest element is not changed.

The second step is to validate the Reference elements in the Manifest itself. As already indicated, the Reference of the message must be validated. For the Reference elements related to the attachments, this can be done by comparing the DigestValue against the expected value or by calculating a Digest on the file that is covered by the signing and to check that it is the same as put in the Manifest.

2.3.2 Availability of the documents signed when verifying the Signature

The documents that are signed can be sent in the same message as the one containing the Signature. It is also possible that the documents are already available, for instance because these are exchanged prior to the message. How the documents are exchanged is less relevant for the Signature processing.

The availability of the documents is not required for the first step in the verification of the Signature itself, as discussed in previous section. It remains important to verify what documents are signed. For documents that do not change, this can be done by comparing the DigestValue and file name with the expected DigestValue. When it concerns documents that are changed by the corporate and afterwards forwarded to the bank, at

least two steps are to be executed. A first step requires to calculate a digest on the file and to compare it with the digest signed. A second step is to check the content of the document signed to determine if it is acceptable.

2.4 Signing and transporting a signed document

The Signature is similar to the previous example when the file is signed and the signed file has to be transported.

There are several ways to create a new file that contains the Signature and the signed file.

This section describes the differences in the Signature and proposes the format of the signed file.

2.4.1 Signature of a file

The Signature of the file is very similar to any other Signature used in EBAM.

```
<ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
  <ds:SignedInfo>
    <ds:CanonicalizationMethod
      ds:Algorithm="http://www.w3.org/TR/2001/12/REC-xml-c14n-20010315"/>
    <ds:SignatureMethod
      ds:Algorithm="http://www.w3.org/2001/04/xmldsig-more#rsa-sha256"/>
    <ds:Reference URI="#Manifest">
      <ds:Transforms>
        <ds:Transform
          ds:Algorithm="http://www.w3.org/TR/2001/12/REC-xml-c14n-20010315"/>
        <ds:Transform
          ds:Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature"/>
      </ds:Transforms>
      <ds:DigestMethod ds:Algorithm="http://www.w3.org/2001/04/xmenc#sha256"/>
      <ds:DigestValue>vg4kjfgJKF545RTR5...</ds:DigestValue>
    </ds:Reference>
  </ds:SignedInfo>
  <ds:SignatureValue>...CMS without certificates...</ds:SignatureValue>
  <ds:KeyInfo>
    <ds:X509Data>
      <ds:X509Certificate>
        MII4ybroe8...
      <ds:X509Certificate>
        MIIqvnippi...
      </ds:X509Certificate>
    </ds:X509Data>
  </ds:KeyInfo>
  <ds:Object>
    <ds:Manifest Id="Manifest">
      <ds:Reference URI="Terms%20and%20Conditions.pdf">
        <ds:DigestMethod ds:Algorithm="http://www.w3.org/2001/04/xmenc#sha256"/>
        <ds:DigestValue>fji4flv5dt581sd55dK...</ds:DigestValue>
      </ds:Reference>
    </ds:Manifest>
  </ds:Object>
</ds:Signature>
```

The difference is in the Manifest. This example shows how one file is signed. When verifying the signature, the SignedInfo is verified to be correct and the DigestValue within the Reference of the Manifest is compared with the digest of the file.

2.4.2 Signed file format

The signed file format is using MIME to format the file. The MIME types used depend on the type of the file. For instance for PDF document the MIME type is well known.

The signed file is a MIME multipart/mixed message.

The following is an example where the MIME encoding is only used for creating a signed file, but where the knowledge of what is signed is application specific, meaning that the URI in the Manifest is interpreted as referring to the first part of the multipart message, where the transfer encoding is reversed, so that the raw file data of the PDF is used for calculating the digest of the file.

```
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary="_999_888_ielsjh080"

--_999_888_ielsjh080
Content-Type: application/pdf;
 name="Terms and Conditions.pdf"
Content-Transfer-Encoding: base64
Content-Disposition: inline;
 filename="Terms and Conditions.pdf"

JVBERi0xLjQNJeLjz9MNCjE1NjAgMcbvYmoNPDwgDS9MaW5lYXJpemVkIDEgDS9PIDE1NjMg
DS9IIFsgMtc1NTUgNTQ4NCBdIA0vTCAxODg3NTQzIA0vRSAxMTMwNzcgDS9OIDI5MyANLlQg
...

--_999_888_ielsjh080
Content-Type: text/xml; charset="utf-8"

<?xml version="1.0" encoding="utf-8" ?>
<ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
  <ds:SignedInfo>
    <ds:CanonicalizationMethod
      ds:Algorithm="http://www.w3.org/TR/2001/12/REC-xml-c14n-20010315"/>
    <ds:SignatureMethod
      ds:Algorithm="http://www.w3.org/2001/04/xmldsig-more#rsa-sha256"/>
    <ds:Reference URI="#Manifest">
      <ds:Transforms>
        <ds:Transform
          ds:Algorithm="http://www.w3.org/TR/2001/12/REC-xml-c14n-20010315"/>
        <ds:Transform
          ds:Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature"/>
      </ds:Transforms>
      <ds:DigestMethod ds:Algorithm="http://www.w3.org/2001/04/xmlenc#sha256"/>
      <ds:DigestValue>vg4kjfgJKF545RTR5...</ds:DigestValue>
    </ds:Reference>
  </ds:SignedInfo>
  <ds:SignatureValue>...CMS without certificates...</ds:SignatureValue>
  <ds:KeyInfo>
    <ds:X509Data>
      <ds:X509Certificate>
        MII4ybroe8...
      </ds:X509Certificate>
      <ds:X509Certificate>
        MIIqvnippi...
      </ds:X509Certificate>
    </ds:X509Data>
  </ds:KeyInfo>
  <ds:Object>
    <ds:Manifest Id="Manifest">
      <ds:Reference URI="Terms%20and%20Conditions.pdf">
        <ds:DigestMethod ds:Algorithm="http://www.w3.org/2001/04/xmlenc#sha256"/>
        <ds:DigestValue>fji4flv5dt581sd55dK...</ds:DigestValue>
      </ds:Reference>
    </ds:Manifest>
  </ds:Object>
</ds:Signature>

--_999_888_ielsjh080--
```

3 Legal notices

Copyright

SWIFT © 2010. All rights reserved.

You may copy this publication within your organisation. Any such copy must include these legal notices.

Confidentiality

This publication may contain SWIFT or third-party confidential information. Do not disclose this publication outside your organisation without the prior written consent of SWIFT.

Disclaimer

SWIFT supplies this publication for information purposes only. The information in this publication may change from time to time. You must always refer to the latest available version on www.swift.com.

Translations

The English version of SWIFT documentation is the only official version.

Trademarks

SWIFT is the trade name of S.W.I.F.T. SCRL. The following are registered trademarks of SWIFT: SWIFT, the SWIFT logo, Sibos, SWIFTNet, SWIFTReady, and Accord. Other product, service, or company names in this publication are trade names, trademarks, or registered trademarks of their respective owners.
