Specyfikacja interfejsów usług Jednolitego Pliku Kontrolnego

Centrum Informatyki Resortu Finansów

13 stycznia 2022 r.

Wersja 3.5

Zmiany

Data	Wersja	Opis
23.05.2016	1.3	Opublikowanie specyfikacji technicznej usług Jednolitego Pliku Kontrolnego.
10.06.2016	1.4	1. Zmiana metody dzielenia spakowanego pliku z metody TAR na binarne dzielenie pliku SPLIT. 2. Metoda Status: - zmiana zwracanej zawartości dla kodu http: 200 i 400 3. Metoda InitUploadSigned w przypadku kodu http: 200 - zmiana typu dla właściwości TimeoutInSec z Timespan na int 4 Zmiany schematu XSD pliku metadanych: - dodanie typu dokumentu JPKAH (JPK ad hoc) dla plików przysłanych w ramach kontroli, - poprawienie nazwy (literówka) EncrypionKey na EncryptionKey, - poprawienie formatu wersji REST API, - poprawienie formatu nazwy pliku, - dodanie całkowitej liczby części podzielonego pliku oraz liczby porządkowej dla poszczególnych części, - usunięcie atrybutów type oraz mode z listy plików cząstkowych FileSignatureList, - dodanie elementu (Packaging) w liście plików cząstkowych FileSignatureList wraz z możliwością wyboru rodzaju podziału i kompresji pliku. Obecnie możliwe jest użycie kompresji zip (deflate) z podziałem binarnym - element SplitZip z atrybutami type (split) oraz mode (zip), - dodanie elementu Encryption w liście plików cząstkowych FileSignatureList wraz z możliwością wyboru algorytmu szyfrowania. Obecnie wykorzystanie algorytmu AES256 - element AES z atrybutami size (256), block (16), mode (CBC), padding (PKCS#7) oraz elementem IV (Initialization Vector) z atrybutami bytes (16) i encoding (Base64).
17.06.2016	1.5	Zmiany schematu XSD pliku metadanych: - ustalenie obsługiwanej wersji REST API - 01.02.01.20160617, - zmiana wyrażenia regularnego elementu FileName. - uzupełnienie zbioru kodów odpowiedzi dla metody Status
04.07.2016	1.6	1. Dodanie opisu specyfikacji szyfrowania klucza szyfrującego. 2. Zmiana w opisie interfejsów - przetłumaczenie komunikatów na język polski. 3. Dodanie identyfikatora żądania (RequestId) w strukturze odpowiedzi dla kodu http: 400 i 500 4. Rozszerzenie zbioru kodów odpowiedzi błędów (400 Bad Request) metody InitUploadSigned. 5. Dodanie informacji o dopuszczalnych transformacjach dla podpisu metadanych. 6. Ograniczenie długości wartości funkcji skrótów w schemacie XSD pliku metadanych.

Data	Wersja	Opis
20.07.2016	1.7	1. Rozszerzenie zbioru kodów odpowiedzi błędów (400 Bad Request) metody InitUploadSigned. 2. Dodanie przykładów prawidłowych odpowiedzi inicjowania sesji metodą InitUploadSigned. 3. Zamieszczenie przykładów wykorzystania narzędzi programistycznych SDK metody Put Blob. 4. Dodanie informacji o parametrze umożliwiającym włączenie weryfikacji podpisu z certyfikatem kwalifikowanym przy inicjowaniu sesji metodą InitUploadSigned na środowisku testowym.
29.07.2016	2.0	1. Doprecyzowanie mechanizmu kompresji ZIP.
30.09.2016	2.1	1. Uzupełnienie zbioru kodów odpowiedzi błędów (400 Bad Request) metody InitUploadSigned. 2. Wyszczególnienie adresów domenowych używanych przestrzeni Azure Storage.
31.01.2017	2.2	1. Zmiana przykładów prawidłowych odpowiedzi inicjowania sesji metodą InitUploadSigned.
31.03.2017	2.3	1. Rozszerzenie opisu funkcjonalności podpisu metadanych o obsługę europejskiego podpisu kwalifikowanego oraz podpisu Profilem Zaufanym. 2. Rozszerzenie opisu adresów domenowych używanych przestrzeni Azure Storage.
11.05.2020	3.0	1. Rozszerzenie opisu przygotowania metadanych uwierzytelniających o możliwość skorzystania z danych autoryzujących (autoryzacja danymi osobowymi oraz wartościami kwot z poprzednich rozliczeń). 2. Aktualizacja kodów statusów: - dodanie nowego kodu 136 zwracanego w metodzie InitUploadSigned - usunięcie niewystępujących kodów (102, 110, 301, 302, 303, 403, 404, 409, 411, 414) - dodanie nowych kodów 417, 418, 419, 420, 422, 423, 424 zwracanych w metodzie Status 3. Dodanie opisu pełnomocnictw 4. Rozszerzenie zakresu adresów magazynów chmurowych (p. 2.2)
25.09.2020	3.1	1. Aktualizacja kodów statusów: - dodanie nowego kodu błędu 155 zwracanego w metodzie InitUploadSigned
06.11.2020	3.2	1. Aktualizacja kodów statusów: - dodanie nowego kodu błędu 411 zwracanego w metodzie Status
21.01.2021	3.3	1. Dodanie obsługi plików CUK(1)
27.05.2021	3.4	1.Dodanie obsługi plików CUK(2) i ALK(1) 2. Aktualizacja kodów statusów - dodanie nowych kodów błędów 99 i 101 zwracanych w metodzie InitUploadSigned - dodanie nowych kodów 425 i 426 zwracanych w metodzie Status
13.01.2022	3.5	1. Dodanie obsługi plików JPK_V7M(2) i JPK_V7K(2)

Spis treści

1 Przygotowanie danych JPK

1.1 Przygotowanie dokumentów JPK

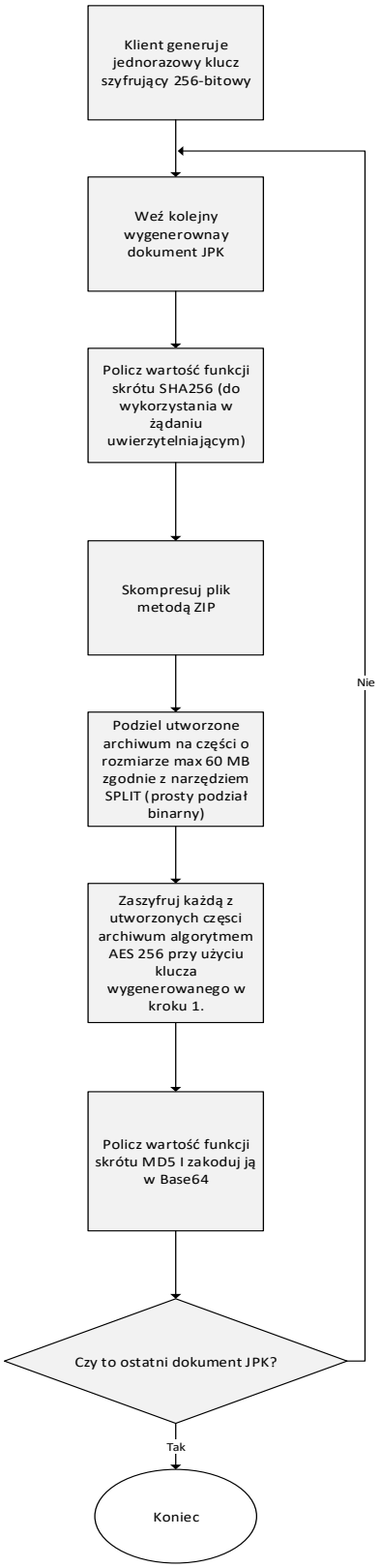
Dane JPK przygotowywane będą po stronie klienta (np. w systemie ERP) w formie plików XML zgodnych ze schematem XSD opublikowanym przez Ministerstwo Finansów na stronie https://epuap.gov.pl/wps/portal/strefa-urzednika/inne-systemy/crwde.

Nazwy schematów opublikowanych w CRWDE ePUAP:

- **JPK_V7M(1), JPK_V7M(2)** DEKLARACJA MIESIĘCZNA I EWIDENCJA DLA PODATKU OD TOWARÓW I USŁUG (W FORMIE JEDNOLITEGO PLIKU KONTROLNEGO),
- **JPK_V7K(1), JPK_V7K(2)** DEKLARACJA KWARTALNA I EWIDENCJA DLA PODATKU OD TOWARÓW I USŁUG (W FORMIE JEDNOLITEGO PLIKU KONTROLNEGO).
- **CUK (1), CUK (2)** INFORMACJA W SPRAWIE OPŁATY OD ŚRODKÓW SPOŻYWCZYCH.
- **ALK (1)** INFORMACJA W SPRAWIE OPŁATY ZA ZEZWOLENIE NA OBRÓT HURTOWY NAPOJAMI ALKOHOLOWYMI W OPAKOWANIACH DO 300 ML

Oprócz ww. schematów opublikowanych w CRWDE e-PUAP obsługiwane są również dokumenty JPK zgodne ze schematami opublikowanymi przez Ministerstwo Finansów na stronie Struktury JPK - Ministerstwo Finansów - Krajowa Administracja Skarbowa - Portal Gov.pl (www.gov.pl).

Każdy z dokumentów opisanych właściwym schematem ma stanowić osobny plik XML. Wygenerowany plik XML powinien być zakodowany w UTF-8. Przygotowanie dokumentów JPK do wysłania odbywa się zgodnie ze schematem zamieszczonym poniżej:



Rysunek 1 Schemat blokowy kroków przygotowywania do wysyłki danych JPK

1.1.1 Kompresja danych JPK

Wygenerowany dokument JPK zostanie skompresowany do pliku w formacie ZIP oraz podzielony binarnie na części o wielkości nie przekraczającej 60 MB. Należy spodziewać się wysokiego stopnia kompresji co spowoduje, że scenariusz w którym będziemy mieli więcej niż jedną część, będzie stosunkowo rzadki.

Wymagana metoda kompresji to format pliku ZIP z użyciem algorytmu DEFLATE, bez stosowania opcji dzielenia (split/multipart). W wyniku kompresji powinien powstać jeden plik ZIP zawierający pojedynczy dokument JPK. Jeżeli rozmiar otrzymanego pliku ZIP przekracza 60MB, należy go podzielić binarnie na odpowiednią liczbę części o wielkości 60MB każda oraz ostatnią część o rozmiarze nie większym niż 60MB. Takie podejście z jednej strony zapewnia wykorzystanie znanych i powszechnie stosowanych narzędzi oraz łatwość implementacji dla różnych platform, z drugiej – efektywność, w szczególności operacji kompresji i prostotę API dla tych operacji.

1.1.2 Szyfrowanie danych JPK

Skompresowane pliki będą szyfrowane. Do szyfrowania plików wykorzystany będzie algorytm AES256, z kluczem szyfrującym wygenerowanym po stronie klienta. W implementacji mechanizmu szyfrowania należy użyć następującej specyfikacji algorytmu AES:

Długość klucza	Key Size	256 bits / 32 bytes
Tryb szyfru	Cipher Mode	CBC (Cipher Block Chaining)
Dopełnienie	Padding	PKCS#7
Rozmiar bloku	Block Size	16 bytes
Wektor inicjujący	Initialization Vector	16 bytes

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 certyfikatu klucza publicznego udostępnionego przez Ministerstwo Finansów,
- zaszyfrowany klucz jest dołączany do pliku metadanych, zgodnie z przedstawionym poniżej opisem tego pliku.

1.1.3 Szyfrowanie klucza szyfrującego

Szyfrowanie klucza szyfrującego należy wykonać algorytmem asymetrycznym RSA z wykorzystaniem certyfikatu klucza publicznego udostępnionego przez Ministerstwo Finansów. W implementacji mechanizmu szyfrowania należy użyć następującej specyfikacji algorytmu RSA:

Długość klucza	Key Size	256 bits / 32 bytes
Tryb szyfru	Cipher Mode	ECB (Electronic Codebook)
Dopełnienie	Padding	PKCS#1
Rozmiar bloku	Block Size	256 bytes

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 uwierzytelniony jedną z technik:

1. użycie:
    a. podpisu kwalifikowanego (polski lub europejski),
    b. podpisu zaufanego
2. umieszczenie elementu AuthData zawierającego zaszyfrowane dane autoryzujące

1.2.1 Podpis kwalifikowany lub podpis zaufany

Plik metadanych musi być podpisany cyfrowo **podpisem kwalifikowanym polskim lub europejskim** albo **podpisem zaufanym** zgodnie z algorytmem XAdES Basic Electronic Signature w postaci pliku XML zgodnego ze schematem http://www.w3.org/2000/09/xmldsig, 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). Przy podpisywaniu można dokonać transformacji obiektu podpisywanego zgodnie z kodowaniem http://www.w3.org/2000/09/xmldsig#base64.

Funkcją skrótu wykorzystywaną w podpisie powinna być RSA-SHA256.

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

1.2.2 Dane autoryzujące

W przypadku korzystania z metody autoryzacji kwotą należy uzupełnić element AuthData:

```
<xs:element name="AuthData" minOccurs="0" maxOccurs="1">
    <xs:annotation>
        <xs:documentation>To opcjonalne pole powinno zawierać
dokument XML zgodny z opublikowaną schemą SIG-2008_v2-0.xsd zaszyfrowany z
wykorzystaniem algorytmu symetrycznego AES256. Powinien zostać wykorzystany ten
sam klucz, który jest wykorzystywany do szyfrowania części skompresowanego
archiwum pliku JPK i załączany do niniejszego pliku metadanych. Algorytm
kodowania zaszyfrowanych danych to Base64.</xs:documentation>
    </xs:annotation>
    <xs:simpleType>
        <xs:restriction base="xs:string"/>
    </xs:simpleType>
</xs:element>
```

Pole to powinno zawierać dokument XML zgodny z opublikowanym schematem SIG-2008_v2-0.xsd zaszyfrowany z wykorzystaniem algorytmu symetrycznego AES256 (generowany po stronie klienta). Powinien zostać wykorzystany **ten sam klucz**, który jest wykorzystywany do szyfrowania części skompresowanego archiwum pliku JPK i załączany do pliku metadanych. Algorytm kodowania zaszyfrowanych danych to Base64.

Parametry szyfrowania danych autoryzujących:

Długość klucza	Key Size	256 bits / 32 bytes
Tryb szyfru	Cipher Mode	CBC (Cipher Block Chaining)
Dopełnienie	Padding	PKCS#7
Rozmiar bloku	Block Size	16 bytes
Wektor inicjujący	Initialization Vector	16 bytes

1.2.3 Pełnomocnictwo

System weryfikuje czy osobie, która składa dokument w imieniu firmy lub podatnika z użyciem podpisu (podpis kwalifikowany lub podpis zaufany) zostało udzielone pełnomocnictwo do podpisywania deklaracji składanej za pomocą środków komunikacji elektronicznej UPL-1.

W przypadku składania dokumentu w imieniu firmy lub podatnika z użyciem podpisu kwalifikowanego lub podpisu zaufanego wymagane jest zarejestrowanie w urzędzie skarbowym pełnomocnictwa do podpisywania deklaracji składanych drogą elektroniczną.

Brak lub błędnie nadane pełnomocnictwo będzie skutkowało zakończeniem przetwarzaniem dokumentu z kodem błędu 420 zwracanym przez metodę Status.

W przypadku podpisu kwalifikowanego wydanego przez polskie centrum certyfikacji, certyfikat powinien zawierać PESEL lub NIP właściciela podpisu.

Pełnomocnictwo nie jest sprawdzane w przypadku użycia danych autoryzujących.

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

Implementacja środowiska testowego dostępna jest pod adresem:

https://test-e-dokumenty.mf.gov.pl/

Natomiast adresy poszczególnych metod przedstawiają się następująco:

https://test-e-dokumenty.mf.gov.pl/api/Storage/InitUploadSigned

https://test-e-dokumenty.mf.gov.pl/api/Storage/Status/{referenceNumber}

https://test-e-dokumenty.mf.gov.pl/api/Storage/FinishUpload

Adresy magazynów chmurowych do których wysyłane są pliki JPK:

https://taxdocumentstorage00tst.blob.core.windows.net

https://taxdocumentstorage01tst.blob.core.windows.net

https://taxdocumentstorage02tst.blob.core.windows.net

…

https://taxdocumentstorage97tst.blob.core.windows.net

https://taxdocumentstorage98tst.blob.core.windows.net

https://taxdocumentstorage99tst.blob.core.windows.net

Implementacja środowiska produkcyjnego dostępna jest pod adresem:

https://e-dokumenty.mf.gov.pl/

Natomiast adresy poszczególnych metod przedstawiają się następująco:

https://e-dokumenty.mf.gov.pl/api/Storage/InitUploadSigned

https://e-dokumenty.mf.gov.pl/api/Storage/Status/{referenceNumber}

https://e-dokumenty.mf.gov.pl/api/Storage/FinishUpload

Adresy magazynów chmurowych do których wysyłane są pliki JPK:

https://taxdocumentstorage00.blob.core.windows.net

https://taxdocumentstorage01.blob.core.windows.net

https://taxdocumentstorage02.blob.core.windows.net

…

https://taxdocumentstorage97.blob.core.windows.net

https://taxdocumentstorage98.blob.core.windows.net

https://taxdocumentstorage99.blob.core.windows.net

wykorzystywane nazwy domenowe można weryfikować za pomocą wyrażenia regularnego:

https:[/]{2}taxdocumentstorage[0-9]{2}.blob.core.windows.net[/](.*)

Poniżej znajduje się szczegółowy opis działania 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 parametrów przekazywanych w adresie metody:

Nazwa	Opis	Typ	Walidacja
enableValidateQualifiedSignature	W przypadku przekazania wartości **true (na środowisku testowym)**, system zweryfikuje czy przesyłany plik został podpisany poprawnym podpisem kwalifikowanym polskim albo europejskim lub podpisem zaufanym.	bool	Opcjonalny – dopuszczalne wartości: **true**, **false**

Adres metody z włączoną weryfikacją podpisu kwalifikowanego:
https://test-e-dokumenty.mf.gov.pl/api/Storage/InitUploadSigned?enableValidateQualifiedSignature=true

Opis struktury XML stanowiącego zawartość żądania (message body):

Nazwa	Opis	Typ	Walidacja
InitUpload	Metadane dla metody InitUpload	Obiekt	Wymagany
DocumentType	Nazwa typu przesyłanego dokumentu.	String	Wymagany - dopuszczalne wartości: **JPK** - dokumenty przesyłane cyklicznie **JPKAH** - dokumenty przesyłane doraźnie w ramach kontroli
Version	Wersja REST API do której adresowane jest zapytanie	String	Wymagany, 01.02.01.20160617
EncryptionKey	Klucz symetryczny zaszyfrowany algorytmem asymetrycznym (RSA)	String	Wymagany
EncryptionKey.algorithm	Algorytm, którym zaszyfrowany jest klucz symetryczny	String – dopuszczalne wartości: **RSA**	Wymagany
EncryptionKey.mode	Tryb szyfrowania	String – dopuszczalne wartości: **ECB**	Wymagany
EncryptionKey.padding	Format dopełnienia klucza szyfrującego	String – dopuszczalne wartości: **PKCS#1**	Wymagany

EncryptionKey.encoding	Algorytm kodowania wartości klucza	String – dopuszczalne wartości: **Base64**	Wymagany
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	String	Wymagany
FormCode.systemCode	Atrybut kodSystemowy elementu KodFormularza z pliku XML	String	Wymagany
FormCode.schemaVersion	Atrybut wersjaSchemy elementu KodFormularza z pliku XML	String	Wymagany
FileName	Nazwa pliku JPK.	String	Wymagany, unikalny, format: [a-zA-Z0-9_\.\-]{5,55} na przykład JPK_VAT_2016-07-01.xml
ContentLength	Całkowity rozmiar dokumentu	Long	Wymagany
HashValue	Skrót całego dokumentu	String	Wymagany

HashValue.algorithm	Nazwa algorytmu funkcji skrótu,	String – dopuszczalne wartości: **SHA-256**	Wymagany
HashValue.encoding	Algorytm kodowania wartości funkcji skrótu	String – dopuszczalne wartości: **Base64**	Wymagany
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 FileSignature	Wymagany. Lista musi zawierać przynajmniej jeden element
FileSignatureList.filesNumber	Liczba wszystkich części pliku	int	Wymagany
Packaging	Możliwe rodzaje podziału i kompresji dokumentu	Lista wyboru	Wymagany
SplitZip	Rodzaj podziału i kompresji dokumentu	Obiekt	Wymagany
SplitZip.type	Rodzaj metody dzielącej dokument na części	String – dopuszczalne wartości: **split**	Wymagany
SplitZip.mode	Rodzaj algorytmu kompresji	String – dopuszczalne wartości: **zip**	Wymagany

Encryption	Możliwe metody szyfrowania plików cząstkowych	Lista wyboru	Wymagany
AES	Metoda szyfrowania plików cząstkowych	Obiekt	Wymagany
AES.size	Rozmiar klucza szyfrującego w bitach	Int – dopuszczalne wartości: **256**	Wymagany
AES.block	Rozmiar bloku szyfrującego w bajtach	Int – dopuszczalne wartości: **16**	Wymagany
AES.mode	Tryb szyfrowania	String – dopuszczalne wartości: **CBC**	Wymagany
AES.padding	Metoda dopełnienia bloku szyfrującego	String – dopuszczalne wartości: **PKCS#7**	Wymagany
IV	Wektor inicjujący algorytmu szyfrującego	String	Wymagany
IV.bytes	Rozmiar wektora inicjującego w bajtach	String – dopuszczalne wartości: **16**	Wymagany
IV.encoding	Metoda kodowania wartość wektora inicjującego	String – dopuszczalne wartości:	Wymagany

		Base64	
FileSignature	Metadane pliku	Obiekt	Wymagany
OrdinalNumber	Liczba porządkowa kolejnej części	Int	Wymagany, unikalny
FileName	Nazwa pliku przesyłanego do Azure Storage.	String	Wymagany, unikalny, format: [a-zA-Z0-9_\.\-]{5,55} na przykład JPK_VAT_2016-07-01.xml.zip.001.aes
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 (nie należy konwertować do hex-a przed konwersją do 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

AuthData	To opcjonalne pole powinno zawierać dokument XML zgodny z opublikowaną schemą SIG-2008_v2-0.xsd zaszyfrowany z wykorzystaniem algorytmu symetrycznego AES256. Powinien zostać wykorzystany ten sam klucz, który jest wykorzystywany do szyfrowania części skompresowanego archiwum pliku JPK i załączany do niniejszego pliku metadanych. Algorytm kodowania zaszyfrowanych danych to Base64.	String	Opcjonalny

Skrót pliku przesyłanego do Storage (element **HashValue** w typie **FileSignatureType**) to wartość funkcji skrótu zgodnie z MD5 zakodowana następnie za pomocą Base64.

Schemat XSD dokumentu XML stanowiącego treść żądania jest udostępniony na stronie https://www.podatki.gov.pl/jednolity-plik-kontrolny/ w sekcji „JPK_VAT z deklaracją". We wskazanej lokalizacji umieszczono przykład metadanych podpisanych w formacie XAdES-BES certyfikatem niekwalifikowanym (self-signed).

Metoda InitUploadSigned zwraca trzy typy odpowiedzi:

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

Opis struktury JSON (application/json) poprawnej odpowiedzi (200 – OK):

Nazwa	Opis	Typ
ReferenceNumber	Identyfikator rozpoczętej sesji	String
TimeoutInSec	Czas życia (w sekundach) klucza uwierzytelniającego do wysłania dokumentów (uzależniony od liczby zadeklarowanych plików do wysyłki)	Int
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 metodą *Put Blob*. Adres jest generowany dynamicznie i jego schemat może ulec zmianie.	String
Method	Metoda przesłania żądania *Put Blob*	String
HeaderList	Lista nagłówków wymaganych do utworzenia żądania *Put Blob*. Zwracane headery są generowane dynamicznie. Ich nazwy jak i ilość elementów może ulec zmianie.	Lista kluczy i wartości
Key	Klucz nagłówka	String
Value	Wartość nagłówka	String

Przykład treści poprawnej odpowiedzi (200 - OK):


```
{
  "ReferenceNumber": "d4fd41850323d2f6000000b013016327",
  "TimeoutInSec": 900,
  "RequestToUploadFileList": [
    {
      "BlobName": "8377ed3d-1b05-4c76-b718-6fddd46fd298",
      "FileName": "jpk_vat_100-01.xml.zip.aes",
      "Url":
"https://taxdocumentstorage09tst.blob.core.windows.net/d4fd41850323d2f6000000b013016327/8377ed3d
-1b05-4c76-b718-6fddd46fd298?sv=2015-07-
08&sr=b&si=d4fd41850323d2f6000000b013016327&sig=yFXyJdsPPkbE0iQwVs5ccLEYEU0lxQHldbVyPf
PciXw%3D",
      "Method": "PUT",
      "HeaderList": [
        {
          "Key": "Content-MD5",
          "Value": "eXkPLHMM+dHB5GCFoeAvsA=="
        },
        {
          "Key": "x-ms-blob-type",
          "Value": "BlockBlob"
        }
      ]
    },
    {
      "BlobName": "0a80a089-bc10-41e1-a74d-70fd45f27aa3",
      "FileName": "jpk_vat_100-02.xml.zip.aes",
      "Url":
"https://taxdocumentstorage09tst.blob.core.windows.net/d4fd41850323d2f6000000b013016327/0a80a089
-bc10-41e1-a74d-70fd45f27aa3?sv=2015-07-
08&sr=b&si=d4fd41850323d2f6000000b013016327&sig=Fj%2BGjn7hCKIM6hSvMBGWBxSOyV7V%2FL
MM9pnenbaoxks%3D",
      "Method": "PUT",
      "HeaderList": [
        {
          "Key": "Content-MD5",
```

```
        "Value": "NZew85QTb16mFLzx9cyKzA=="
       },
       {
        "Key": "x-ms-blob-type",
        "Value": "BlockBlob"
       }
     ]
   }
 ]
}
```


Odpowiedź dla przykładu pliku podpisanego certyfikatem niekwalifikowanym w formacie XAdES-BES (enveloping) zamieszczonego na stronie w archiwum JPK-VAT-TEST-0001.ZIP:


```
{
  "ReferenceNumber": " ef7d17780087346e0000004c0c7982ec",
  "TimeoutInSec": 900,
  "RequestToUploadFileList": [
   {
    "BlobName": "094951bc-ba54-404e-b2c8-df2591ad0e17",
    "FileName": "JPK-VAT-TEST-0001.xml.zip.aes",
    "Url":
"https://taxdocumentstorage03tst.blob.core.windows.net/ef7d17780087346e0000004c0c7982ec/094951bc-ba54-404e-b2c8-df2591ad0e17?sv=2015-07-08&sr=b&si=ef7d17780087346e0000004c0c7982ec&sig=kN7LlprYkIP9uxod%2F1gcaDGN8WjbEbfDIA4GXuuzOmk%3D",
    "Method": "PUT",
    "HeaderList": [
     { "Key": "Content-MD5", "Value": "5YnivEH4gz5Wg5E8M2XwAQ==" },
     { "Key": "x-ms-blob-type","Value": "BlockBlob" }
    ]
   }
 ]
}
```


Odpowiedź dla przykładu pliku podpisanego certyfikatem niekwalifikowanym w formacie XAdES-BES (enveloped) zamieszczonego na stronie w archiwum JPK-VAT-TEST-0000.ZIP:


```
{
```

```
  "ReferenceNumber": " ef81ecf9011a546c0000004d72be8011",
  "TimeoutInSec": 900,
  "RequestToUploadFileList": [
   {
    "BlobName": "55a19799-5f1d-4336-9051-197dc53e5adf",
    "FileName": "JPK-VAT-TEST-0001.xml.zip.aes",
    "Url":
"https://taxdocumentstorage02tst.blob.core.windows.net/ef81ecf9011a546c0000004d72be8011/55a19799
-5f1d-4336-9051-197dc53e5adf?sv=2015-07-
08&sr=b&si=ef81ecf9011a546c0000004d72be8011&sig=HeLYQd8RfRucs4KGgWxITEU36OgQuqSe1RU
XZ10n8%2Bs%3D",
    "Method": "PUT",
    "HeaderList": [
     { "Key": "Content-MD5", "Value": "5YnivEH4gz5Wg5E8M2XwAQ=="},
     { "Key": "x-ms-blob-type", "Value": "BlockBlob" }
    ]
   }
 ]
}
```


Opis struktury JSON (application/json) odpowiedzi (400 – Bad Request):

Nazwa	Opis	Typ
Message	Komunikat błędu	String
Code	Kod błędu	String
Errors	Opcjonalnie. Tablica błędów	Lista stringów
RequestId	Unikalny identyfikator błędnego żądania	GUID

Wyszczególnienie kodów zawartych w odpowiedzi (400 – Bad Request):

Code	Komunikat	Opis
99	Nieprawidłowe kodowanie znaków w pliku xml	Podany dokument nie jest zakodowany w formacie UTF-8
100	Niepoprawny XML	Podany dokument nie jest dokumentem XML
101	Nieprawidłowa deklaracja kodowania znaków w pliku xml	Podany dokument zawiera nieprawidłową deklarację kodowania znaków (inną niż <?xml version="1.0" encoding="utf-8"?>)
110	Niepodpisany dokument	Podany dokument jest niepodpisany zgodnie ze specyfikacją
111	Podpis jest złożony w innym formacie niż XAdES-BES	
112	Niepoprawnie złożony podpis. Niemożliwa weryfikacja	W trakcie weryfikacji podpisu wystąpił nieoczekiwany błąd
113	Podpis złożony w nieobsługiwanym formacie zewnętrznym (detached)	Obsługiwane formaty podpisu to enveloped i enveloping
114	Problem z odczytaniem podpisanego obiektu	
120	Podpis negatywnie zweryfikowany	Nie udało się poprawnie zweryfikować podpisu
130	Referencje w podpisie zostały negatywnie zweryfikowane. Dane prawdopodobnie zostały zmodyfikowane	
135	Dokument z podpisem niekwalifikowanym	Na środowisku produkcyjnym sprawdzana jest autentyczność podpisu kwalifikowanego.

136	Dokument zawiera więcej niż jeden podpis	Dokument może być podpisany wyłącznie jednym podpisem
140	Przesłany plik jest niezgody ze schematem XSD	Nie udało się zweryfikować dokumentu zgodnie ze schematem InitUpload.xsd
150	Nieobsługiwany kod formularza : „konkretny systemCode”	Kod formularza jest nieobsługiwany
155	Przesłany plik jest niepoprawny. Zadeklarowano co najmniej dwa pliki cząstkowe o takim samym skrócie.	Błąd dotyczy zadeklarowania w pliku initupload co najmniej dwóch plików cząstkowych o takim samym skrócie.
160	Wartość „konkretny HashValue” nie jest zakodowana w Base64	Skrót plików zadeklarowanych do przesłania musi być zakodowany w Base64.
170	Przesłano duplikat przetworzonego dokumentu. Numer referencyjny oryginału: XXXXXXXX	Duplikaty są sprawdzane na podstawie wartości skrótu SHA-256 zadeklarowanego dokumentu JPK

Przykład odpowiedzi:


```
{
 "Message": "Podpis negatywnie zweryfikowany",
 "Code": 120,
 "RequestId": "172dc3cc-5b97-48de-91dd-6903587cba19"
}
```


Opis struktury JSON (application/json) odpowiedzi (500 – Internal Server Error):

Nazwa	Opis	Typ
Message	Komunikat błędu	String
RequestId	Unikalny identyfikator błędnego żądania	GUID

Przykład odpowiedzi:


```
{
```

```
 "Message": "Wewnętrzny błąd systemu ",
 "RequestId": "172dc3cc-5b97-48de-91dd-6903587cba19"
}
```


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

2.2.2.1 Wysłanie za pomocą klienta Http

Adres żądania:

https://<nazwa_konta_storage>.blob.core.windows.net/<reference_number>/<nazwa _bloba>

Pełny adres, do którego klient ma wysłać dokumenty JPK jest zwracany przez metodę InitUploadSigned. 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 w zadanych ramach czasowych i w zadanym fragmencie przestrzeni Azure Storage – zapewnia więc wysoki poziom bezpieczeństwa.

Metoda żądania:

Zwracana jest przez InitUploadSigned.

Nagłówek żądania

Zwracane są przez InitUploadSigned. Wykorzystywane nagłówki żądań:

Nagłówek żądania	Opis
x-ms-blob-type	Wymagany. Określa rodzaj bloba. Dopuszczalna wartość to **BlockBlob**.
Content-MD5	Opcjonalny. 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).

Treść żądania

W treści żądania zawarty jest wysyłany plik.

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

Metoda Put Blob zwraca odpowiedzi:

Kod odpowiedzi	Opis
201 – Created	Poprawnie przesłano plik do przestrzeni Azure.
4xx	Błędne wywołanie usługi
5xx	Błędne przetwarzanie zapytania

Odpowiedź (201 – Created):

Pusta zawartość odpowiedzi

Odpowiedzi 4xx oraz 5xx zwracają informację o błędzie w postaci XML (**application/xml**):

Nazwa	Opis	Typ
Error	Element główny struktury	Object
Code	Opisowy kod błędu	String
Message	Komunikat błędu	String

Przykład:

```
<?xml version="1.0" encoding="utf-8"?>
<Error>
  <Code>AuthenticationFailed</Code>
  <Message>Server failed to authenticate the request. Make sure the value of Authorization header is formed correctly including the signature.
RequestId:a5124e1c-0001-0056-06b3-ddc62c000000
Time:2016-07-14T09:40:13.7833645Z</Message>
  <AuthenticationErrorDetail>SAS identifier cannot be found for specified signed identifier</AuthenticationErrorDetail>
</Error>
```

2.2.2.2 Wysłanie za pomocą SDK

Dostępne implementacje: .NET, Node.js, Java, C++, PHP, Ruby, Python, iOS, Xamarin.

https://azure.microsoft.com/en-gb/documentation/articles/storage-dotnet-how-to-use-blobs/

Przykład:

Wiadomość zwrócona przez InitUploadSigned:


```
{
  "ReferenceNumber": "d8cb2f0f014381ab000000b012f8a3d6",
  "TimeoutInSec": 900,
  "RequestToUploadFileList": [
    {
      "BlobName": "b42748d3-0660-4d81-afc2-3c250fbcdbef",
      "FileName": "jpk_vat_100.xml.zip.aes",
      "Url":
"https://taxdocumentstorage10tst.blob.core.windows.net/d8cb2f0f014381ab000000b012f8a3d6/b42748d3-0660-4d81-afc2-3c250fbcdbef?sv=2015-07-08&sr=b&si=d8cb2f0f014381ab000000b012f8a3d6&sig=2y%2BZ3cjcyBbBnCM6Mw9a4EPN2KA%2B01kgf9fro%2FK6Xgw%3D",
      "Method": "PUT",
      "HeaderList": [
        { "Key": "Content-MD5", "Value": "eXkPLHMM+dHB5GCFoeAvsA==" },
        { "Key": "x-ms-blob-type", "Value": "BlockBlob" }
      ]
    }
  ]
}
```


Wysyłka pliku w .NET:

```
var absoluteUri =
"https://taxdocumentstorage10tst.blob.core.windows.net/d8cb2f0f014381ab000000b012f8a3d6/b42748d3-0660-4d81-afc2-3c250fbcdbef";
var sas = "sv=2015-07-08&sr=b&si=d8cb2f0f014381ab000000b012f8a3d6&sig=2y%2BZ3cjcyBbBnCM6Mw9a4EPN2KA%2B01kgf9fro%2FK6Xgw%3D";
var blob = new CloudBlockBlob(new Uri(absoluteUri), new StorageCredentials(sas));
using (var stream = new FileStream("jpk_vat_100-01.xml.zip.aes", FileMode.Open))
      {
            blob.UploadFromStream(stream);
```

}

2.2.3 FinishUpload

Metoda kończąca sesję. Jej wywołanie jest warunkiem koniecznym prawidłowego zakończenia procedury wysyłania dokumentów. Sprawdzane są wtedy wymagane pliki używając nazwy i MD5 wartości zadeklarowanych w InitUploadSigned. 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 struktury JSON (application/json) stanowiącego zawartość żądania (message body):

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

Przykład:

```
{
 "ReferenceNumber": "e8505c4703e5fd5b000000b04bc6f43f"
 "AzureBlobNameList": [
  "d1eadd0e-ccd5-44ab-85e7-2f2a552e7f17",
  "5c3ceb5f-8c5d-4720-9005-7c7d1d88f121"
```

```
  ],
}
```

Metoda FinishUpload zwraca trzy typy odpowiedzi:

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

Odpowiedź (200 – OK):

Pusta zawartość odpowiedzi

Opis struktury JSON (application/json) odpowiedzi (400 – Bad Request):

Nazwa	Opis	Typ
Message	Komunikat błędu	String
Errors	Opcjonalnie. Tablica błędów	Lista stringów
RequestId	Unikalny identyfikator błędnego żądania	GUID

Przykład:

```
{
  "Message": "Żądanie jest nieprawidłowe"
  "Errors": "[‘Reference number jest wymagany’]"
  "RequestId": "172dc3cc-5b97-48de-91dd-6903587cba19"
}
```

Opis struktury JSON (application/json) odpowiedzi (500 – Internal Server Error):

Nazwa	Opis	Typ
Message	Komunikat błędu	String

RequestId	Unikalny identyfikator błędnego żądania	GUID

Przykład:

```
{
  "Message": "Wewnętrzny błąd systemu ",
  "RequestId": "172dc3cc-5b97-48de-91dd-6903587cba19"
}
```

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.

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

Opis przesyłanego parametru

Nazwa	Opis	Typ	Walidacja
ReferenceNumber	ReferenceNumber - Identyfikator sesji	String	Wymagany

Metoda Status zwraca trzy typy odpowiedzi:

Kod odpowiedzi	Opis
200 – OK	Poprawnie zwrócono potwierdzenie

400 – Bad Request	Nieprawidłowe zapytanie. Błędne wywołanie usługi
500 – Server Error	Błędne przetwarzanie zapytania

Opis struktury JSON (application/json) poprawnej odpowiedzi (200 – OK):

Nazwa	Opis	Typ
Code	Kod statusu	String
Description	Opis	String
Details	Szczegóły zdarzenia	String
Upo	Opcjonalne. Urzędowe poświadczenie odbioru	String
Timestamp	Znacznik czasu	Datetime

Przykład:


```
{
  "Code": 300,
  "Description": "Nieprawidłowy numer referencyjny",
  "Upo": ""
  "Details": ""
  "Timestamp": "2016-06-17T09:37:40.773976+00:00"
}
```

Lista statusów:

Poniższa tabela prezentuje kody statusów wraz z ich opisami zwracanych w poprawnej odpowiedzi przez metodę Status. Statusy są pogrupowane w poniższy sposób:

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

2xx – Kody określające prawidłowe zakończenie procesu przetwarzania dokumentu

3xx – Kody informujące o fazie przetwarzania dokumentu

4xx – Kody określające niewłaściwe zakończenie procesu przetwarzania dokumentu

Kod status	Opis
100	Rozpoczęto sesję przesyłania plików.
101	Odebrano X z Y zadeklarowanych plików.
120	Sesja została poprawnie zakończona. Dane zostały poprawnie zapisane. Trwa weryfikacja dokumentu.
200	Przetwarzanie dokumentu zakończone poprawnie, pobierz UPO.
300	Nieprawidłowy numer referencyjny.
401	Weryfikacja negatywna – dokument niezgodny ze schematem XSD.
405	Dokument z odwołanym certyfikatem.
406	Dokument z certyfikatem z nieobsługiwanym dostawcą.
407	Przesłałeś duplikat dokumentu. Numer referencyjny oryginału to XXXXXXXX
408	Dokument zawiera błędy uniemożliwiające jego przetworzenie.
410	Przesłane pliki nie są prawidłowym archiwum ZIP.

411	Weryfikacja negatywna - w systemie jest już złożona identyczna deklaracja VAT
412	Dokument nieprawidłowo zaszyfrowany.
413	Suma kontrolna dokumentu niezgodna z deklarowana wartością.
415	Przesłany rodzaj dokumentu nie jest obsługiwany w systemie.
417	Dokument nieprawidłowo zaszyfrowany. Błąd odszyfrowania danych autoryzujących
418	Weryfikacja negatywna - dane autoryzujące niezgodne ze schematem XSD
419	Weryfikacja negatywna – błąd w danych autoryzujących
420	Brak aktualnego pełnomocnictwa/upoważnienia do podpisywania dokumentu
422	Weryfikacja negatywna – dokument złożony z użyciem danych autoryzujących może złożyć wyłącznie podatnik, będący osobą fizyczną
423	Dokument z certyfikatem bez wymaganych atrybutów
424	Weryfikacja negatywna – dokument nie może być podpisany z użyciem danych autoryzujących
425	Weryfikacja negatywna - niespójne dane
426	Nieprawidłowe kodowanie znaków w danych autoryzujących

Opis struktury JSON (application/json) odpowiedzi (400 – Bad Request):

Nazwa	Opis	Typ
Message	Komunikat błędu	String
Errors	Opcjonalnie. Tablica błędów	Lista stringów
RequestId	Unikalny identyfikator błędnego żądania	GUID

Przykład:

```
{
  "Message": "Żądanie jest nieprawidłowe",
  "RequestId": "172dc3cc-5b97-48de-91dd-6903587cba19"
}
```


Opis struktury JSON (application/json) odpowiedzi (500 – Internal Server Error):

Nazwa	Opis	Typ
Message	Komunikat błędu	String
RequestId	Unikalny identyfikator błędnego żądania	GUID

Przykład:


```
{
  "Message": "Wewnętrzny błąd systemu ",
  "RequestId": "172dc3cc-5b97-48de-91dd-6903587cba19"
}
```