web service for reception of KIR KPR

Field: VAT

Verzija: 0.05

content

[1. INTRODUCTION 4](#_Toc199419263)

[2. FUNCTIONALITIES OF THE SERVICE 4](#_Toc199419264)

[3. ACCESS TO THE SERVICE 4](#_Toc199419265)

[4. FORMAT OF THE BOOK OF INVOICES 5](#_Toc199419266)

[5. DEFINITION OF THE REST INTERFACE OF THE SERVICE 5](#_Toc199419267)

[5.1. GENERAL CALL PARAMETERS 5](#_Toc199419268)

[Authentication 5](#_Toc199419269)

[Media type of input parameters 5](#_Toc199419270)

[Media type of output parameters 5](#_Toc199419271)

[5.2. GENERAL HTTP CODES OF REPLIES 6](#_Toc199419272)

[5.3. METHODS FOR LOGIN – /api/v1/Auth 7](#_Toc199419273)

[Method /api/v1/Auth/Representing/{taxPayerId}/{taxPayerType} 7](#_Toc199419274)

[Method /api/v1/Auth/Certificate/Representing/{taxPayerId}/{taxPayerType} 8](#_Toc199419275)

[5.4. METHODS FOR WORK WITH BOOKS OF INVOICES - /api/v1/InvoiceBook 8](#_Toc199419276)

[Method /api/v1/InvoiceBook 8](#_Toc199419277)

[Metoda Method /api/v1/InvoiceBook/Status/{edpId} 10](#_Toc199419278)

[5.5. Other methods in connection with books - /api/v1/Documents 11](#_Toc199419279)

[Metoda Method /api/v1/Documents/OpenPeriods/{formCode} 11](#_Toc199419280)

[Metoda Method /api/v1/Documents/Permissions 13](#_Toc199419281)

[5.6. Methods in connection with operations of the service - /api/v1/Heartbeat 15](#_Toc199419282)

[Method /api/v1/Heartbeat 15](#_Toc199419283)

[6. Definition of interface of SOAP service 16](#_Toc199419284)

[6.1. consideration of errors 16](#_Toc199419285)

[6.2. login service 16](#_Toc199419286)

[Metoda Method LoginUsingClientCertificate 16](#_Toc199419287)

[6.3. service for content functionalities 17](#_Toc199419288)

[Metoda Method UploadInvoiceBook 17](#_Toc199419289)

[Method GetInvoiceBookStatus 19](#_Toc199419290)

[Method GetOpenPeriods 19](#_Toc199419291)

[Method GetPermissions 19](#_Toc199419292)

[7. Validation 19](#_Toc199419293)

[8. Annexes 21](#_Toc199419294)

**History of the document**

|  |  |
| --- | --- |
| **Date** | **Notes** |
| 13.1.2025 | Document |
| 6.3.2025 | Link added to the published schemes |
| 26.3.2025 | Completion of the document and new annex (examples of invoice books for each scheme) |
| 2.4.2025 | Correction of JSON scheme (Annex 8). |
| 19.6.2025 | Minor additions and corrections |

# INTRODUCTION

The online service described in this document enables the submission of data from KIR (the book of outgoing invoices) and KPR (the book of incoming invoices) for users of the eDavki system.

The web service has a REST (supports data encoded in JSON) and SOAP interfaces.

# FUNCTIONALITIES OF THE SERVICE

The web service has the following functionalities:

* Registration, which is possible with:
  + digital certificates, supported by eDavki;
  + Authentication tokens issued by the OAuth, which is eDavki web service.
* Checking the permissions and persons authorized for the submission of books.
* Submission of books (transfer of files into eDavki system).
* Checking the status of the submitted book.
* Checking open periods for submission of books.

# ACCESS TO THE SERVICE

The web service is available on the following URLs:

* REST service:
  + Production environment: <https://edavki.durs.si/InvoiceBookService/>
  + Test environment: <https://beta.edavki.durs.si/InvoiceBookService/>
* SOAP service for login:
  + Production environment: <https://edavki.durs.si/InvoiceBookService/SoapService/Login/>
  + Test environment: <https://beta.edavki.durs.si/InvoiceBookService/SoapService/Login/>
* SOAP service for content functionalities:
  + Production environment: <https://edavki.durs.si/InvoiceBookService/SoapService/>
  + Test environment: <https://beta.edavki.durs.si/InvoiceBookService/SoapService/>

The REST interface of the web service can be viewed and tested using the Swagger web interface at the following addresses:

* Production environment: <https://edavki.durs.si/InvoiceBookService/swagger/index.html>
* Test environment: <https://beta.edavki.durs.si/InvoiceBookService/swagger/index.html>

WSDL of SOAP service is available at the following addresses:

* SOAP service for login:
  + Production environment: <https://edavki.durs.si/InvoiceBookService/SoapService/Login/?singleWsdl>
  + Test environment: <https://beta.edavki.durs.si/InvoiceBookService/SoapService/Login/?singleWsdl>
* SOAP service for content functionalities:
  + Production environment: <https://edavki.durs.si/InvoiceBookService/SoapService/?singleWsdl>
  + Test environment: <https://beta.edavki.durs.si/InvoiceBookService/SoapService/?singleWsdl>

# FORMAT OF THE BOOK OF INVOICES

The service accepts books formatted in one of the following three formats: XML, JSON, CSV. The format specifications are included in the chapter 'Appendices'.

# DEFINITION OF THE REST INTERFACE OF THE SERVICE

This chapter describes the individual methods of the REST service, grouped by functionality.

## GENERAL CALL PARAMETERS

In all service calls, some general parameters are used that are sent in the header of the request.

### Authentication

To access all functionalities of the service (except for login itself), the user must authenticate. This is done by sending an authentication token in the method call, which is obtained during login (see subsection 5.3). The token shall be sent in the header of the request in the field Authorisation. The value in the field shall have the following form:

Bearer {token}

Example of the whole field:

Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiI1NDYxMTMxMCIsImp0aSI6IjA0ZDgwYTYxLTk1MTEtNDEzNy04ODI1LTEwNTc4MzllZTkxYSIsImlhdCI6MTcyMjUwNTExMSwiTG9naW5JZCI6IjI3OSIsIkxvZ2luVHlwZSI6IkNlcnRpZmljYXRlIiwiVXNlclByb2ZpbGVJZCI6IjQzMCIsIkluZGl2aWR1YWxQZXJzb25JZCI6IjU0NjExMzEwIiwiVGF4UGF5ZXJJZCI6IjU0NjExMzEwIiwiVGF4UGF5ZXJUeXBlIjoiRk8iLCJSZXByZXNlbnRpbmdJZCI6IjU0NjExMzEwIiwiUmVwcmVzZW50aW5nVHlwZSI6IkZPIiwiZXhwIjoxNzIyNTkxNTExLCJpc3MiOiJodHRwczovL2R1cnN3ZWIuZW5kYXZhLm5ldDo0NjIvIiwiYXVkIjoiaHR0cHM6Ly9kdXJzd2ViLmVuZGF2YS5uZXQ6NDYyLyJ9.pl4AmJkVRTZkmYKuKCz13xkvS5V2cuIhRr0qvAnb5bM

### Media type of input parameters

The media type of input parameters is defined in the header of the request in the field Content-Type. Supported types are »application/json« for JSON (an exception is the method /api/v1/InvoiceBook – see below).

Example of the whole field:

Content-Type: application/json

### Media type of output parameters

The media type of output parameters is defined in the header of the request in the field accept. The supported types are »application/json« for JSON.

Example of the whole field:

accept: application/json

## GENERAL HTTP CODES OF REPLIES

In the case of a successful service call, it generally responds with an HTTP code 200 OK and a method-specific response in the form of JSON.

In the event that an error occurs when calling the service, the service responds with an appropriate HTTP code. In most cases, it also returns a JSON response with more detailed information about the cause for error of the type *ErrorResponse*:

ErrorResponse{

|  |  |
| --- | --- |
| errorReason | ErrorReason integer($int32)  Tip napake Error type:  0 = BadRequest  1 = Unauthorized  2 = Forbidden  3 = NotFound  4 = ServerError  Enum: [ 0, 1, 2, 3, 4 ] |
| errorMessage | string nullable: true  Opis napake. Description of the error. |
| incidentId | string nullable: true  Opcijski ID incidenta. Optional ID of the incident. |

}

In all service calls, the following HTTP response codes are possible in case of errors:

* 400 Bad Request: wrong input data
* 401 Unauthorized: The user failed to authenticate with a valid login mechanism. In most cases, the response body is empty; in exceptional cases, it may contain a JSON response.
* 500 InternalServerError internal error on the service. The optional field is filled in for this error incidentId, which is intended for easier tracking of errors.

In the event of errors, the service may return also other HTTP codes that are specific for individual service methods.

## METHODS FOR LOGIN – /api/v1/Auth

By calling one of the methods for login, the user simultaneously authenticates and selects the represented taxable person. The service supports two methods of authentication:

* With digital certificates supported in eDavki. This is the recommended method of authentication.
* With authentication tokens issued by the OAuth web service of eDavki. This method should be used exceptionally when authentication with digital certificates is not possible for technical reasons.

### Method /api/v1/Auth/Representing/{taxPayerId}/{taxPayerType}

* Description: the method enables the selection of the represented taxable person, the user authenticates with an authentication token. There are two options:
  + The user uses a token issued by the eDavki OAuth service. This issues a token in which the represented taxable person has not yet been selected.
  + The user uses a previously issued token that he or she has obtained by calling this method or method /api/v1/Auth/Certificate/Representing/{taxPayerId}/{taxPayerType} (see below). In this way, the user can change the represented taxable person.
* HTTP method: GET
* Input parameters:
  + taxPayerId (integer($int32)): tax number of the represented taxable person
  + taxPayerType (integer($int32):type of the represented taxable person:
    - *0 = FO*
    - *1 = SP*
    - *2 = PO*
* Specific HTTP response codes:
  + 403 Forbidden: access is not allowed (for example, the user does not have permissions for the selected represented taxable person)
* Output parameters:
  + Status of the reply 200 OK:

TokenResponse{

|  |  |
| --- | --- |
| expires | string($date-time)  . The date and time when the token expires. |
| token | string nullable: true  Authentication token. |

}

* Examples:
  + Input parameters:
  + taxPayerId: 54611310

taxPayerType: 0

* + Reply for HTTP status 200 OK:

{

"expires": "2024-01-01T01:12:01",

"token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiI1NDYxMTMxMCIsImp0aSI6IjA0ZDgwYTYxLTk1MTEtNDEzNy04ODI1LTEwNTc4MzllZTkxYSIsImlhdCI6MTcyMjUwNTExMSwiTG9naW5JZCI6IjI3OSIsIkxvZ2luVHlwZSI6IkNlcnRpZmljYXRlIiwiVXNlclByb2ZpbGVJZCI6IjQzMCIsIkluZGl2aWR1YWxQZXJzb25JZCI6IjU0NjExMzEwIiwiVGF4UGF5ZXJJZCI6IjU0NjExMzEwIiwiVGF4UGF5ZXJUeXBlIjoiRk8iLCJSZXByZXNlbnRpbmdJZCI6IjU0NjExMzEwIiwiUmVwcmVzZW50aW5nVHlwZSI6IkZPIiwiZXhwIjoxNzIyNTkxNTExLCJpc3MiOiJodHRwczovL2R1cnN3ZWIuZW5kYXZhLm5ldDo0NjIvIiwiYXVkIjoiaHR0cHM6Ly9kdXJzd2ViLmVuZGF2YS5uZXQ6NDYyLyJ9.pl4AmJkVRTZkmYKuKCz13xkvS5V2cuIhRr0qvAnb5bM"

}

### Method /api/v1/Auth/Certificate/Representing/{taxPayerId}/{taxPayerType}

* Description: the method enables the selection of a represented taxable person, the user authenticates with a digital certificate, which he or she used to establish the TLS connection.
* In all other aspects, the method behaves the same as the method /api/v1/Auth/Representing/{taxPayerId}/{taxPayerType} (see above).

## METHODS FOR WORK WITH BOOKS OF INVOICES - /api/v1/InvoiceBook

### Method /api/v1/InvoiceBook

* Description: the method enables the upload of a new invoice book for the represented taxable person. The body of the service call must be formatted as »multipart/form-data«. It shall include exactly two sections:
  + The first section must contain metadata about the invoice book in JSON format. The media type must be one of "application/json", "application/\*+json", "text/json".
  + The second section should contain a zip archive with the invoice book in binary format. The media type must be "application/octet-stream" ali or "application/x-zip-compressed".
* HTTP method: POST
* Input parameters (metapodatki o knjigi v JSON obliki v prvi sekciji metadata about the book in JSON format in the first section):

UploadInvoiceBookRequest{

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| period | Period{   |  |  | | --- | --- | |  | Period of the book | | begin | string($date-time) nullable: true  Beginning of the period | | end | string($date-time) nullable: true  End of the period |   } |
| format | InvoiceBookFormat integer($int32)  Format of the book.  0 = XML  1 = JSON  2 = CSV  Enum: [ 0, 1, 2 ] |
| schemaVersion | integer($int32)  Version of the scheme of invoice book. |
| correlationId | string nullable: true  Client correlation ID. Intended for easier tracking of the book processing. |

}

* Specific HTTP response codes:
* Output parameters:
  + Status of reply 200 OK:

UploadInvoiceBookResponse{

|  |  |
| --- | --- |
| edpId | string nullable: true  ID of the book in eDavki system. |
| status | InvoiceBookStatus integer($int32)  Status of the book (see below).  Enum: [ 0, 1, 2, 3, 4, 5, 6 ] |

}

* Examples:
  + Input parameters (first section):

{

"period": {

"begin": "2024-01-01",

"end": "2024-01-31"

},

"format": 0,

"schemaVersion": 1,

"correlationId": "098sdkljf8907asd8f"

}

* + Reply for HTTP status 200 OK:

{

"edpId": "KIR-12345678-123",

"status": 0

}

* Example of the whole call:

*Content-Length: 24215*

*Content-Type: multipart/form-data; boundary='0d2377d3-0b22-4a46-8879-819381d5200d'*

*Accept-Encoding: gzip, deflate, br*

*User-Agent: Mozilla/5.0 (Windows NT 10.0; Microsoft Windows 10.0.22621; sl-SI) PowerShell/7.5.0*

*--0d2377d3-0b22-4a46-8879-819381d5200d*

*Content-Type: application/json*

*Content-Disposition: form-data*

*{*

*"period": {*

*"begin": "2024-01-01T00:00:00",*

*"en d": "2024-01-31T00:00:00"*

*},*

*"format": 0,*

*"type": 0,*

*"schemaVersion": 1,*

*"correlationId": "ENDAUTOKSPICXCA-20250324T1051055652Z"*

*}*

*--0d2377d3-0b22-4a46-8879-819381d5200d*

*Content-Type: application/octet-stream*

*Content-Disposition: form-data*

*….( binary content omitted)…*

*--0d2377d3-0b22-4a46-8879-819381d5200d--*

### Metoda Method /api/v1/InvoiceBook/Status/{edpId}

* Description: the method returns data about the status of the previously uploaded invoice book.
* HTTP method: GET
* Input parameters:
  + *edpId*: ID of the book in eDavki system
* Specific HTTP response codes:
  + 403 Forbidden: The book of invoices with the given ID does not belong to the represented taxable person.
  + 404 Not Found: The book of invoices with the given ID does not exist in the eDavki system.
* Output parameters:
  + Response status 200 OK:

InvoiceBookStatusResponse{

|  |  |
| --- | --- |
| status | InvoiceBookStatus integer($int32)  Marks for the status of the book:  0 = Uploaded –the book is uploaded  1 = Validating –the book is in the validation process in eDavki system  2 = ValidatedOk – the book has been successfully validated in eDavki system  3 = ValidationError –the book has been validated in eDavki system, but it includes errors. The book will not be transmitted to the backend system. The book should be uploaded once more without errors.  4 = InProcess – the book is being processed in backend systems  5 = ProcessedOk – the book has been successfully processed in backend systems  6 = ProcessedError –the book has not been successfully processed in backend systems  Enum: [ 0, 1, 2, 3, 4, 5, 6 ] |
| validationMessage | string nullable: true  Optional report on found errors during validation in eDavki system. The report is in JSON format (see chapter 7). |
| backendMessage | string nullable: true  Optional message is being processed in backend systems. |

}

* Examples:
  + Response JSON for HTTP status 200 OK:

{

"status": 3,

"validationMessage": "[{\"Type\":2,\"Code\":5,\"Message\":\"Oznaka države 'AB' ne obstaja v seznamu držav.\",\"BookId\":null,\"BookArrayIndex\":null},{\"Type\":2,\"Code\":7,\"Message\":\"Knjig ne morete oddati, ker v izbranem obdobju '1.6.2023 - 30.6.2023' niste zastopnik tujca v 'string'.\",\"BookId\":null,\"BookArrayIndex\":null}]",

"backendMessage": null

}

A response with HTTP status 200 OK does not necessarily mean that the uploaded book has been accepted. This also depends on the validation of the book, which is performed asynchronously (see Chapter 7) and is evident from the book status in the response.

## Other methods in connection with books - /api/v1/Documents

### Metoda Method /api/v1/Documents/OpenPeriods/{formCode}

* Description: the method returns a list of open periods for the represented taxable person and the specified form code.
* HTTP method: GET
* Input parameters:
  + *formCode*: form type . Currently, the supported types are 'DDV\_EVID' (for invoice books) and informatively also 'DDV\_O' (for DDV-O forms).
* Specific HTTP reply codes: /
* Output parameters:
  + Response status 200 OK:

OpenPeriodsResponse{

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| openPeriods | [ nullable: true  List of open periods.  OpenPeriod{   |  |  | | --- | --- | |  | Open period | | period | Period{   |  |  | | --- | --- | |  | Period | | begin | string($date-time) nullable: true  Beginning of the period | | end | string($date-time) nullable: true  End of the period |   } | | representedForeignerId | string nullable: true  Optional identification mark for the represented foreign person. | | dueDate | string($date-time) nullable: true  Time limit for submission |   }] |

}

* Examples:
  + Reply for HTTP status 200 OK:

{

"openPeriods": [

{

"period": {

"begin": "2024-01-01T00:00:00",

"end": "2024-01-31T00:00:00"

},

"representedForeignerId": null,

"dueDate": "2024-01-31T00:00:00"

},

{

"period": {

"begin": "2024-02-01",

"end": "2024-02-29"

},

"representedForeignerId": "XY1234567890",

"dueDate": null

}

]

}

### Metoda Method /api/v1/Documents/Permissions

* Description: the method returns a list of permissions that the user has in the eDavki system.
* HTTP method: GET
* Input parameters:
  + *formCode*: optional form type label. Currently, only the type 'DDV\_O' is supported.
* Specific HTTP response codes: /
* Output parameters:
  + Response status 200 OK:

DocumentPermissionsResponse{

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| permissions | [ nullable: true  List of permissions.  DocumentPermission{   |  |  | | --- | --- | |  | Specification of permissions for the type of form. | | mandator | TaxPayer{   |  |  | | --- | --- | |  | Data about the authorising person. | | id | integer($int32)  Tax number | | type | TaxPayerType integer($int32)  Type of taxable person.  0 = FO  1 = SP  2 = PO  Enum: [ 0, 1, 2 ] |   } | | formCode | string nullable: true  Mark of form type. | | fill | boolean  Permission to enter a form. | | sign | boolean  Permission to file a form. | | viewSent | boolean  Permission to check the filed form. |   }] |

}

* Examples:
  + Reply for HTTP status 200 OK:

{

"permissions": [

{

"mandator": {

"id": 12345678,

"type": 0

},

"formCode": "DDV\_O",

"fill": true,

"sign": false,

"viewSent": true

},

{

"mandator": {

"id": 23456789,

"type": 2

},

"formCode": "DDV\_O",

"fill": true,

"sign": true,

"viewSent": true

}

]

}

## Methods in connection with operations of the service - /api/v1/Heartbeat

### Method /api/v1/Heartbeat

* Description: the method is intended for checking the status and availability of the service.
* HTTP method: GET
* Input parameters: *-*
* Specific HTTP reply codes: /
* Output parameters:
  + Response status 200 OK:

*HeartbeatResponse{*

|  |  |
| --- | --- |
|  | *Reply to the inquiry for status and availability of the service* |
| *hostingEnvironment* | *string nullable: true*  *Mark of the implementation environment* |
| *apiVersion* | *string nullable: true*  *Version of service interface.* |
| *productVersion* | *string nullable: true*  *Service version.* |
| *serverTime* | *string($date-time)*  *Current time at the service.* |

*}*

# Definition of interface of SOAP service

For technical reasons, the SOAP service is divided into two separate services, a login service and a service for content functionalities.

Funkcionalno SOAP servis podvaja funkcionalnosti REST servisa, zato se pri tehničnih podrobnostih metod (vhodni in izhodni parametri, …) večinoma sklicujemo na ustrezne metode REST servisa. From the functional point of view, SOAP service duplicates the functionalities of the REST service, so in the technical details of the methods (input and output parameters, etc.) we mostly refer to the corresponding methods of REST service. [[1]](#footnote-1)

## consideration of errors

In the event that the SOAP service call fails, it returns an object of type Fault. Details of the error are located in the detail element, which correspond to the HTTP codes or responses returned by the REST service in case of errors.

Primer Example:

<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">

<s:Body>

<s:Fault>

<faultcode>s:Client</faultcode>

<faultstring xml:lang="en-US">Unauthorized</faultstring>

<detail>

<UnauthorizedFault xmlns="http://schemas.datacontract.org/2004/07/Endava.Edp.InvoiceBook.Web.Soap" xmlns:i="http://www.w3.org/2001/XMLSchema-instance">

<Message>Niste prijavljeni oziroma ni izbrana zastopana oseba.</Message>

</UnauthorizedFault>

</detail>

</s:Fault>

</s:Body>

</s:Envelope>

## login service

### Metoda Method LoginUsingClientCertificate

* Description: The method enables the selection of the represented taxable person, the user authenticates with a digital certificate, which is used to establish the TLS connection. The result of a successful call to the method contains an authentication token with which the user authenticates when calling the service methods for content functionalities.
* Input parameters:
  + tax number of the represented taxable person
  + type of the represented taxable person
* Output parameters at a successful call: object of type TokenResponse
* Technical details: see Method »Method /api/v1/Auth/Certificate/Representing/{taxPayerId}/{taxPayerType}«

## service for content functionalities

### Metoda Method UploadInvoiceBook

* Description: The method enables the upload of a new book of invoices for the represented taxable person. The method supports MTOM (Message Transmission Optimization Mechanism) with XOP (XML-binary Optimized Packaging) and thus enables the transfer of books of invoices in binary form (without encoding in base64). The use of MTOM when calling the method is highly recommended, especially for large books.
* Input parameters: for technical reasons, the structure of the request differs according to the equivalent REST service method: part of the input parameters is located in the request header (element soapenv:Header), while the other part is in the request body (element soapenv:Body).
  + About the parameters in the SOAP request header: the data on the uploaded invoice book correspond to the parameters schemaVersion, period, format, and correlationId in the UploadInvoiceBookRequest type.
  + Parameters in the body of the SOAP request (element soapenv:Body): data (byte[]): zip archive with the invoice book in binary format (reference to the binary part of the message if MTOM is used or base64 encoded content of the book if MTOM is not used)
* Output parameters at a successful call: object of the type UploadInvoiceBookResponse
* Technical details: see Method »Method /api/v1/InvoiceBook«
* Example of call using MTOM:

*POST http://localhost:5288/SoapService/ HTTP/1.1*

*Accept-Encoding: gzip,deflate*

*Content-Type: multipart/related; type="application/xop+xml"; start="<rootpart@soapui.org>"; start-info="text/xml"; boundary="----=\_Part\_41\_364276937.1727935652947"*

*SOAPAction: "http://tempuri.org/ISoapService/UploadInvoiceBook"*

*MIME-Version: 1.0*

*Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiI1NDYxMTMxMCIsImp0aSI6IjcwZmU0YzQ0LWRkZGQtNDk4MS05MzcwLTE2MGMwNzg0ZjNhNyIsImlhdCI6MTcyNzg3NTE0MCwiTG9naW5JZCI6IjM0MCIsIkxvZ2luVHlwZSI6IkNlcnRpZmljYXRlIiwiVXNlclByb2ZpbGVJZCI6Ijk3IiwiSW5kaXZpZHVhbFBlcnNvbklkIjoiNTQ2MTEzMTAiLCJUYXhQYXllcklkIjoiNTQ2MTEzMTAiLCJUYXhQYXllclR5cGUiOiJGTyIsIlJlcHJlc2VudGluZ0lkIjoiNTQ2MTEzMTAiLCJSZXByZXNlbnRpbmdUeXBlIjoiRk8iLCJleHAiOjE3Mjc5NjE1NDAsImlzcyI6Imh0dHA6Ly9sb2NhbGhvc3Q6NjM5MzkvIiwiYXVkIjoiaHR0cDovL2xvY2FsaG9zdDo2MzkzOS8ifQ.bvhMdskZDRgPKFhEaanIRrTb-J50XZ99tuO9toCNpgY*

*Content-Length: 1538*

*Host: localhost:5288*

*Connection: Keep-Alive*

*User-Agent: Apache-HttpClient/4.5.5 (Java/12.0.1)*

*------=\_Part\_41\_364276937.1727935652947*

*Content-Type: application/xop+xml; charset=UTF-8; type="text/xml"*

*Content-Transfer-Encoding: 8bit*

*Content-ID: <rootpart@soapui.org>*

*<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:tem="http://tempuri.org/" xmlns:end="http://schemas.datacontract.org/2004/07/Endava.Edp.InvoiceBook.Web.Models">*

*<soapenv:Header>*

*<tem:SchemaVersion>0</tem:SchemaVersion>*

*<tem:Period>*

*<!--Optional:-->*

*<end:Begin>2024-01-01</end:Begin>*

*<!--Optional:-->*

*<end:End>2024-01-31</end:End>*

*</tem:Period>*

*<tem:Format>0</tem:Format>*

*<tem:CorrelationId>lfkadjsfklj</tem:CorrelationId>*

*</soapenv:Header>*

*<soapenv:Body>*

*<tem:UploadInvoiceBookRequest>*

*<tem:data><inc:Include href="cid:1405716680238" xmlns:inc="http://www.w3.org/2004/08/xop/include"/></tem:data>*

*</tem:UploadInvoiceBookRequest>*

*</soapenv:Body>*

*</soapenv:Envelope>*

*------=\_Part\_41\_364276937.1727935652947*

*Content-Type: application/zip; name=test\_KIR\_CSV.zip*

*Content-Transfer-Encoding: binary*

*Content-ID: <1405716680238>*

*Content-Disposition: attachment; name="test\_KIR\_CSV.zip"; filename="test\_KIR\_CSV.zip"*

*….( binary content omitted)…*

*------=\_Part\_41\_364276937.1727935652947--*

### Method GetInvoiceBookStatus

* Description: The method returns data about the status of the previously uploaded invoice book.
* Input parameters:
  + edpId (string): ID of the book in eDavki system
* Output parameters at a successful call: object of the type InvoiceBookStatusResponse
* Technical details: see Method »Method /api/v1/InvoiceBook/Status/{edpId}«

### Method GetOpenPeriods

* Description: The method returns a list of open periods for the represented taxable person and the specified form code.
* Input parameters:
  + formCode (string): form type. Currently, only type 'DDV\_O'is supported.
* Output parameters at a successful call: object of the type OpenPeriodsResponse
* Technical details: see Method »Method /api/v1/Documents/OpenPeriods/{formCode}«

### Method GetPermissions

* Description: the method returns a list of permissions that the user has in the eDavki system.
* Input parameters:
  + formCode (string): optional form type label. Currently, only type 'DDV\_O'is supported.
* Output parameters at a successful call: object of the type DocumentPermissionsResponse
* Technical details: see Method »Method /api/v1/Documents/Permissions«

# Validation and processing of uploaded books

The uploaded invoice books are firstly validated by the service for syntactic and content errors. Books for which validation does not find any critical errors go into the ValidatedOk status and await transfer to the backend (status of the books can be changed after processing in the backend system). Books that contain critical errors go into the ValidationError status and consequently are not transferred to the backend; the user must upload a new book with the corrected errors.

After the validation of the uploaded book is completed, the user can obtain the validation result by calling the method /api/v1/InvoiceBook/Status/{edpId}) on REST interface (see Method /api/v1/InvoiceBook/Status/{edpId}) or Method GetInvoiceBookStatus on SOAP interface (see Method GetInvoiceBookStatus). List of errors is written in the filed ValidationMessage.

Syntax of the error list in JSON format:

[ValidationError{

|  |  |
| --- | --- |
| Type | ValidationErrorType string  Info  Warning  Error  Enum: Array [ 3 ]  ***Type of error. Errors of type Error are considered critical.*** |
| Code | ValidationErrorMessaage string  Enum: Array [ 48 ]  ***Code of error in eDavki system.*** |
| Message | string nullable: true  ***Description of error.*** |
| BookId | integer($int32) nullable: true  *Sequence number of entry in the list of invoices, which includes an error. Not defined for errors in the header.* |
| BookArrayIndex | integer($int32) nullable: true  ***Sequence number of the field in the entry, which includes an error if it is possible to define.*** |
| BookIds | [integer($int32)] nullable: true  List of sequential numbers of an entry in the invoice list that contains a warning type error. Not defined for header errors. |

}]

Depending on the result of the validation of the book, we distinguish three cases:

1. The book was validated without errors. The book gets status 2 = ValidatedOk and is transferred to the backend. The error list is empty. (it is equal »[]«).
2. The book is validated with warnings ("Type": "Warning") and without critical errors. The book gets status 2 = ValidatedOk and is transferred to the backend. If the same warning appears in multiple entries, it is recorded only once, and the corresponding sequential numbers of the entries are listed in the BookIds field. Example of a list of warnings:  
     
   [{  
    "Type": "Warning",  
    "Code": "KirFieldWarning1",  
    "Message": "Preverite obračunani DDV v poljih P14, P15 in P16 glede na vpisano vrednost v polju P7.",  
    "BookId": null,  
    "BookArrayIndex": null,  
    "BookIds": [1, 2]  
    }, {  
    "Type": "Warning",  
    "Code": "KirFieldWarning3",  
    "Message": "Glede na vpisano vrednost v polju P7 se pričakuje obračunani DDV v poljih P14, P15 in/ali P16.",  
    "BookId": null,  
    "BookArrayIndex": null,  
    "BookIds": [1]  
    }  
   ]
3. The book is validated with one or more critical errors ("Type": "Error"). Each critical error is recorded separately, even if the same error occurs in multiple entries (unlike warnings, which are combined - see above). The book gets status 3 = ValidationError and is not uploaded to the backend. Example of a list of errors:  
     
   [{  
    "Type": "Error",  
    "Code": "KirFieldP6P6DsNotMatching",  
    "Message": "Izpolnite obe polji 'P6' in 'P6DS' ali nobenega od njiju.",  
    "BookId": 1,  
    "BookArrayIndex": null  
    }, {  
    "Type": "Error",  
    "Code": "KprFieldMandatory",  
    "Message": "Za prejete račune je vrednost v polju 'P7' obvezna.",  
    "BookId": 1,  
    "BookArrayIndex": null  
    }, {  
    "Type": "Error",  
    "Code": "KirFieldP6P6DsNotMatching",  
    "Message": "Izpolnite obe polji 'P6' in 'P6DS' ali nobenega od njiju.",  
    "BookId": 2,  
    "BookArrayIndex": null  
    }, {  
    "Type": "Error",  
    "Code": "KprFieldMandatory",  
    "Message": "Za prejete račune je vrednost v polju 'P7' obvezna.",  
    "BookId": 2,  
    "BookArrayIndex": null  
    }  
   ]

# Annexes

|  |  |  |
| --- | --- | --- |
| XML scheme of invoice books:    [eDavki - XML strukt.](https://edavki.durs.si/EdavkiPortal/OpenPortal/pages/technicals/formsxml.aspx) |  | OpenApi (OAS 3.0) file of REST web service: |
| Example of an invoice book in XML format (the file is syntactically correct, but not semantically) |  | One-time login with OAuth protocol |
| JSON scheme of invoice books:    [eDavki - XML strukt.](https://edavki.durs.si/EdavkiPortal/OpenPortal/pages/technicals/formsxml.aspx) |  | WSDL SOAP of login web service |
| Example of an invoice book in JSON format (the file is syntactically correct, but not semantically) |  | WSDL SOAP of web service for functionalities |
| Specification of CSV book format: |  | List of fields: |
| Example of a file with an invoice book in CSV format (the file is syntactically correct, but not semantically). We advise you to open file with Notepad or Notepad++ |  | List of rules: |

All annexes are available also in zip file on the following link: <https://edavki.durs.si/OpenPortal/Dokumenti/DDV_KIR_KPR_en.zip>

1. An important difference is in the serialization of enum values. In REST service, they are serialized as an integer type, while in SOAP service, they are serialized as a string type. [↑](#footnote-ref-1)