a. 

Specyfikacja interfejsów usług Jednolitego Pliku Kontrolnego

Ministerstwo Finansów

Departament Informatyzacji

23 May 2016

Version 1.3

Spis treści

1 Przygotowanie danych JPK

1.1 Przygotowanie dokumentów JPK

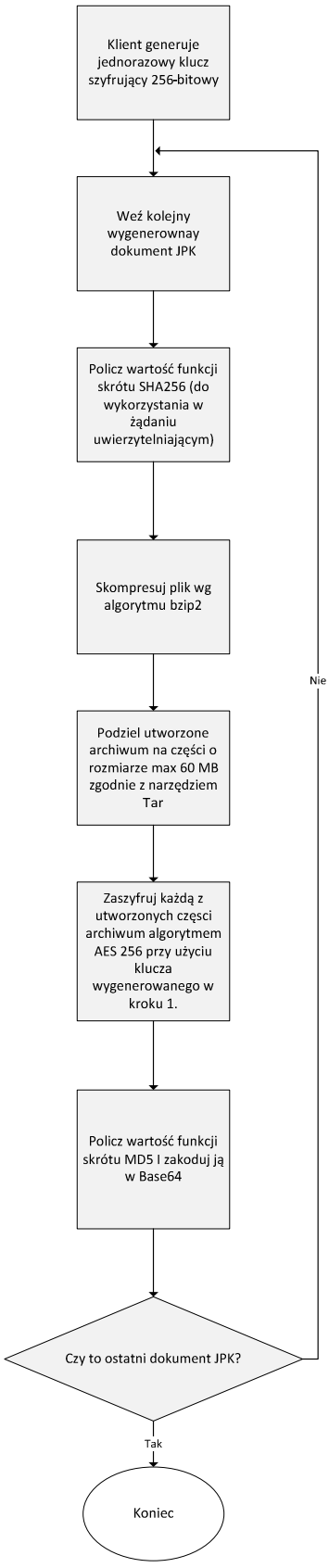
Dane JPK przygotowywane będą po stronie klienta (np. systemu ERP) w formie plików XML zgodnych ze schematem XSD opublikowanym przez Ministerstwo Finansów:

http://www.mf.gov.pl/kontrola-skarbowa/dzialalnosc/jednolity-plik-kontrolny/-/asset_publisher/2NoO/content/struktury-jpk

Każdy z dokumentów opisanych właściwym schematem ma stanowić osobny plik XML.

Wygenerowany plik XML powinien być zakodowany w UTF-8.

Dla każdego z plików JPK zostaną wykonane następujące operacje:


Klient generuje jednorazowy klucz szyfrujący 256-bitowy
Weź kolejny wygenerownay dokument JPK
Policz wartość funkcji skrótu SHA256 (do wykorzystania w żądaniu uwierzytelniającym)
Skompresuj plik wg algorytmu bzip2
Podziel utworzone archiwum na części o rozmiarze max 60 MB zgodnie z narzędziem Tar
Zaszyfruj każdą z utworzonych częsci archiwum algorytmem AES 256 przy użyciu klucza wygenerowanego w kroku 1.
Policz wartość funkcji skrótu MD5 I zakoduj ją w Base64
Czy to ostatni dokument JPK?
Nie
Tak
Koniec

Rysunek 1 Schemat blokowy kroków przygotowywania do wysyłki danych JPK

1.1.1 Kompresja danych JPK

Wygenerowany dokument JPK zostanie skompresowany algorytmem bzip2 oraz dzielony na części o wielkości nie przekraczającej 60 MB (w praktyce należy spodziewać się wysokiego stopnia kompresji, dochodzącej nawet do 1:50, co sprawia, że scenariusz w którym będziemy mieli więcej niż jedną część, będzie stosunkowo rzadki)

Proponowana metoda kompresji to algorytm bzip2, natomiast dzielenie na części jest zgodne z programem Tar. Takie podejście z jednej strony zapewnia wykorzystanie znanych i powszechnie stosowanych standardów o powszechnie występujących implementacjach dla różnych platform, z drugiej – efektywność, w szczególności operacji kompresji i prostotę API dla tych operacji. Są one dostępne w narzędziu Tar oraz poprzez bibliotekę libtar.

Dokumentacja programu Tar:

http://www.gnu.org/software/tar/manual/tar.html

1.1.2 Szyfrowanie danych JPK

Skompresowane pliki będą szyfrowane. Do szyfrowania plików wykorzystany będzie algorytm AES256, z kluczem szyfrującym wygenerowanym po stronie klienta.

Algorytm procesu szyfrowania będzie wyglądał następująco:

- Klient generuje losowy, 256 bitowy klucz
- Wygenerowanym kluczem szyfrowane są wszystkie części skompresowanego archiwum (zgodnie z pkt. 1.1) . Algorytmem szyfrującym jest AES256.
- Klucz szyfrujący jest szyfrowany z wykorzystaniem algorytmu asymetrycznego RSA, z wykorzystaniem kryptografii (klucz publiczny) dostarczonej podatnikowi przez Ministerstwo
- Tak zaszyfrowany klucz jest dołączany do pliku metadanych, zgodnie z przedstawionym dalej opisem tego pliku.

1.2 Przygotowanie metadanych uwierzytelniających

Po przygotowaniu zasadniczych dokumentów zgodnych ze schematem Jednolitego Pliku Kontrolnego (JPK), klient, w celu wysłania danych, musi przygotować dane uwierzytelniające,

mające postać odpowiedniego XML, przesłane w metodzie InitUploadSigned (opisanej w następnym rozdziale).

Plik metadanych musi być podpisany cyfrowo zgodnie z algorytmem XAdES Basic Electronic Signature, w skrócie XAdES-BES w wersji **Enveloped** (podpis jako dodatkowy element ds:Signature w oryginalnym XML) lub **Enveloping** (oryginalny dokument zawarty jako element w podpisanej strukturze).

Funkcją skrótu wykorzystywaną w podpisie powinna być RSA-SHA256 lub RSA-SHA1

Przykład metadanych uwierzytelniających można znaleźć w p. 2.2.1, gdzie omówiona jest metoda InitUploadSigned, przyjmująca metadane uwierzytelniające.

2 Specyfikacja interfejsu przyjmującego dokumenty JPK dla klientów

2.1 Wstęp

Mechanizm przyjmowania dokumentów oparty jest o usługi REST, działające w oparciu o protokół https. Takie podejście zapewnia zarówno efektywność i sprawność interfejsu (choćby w porównaniu np. do interfejsów typu SOAP), jak i łatwość integracji z rozwiązaniami ERP i innymi, napisanymi w różnych technologiach.

2.2 Opis interfejsu

Zasadnicza część interfejsu dla klientów ERP składa się z następujących metod:

- InitUploadSigned
- Put Blob
- FinishUpload
- Status

Poniżej znajduje się szczegółowy opis działania tych metod.

2.2.1 InitUploadSigned

Metoda inicjująca sesję klienta. Jej wywołanie jest warunkiem koniecznym do przesłania danych metodą Put Blob usługi Azure.

Nazwa	InitUploadSigned
Typ metody	Post
Typ przesyłanej zawartości	application/xml
Typ zwracanej zawartości	application/json
Maksymalny rozmiar żądania	100KB

Opis XML stanowiącego zawartość (body) żądania.

Nazwa	Opis	Typ	Walidacja
InitUpload	Metadane dla metody InitUpload	Obiekt	Wymagany
DocumentType	Nazwa typu przesyłanego dokumentu.	String	Wymagany. Dopuszczalne wartości [JPK]
Version	Wersja REST API do której adresowane jest zapytanie	String	Wymagany. Format [0-9][0-9].[0-9][0-9].[0-9][0-9].[0-9]{8} , np. 01.01.01.20160519.
EncryptionKey	Klucz symetryczny zaszyfrowany algorytmem asymetrycznym (RSA)	String	Wymagany
EncryptionKey.algorithm	Algorytm, którym zaszyfrowany jest klucz symetryczny	String – dopuszczalne wartości: **RSA**	Wymagany
EncryptionKey.transformation	Metoda transformacji wykorzystywana w szyfrowaniu	String – dopuszczalne wartości: **RSA/ECB/NoPadding**	Wymagany
EncryptionKey.encoding	Algorytm kodowania	String – dopuszczalne	Wymagany

	wartości klucza	wartości: **Base64**	
DocumentList	Lista przesłanych dokumentów	Lista obiektów typu Document	Wymagany. Lista musi zawierać przynajmniej jeden dokument.
Document	Metadane przesyłanego dokumentu	Obiekt	Wymagany
FormCode	KodFormularza zawarty w nagłówku pliku XML	Brak	Wymagany
FormCode.systemCode	Atrybut kodSystemowy elementu KodFormularza z pliku XML	String	Wymagany
Document.schemaVersion	Atrybut wersjaSchemy elementu KodFormularza z pliku XML	String	Wymagany
HashValue	Skrót całego dokumentu	String	Wymagany
HashValue.algorithm	Nazwa algorytmu funkcji skrótu,	String – dopuszczalne wartości: **SHA-256**	Wymagany
HashValue.encoding	Algorytm	String –	Wymagany

	kodowania wartości funkcji skrótu	dopuszczalne wartości: **Base64**	
FileSignatureList	Metadane plików wchodzących w skład dokumentu. W przypadku gdy rozmiar przesyłanego dokumentu jest mniejszy niż 60MB to lista składa się tylko z jednego pliku	Lista obiektów typu FileSignatureList	Wymagany. Lista musi zawierać przynajmniej jeden element
FileSignatureList.type	Rodzaj metody dzielącej dokument na części	String – dopuszczalne wartości: **tar**	Wymagany
FileSignatureList.mode	Rodzaj algorytmu kompresji	String – dopuszczalne wartości: **bz2**	Wymagany
FileSignature	Metadane pliku	Obiekt	Wymagany
FileName	Nazwa pliku przesyłanego do Azure Storage. Nazwa musi być zgodna z wyrażeniem regularnym: [_a-zA-Z0-9-]{5,55}	String	Wymagany

ContentLength	Długość pliku przesyłanego do Azure Storage	Int	Wymagany. Maksymalny rozmiar to 62914560 bajtów (60MB)
HashValue	Wartość funkcji skrótu pliku przesyłanego do Azure Storage, zakodowana w Base64.	String	Wymagany. Długość: 24 znaki
HashValue.algorithm	Nazwa algorytmu funkcji skrótu,	String – dopuszczalne wartości: **MD5**	Wymagany
HashValue.encoding	Algorytm kodowania wartości funkcji skrótu	String – dopuszczalne wartości: **Base64**	Wymagany

Skrót pliku przesyłanego do Storage (atrybut **HashValue** w type **FileSignature**) to wartość funkcji skrótu zgodnie z MD5 zakodowana następnie za pomocą Base64. Poniższy fragment kodu ilustruje to podejście:

```
var md5 = new MD5CryptoServiceProvider().ComputeHash(Encoding.Default.GetBytes(str));
var md5ToBase64 = Convert.ToBase64String(md5);
```

Schemat XSD dokumentu XML stanowiącego treść żądania:

initupload.xsd

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema xmlns="http://e-dokumenty.mf.gov.pl" xmlns:mf="http://e-dokumenty.mf.gov.pl" xmlns:xs="http://www.w3.org/2001/XMLSchema" targetNamespace="http://e-dokumenty.mf.gov.pl" elementFormDefault="qualified">
```

```
    <xs:element name="InitUpload" type="mf:InitUploadType"/>
    <xs:complexType name="InitUploadType">
        <xs:sequence>
            <xs:element name="DocumentType" minOccurs="1" maxOccurs="1">
                <xs:simpleType>
                    <xs:restriction base="xs:string">
                        <xs:enumeration value="JPK"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:element>
            <xs:element name="Version" minOccurs="1" maxOccurs="1">
                <xs:simpleType>
                    <xs:restriction base="xs:string">
                        <xs:pattern value="[0-9][0-9].[0-9][0-9].[0-9][0-
9].[0-9]{8}"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:element>
            <xs:element name="EncrypionKey" maxOccurs="1">
                <xs:complexType>
                    <xs:simpleContent>
                        <xs:extension base="xs:string">
                            <xs:attribute name="algorithm" use="required"
fixed="RSA"/>
                            <xs:attribute name="transformation"
use="optional" fixed="RSA/ECB/NoPadding"/>
                            <xs:attribute name="encoding" use="required"
fixed="Base64"/>
                        </xs:extension>
                    </xs:simpleContent>
                </xs:complexType>
            </xs:element>
            <xs:element name="DocumentList" type="mf:ArrayOfDocumentType"
minOccurs="1" maxOccurs="1">
                <xs:unique name="UniqueDocumentFileName">
                    <xs:selector xpath="mf:Document"/>
```

```
                    <xs:field xpath="mf:FileName"/>
                </xs:unique>
            </xs:element>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="ArrayOfDocumentType">
        <xs:sequence>
            <xs:element name="Document" minOccurs="1">
                <xs:complexType>
                    <xs:complexContent>
                        <xs:extension base="mf:DocumentType"/>
                    </xs:complexContent>
                </xs:complexType>
            </xs:element>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="DocumentType">
        <xs:sequence>
            <xs:element name="FormCode">
                <xs:annotation>
                    <xs:documentation>KodFormularza zawarty w nagłówku pliku
XML.</xs:documentation>
                </xs:annotation>
                <xs:complexType>
                <xs:simpleContent>
                <xs:extension base="xs:string">
                    <xs:attribute name="systemCode" type="xs:string"
use="required">
                        <xs:annotation>
                            <xs:documentation>Atrybut kodSystemowy elementu
KodFormularza z pliku XML.</xs:documentation>
                        </xs:annotation>
                    </xs:attribute>
                    <xs:attribute name="schemaVersion" type="xs:string"
use="required">
```

```
                        <xs:annotation>
                            <xs:documentation>Atrybut wersjaSchemy elementu
KodFormularza z pliku XML.</xs:documentation>
                        </xs:annotation>
                    </xs:attribute>
                </xs:extension>
                </xs:simpleContent>
                </xs:complexType>
            </xs:element>
            <xs:element name="FileName">
                <xs:simpleType>
                    <xs:restriction base="xs:string">
                        <xs:pattern value="[_a-zA-Z0-9-]{5,55}"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:element>
            <xs:element name="HashValue" type="HashValueSHAType"
minOccurs="1" maxOccurs="1"/>
            <xs:element name="FileSignatureList"
type="mf:ArrayOfFileSignatureType" minOccurs="1" maxOccurs="1">
                <xs:unique name="UniqueFileSignatureFileName">
                    <xs:selector xpath="mf:FileSignature"/>
                    <xs:field xpath="mf:FileName"/>
                </xs:unique>
            </xs:element>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="ArrayOfFileSignatureType">
        <xs:sequence>
            <xs:element name="FileSignature" type="mf:FileSignatureType"
nillable="true" minOccurs="1" maxOccurs="unbounded"/>
        </xs:sequence>
        <xs:attribute name="type" use="required">
            <xs:simpleType>
                <xs:restriction base="xs:string">
```

```
                    <xs:enumeration value="tar"/>
                </xs:restriction>
            </xs:simpleType>
        </xs:attribute>
        <xs:attribute name="mode" use="required">
            <xs:simpleType>
                <xs:restriction base="xs:string">
                    <xs:enumeration value="bz2"/>
                </xs:restriction>
            </xs:simpleType>
        </xs:attribute>
    </xs:complexType>
    <xs:complexType name="FileSignatureType">
        <xs:sequence>
            <xs:element name="FileName" minOccurs="1" maxOccurs="1">
                <xs:simpleType>
                    <xs:restriction base="xs:string">
                        <xs:pattern value="[_a-zA-Z0-9-]{5,55}"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:element>
            <xs:element name="ContentLength" type="xs:int" minOccurs="1"
maxOccurs="1"/>
            <xs:element name="HashValue" type="HashValueMD5Type"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="HashValueSHAType">
        <xs:simpleContent>
            <xs:extension base="xs:string">
                <xs:attribute name="algorithm" use="required" fixed="SHA-
256"/>
                <xs:attribute name="encoding" use="required" fixed="Base64"/>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
```

```
    <xs:complexType name="HashValueMD5Type">
        <xs:simpleContent>
            <xs:extension base="xs:string">
                <xs:attribute name="algorithm" use="required" fixed="MD5"/>
                <xs:attribute name="encoding" use="required" fixed="Base64"/>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
</xs:schema>
```

Przykładowa treść (body) żądania (dla czytelności pokazana jest treść bez elementów związanych z podpisem cyfrowym):

```
<?xml version="1.0" encoding="utf-8"?>
<InitUpload xmlns="http://e-dokumenty.mf.gov.pl">
  <DocumentType>JPK</DocumentType>
  <Version>01.01.01.20160430</Version>
  <EncrypionKey algorithm="RSA" transformation="RSA/ECB/NoPadding"
encoding="Base64">EncrypionKey1</EncrypionKey>
  <DocumentList>
    <Document>
      <FormCode systemCode="JPK_VAT (1)" schemaVersion="1-0">
JPK_VAT<FormCode>
      <FileName>FileName1</FileName>
      <HashValue algorithm="SHA-256" encoding="Base64">HashValue1</HashValue>
      <FileSignatureList type="tar" mode="bz2">
        <FileSignature>
          <FileName>FileName1</FileName>
          <ContentLength>103432</ContentLength>
          <HashValue algorithm="MD5" encoding="Base64">HashValue1</HashValue>
        </FileSignature>
      </FileSignatureList>
    </Document>
  </DocumentList>
</InitUpload>
```

Przykładowa treść (body) żądania (wraz z elementami związanymi z podpisem cyfrowym wg– zgodnie z wymaganiami przedstawionymi w p. 1.2

```
<?xml version="1.0" encoding="utf-8"?>
<InitUpload xmlns="http://e-dokumenty.mf.gov.pl">
  <DocumentType>JPK</DocumentType>
  <Version>01.01.01.20160430</Version>
  <EncrypionKey algorithm="RSA" transformation="RSA/ECB/NoPadding"
encoding="Base64">EncrypionKey1</EncrypionKey>
```

```
  <DocumentList>
    <Document>
      <FormCode systemCode="JPK_VAT (1)" schemaVersion="1-0">JPK_VAT
<FormCode>
      <FileName>FileName1</FileName>
      <HashValue algorithm="SHA-256" encoding="Base64">HashValue1</HashValue>
      <FileSignatureList type="tar" mode="bz2">
        <FileSignature xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:nil="true"/>
        <FileSignature>
          <FileName>FileName1</FileName>
          <ContentLength>103432</ContentLength>
          <HashValue algorithm="MD5" encoding="Base64">HashValue1</HashValue>
        </FileSignature>
        <FileSignature xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:nil="true"/>
      </FileSignatureList>
    </Document>
  </DocumentList><ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
    <ds:SignedInfo>
        <ds:CanonicalizationMethod Algorithm="http://www.w3.org/TR/2001/REC-
xml-c14n-20010315"/>
        <ds:SignatureMethod Algorithm="http://www.w3.org/2001/04/xmldsig-
more#rsa-sha256"/>
        <ds:Reference URI="">
            <ds:Transforms>
                <ds:Transform
Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature"/>
            </ds:Transforms>
            <ds:DigestMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
            <ds:DigestValue>XT5th/g03u9CpJJNNPdKzYHg+sA=</ds:DigestValue>
        </ds:Reference>
    </ds:SignedInfo>
    <ds:SignatureValue>
JJ1G4Ll4bnd8jE9H4gycyhp4imLvvEUqqjI8AUqdFWswvNFsJQJY3hp/8r8P7E3f
3/ly8E0njEeSG7RFp0F99xmQBCjkJhJ6Ha/MmdTkioSV5ZmUn6rljlusikjAxgdG
Y2mW/p8IoMJRR8GllOmQdPHZuqpCc6GuLEeoxD/8GUN52FU+wIAbSnoYO5S9bpW+
KO5wfEfO0k1Uo/dDfoNQlOZt5WSLqqZYq9jaiBBPOnRN/nXHa8dao961CgR/kiJc
xJ+3J9iHMdfXVHt05iQv15OIpcuMS9AZePpazxVKVXmH3HfF6BqirNXWyogXje+x
mK0HbnbWZCewofZb4Sn2eA==
    </ds:SignatureValue>
    <ds:KeyInfo>
        <ds:KeyValue>
            <ds:RSAKeyValue>
                <ds:Modulus>
tVEy4LmCs7znT8V0Vnzu4VnMssRoM3Yblr9RbK33GtJAwiiMFBW+e+jXPQrIhqkN
HUxkdRphA/Il81UgX5BOzgULeDcDitMFVqHMcOVaeZPK5AmTJeGDvVjcZ5g8PRRa
HfbP+wei7zUDt9Lt2lMccWFWSg7z7UQwPsBj83Gj6ahzgq+Pu1W7Gz5stVgeAQN3
zq++XNACulxT0kgY58NlZGqCov61ksT6W/MgRx3Bo12LcWnfc1r0GhZiQfqWZXdc
DPhhFosB/HgkJ8vm/0VB9Jg0dVb4fm4CPBPhNKKrxdxrHzRV8g6qd5Ro0gxfM12x
T+yK8u3MDWe/MpB5Q7dZ2Q==
                </ds:Modulus>
                <ds:Exponent>AQAB</ds:Exponent>
            </ds:RSAKeyValue>
        </ds:KeyValue>
        <ds:X509Data>
```

```
            <ds:X509IssuerSerial>
                <ds:X509IssuerName>Issuer</ds:X509IssuerName>

<ds:X509SerialNumber>265379454857972297697887687165046992692322305 6</ds:X509S
erialNumber>
            </ds:X509IssuerSerial>
            <ds:X509SubjectName>Subject</ds:X509SubjectName>
            <ds:X509Certificate>
MIIGBDCCBOygAwIBAgITdwARQBAhKoGbpNyrHQAAABFAEDANBgkqhkiG9w0BAQsF
ADA5MR4wHAYDVQQKExVNaWNyb3NvZnQgQ29ycG9yYXRpb24xFzAVBgNVBAMTDk1T
SVQgTkRFUyBDQSA0MB4XDTE2MDQyMjE3NTE0M1oXDTE2MDcyMTE3NTE0M1owgY4x
EzARBgoJkiaJk/IsZAEZFgNjb20xGTAXBgoJkiaJk/IsZAEZFgltaWNyb3NvZnQx
FDASBgoJkiaJk/IsZAEZFgRjb3JwMRYwFAYKCZImiZPyLGQBGRYGZXVyb3BlMRUw
EwYDVQQLEwxVc2VyQWNjb3VudHMxFzAVBgNVBAMTDlBpb3RyIEJvbmluc2tpMIIB
IjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAtVEy4LmCs7znT8V0Vnzu4VnM
ssRoM3Yblr9RbK33GtJAwiiMFBW+e+jXPQrIhqkNHUxkdRphA/Il81UgX5BOzgUL
eDcDitMFVqHMcOVaeZPK5AmTJeGDvVjcZ5g8PRRaHfbP+wei7zUDt9Lt2lMccWFW
Sg7z7UQwPsBj83Gj6ahzgq+Pu1W7Gz5stVgeAQN3zq++XNACulxT0kgY58NlZGqC
ov61ksT6W/MgRx3Bo12LcWnfc1r0GhZiQfqWZXdcDPhhFosB/HgkJ8vm/0VB9Jg0
dVb4fm4CPBPhNKKrxdxrHzRV8g6qd5Ro0gxfM12xT+yK8u3MDWe/MpB5Q7dZ2QID
AQABo4ICrTCCAqkwCwYDVR0PBAQDAgeAMCsGA1UdJQQkMCIGCisGAQQBgjcqAgEG
CisGAQQBgjcUAgIGCCsGAQUFBwMCMB0GA1UdDgQWBBQ3YhTnGyptfSNGP7N0WvJC
ZfOvfDAvBgNVHREEKDAmoCQGCisGAQQBgjcUAgOgFgwUcGlvdHJiQG1pY3Jvc29m
dC5jb20wHwYDVR0jBBgwFoAUEcADpofSlyPZGKy2kl6mwiIn4+4wgccGA1UdHwSB
vzCBvDCBuaCBtqCBs4YraHR0cDovL2NvcnBwa2kvY3JsL01TSVQlMjBOREVTJTIw
CisGAQQBgjcUAgIGCCsGAQUFBwMCMB0GA1UdDgQWBBQ3YhTnGyptfSNGP7N0WvJC
cC9jcmwvTVNJVCUyME5ERVMlMjBDQSUyMDQuY3JshkBodHRwOi8vY3JsLm1pY3Jv
c29mdC5jb20vcGtpL21zY29ycC9jcmwvTVNJVCUyME5ERVMlMjBDQSUyMDQuY3Js
MIGTBggrBgEFBQcBAQSBhjCBgzA3BggrBgEFBQcwAoYraHR0cDovL2NvcnBwa2kv
YWlhL01TSVQlMjBOREVTJTIwQ0ElMjA0LmNydDBIBggrBgEFBQcwAoY8aHR0cDov
L3d3dy5taWNyb3NvZnQuY29tL3BraS9tc2NvcnAvTVNJVCUyME5ERVMlMjBDQSUy
MDQuY3J0MDwGCSsGAQQBgjcVBwQvMC0GJSsGAQQBgjcVCIPPiU2t8gKFoZ8MgvrK
fYHh+3SBT4bBwlTsrWYCAWQCAS0wNwYJKwYBBAGCNxUKBCowKDAMBgorBgEEAYI3
KgIBMAwGCisGAQQBgjcUAgIwCgYIKwYBBQUHAwIwJQYDVR0gBB4wHDAMBgorBgEE
AYI3KgEFMAwGCisGAQQBgjcqARQwDQYJKoZIhvcNAQELBQADggEBAKfR7U4NaXk4
xNRo/tMmb2OMTr4ofiHqD/66lSH6esJ0Ap+9TOMxfXGnVa0B8H5A1lfW/HndGl8K
mWuItHPPZIqJLTuwxIRETWFMmJWuLllqn/BfLUB+4DWtcjZDTWvgET4gcX2VOr3u
tXthKd0kgfb1AyJY3Tw2cuqRvymBFuDC6s+jeg0L+NLI2ZWkV/MUoiH7Tpy265rv
28tJrQvhoFJQSanbUQOMhG3chfY/3kMhz2pOjKaYZqWxlANuzxJpRVSo1aTyWbCV
kFeDy7EGYzpH8pQHr56MD6qUX+hEYBNl5/CrJJVfMsY2wJvyTOwLnmIrevgKlaEI
5CWuHnfp2IA=
            </ds:X509Certificate>
        </ds:X509Data>
    </ds:KeyInfo>
</ds:Signature>
</InitUpload>
```

Zwracane dane

Odpowiedzi

Kod odpowiedzi	Opis

200 – OK	Poprawnie rozpoczęto sesję
400 – Bad Request	Nieprawidłowe zapytanie. Błędne wywołanie usługi
500 – Server Error	Błędne przetwarzanie zapytania

Odpowiedź 200 - OK:

Nazwa	Opis	Typ
ReferenceNumber	Identyfikator rozpoczętej sesji	String
TimeoutInSec	Czas życia (w sekundach) klucza uwierzytelniającego do wysłania dokumentów	Timespan
RequestToUploadFileList	Lista metadanych wykorzystywanych do zbudowania żądania wysłania plików do Azure Storage	Lista obiektów typu RequestToUploadFile
RequestToUploadFile	Metadane wykorzystywane do zbudowania żądania wysłania pliku do Azure Storage	Obiekt
BlobName	Nazwa bloba do którego będzie zapisany plik	String
FileName	Nazwa pliku	String
Url	Adres do którego nastąpi wysłanie pliku	String
Method	Metoda przesłania żądania	String

HeaderList	Lista nagłówków wymaganych do utworzenia żądania	Lista kluczy i wartości
Key	Klucz nagłówka	String
Value	Wartość nagłówka	String

Przykład odpowiedzi:


```
{
 "ReferenceNumber": "ba96951d00635700000001726b6ec621",
 "TimeoutInSec": "7200",
 "RequestToUploadFileList": [
  {
   "BlobName": "a8b6f7db-e5f4-4541-b232-0e6a9017ca3f",
   "FileName": "jpkfile01.xml",
   "Url": "https://jpkstorageaccount03dev.blob.core.windows.net/container-004/a8b6f7db-e5f4-4541-b232-0e6a9017ca3f",
   "Method": "PUT",
   "HeaderList": [
    {
     "Key": "x-ms-date",
     "Value": "Mon, 16 May 2016 17:21:51 GMT"
    },
    {
     "Key": "x-ms-version",
     "Value": "2015-04-05"
    },
    {
     "Key": "Content-MD5",
     "Value": "eu/k4pzvymH+SYVs1F8MAg=="
```

```
      },
      {
        "Key": "x-ms-blob-type",
        "Value": "BlockBlob"
      },
      {
        "Key": "Content-Type",
        "Value": "application/xml"
      },
      {
        "Key": "Authorization",
        "Value": "SharedKey jpkstorageaccount03dev:Gf565UNo7q7ymIw2rGdg4LDM4z+M3BbTbXedg+Xt7Mk="
      }
    ]
  },
  {
    "BlobName": "2a3bfb5d-e817-404c-9e7a-5d819fdd4df7",
    "FileName": "jpkfile02.txt",
    "Url": "https://jpkstorageaccount03dev.blob.core.windows.net/container-004/2a3bfb5d-e817-404c-9e7a-5d819fdd4df7",
    "Method": "PUT",
    "HeaderList": [
      {
        "Key": "x-ms-date",
        "Value": "Mon, 16 May 2016 17:21:51 GMT"
      },
      {
        "Key": "x-ms-version",
        "Value": "2015-04-05"
      },
```

```
      {
        "Key": "Content-MD5",
        "Value": "eu/PE54vymH+SYVs238MAg=="
      },
      {
        "Key": "x-ms-blob-type",
        "Value": "BlockBlob"
      },
      {
        "Key": "Content-Type",
        "Value": "application/xml"
      },
      {
        "Key": "Authorization",
        "Value": "SharedKey jpkstorageaccount03dev:Tz7EqAl6OszIxGjBUk2qcxs82Af4Xq9CxyFx6u34LEI="
      }
    ]
  }
 ]
}
```


Odpowiedź 400 – Bad Request:

Nazwa	Opis	Typ
Message	Komunikat błędu	String
ModelState	Szczegółowe informacje na temat wykrytych błędów	Obiekt
initUpload.X	Szczegółowa walidacja pola X	Lista błędów

Przykład odpowiedzi:


```
{
  "Message": "The request is invalid.",
  "ModelState": {
     "initUpload.Version": [
   "Pole Version jest wymagane."
  ],
     "initUpload.EncrypionKey": [
   "Pole EncrypionKey jest wymagane."
  ]
 }
}}
```

2.2.2 Put Blob

Metoda wysyłająca zasadnicze dokumenty JPK. Jest to metoda bezpośrednio implementowana przez usługę przestrzeń magazynową Azure (Azure Storage).

Jej pełna dokumentacja dostępna jest pod adresem:

https://msdn.microsoft.com/en-us/library/azure/dd179451.aspx

Schemat żądania http:

https://<nazwa_konta_storage>.blob.core.windows.net/<nazwa_kontenera>/<nazwa _blobu>

Dla przypomnienia – pełny adres, do którego klient ma wysłać dokumenty JPK jest zwracany przez metodę InitUpload. Częścią zwracanego adresu jest Shared Access Signature (SAS), jednorazowy klucz, umożliwiający klientowi na umieszczenie dokumentów we wskazanym kontenerze. Klucz SAS jest generowany jednorazowo i jest ważny tylko dla konkretnego przesyłanego pliku (weryfikacja wartości funkcji skrótu), w zadanych ramach czasowych i w zadanym fragmencie przestrzeni Azure Storage – zapewnia więc wysoki poziom bezpieczeństwa i gwarantuje, że wysłane zostaną pliki, dla których klucz SAS został wygenerowany

Nagłówek żądania

Wykorzystywane nagłówki żądań:

Nagłówek żądania	Opis
Authorization	Wymagany. Określa schemat uwierzytelniania, nazwę konta i podpis. Więcej informacji: Authentication for the Azure Storage Services.
Date* or *x-ms-date	Wymagany. Określa Coordinated Universal Time (UTC) dla żądania. Więcej informacji: Authentication for the Azure Storage Services.
x-ms-version	Wymagany dla wszystkich uwierzytelnionych żądań. Określa wersję interfejsu po stronie Azure dla operacji. Więcej informacji: Versioning for the Azure Storage Services.
x-ms-blob-type:	Wymagany. Określa rodzaj bloba. Dopuszczalna wartość to BlockBlob.

BlockBlob	
Content-MD5	Wymagany. Wartość funkcji skrótu MD5. Ten skrót jest używany do weryfikacji integralności danych podczas transportu. Wykorzystując tę wartość, Azure Storage automatycznie sprawdza wartość skrótu danych które otrzymał z zadeklarowanymi. Jeśli obie wartości się różną, operacja zakończy się niepowodzeniem z kodem błędu 400 (Bad Request).
Content-Type	MIME typ przesyłanego pliku

Pełna dokumentacja dotycząca nagłówków żądań – i innych szczegółów interakcji z Azure Storage – dostępna jest po wskazywanym już adresem:

https://msdn.microsoft.com/en-us/library/azure/dd179451.aspx

Treść żądania

W treści żądania zawarty jest wysyłany plik.

2.2.3 FinishUpload

Metoda kończąca sesję. Jej wywołanie jest warunkiem koniecznym prawidłowego zakończenia procedury wysyłania dokumentów. Brak jej wywołania jest tożsamy z uznaniem, że sesja została przerwana.

Nazwa	FinishUpload
Typ metody	Post
Typ przesyłanej zawartości	application/json
Typ zwracanej zawartości	application/json
Maksymalny rozmiar żądania	100KB

Opis treści (body) żądania:

Nazwa	Opis	Typ	Walidacja
ReferenceNumber	Identyfikator sesji	String	Wymagany
AzureBlobNameList	Lista nazw blobów, które znajdują się w Azure Storage	List stringów	Wymagany. Lista musi zawierać tyle elementów ile plików wysłaliśmy do Azure Storage

Zwracane dane

Odpowiedzi

Kod odpowiedzi	Opis
200 – OK	Poprawnie zakończona sesja
400 – Bad Request	Nieprawidłowe zapytanie. Błędne wywołanie usługi
500 – Server Error	Błędne przetworzanie zapytania

Odpowiedź 200 – Ok

Pusta zawartość odpowiedzi

Odpowiedz 400 – Bad Request:

Nazwa	Opis	Typ
Message	Komunikat błędu	String
ModelState	Szczegółowe informacje na temat wykrytych błędów	Obiekt
finishUpload.X	Szczegółowa walidacja pola X	Lista błędów

Przykład:

```
{
  "Message": "The request is invalid.",
  "ModelState": {
     "finishUpload.ReferenceNumber": [
   "Pole ReferenceNumber jest wymagane."
  ]
 }
}}
```

2.2.4 Status

Metoda zwraca Urzędowe Potwierdzenie Odbioru wysłanych dokumentów. Metoda ta jest częścią API dla klientów, dostępną z tej samej usługi co inne metody, natomiast – w przeciwieństwie do tych innych – jej działanie ogranicza się do wywołania odpowiedniej metody wystawianej przez CPD, gdzie fizycznie dostępne jest wygenerowane UPO – i przekazanie tego UPO do klienta.

Nazwa	Status
Typ metody	Get
Typ przesyłanej zawartości	Query String
Typ zwracanej zawartości	application/json
Maksymalny rozmiar żądania	100KB
Format	Status/ba96951d00635700000001726b6ec621

Opis przesyłanego json-a w Body

Nazwa	Opis	Typ	Walidacja
ReferenceNumber	ReferenceNumber - Identyfikator sesji	String	Wymagany

Odpowiedzi

Kod odpowiedzi	Opis
200 – OK	Poprawnie zwrócono potwierdzenie
400 – Bad Request	Nieprawidłowe zapytanie. Błędne wywołanie usługi
500 – Server Error	Błędne przetwarzanie zapytania

Odpowiedz 200 – Ok

Nazwa	Opis	Typ
Code	Kod statusu	String
Description	Opis	String
Upo	Urzędowe potwierdzenie odbioru	Obiekt

Odpowiedź 400 – Bad Request:

Nazwa	Opis	Typ
Message	Komunikat błędu	String

ModelState	Szczegółowe informacje na temat wykrytych błędów	Obiekt
upoNumber.X	Szczegółowa walidacja pola X	Lista błędów

Przykład:

```
{
  "Message": "The request is invalid.",
  "ModelState": {
    "upoNumber.ReferenceNumber": [
   "Pole ReferenceNumber jest wymagane."
  ]
 }
}}
```

Lista statusów:

Poniższa tabela prezentuje kody statusów wraz z ich opisami.

Statusy są pogrupowane w poniższy sposób:

1xx – Kody określające sytuacje związane ze stanem sesji (np. rozpoczęta, wygasła)

2xx – Kody określające sytuacje, w których przetwarzanie dokumentów zakończyło się powodzeniem

3xx – Kody informujące o fazie przetwarzania dokumentu

4xx- 5xx Kody określające sytuacje, w których proces przetwarzania dokumentów zakończył się błędem..

Kod status	Opis
100	Sesja rozpoczęta - czekamy na wysłanie dokumentów
110	Sesja wygasła

200	Przetwarzanie dokumentu zakończone poprawnie, pobierz UPO
300	Nieprawidłowy numer referencyjny
401	Weryfikacja negatywna – dokument niezgodny ze schematem xsd
403	Dokument z niepoprawnym podpisem