How to make invoicing software e-invoice compliant

with a few simple REST API calls

Basic introduction for transformers

Service e-Invoices Online – <https://e-invoices.online>

# Revision

Initial revision: *2022-03-03*

Lastest revision: Vt0.84, *2022-02-03*

# Scope of this guide

This document is trying to show REST integration of e-invoice APIs with workflow examples and minimal overhead. Examples of HTML POST request are all using curl[[1]](#footnote-1) in terminal[[2]](#footnote-2). And there will be some examples in c#.

Contents

[I. Revision 1](#_Toc128996174)

[II. Scope of this guide 1](#_Toc128996175)

[1. Introduction: 4](#_Toc128996176)

[2. Interactive registration of your IT company 4](#_Toc128996177)

[3. Authentication 4](#_Toc128996178)

[4. WorkFlows 5](#_Toc128996179)

[Send an invoice to e-invoice service 5](#_Toc128996180)

[GetInitializedNewInvoice extension 5](#_Toc128996181)

[SaveInvoice extension 5](#_Toc128996182)

[SaveInvoice extension with additional steps 6](#_Toc128996183)

[Invoice special Types 7](#_Toc128996184)

[Save Invoice extension modifying an existing invoice 8](#_Toc128996185)

[File upload and download API 8](#_Toc128996186)

[1. Standard objects 9](#_Toc128996187)

[Invoice object 9](#_Toc128996188)

[Partial 9](#_Toc128996189)

[Invoice 9](#_Toc128996190)

[ObjectInfo 11](#_Toc128996191)

[JSON Schema 12](#_Toc128996192)

[Standard response 12](#_Toc128996193)

[Status 13](#_Toc128996194)

[Reason 13](#_Toc128996195)

[ResultType: 13](#_Toc128996196)

[1. Additional country specifics 14](#_Toc128996197)

[Montenegro (ME) 14](#_Toc128996198)

[VatCodeSC – VAT codes and other VAT exemptions 14](#_Toc128996199)

[BuyerType – Types of buyer 14](#_Toc128996200)

[BuyerLegalFormSC\_\_LegalForm - Buyers legal form 14](#_Toc128996201)

[InvSaleTypeSC\_\_Code - standard invoice sale type 14](#_Toc128996202)

[InvPaymentType – Invoice payment type 14](#_Toc128996203)

[InvType – Invoice type 15](#_Toc128996204)

[PaymentMeans - Payment means 15](#_Toc128996205)

[Invoice 15](#_Toc128996206)

[Comments 15](#_Toc128996207)

[Appendix: Standard codes 17](#_Toc128996208)

[UMCodeNameSC 17](#_Toc128996209)

[Appendix: Example of a response to GetInitializedNewInvoice API call (only invoice object included) 18](#_Toc128996210)

[Appendix: Example of a properly prepared corrective invoice object for invoice T-001932/2022-3 20](#_Toc128996211)

[Appendix: Example of a curl call to save invoice. 23](#_Toc128996212)

[Appendix: Example of a properly prepared Invoice object 24](#_Toc128996213)

[Appendix: Simplified implementation of the "Send an invoice to a service" example in C# 26](#_Toc128996214)

[Appendix: Simplified implementation of uploading a file to a service using the HTTP POST method with the multipart/form-data content type in C# 27](#_Toc128996215)

# Introduction:

Although cUrl can implement HTTP post requests to our service, this is not the preferred way. Readers of this document are expected to have the knowledge of how to implement HTTP post requests in the technology best for their invoicing software solution. But - if there is no other possibility, the curl is also an option.  
Requests to the service e-Invoices Online is generally sent in the form like this:

curl -X POST https://e-invoices.online/api/service/<extension-name>

-H "Authorization: IoT <connectionKey>:EUeInvoices"

-H "Content-Type: application/json"

-d "{<parameters>}"

Note that all apostrophes in the data object must be escaped.

1. Interactive registration of your IT company

You can register on the web page <https://e-invoices.online/> with click to signup button and choose our company is software provider with transformation option.

Interactive access to the service has two purposes:

1. In the interactive session, you can obtain[[3]](#footnote-3) your ConnectionKey, that is used in REST API calls.

2. With interactive authentication, you get access to the admin component, where you can interactively solve

the possible problems with sent invoices.

Note: ConnectionKey of your customers can also be obtained via AddCustomer API.

# Authentication

Authenticate using connection key with each call in authorization header "Authorization IoT <connectionKey>:<projectName>. As authentication is sent in plaintext, you should make sure to use encrypted connection (i.e.: https) to service.

While REST API is simpler (no cookie handling), it does not have all of the features d3.client implements, like preempting.

Example:

curl -X POST https://e-invoices.online/api/service/EUeInvoices.GetInitializedNewInvoice -H "Authorization: IoT f24d5eb8-d5bb-11ec-90fd-b2e2095ceb4f:EUeInvoices" -H "Content-Type: application/json" -d "{\"Partial\":\"APIMinimal\"}"

# WorkFlows

## Send an invoice to e-invoice service

Steps can be summaries as a three-part process. First retrieve invoice object with all three properties using GetInitializedNewInvoice extension, then add data to invoice property according to the objectinfo and JSON schema, finally send invoice to the service using SaveInvoice extension which will return id of the invoice on the service.

Once the invoice is saved, it can be viewed on the interactive e-invoices page. Depending on your package, it will generate a pdf, xml and will be sent to the TAP (tax authority portal) or Peppol. Details on the setup and interactive page layout can be found in a separate manual.

### GetInitializedNewInvoice extension

Example[[4]](#footnote-4):

curl -X POST https://e-invoices.online/api/service/EUeInvoices.GetInitializedNewInvoice -H "Authorization: IoT f24d5eb8-d5bb-11ec-90fd-b2e2095ceb4f:EuEInvoices" -H "Content-Type: application/json" -d "{"\"Partial\":\"ApiMinimal\",\"OnlyStructure\":false"}"

In response to this call, you will receive an invoice object. It contains an initialized Invoice, objectInfo and JSONSchema objects. Where Invoice object contains one initialized object for invoice and all details.

Note: In the above and the following samples, the Connection key of the demo company is used. Therefore, the sent data will transform into an e-invoice in the demo company and could be seen by all the users entered in the demo.

If you don’t want this effect, register your company, and do an integration test with your Connection key.

### SaveInvoice extension

Example:

curl -X POST https://e-invoices.online/api/service/EUeInvoices.SaveInvoice -H "Authorization: IoT f24d5eb8-d5bb-11ec-90fd-b2e2095ceb4f:EuEInvoices" -H "Content-Type: application/json" -d "{\"Partial\":\"APIMinimal\", \"Invoice\":{<InvoiceDataJSONObject>}}"

If the sent object has no errors, in response there will be id of the invoice in e-invoices service.   
An example of the response:

{

"Status": "OK",

"Result": {

"Status": "OK",

"Result": {

"Id": "2a79d755-796a-11ed-9102-c49e7815da59"

},

"ResultType": "Custom"

}

Save the Id locally as it is a mandatory field in eu\_invoices object when sending a correction of existing invoice.

### SaveInvoice extension with additional steps

Example:

curl -X POST https://e-invoices.online/api/service/EUeInvoices.SaveInvoice -H "Authorization: IoT f24d5eb8-d5bb-11ec-90fd-b2e2095ceb4f:EuEInvoices" -H "Content-Type: application/json" -d "{\"Partial\":\"APIMinimal\", \"AdditionalSteps\":[999], \"Invoice\":{<InvoiceDataJSONObject>}}"

while the the regular call just saves invoice to e-invoices, 'AdditionalSteps' allows you to get multiple results and have greater control over workflow. (e.g.: Save result from step 40 locally, before sending invoice to peppol).

|  |  |
| --- | --- |
| Id | Description |
| 35 | Invoice confirmed and locked for editing |
| 40 | Fiscalized |
| 43 | Original PDF modified (with fiscalization data) |
| 45 | Generate PDF, before sending to TAP |
| 47 | PDF set as primary attachment to invoice |
| 50 | Generate UBL specific to country |
| 55 | Generate Peppol UBL |
| 70 | Send to tax authority (TAP) |
| 80 | Send to Peppol |
| 83 | Modify pdf document wih fiscalization and sending data |
| 85 | Generate pdf document wih fiscalization and sending data |
| 999 | All steps set as default for specific country |

When saving invoice without additional steps set, the invoice is set to step 30, that represent a well formed invoice. And the result is id of invoice in database.

Steps may be skipped, but lower numbered steps can't be run after higher numbered steps. In result you will get invoice id in database and the result of last step. If error is thrown during any of the steps, you will get invoice id (if saved to database), error description and step number that has thrown it.

Some examples:

[35, 40] - confirm invoice and send it to fiscalization service. In result you will get id and all fields containing fiscalization data, as date time of fiscalization and fiscalization id.

[35, 45,47,50,70,83] - confirm invoice, generate PDF file of invoice, make it a primary attachment to be sent to TAP, generate UBL send it to TAP and modify PDF with data from TAP. As a result you will receive: id and link to PDF document with document.

[35, 47, 50, 70] - save invoice and send it to TAP. Get all data, that is important from sending invoice to TAP, like sequential number, TAP id, etc.

You can also call SaveInvoice to execute steps on existing invoice. For that you must use invoice id and Additionalsteps parameter.

curl -X POST https://e-invoices.online/api/service/EUeInvoices.SaveInvoice -H "Authorization: IoT f24d5eb8-d5bb-11ec-90fd-b2e2095ceb4f:EuEInvoices" -H "Content-Type: application/json" -d "{\"AdditionalSteps\":[70], \"Id\":\"9337614d-8764-11ed-9102-c49e7815da59\"}}"

example of error response:

{

"Status": "OK",

"Result": {

"Status": "Error",

"Step": 70,

"Reason": "'Invoice' object was expected to contain a document URL, but was not found under expected name 'XMLSpecific'.",

"Id": "9337614d-8764-11ed-9102-c49e7815da59",

"Result": {

"Invoice": null,

"ObjectInfo": null,

"JSONSchema": null,

"Id": "9337614d-8764-11ed-9102-c49e7815da59"

},

"ResultType": "Invoice"

}

}

First, Status and Result are the REST API response (API wrapper). Second, the response of the extension.

In this instance, step 70 was called before step 50 and the UBL file was not manually added to the invoice. The status is "Error" and the Reason has a detailed response from the extension. Also, Result and Id are included in the response. We can determine that step 70 has thrown an error because we didn't add the UBL to the invoice.

### Invoice special Types

This section will focus on differences of special types as of standard INVOICE type. Type is defined in InvTypeCodeNameSC\_\_Code field. You will also need ApiNormal Invoice object, as ApiMinimal does not have all needed fields. You also send them via SaveInvoice extension.

#### Corrective Invoice

Corrective invoice is currently supported only as a full cancelation of a previous invoice. There are several types of corrective invoice CORRECTIVE, CORRECTIVEADVANCE and CORRECTIVECREDITNOTE. The only difference between them is what document they reference. To cancel a previous invoice you will need to add a reference to it in InvPreviousIssueDate and InvPreviousNum[[5]](#footnote-5) fields.

#### Creditnote

A credit note is set as the CREDITNOTE type. You will need to fill in an additional EU\_Invoices\_CN\_IICRefs section, which contains information about previous invoices and their items that this credit note references.

#### Advance

An advance is a prepayment invoice and is set as the ADVANCE type. There are no special fields specifically designated for this type.

### Save Invoice extension modifying an existing invoice

If you have content errors in invoice object, that was sent to e-invoices, but the invoice was saved to database (you got Id in response), you can correct them by calling SaveInvoice extension again with the correct invoice object, that also contains e-invoices 'Id' property in 'eu\_invoices' object. This will call MODIFY function instead of CREATE.

## File upload and download API

This API endpoint is not tied to JSON-RPC. It responds to an ordinary HTTP POST request for file upload.

Example:

curl -X POST https://e-invoices.online/file?category=common -H "Authorization: IoT f24d5eb8-d5bb-11ec-90fd-b2e2095ceb4f:EUeInvoices" -H "Content-Type: multipart/form-data" -F "file=@invoice.pdf;type=application/pdf"

In response to a successful upload request, the service will return a JSON structure, similar to this:

{ "Status": "OK", "Result": { "Id": "18c8c4b4-72f2-48a2-9f15-53b8e415b946" } }

The "Id" attribute contains a unique file identifier for the newly created BLOB, which can be used to address the BLOB by URL. To download the new BLOB as file in the future, we would simply issue a HTTP GET request to the appropriate File URL:

[https://e-invoices.online/file/18c8c4b4-72f2-48a2-9f15-53b8e415b946](https://test.dopinus.com/file/18c8c4b4-72f2-48a2-9f15-53b8e415b946)

Example:

curl -X GET https://e-invoices.online/file/f2e83764-68e9-401a-8ed5-afa7cbce26eb -H "Authorization: IoT f24d5eb8-d5bb-11ec-90fd-b2e2095ceb4f:EUeInvoices" -o invoice\_dl.pdf

NOTE: You can find a more complete and fail-safe file upload example code with boundary header in appendix.

# Standard objects

## Invoice object

Is a JSON object, that has contains three properties:

{

    "Invoice" : invoice data,

    "ObjectInfo": Additional information of invoice data enteties ie. EU\_Invoices, EU\_Invoice\_Items, etc.,

    "JSONSchema" : Json schema object, that can be used for invoice object validation.

}

### Partial

Partial parameter defines scope of JSON schema and invoice data and ObjectInfo. It string value represents enum values ApiMinimal, APINormal.

ApiMinimal: represents an invoice object that has minimal fields for INVOICE and ADVANCE documents.

ApiNormal: has additional fields to ApiMinimal, for more complex invoices, like CREDITNOTE and CORECTIVE.

This document will focus exclusively on ApiMinimal object, that is used to define minimal set of fields for invoice that is sent to API[[6]](#footnote-6).

### Invoice

Invoice object contains an information about the invoice such as its number, type, issue date, currency, and total amount, as well as details about the buyer and seller. In the details section, there is also an array of items on the invoice called EU\_Invoices\_Items, and an array of payment instructions for the invoice called EU\_Invoices\_PaymentInstrs.

Contains invoice data object, that must be validated with JSON schema object. Detailed definition of fields are found in ObjectInfo object.

|  |  |  |  |
| --- | --- | --- | --- |
| Property | | | Description |
| EU\_Invoices | | | Main invoice object (master entity) |
|  | FiscOperatorRegCode | | Operator code in fiscalization service (can be overwritten if registry is maintained manually) |
|  | BuyerFormalName | | Buyer formal name |
|  | BuyerTaxNum | | Buyer tax number |
|  | BuyerRegNum | | Buyer registration number |
|  | BuyerIsBudget | | Buyer is registered as public budget user |
|  | BuyerBudgetNum | | Buyer public budget number (mandatory if buyer is registered as public budget user) |
|  | BuyerUndefinedNum | | Optional buyer identifier when buyer doesn't have tax number. (e.g.: passport number, car registration number, ... ). Number isn't checked against any TAP registry, but should be unique buyer identifier. |
|  | BuyerAddress1 | | Buyer street and house number |
|  | BuyerPostCode | | Buyer postal code |
|  | BuyerCityName | | Buyer city name |
|  | BuyerCounCode | | Buyer country code |
|  | BuyerOrderRef | | Buyer order reference number |
|  | BuyerTypeSC\_\_BuyerType | | Type of buyer (Domestic, ForeignEU, ForeignOutEU) |
|  | BuyerLegalFormSC\_\_LegalForm | | Buyers legal form (LegalEntity, SoleProprietor, Unidentified) |
|  | InvNum | | Invoice number |
|  | InvSaleTypeSC\_\_Code | | Standard invoice sale type code (Wholesale, Retailsale) |
|  | InvPaymentTypeCodeNameSC\_\_Code | | Invoice payment type (CASH, NONCASH, COMBINE) |
|  | InvDueDate | | Invoice due date |
|  | InvEndDate | | Invoice end date |
|  | InvIssueDate | | Invoice issue date |
|  | DeliveryDateActual | | Invoice delivery date |
|  | InvAllowPercent | | Invoice discount percent |
|  | InvAmountInclVatInp | | Invoice amount including VAT |
|  | InvCurrencyCode | | Invoice currency code (e.g.: EUR) |
|  | ExchangeRate | | Exchange rate between invoice currency and countries main currency |
|  | InvNote | | Invoice additional notes |
|  | InvSigner | | Invoice signers name (e.g.: organization CEO's name) |
|  | InvStartDate | | Invoice start date |
|  | InvTotalVatAmountCCInp | | Total invoice VAT amount |
|  | InvTypeCodeNameSC\_\_Code | | Invoice type (INVOICE, CREDITNOTE, CORECTIVE, ADVANCE, ...) |
|  | PDFOriginal | | File GUID link in key value (KV) store (must be uploaded to KV store before save invoice call) |
|  | Id | | Id of the invoice in e-invoices service |
|  | EU\_Invoices\_Items | | Invoice line items |
|  |  | ItemName | Item Name |
|  |  | ItemNetPrice | Item NET price |
|  |  | ItemVatCodeSC\_\_VatCode | Item VAT code |
|  |  | ItemVatRate | Item VAT rate |
|  |  | Quantity | Item quantity |
|  |  | UMCodeNameSC\_\_Code | Item unit of measure code (standard units described in appendix) |
|  |  | LineAllowPercent | Line Item discount percent |
|  |  | LineExciseAmount | Line Item excise amount |
|  |  | ItemRetailPriceInp | Line Item retail price |
|  | EU\_Invoices\_PaymentInstrs | | Invoice payment instructions/options |
|  |  | PayDocDate | Payment document date |
|  |  | PayeeAccountType | Payment account type (IBAN, BBAN) for payment by CREDITTRANSFER means |
|  |  | PayHolderName\_Ref | Account holder name reference (e.g.: Credit card owner when payment by CARD means) |
|  |  | PaymentAmount | Payment amount |
|  |  | PaymentMeansCodeNSC\_\_Code | Payment means code (CREDITTRANSFER, CARD, CASH, PAYPAL, ADVANCE) |
|  |  | PayNetworkProvider | Payment network provider (e.g.: BIC number for IBAN account type) |
|  |  | PayNumber | Payment account number (e.g.: IBAN when payment by CREDITTRANSFER means) |

Additional fields in ApiNormal

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Property | | | | Description |
| EU\_Invoices | | | | Main invoice object (master entity) |
|  | InvPreviousIssueDate | | | Original invoice issue date |
|  | InvPreviousNum | | | Original invoice reference number |
|  | EU\_Invoices\_AddRefDocs | | | Additional reference documents |
|  |  | AddDocDesc | | Document description |
|  |  | AddDocRef | | Document reference (e.g.: document number) |
|  |  | AttFileName | | Filename without path |
|  |  | AttFileURL | | Short file URL to document on KV store (e.g: "File/0821370c-5646-4590-94b1-b4ba12689bfb?inline=true"). |
|  |  | AttMimeCode | | MIME type code (e.g.: "application/pdf") |
|  | EU\_Invoices\_CN\_IICRefs | | | Credit note reference details |
|  |  | InvAmountInclVat | | Reference invoice amount including VAT |
|  |  | InvIssueDate | | Reference invoice issue date |
|  |  | InvNum | | Reference invoice number |
|  |  | EU\_Invoices\_CN\_IICRefs\_Items | | Referenced credit note line items |
|  |  |  | LineNetAmount | Reference invoice line item NET amount |
|  |  |  | ItemVatCodeSC\_\_VatCode | Reference invoice line item VAT code (20, 22, ...) |

These fields are used for advanced invoice types (e.g.: CREDITNOTE)

### ObjectInfo

Defines every field in invoice data object with its name, entity, mandatory flag, internal type, .NET type, JSON type, etc.

|  |  |  |  |
| --- | --- | --- | --- |
| Property | | | Description |
| Entity Name | | |  |
|  | Field Name | | Object containing field info |
|  |  | Source | Info about source from which field is generated (SYSTEM, INVOICE, MAPPEDINV, CONNECTED, MAPPEDSYS) |
|  |  | Mandatory | Boolean property that signals if field is mandatory |
|  |  | Type | Field type (string, decimal, bool, integer, date, datetime) |
|  |  | TypeNet | Standard type in .NET (bool, DateTime, decimal, int, string) |
|  |  | TypeDDD | Internal type in DDD (Connection, Date, DateTime, Logical, Numeric, Text) |
|  |  | Pattern | Regex pattern if defined. |

outline:

{

    "ObjectInfo": {

        "table name 1" : {

            "TableField1" : {

                "Property1" : value1,

                "Property12 : value2,

                ...

            },

            "TableField2" : {

                "Property1" : value1,

                "Property12 : value2,

                ...

            }

            ...,

        },

        "table name 2" : {

            "TableField1" : {

                "Property1" : value1,

                "Property12 : value2,

                ...

            },

            ...,

        }

    }

}

Example of a DeliveryDateActual field in eu\_invoices entity.

{

"ObjectInfo": {

"EU\_Invoices": {

"DeliveryDateActual": {

"Source": "SYSTEM",

"Mandatory": false,

"Type": "date",

"TypeNET": "DateTime",

"TypeDDD": "Date",

"Pattern": "[0-9]{4}-[0-9]{2}-[0-9]{2}"

}, ... /\*other fields\*/

}, ... /\*other entities\*/

}}

### JSON Schema

A Newtonsoft.Json.Schema.JSchema object, that can be used for invoice object validation before sending it to API.

Example of validation.

var JSONSchema = (JObject)Result["Invoice.JSONSchema"];

var schema = JSchema.Parse(JSONSchema.ToString());

if(!((JObject)Result["Invoice.Invoice"]).IsValid(schema, out errors))

{

    throw new Exception("Invalid schema."+string.Join(", ",errors));

}

## Standard response

Extensions have a standard response.

{

    "Status": Response status that determines success of extension.

    "Reason": Reason why extension has failed.

    "Result": Result of an extension in result type.

    "ResultType": Defines result type

}

A call to REST API also has standard response

{

    "Status": Response status that determines success of API.

    "Reason": Reason why API has failed.

    "Result": Result that is returned from extension.

    "Code": Error code

}

Typical response from REST API looks like double wrapped standard response. Extension response wrapped by REST API response.

{

"Status": "OK", //response of server that auth are ok, and that extension is found and has been executed.

"Result": {

"Status": "Error", //response of extension, that describes an error on step 70

"Step": 70,

"Reason": "'Invoice' object was expected to contain a document URL, but was not found under expected name 'XMLSpecific'.",

}

}

example of response with invalid credentials

{

"Status": "Error",

"Reason": "Invalid device ID or project name.",

"Code": 401

}

See appendix for more examples.

### Status

Response status. Its string represents enum value OK, Warning and Error.

OK: API successfully completed. Result and result type are defined.

Warning: API successfully completed, but with some warnings. Result, result type and reason are defined.

Error: API didn't complete. Reason is defined.

### Reason

Reason for warning or error.

### ResultType:

Defines type of result and is represented with string value of enum Custom, Invoice, Registration.

Custom: Contains key value pairs that are consistent with fields in eu\_invoices entity (e.g.: Id).

Invoice : A standard Invoice object with Invoice, ObjectInfo and JSONSchema objects.

Registration: Response of new customer registration. (defined in AddCustomer extension description)

# Additional country specifics

## Montenegro (ME)

### VatCodeSC – VAT codes and other VAT exemptions

|  |  |  |
| --- | --- | --- |
| VatCode | Description in ME language | Rate |
| 21 | Opšta stopa 21% | 21 |
| 7 | Snižena stopa 7% | 7 |
|  | PDV nije obračunat (Mali obveznici - čl. 42. Zakona o PDV-u). | 0 |
| 0 | (Oslobođeno) Nulta stopa. | 0 |
| 0-Export | Oslobođen izvoz 0% | 0 |
| VAT\_CL17 | Oslobođeno PDV-a prema čl. 17. st. 4. Zakona o PDV-u. Mjesto prometa usluga. | 0 |
| VAT\_CL20 | Oslobođeno PDV-a prema čl. 20. Zakona o PDV-u. Poreska osnovica i ispravka poreske osnovice. | 0 |
| VAT\_CL26 | Oslobođeno PDV-a prema čl. 26. Zakona o PDV-u. Oslobođenja od javnog interesa. | 0 |
| VAT\_CL27 | Oslobođeno PDV-a prema čl. 27. Zakona o PDV-u. Ostala oslobođenja. | 0 |
| VAT\_CL28 | Oslobođeno PDV-a prema čl. 28. Zakona o PDV-u.Oslobođenja kod uvoza proizvoda. | 0 |
| VAT\_CL29 | Oslobođeno PDV-a prema čl. 29. Zakona o PDV-u. Oslobođenja kod privremenog uvoza proizvoda. | 0 |
| VAT\_CL30 | Oslobođeno PDV-a prema čl. 30. Zakona o PDV-u. Posebna oslobođenja. | 0 |
| VAT\_CL44 | Oslobođeno PDV-a prema čl. 44. Zakona o PDV-u. Usluge putničkih agencija. | 0 |

### BuyerType – Types of buyer

|  |  |
| --- | --- |
| BuyerType | Description in ME language |
| Domestic | Domaci |
| ForeignEU | Strani EU |
| ForeignOutEU | Strani izvan EU |

### BuyerLegalFormSC\_\_LegalForm - Buyers legal form

|  |  |
| --- | --- |
| BuyerLegalForm | Description in ME language |
| LegalEntity | Pravno lice |
| SoleProprietor | Preduzetnik |
| Unidentified | Neidentifikovani |

### InvSaleTypeSC\_\_Code - standard invoice sale type

|  |  |
| --- | --- |
| InvSaleType | Description in ME language |
| Wholesale | Veleprodaja |
| Retailsale | Maloprodaja |

### InvPaymentType – Invoice payment type

|  |  |
| --- | --- |
| InvPaymentType | Description in ME language |
| CASH | Gotovinski |
| NONCASH | Bezgotovinski |

### InvType – Invoice type

|  |  |
| --- | --- |
| InvType | Description in ME language |
| ADVANCE | Avansni račun |
| CORRECTIVE | Korektivni račun |
| CORRECTIVEADVANCE | Korektivni avansni račun |
| CORRECTIVECREDITNOTE | Korektivno knjižno odobrenje |
| CREDITNOTE | Knjižno odobrenje |
| INVOICE | Račun |
| PROFORMA | Predračun |

### PaymentMeans - Payment means

|  |  |
| --- | --- |
| Payment Means | Description in ME language |
| CARD | Kartica fizičkog lica |
| CARDBUSINESS | Kartica preduzeća |
| CASH | Gotovina |
| CASHOTH | Ostalo-gotovina |
| CREDITTRANSFER | Virman |
| FACTORING | Factoring |
| GIFTCARD | Poklon kartica |
| ORDER | Narudžba |
| OTHER | Ostalo |
| VOUCHER | Vaučer |

### Invoice

|  |  |  |  |
| --- | --- | --- | --- |
| Property | | | Description |
| EU\_Invoices | | | Main invoice object (master entity) |
|  | InvSubseqDelTypeCodeNameSC\_\_Code | | Fiscalization delay reason type. TECHNICALERROR/NOINTERNET/BOUNDBOOK/SERVICE/BUSINESSNEEDS |
|  | InvParagonBlockNum | | Number of bound book. Must be filled when reason for delayed input is BOUNDBOOK |
|  | InvPreviousFiscCalcNum | | Original invoice fiscalization number |
|  | EU\_Invoices\_CN\_IICRefs | | Credit note reference details |
|  |  | FiscCalcNum | Reference invoice fiscalization number |

### Comments

**EU\_Invoices**

**FiscOperatorRegCode**

Normally the key is evaluated based on connection key used. But if operators are maintained in local registry, the field must contain the number of the operator obtained from TAP.

**InvSubseqDelTypeCodeNameSC\_\_Code**

If the invoice was sent to fiscalization with delay greater than 10 minutes, this field is required and subject to tax inspection.

|  |  |
| --- | --- |
| Code | Description |
| TECHNICALERROR | Technical error (up to 48 hours) |
| NOINTERNET | Device operates in area without internet service (up to 7 days) |
| BOUNDBOOK | Bound book is used, because device isn't working and fiscalization request couldn't be generated. (up to 7 days) |
| SERVICE | TAP service error. (up to 48 hours) |
| BUSINESSNEEDS | Subsequent sending is conditioned by business needs (up to 7 days) |

**EU\_Invoices\_PaymentInstrs**

**PaymentMeansCodeNSC\_\_Code**

Montenegro is not in SEPA system, thusCREDITTRANSFERSEPA and DEBITTRANSFERSEPA aren’t valid values.

# Appendix: Standard codes

## UMCodeNameSC

|  |  |
| --- | --- |
| Id | Description |
| box | Box |
| cm | centimetre |
| dag | decagram |
| day | day |
| g | gram |
| hl | hectolitre |
| hour | hour |
| kg | kilogram |
| km | kilometre |
| kW | kilowatt |
| kWh | kilowatt hour |
| l | litre |
| m | metre |
| m2 | square metre |
| m3 | cubic metre |
| min | minute [unit of time] |
| mm | millimetre |
| month | month |
| pack | piece |
| pal | Pallet |
| piece | piece |
| s | second [unit of time] |
| size | piece |
| t | tonne (metric ton) |
| value | monetary value |
| week | week |
| year | year |

# Appendix: Example of a response to GetInitializedNewInvoice API call (only invoice object included)

{"id":66355471,"result":{

"Status": "OK",

"Result": {

"Invoice": {

"EU\_Invoices": {

"BuyerCounCode": "RS",

"InvNote": null,

"InvAmountInclVatInp": 0,

"DeliveryTaxNum": null,

"PDFOriginal": null,

"InvDueDate": "2023-02-25",

"BuyerVatNum": null,

"InvAllowPercent": 0.0,

"BuyerBudgetNum": null,

"BuyerIsBudget": false,

"BuyerUndefinedNum": null,

"InvCurrencyCode": "RSD",

"BuyerIsVat": false,

"BuyerRegNum": null,

"BuyerTaxNum": null,

"BuyerCityName": null,

"BuyerPostCode": null,

"ExchangeRate": 1.0,

"InvEndDate": "2023-02-10",

"InvTotalVatAmountCCInp": 0,

"BuyerFormalName": null,

"InvStartDate": "2023-02-10",

"BuyerAddress1": null,

"BuyerSendToCIR": false,

"DeliveryName": null,

"DeliveryAddress1": null,

"DeliveryCityName": null,

"DeliveryPostCode": null,

"DeliveryCounCode": null,

"BuyerOrderRef": "Nar:",

"InvSigner": "Testni potpisnik",

"InvPreviousNum": null,

"InvPreviousIssueDate": null,

"DeliveryRegNum": null,

"InvSaleTypeSC\_\_Code": "Wholesale",

"InvPaymentTypeCodeNameSC\_\_Code": "NONCASH",

"BuyerLegalFormSC\_\_LegalForm": "LegalEntity",

"InvTypeCodeNameSC\_\_Code": "INVOICE",

"BuyerTypeSC\_\_BuyerType": "Domestic",

"FiscOperatorRegCode": null,

"DeliveryDateActual": "2023-02-10",

"InvIssueDate": null,

"InvNum": null,

"\_details": {

"EU\_Invoices\_Items": [

{

"ItemNetPrice": 1.0,

"ItemName": "Stavka",

"Quantity": 1.0,

"ItemVatRate": null,

"LineExciseAmount": 0.0,

"LineAllowPercent": 0.0,

"ItemVatCodeSC\_\_VatCode": "20",

"UMCodeNameSC\_\_Code": "piece",

"ItemRetailPriceInp": null

}

],

"EU\_Invoices\_PaymentInstrs": [

{

"PayDocDate": null,

"PaymentAmount": 0,

"PayHolderName\_Ref": null,

"PaymentMeansCodeNSC\_\_Code": "CREDITTRANSFER",

"PayNetworkProvider": "CONARS22XXX",

"PayeeAccountType": "BBAN",

"PayNumber": "603-123-56666"

}

],

"EU\_Invoices\_AddRefDocs": [

{

"AddDocRef": null,

"AddDocDesc": null,

"AttMimeCode": null,

"AttFileName": null,

"AttFileURL": null

}

],

"EU\_Invoices\_CN\_IICRefs": [

{

"InvAmountInclVat": 0,

"InvIssueDate": null,

"InvNum": null

}

]

}

}

}

},

"ResultType": "Invoice"

}}

# Appendix: Example of a properly prepared corrective invoice object[[7]](#footnote-7) for invoice T-001932/2022-3

{

"EU\_Invoices": {

"InvPaymentTypeCodeNameSC\_\_Code": "NONCASH",

"InvNum": "T-001932/2022-3-S1",

"InvTypeCodeNameSC\_\_Code": "CORRECTIVE",

"InvIssueDate": "2022-12-28T17:06:22.000",

"InvSaleTypeSC\_\_Code": "Retailsale",

"DeviceId": "fa199eac-a5d1-11ec-90fb-faa8f752a7b9",

"InvStartDate": "2022-12-28T00:00:00.000",

"InvEndDate": "2022-12-28T00:00:00.000",

"InvDueDate": "2023-01-12T00:00:00.000",

"InvNote": "napomena 212",

"InvCurrencyCode": "RSD",

"BuyerOrderRef": "Nar:",

"BuyerSendToCIR": false,

"BuyerLegalFormSC\_\_LegalForm": "LegalEntity",

"BuyerTypeSC\_\_BuyerType": "Domestic",

"BuyerFormalName": "AGRO-BANEKS DOO ",

"BuyerAddress1": "KOCELJEVA",

"BuyerCityName": "KOCELJEVA",

"BuyerPostCode": "15220",

"BuyerCounCode": "RS",

"BuyerIsVat": false,

"BuyerIsBudget": false,

"BuyerTaxNum": "104512861",

"BuyerRegNum": "20175745",

"DeliveryDateActual": "2022-12-28T00:00:00.000",

"DeliveryCounCode": "RS",

"InvTotalVatAmountCCInp": -620.87,

"InvAmountInclVatInp": -3725.19,

"InvAllowPercent": 2,

"ExchangeRate": 1,

"InvSigner": "Testni potpisnik",

"InvPreviousNum": "T-001932/2022-3",

"InvPreviousIssueDate": "2022-12-28T17:05:54.000",

"BuyerVatNum": null,

"BuyerUndefinedNum": null,

"BuyerBudgetNum": null,

"DeliveryAddress1": null,

"DeliveryCityName": null,

"DeliveryPostCode": null,

"DeliveryName": null,

"DeliveryTaxNum": null,

"DeliveryRegNum": null,

"PDFOriginal": null,

"FiscOperatorRegCode": null,

"\_details": {

"EU\_Invoices\_Items": [

{

"ItemName": "Stavka 1",

"Quantity": -1,

"UMCodeNameSC\_\_Code": "piece",

"ItemNetPrice": 1,

"ItemRetailPriceInp": 1.2,

"ItemVatRate": 0,

"ItemVatCodeSC\_\_VatCode": "20",

"LineAllowPercent": 0,

"LineExciseAmount": 0

},

{

"ItemName": "Stavka 2",

"Quantity": -2,

"UMCodeNameSC\_\_Code": "piece",

"ItemNetPrice": 1666.6667,

"ItemRetailPriceInp": 2000,

"ItemVatRate": 0,

"ItemVatCodeSC\_\_VatCode": "20",

"LineAllowPercent": 5,

"LineExciseAmount": 0

}

],

"EU\_Invoices\_PaymentInstrs": [

{

"PaymentMeansCodeNSC\_\_Code": "ADVANCEOTH",

"PayeeAccountType": "BBAN",

"PayNumber": "123123",

"PayDocDate": "2022-12-30T00:00:00.000",

"PaymentAmount": -3725.19,

"PayNetworkProvider": null,

"PayHolderName\_Ref": null

}

],

"EU\_Invoices\_AddRefDocs": [

{

"AddDocRef": "T-001932/2022-3-S1PDF",

"AttFileName": "2dba2e1af7ab4064b0b40c094c37e547.pdf",

"AttMimeCode": "application/pdf",

"AttFileURL": "File/67d83353-2b4d-4bff-a787-195144cc2704?inline=true",

"AddDocDesc": null

}

],

"EU\_Invoices\_CN\_IICRefs": [

{

"InvAmountInclVat": 0,

"InvNum": null,

"InvIssueDate": null

}

]

}

}

}

# Appendix: Example of a curl call to save invoice.

curl -X POST https://e-invoices.online/api/service/EUeInvoices.SaveInvoice -H "Authorization: IoT f24d5eb8-d5bb-11ec-90fd-b2e2095ceb4f:EuEInvoices" -H "Content-Type: application/json" -d "{\"Invoice\":{\"EU\_Invoices\":{\"InvPaymentTypeCodeNameSC\_\_Code\":\"NONCASH\",\"InvNum\":\"2022/001927\",\"InvTypeCodeNameSC\_\_Code\":\"INVOICE\",\"InvIssueDate\":\"2022-12-27T11:18:03.000\",\"InvSaleTypeSC\_\_Code\":\"Wholesale\",\"InvStartDate\":\"2022-12-01T00:00:00.000\",\"InvEndDate\":\"2022-12-27T00:00:00.000\",\"InvDueDate\":\"2023-01-11T00:00:00.000\",\"InvCurrencyCode\":\"RSD\",\"BuyerOrderRef\":\"Nar: 123\",\"BuyerLegalFormSC\_\_LegalForm\":\"LegalEntity\",\"BuyerTypeSC\_\_BuyerType\": \"Domestic\",\"BuyerFormalName\":\"TailoredSaas d.o.o., Beograd\",\"BuyerAddress1\":\"Jurija Gagarina 151a T.C. Piramida, sprat 1, lokal 57 \",\"BuyerCityName\":\"Novi Beograd\",\"BuyerPostCode\":\"11177\",\"BuyerCounCode\":\"RS\",\"BuyerIsBudget\":false,\"BuyerTaxNum\":\"112742026\",\"BuyerRegNum\":\"21730327\",\"DeliveryDateActual\":\"2022-12-27T00:00:00.000\",\"InvTotalVatAmountCCInp\":51.2,\"InvAmountInclVatInp\":307.2,\"InvAllowPercent\":0,\"ExchangeRate\":1,\"InvSigner\":\"Testni potpisnik\",\"InvNote\":null,\"BuyerUndefinedNum\":null,\"BuyerBudgetNum\":null,\"InvPreviousNum\":null,\"InvPreviousIssueDate\":null,\"PDFOriginal\":null,\"FiscOperatorRegCode\":null,\"\_details\":{\"EU\_Invoices\_Items\":[{\"ItemName\":\"Stavka\",\"Quantity\":1,\"UMCodeNameSC\_\_Code\":\"piece\",\"ItemNetPrice\":256,\"ItemVatRate\":0,\"ItemVatCodeSC\_\_VatCode\":\"20\",\"LineAllowPercent\":0,\"LineExciseAmount\":0,\"ItemRetailPriceInp\":null}],\"EU\_Invoices\_PaymentInstrs\":[{\"PaymentMeansCodeNSC\_\_Code\":\"CREDITTRANSFER\",\"PayeeAccountType\":\"BBAN\",\"PayNumber\":\"603-123-56666\",\"PayNetworkProvider\":\"CONARS22XXX\",\"PayHolderName\_Ref\":\"2323\",\"PaymentAmount\":307.2,\"PayDocDate\":null}]}}},\"XML\":null,\"Partial\":\"ApiMinimal\",\"Id\":null,\"AdditionalSteps\":null,\"BackendStep\":null,\"Destination\":\"IEI\",\"RequestId\":null}"

# Appendix: Example of a properly prepared Invoice object[[8]](#footnote-8) [[9]](#footnote-9)

{

"EU\_Invoices": {

"InvPaymentTypeCodeNameSC\_\_Code": "NONCASH",

"InvNum": "2022/001927",

"InvTypeCodeNameSC\_\_Code": "INVOICE",

"InvIssueDate": "2022-12-27T11:18:03.000",

"InvSaleTypeSC\_\_Code": "Wholesale",

"DeviceId": "45413c22-d690-11ec-90fd-b2e2095ceb4f",

"InvStartDate": "2022-12-01T00:00:00.000",

"InvEndDate": "2022-12-27T00:00:00.000",

"InvDueDate": "2023-01-11T00:00:00.000",

"InvCurrencyCode": "RSD",

"BuyerOrderRef": "Nar: 123",

"BuyerSendToCIR": false,

"BuyerLegalFormSC\_\_LegalForm": "LegalEntity",

"BuyerTypeSC\_\_BuyerType": "Domestic",

"BuyerFormalName": "TailoredSaas d.o.o., Beograd",

"BuyerAddress1": "Jurija Gagarina 151a T.C. Piramida, sprat 1, lokal 57 ",

"BuyerCityName": "Novi Beograd",

"BuyerPostCode": "11177",

"BuyerCounCode": "RS",

"BuyerIsVat": false,

"BuyerIsBudget": false,

"BuyerTaxNum": "112742026",

"BuyerRegNum": "21730327",

"DeliveryDateActual": "2022-12-27T00:00:00.000",

"InvTotalVatAmountCCInp": 51.2,

"InvAmountInclVatInp": 307.2,

"InvAllowPercent": 0,

"ExchangeRate": 1,

"InvSigner": "Testni potpisnik",

"InvNote": null,

"BuyerVatNum": null,

"BuyerUndefinedNum": null,

"BuyerBudgetNum": null,

"DeliveryAddress1": null,

"DeliveryCityName": null,

"DeliveryPostCode": null,

"DeliveryCounCode": null,

"DeliveryName": null,

"DeliveryTaxNum": null,

"DeliveryRegNum": null,

"InvPreviousNum": null,

"InvPreviousIssueDate": null,

"PDFOriginal": null,

"FiscOperatorRegCode": null,

"\_details": {

"EU\_Invoices\_Items": [

{

"ItemName": "Stavka",

"Quantity": 1,

"UMCodeNameSC\_\_Code": "piece",

"ItemNetPrice": 256,

"ItemVatRate": 0,

"ItemVatCodeSC\_\_VatCode": "20",

"LineAllowPercent": 0,

"LineExciseAmount": 0,

"ItemRetailPriceInp": null

}

],

"EU\_Invoices\_PaymentInstrs": [

{

"PaymentMeansCodeNSC\_\_Code": "CREDITTRANSFER",

"PayeeAccountType": "BBAN",

"PayNumber": "603-123-56666",

"PayNetworkProvider": "CONARS22XXX",

"PayHolderName\_Ref": "2323",

"PaymentAmount": 307.2,

"PayDocDate": null

}

],

"EU\_Invoices\_AddRefDocs": [

{

"AddDocRef": "2022/001927PDF",

"AttFileName": "593dd40493134fae8e4b6e1fd2f27232.pdf",

"AttMimeCode": "application/pdf",

"AttFileURL": "File/6004bb85-867d-4cd2-8fb8-af2eec6b2100?inline=true",

"AddDocDesc": null

}

],

"EU\_Invoices\_CN\_IICRefs": [

{

"InvAmountInclVat": 0,

"InvNum": null,

"InvIssueDate": null

}

]

}

}

}

# Appendix: Simplified implementation of the "Send an invoice to a service" example in C#

// JsonRPCClient and InvoiceBuilder are left to the user to implement

// call get new initialized invoice

info = (JObject)client.CallSingle("Extension.Execute", new Dictionary<string, object>

{

    { "extension", "EUeInvoices.GetInitializedNewInvoice" },

    {

        "args", new Dictionary<string, object>

        {

            { "Partial", "APIMinimal" },

        }

    }

});

if (info["Status"] != "OK")

{

    throw new Exception("Error retrieving new initialized invoice");

}

var objectInfo = (JObject)info["Result.ObjectInfo"];

var jsonSchema = (JObject)info["Result.JSONSchema"];

var newInvoice = (JObject)info["Result.Invoice"];

//this is custom mapping an parsing of objects.

var invoiceBuilder = new InvoiceBuilder(newInvoice, data, objectInfo);

invoiceBuilder.Build();

var invoice = invoiceBuilder.FinalInvoice;

//we can check invoice with schema before sending it to service.

var schema = JSchema.Parse(jsonSchema.ToString());

if(!invoice.IsValid(schema, out errors))

{

    throw new Exception("Invalid schema."+string.Join(", ",errors));

}

//if we have a PDF invoice file that we would like to add to invoice we must first upload blob to server.

//d3.client uses a simple http REST wrapper function to upload files to KV store

var uploadResult = RestClient.UploadFileAsMultipartFormData(ServerUrl, cookie, inputFileName, contentType: "application/pdf", metadata: null);

if (!uploadResult.Succeeded)

{

    throw new Exception("File wasn't uploaded successfully!");

}

AddToInvoice(invoice, "OriginalPDF",uploadResult.Value.Id)

//Save invoice to service

info = (JObject)client.CallSingle("Extension.Execute", new Dictionary<string, object>

{

    { "extension", "EUeInvoice.SaveInvoice" },

    {

        "args", new Dictionary<string, object>

        {

            { "Invoice" , invoice },

            { "Partial", "APIMinimal" }

        }

    }

});

if (info["Status"] != "OK" && info["Status"] != "Warning")

{

    throw new Exception("Error saving invoice" + info["Reason"]);

}

//In result we have a complete Invoice object, that has recalculated fields needed for invoice service. Including id, that can be used to reference in future calls.

var CompleteInvoice = (string)info["Result.Invoice"];

//Do something, like save ID to data.

# Appendix: Simplified implementation of uploading a file to a service using the HTTP POST method with the multipart/form-data content type in C#

public static string UploadFileAsMultipartFormData(string serverBaseUrl, string connectionKey, string fileName, string contentType, NameValueCollection metadata)

{

    if (!File.Exists(fileName))

    {

        throw new Exception("File not found: " + fileName);

    }

    if (string.IsNullOrEmpty(connectionKey))

    {

        throw new ArgumentNullException(nameof(connectionKey));

    }

    var fileApiUrl = serverBaseUrl.TrimEnd('/') + "/file";

    var boundary = "--------------------" + DateTime.UtcNow.Ticks.ToString("x");

    var startBytes = Encoding.ASCII.GetBytes("--" + boundary + "\r\n");

    var endBytes = Encoding.ASCII.GetBytes("\r\n");

    // Prepare the authenticated request.

    var request = (HttpWebRequest)WebRequest.Create(fileApiUrl);

    request.ContentType = "multipart/form-data; boundary=" + boundary;

    request.Method = "POST";

    request.KeepAlive = true;

    request.Headers.Add("Authorization", connectionKey.Trim());

    using (var stream = request.GetRequestStream())

    {

        // Write form metadata, if any.

        if (metadata?.Count > 0)

        {

            const string itemTemplate = "Content-Disposition: form-data; name=\"{0}\"\r\n\r\n{1}";

            foreach (string key in metadata.Keys)

            {

                var item = String.Format(itemTemplate, key, metadata[key]);

                var bytes = Encoding.UTF8.GetBytes(item);

                stream.Write(startBytes, 0, startBytes.Length);

                stream.Write(bytes, 0, bytes.Length);

                stream.Write(endBytes, 0, endBytes.Length);

            }

        }

        using (var file = new FileStream(fileName, FileMode.Open, FileAccess.Read, FileShare.Read))

        {

            // Load the file content into upload stream.

            var chunk = 0;

            var buffer = new byte[4096];

            int read;

            while ((read = file.Read(buffer, 0, buffer.Length)) > 0)

            {

                if (chunk++ == 0)

                {

                    // Make sure we know the content type.

                    if (String.IsNullOrEmpty(contentType))

                    {

                        contentType = MimeHelper.GetMimeTypeFromBuffer(buffer, read);

                        if (String.IsNullOrEmpty(contentType))

                        {

                            contentType = MimeHelper.BinaryMimeType;

                        }

                    }

                    // Write file header.

                    const string headerTemplate = "Content-Disposition: form-data; name=\"file\"; filename=\"{0}\"\r\nContent-Type: {1}\r\n\r\n";

                    var header = String.Format(headerTemplate, Path.GetFileName(fileName), contentType);

                    var bytes = Encoding.UTF8.GetBytes(header);

                    stream.Write(startBytes, 0, startBytes.Length);

                    stream.Write(bytes, 0, bytes.Length);

                }

                // Write the chunk.

                stream.Write(buffer, 0, read);

            }

            // Finish up the file content.

            stream.Write(endBytes, 0, endBytes.Length);

            // Write the end trailer.

            var trailer = Encoding.ASCII.GetBytes("--" + boundary + "--");

            stream.Write(trailer, 0, trailer.Length);

        }

    }

    // Issue the request and wait for the response.

    using (var response = (HttpWebResponse)request.GetResponse())

    {

        if (response.StatusCode == HttpStatusCode.OK)

        {

            using (var rstream = response.GetResponseStream())

            {

                if (rstream == null)

                {

                    throw new Exception("No response from server. File upload status unknown.");

                }

                using (var reader = new StreamReader(rstream))

                using (var jsonReader = new JsonTextReader(reader)

                {

                    DateParseHandling = DateParseHandling.None,

                    DateTimeZoneHandling = DateTimeZoneHandling.RoundtripKind

                })

                {

                    var json = JObject.Load(jsonReader);

                    return (string)json["id"];

                }

            }

        }

        throw new Exception(String.Format("{0} - Upload failed.{1}", (int)response.StatusCode, !String.IsNullOrEmpty(response.StatusDescription) ? " " + response.StatusDescription : null));

    }

}

1. cURL is a popular Open Source command-line utility for sending HTTP requests, available here: <https://curl.se>. [↑](#footnote-ref-1)
2. **terminal must run in UTF8 codepage**. [↑](#footnote-ref-2)
3. ConnectionKey can be obtained interactively only. [↑](#footnote-ref-3)
4. You can copy any of the curl code directly into command prompt and look at the response. [↑](#footnote-ref-4)
5. Some additional country specific fields may be required. [↑](#footnote-ref-5)
6. The APIMinimal object may change and the default structure should always derive from the accompanying objectinfo object. [↑](#footnote-ref-6)
7. Remember to escape quotes and remove newline characters when copying a command to the command prompt. [↑](#footnote-ref-7)
8. Remember to escape quotes and remove newline characters when copying a command to the command prompt. [↑](#footnote-ref-8)
9. If we omit the Id property, operation will be CREATE instead of MODIFY. NOTE: Error is thrown if Id is not found when included in object [↑](#footnote-ref-9)