How to make invoicing software e-invoice compliant with a few simple REST API calls

Basic introduction for integrators

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

I. Revision

Initial revision: *2022-03-03*

Lastest revision: Vi0.81, *2022-12-11*

II. Scope of this guide

This document is trying to show REST integration of invoice2einvoice APIs with workflow examples and minimal overhead. Examples of HTML POST request are all using curl[1] in terminal[2]. And there will be some examples in c#.

[1] cURL is a popular Open Source command-line utility for sending HTTP requests, available here: https://curl.se.
[2] **terminal must run in UTF8 codepage**.

Contents

1. 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.Execute
  -H "Authorization: IoT <connectionKey>:EUeInvoices"
  -H "Content-Type: application/json"
  -d "{\"extension\":\"<extension-name>\", \"args\": {<parameters>}}"
```

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

2. Interactive registration of your IT company

You can register on the web page https://e-invoices.online/ with click to signup button and choose integrator option as payment plan.

Interactive access to the service has two purposes:
1. In the interactive session, you can obtain[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.

3. 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/Extension.Execute
      -H "Authorization: IoT f24d5eb8-d5bb-11ec-90fd-b2e2095ceb4f:EUeInvoices"
      -H "Content-Type: application/json"
      -d "{\"extension\":\"EUeInvoices.GetInitializedNewInvoice\",
           \"args\":{\"Partial\":\"APIMinimal\"}}"
```

[3] ConnectionKey can be obtained interactively only.

4. Workflows

4.a Register your customers via API and obtain the Connection key for them

Steps can be summarized as a single call to AddCustomer extension, that returns when successful a newly generated ConnectionKey for added customer.

AddCustomer extension

extension call:

```
curl -X POST https://e-invoices.online/api/service/Extension.Execute
-H "Authorization: IoT <YourConnectionKey>:EUeInvoices"
-H "Content-Type: application/json"
-d "{\"extension\":\"EUeInvoices.AddCustomer\",
      \"args\": \"CustomerObj\": {<Data about your Customer>}}"
```

Property		Description
CustomerObj		Object that describes addCustumer object
	IntegratorId	Your integrator ID, that can be found in settings section of interactive session. (not the same as ConnectionKey)
	CustomerPhone	Phone number
	CustomerName	Formal name
	CustomerAddress	Street and house number
	CustomerCityName	City name
	CustomerPostCode	Postal code
	CustomerCountryCode	International code of country according to ISO 3166. Alpha-2
	CustomerVatNum	Vat number
	CustomerTaxNum	Tax number
	CustomerEmail	e-mail
	CustomerBudgetNum	Budget number
	CustomerRegNum	Registration number

The response will contain a ConnectionKey in the Result object. A detailed description can be found in section on standard extension response. An example of the response:


```
{
 "id": 44292081,
 "result": {
  "Status": "OK",
  "Result": {
   "ConnectionKey": "d969e2fe-c4f7-47e0-a41b-4e53479cc03a"
  },
  "ResultType": "Registration"
 }
}
```

4.b 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

Extension call[4]:

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

In response to this call, you will receive an invoice object. It contains an initialized Invoice, objectInfo and JSONSchema objects that are detailed later in Invoice object section. A detailed example can be found in the appendix.

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

Extension call:

```
curl -X POST https://e-invoices.online/api/service/Extension.Execute
-H "Authorization: IoT f24d5eb8-d5bb-11ec-90fd-b2e2095ceb4f:EUeInvoices"
-H "Content-Type: application/json"
-d "{\"extension\":\"EUeInvoices.SaveInvoice\",
\"args\":{   \"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"
```


[4] You can copy any of the curl code directly into command prompt and look at the response.

```
}
```

File upload API

This API endpoint is not tied to JSON-RPC. It responds to an ordinary HTTP POST request for file upload, which can be achieved directly from any web browser using the following simple HTML `<form>`:

```
<form method="post" enctype="multipart/form-data" action="/file?category=common">
  <input type="file" name="file" id="file" />
  <input type="submit" name="submit" value="Submit" />
</form>
```

The raw HTTP request that uploads a new file into a `common` category[5] looks similar to this (showing only relevant headers):

```
POST /file?category=common HTTP/1.1
Host: test.dopinus.com
Content-Length: 23652
Content-Type: multipart/form-data; boundary=----RandomBoundaryText
Authorization: IoT f24d5eb8-d5bb-11ec-90fd-b2e2095ceb4f:EUeInvoices
```

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://test.dopinus.com/file/18c8c4b4-72f2-48a2-9f15-53b8e415b946
```

Executed from a web browser, the request will trigger a download of the BLOB contents as a file and store it to the local disk.

[5] Each BLOB has a category associated with it. This enables D3.PRO to group them by their intended use and make it easier for back-end maintainers to link them back to their corresponding application.

5. Standard objects

5.a 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. 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].

Invoice

Contains invoice data object, that must validate 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 with local code if maintained locally)
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)
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

[6] The APIMinimal object may change and the default structure should always derive from the accompanying objectinfo object.

	currency
InvNote	Invoice additional notes
InvPreviousIssueDate	Original invoice issue date
InvPreviousNum	Original invoice reference number
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)
EU_Invoices_CN_IICRefs	Credit note reference details
InvAmountInclVat	Reference invoice amount including VAT
InvIssueDate	Reference invoice issue date
FiscCalcNum	Reference invoice fiscalization number
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, ...)
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 (e.g.: kg, piece, ...)
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)

See appendix for an example.

ObjectInfo

Defines every field in invoice data object with its JSON type, name, entity, required flag, .NET type, XML 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	Logical 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));
}
```

5.b Standard response

Extensions have a standard response.

```
{
  "Status": Response status that determines success of API.
  "Reason": Reason why API has failed.
  "Result": Result of an API in result type.
  "ResultType": Defines result type
}
```

See appendix for an example.

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 and Invoice.

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.

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


```
{"id":38517511,"result":{
  "Status": "OK",
  "Result": {
    "Invoice": {
      "EU_Invoices": {
        "BuyerCounCode": "RS",
        "InvAmountInclVatInp": 0,
        "PDFOriginal": null,
        "InvDueDate": "2022-12-26",
        "SellerAddress1": "Srbska 22",
        "InvAllowPercent": 0.0,
        "BuyerBudgetNum": null,
        "BuyerIsBudget": false,
        "InvCurrencyCode": "RSD",
        "BuyerRegNum": null,
        "BuyerTaxNum": null,
        "BuyerCityName": null,
        "SaleType": "Wholesale",
        "BuyerPostCode": null,
        "InvNote": null,
        "ExchangeRate": 1.0,
        "InvEndDate": "2022-12-11",
        "InvTotalVatAmountCCInp": 0,
        "BuyerFormalName": null,
        "InvStartDate": "2022-12-11",
        "BuyerAddress1": null,
        "BuyerOrderRef": "Nar:",
        "InvSigner": "Testni potpisnik",
        "InvPreviousNum": null,
        "InvPreviousIssueDate": null,
        "InvSaleTypeSC__Code": "Wholesale",
        "InvPaymentTypeCodeNameSC__Code": null,
        "BuyerLegalFormSC__LegalForm": "LegalEntity",
        "InvTypeCodeNameSC__Code": "INVOICE",
        "BuyerTypeSC__BuyerType": "Domestic",
        "DeliveryDateActual": "2022-12-11",
        "InvIssueDate": null,
        "InvNum": null,
        "FiscOperatorRegCode": 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_CN_IICRefs": [
         {
          "InvAmountInclVat": 0,
          "InvIssueDate": null,
          "FiscCalcNum": null,
          "InvNum": null,
          "_details": {
           "EU_Invoices_CN_IICRefs_Items": [
            {
             "LineNetAmount": null,
             "ItemVatCodeSC__VatCode": null
            }
           ]
          }
         }
        ]
       }
      }
     },
   "ResultType": "Invoice"
}}
```

Appendix B: Example of a properly prepared Invoice object[7]


```
{
 "EU_Invoices": {
  "InvPaymentTypeCodeNameSC__Code": "COMBINE",
  "InvTypeCodeNameSC__Code": "INVOICE",
  "InvIssueDate": "2022-12-09T21:02:56.000",
  "SaleType": "Wholesale",
  "InvSaleTypeSC__Code": "Wholesale",
  "InvStartDate": "2022-12-09T00:00:00.000",
  "InvEndDate": "2022-12-09T00:00:00.000",
  "InvDueDate": "2022-12-24T00:00:00.000",
  "InvNote": "Faktura je bila poslata II.",
  "InvCurrencyCode": "RSD",
  "BuyerOrderRef": "Nar:",
  "SellerAddress1": "Srbska 22",
  "BuyerTypeSC__BuyerType": "ForeignOutEU",
  "BuyerLegalFormSC__LegalForm": "SoleProprietor",
  "BuyerFormalName": "erertertret",
  "BuyerAddress1": "tzrz",
  "BuyerCityName": "dfgf",
  "BuyerPostCode": "dg",
  "BuyerCounCode": "AE",
  "BuyerIsBudget": false,
  "BuyerTaxNum": "rztrzrtztr",
  "DeliveryDateActual": "2022-12-09T00:00:00.000",
  "InvTotalVatAmountCCInp": 10.53,
  "InvAmountInclVatInp": 63.17,
  "InvAllowPercent": 6.0,
  "ExchangeRate": 1.0,
  "InvSigner": "Testni potpisnik",
  "InvNum": null,
  "BuyerBudgetNum": null,
  "BuyerRegNum": null,
  "InvPreviousNum": null,
  "InvPreviousIssueDate": null,
  "PDFOriginal": null,
  "FiscOperatorRegCode": null,
  "_details": {
   "EU_Invoices_Items": [
    {
     "ItemName": "Stavka",
     "Quantity": 1.0,
     "UMCodeNameSC__Code": "piece",
     "ItemNetPrice": 56.0,
     "ItemVatRate": 0.0,
     "ItemVatCodeSC__VatCode": "20",
```


[7] Remember to escape quotes and remove newline characters when copying a command to the command prompt.

```
    "LineAllowPercent": 0.0,
    "LineExciseAmount": 0.0,
    "ItemRetailPriceInp": null
   }
  ],
  "EU_Invoices_PaymentInstrs": [
   {
    "PaymentMeansCodeNSC__Code": "CREDITTRANSFER",
    "PayeeAccountType": "BBAN",
    "PayNumber": "603-123-56666",
    "PayNetworkProvider": "CONARS22XXX",
    "PaymentAmount": 63.17,
    "PayDocDate": null,
    "PayHolderName_Ref": null
   }
  ],
  "EU_Invoices_CN_IICRefs": [
   {
    "InvAmountInclVat": 0.0,
    "InvNum": null,
    "FiscCalcNum": null,
    "InvIssueDate": null,
    "_details": {
     "EU_Invoices_CN_IICRefs_Items": [
      {
       "LineNetAmount": null,
       "ItemVatCodeSC__VatCode": null
      }
     ]
    }
   }
  ]
 }
}
```

Appendix C: 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 D: 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));
    }
}
```