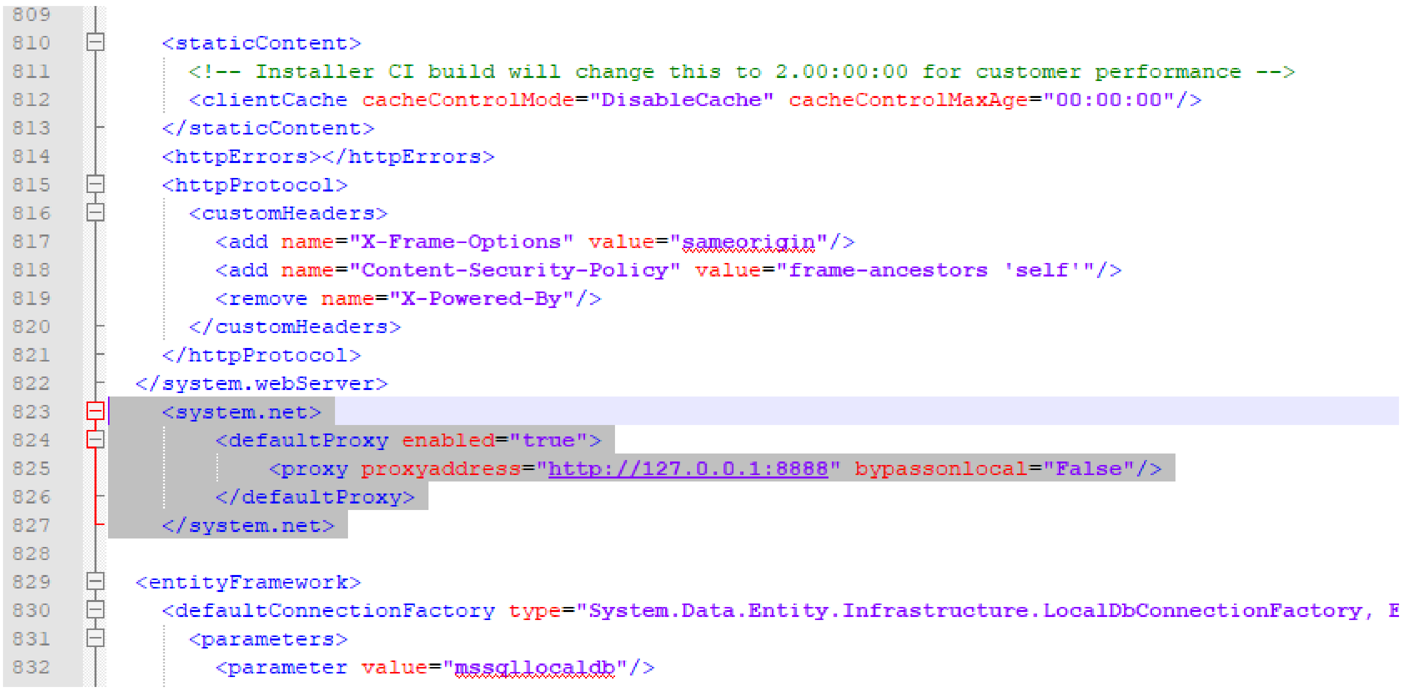
Using Fiddler and Postman with the iMIS REST API

Use Fiddler to capture iMIS traffic

To capture traffic from iMIS go into the Web.Config for AsiWebAppRoot and add this

```
<system.net>
   <defaultProxy enabled="true">
      <proxy proxyaddress="http://127.0.0.1:8888" bypassonlocal="False"/>
   </defaultProxy>
</system.net>
```

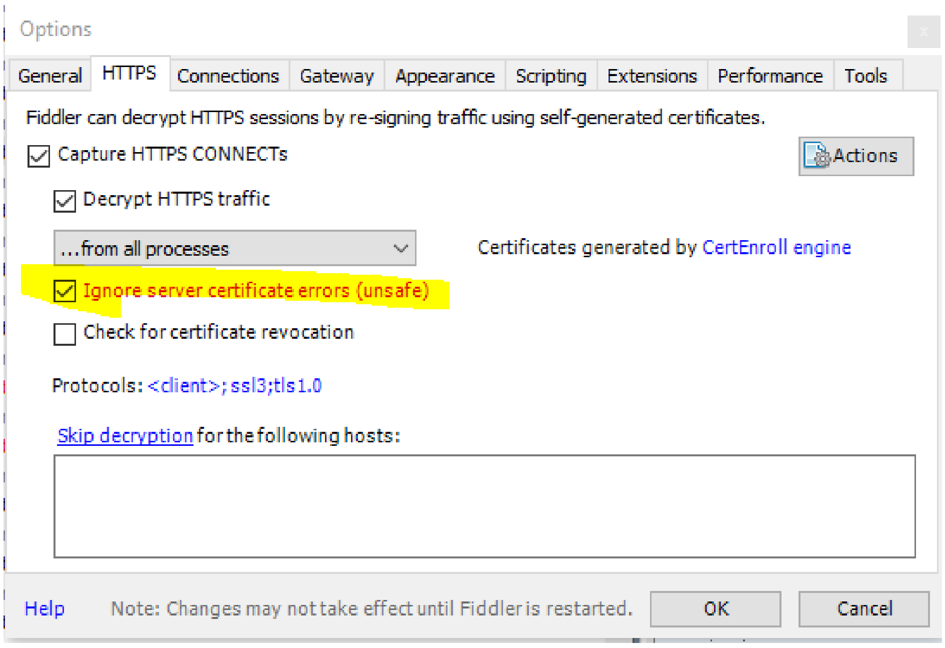


```
    <staticContent>
      <!-- Installer CI build will change this to 2.00:00:00 for customer performance -->
      <clientCache cacheControlMode="DisableCache" cacheControlMaxAge="00:00:00"/>
    </staticContent>
    <httpErrors></httpErrors>
    <httpProtocol>
      <customHeaders>
        <add name="X-Frame-Options" value="sameorigin"/>
        <add name="Content-Security-Policy" value="frame-ancestors 'self'"/>
        <remove name="X-Powered-By"/>
      </customHeaders>
    </httpProtocol>
  </system.webServer>
    <system.net>
        <defaultProxy enabled="true">
            <proxy proxyaddress="http://127.0.0.1:8888" bypassonlocal="False"/>
        </defaultProxy>
    </system.net>

  <entityFramework>
    <defaultConnectionFactory type="System.Data.Entity.Infrastructure.LocalDbConnectionFactory, E
      <parameters>
        <parameter value="mssqllocaldb"/>
```

The web.config may not already have a <system.net> in which case add the whole section.

Next open Fiddler and under Tools > Options > HTTPS set to ignore server certificate errors if required



Fiddler can decrypt HTTPS sessions by re-signing traffic using self-generated certificates.

☑ Capture HTTPS CONNECTs

☑ Decrypt HTTPS traffic

...from all processes

Certificates generated by CertEnroll engine

☑ Ignore server certificate errors (unsafe)

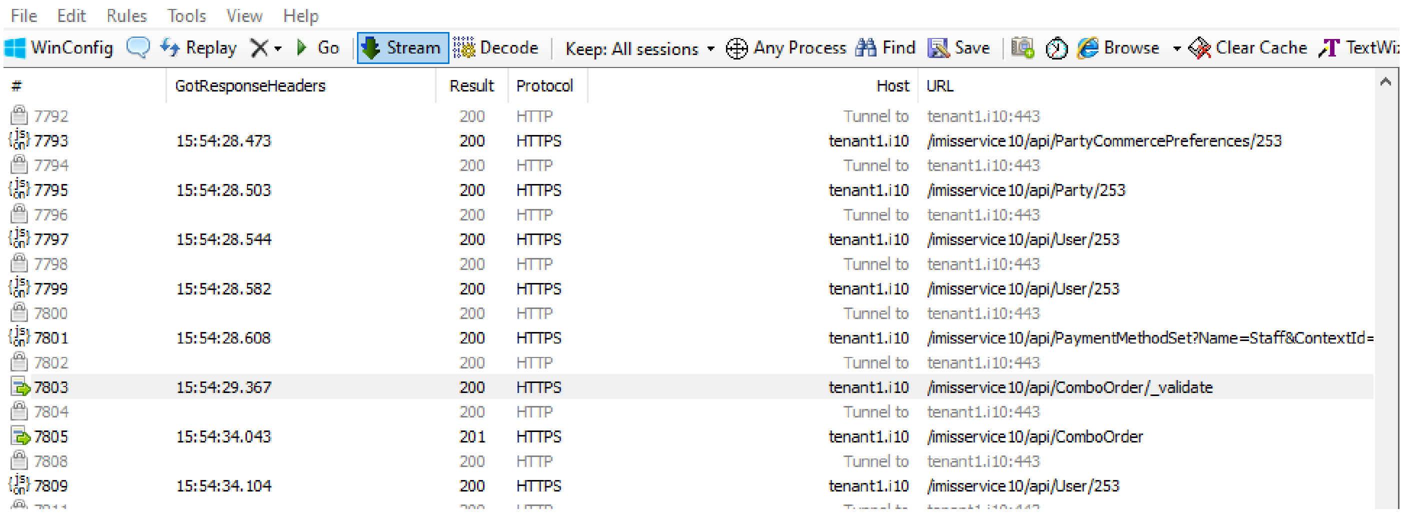
☐ Check for certificate revocation

Protocols: <client>; ssl3;tls1.0

Skip decryption for the following hosts:

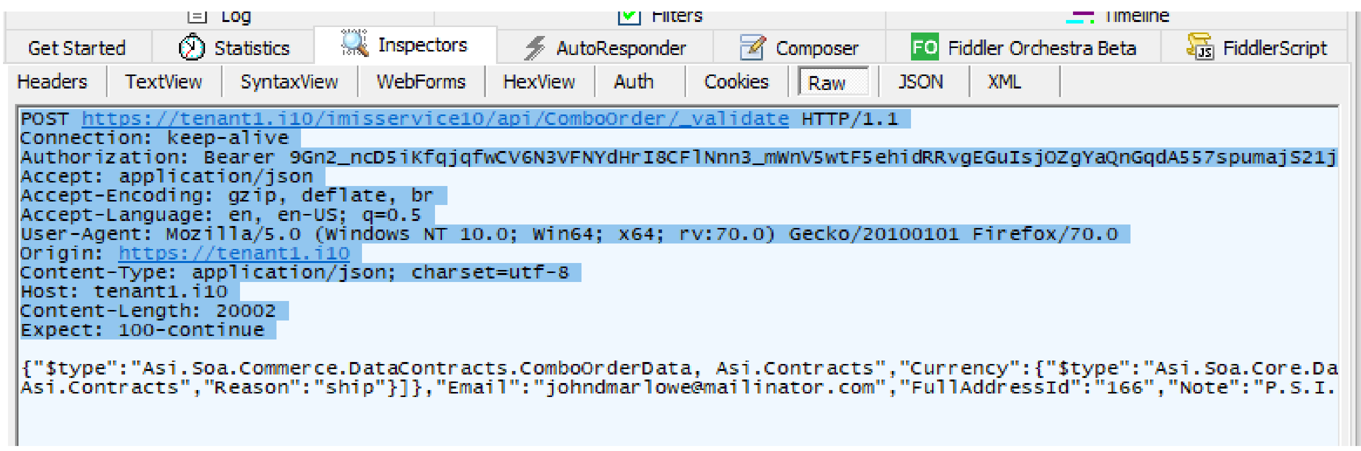
Note: Changes may not take effect until Fiddler is restarted.

Now you should be able to see REST traffic being passed from the iMIS website to the service. For example here it is submitting a donation:



#	GotResponseHeaders	Result	Protocol	Host	URL
7792		200	HTTP	Tunnel to	tenant1.i10:443
7793	15:54:28.473	200	HTTPS	tenant1.i10	/imisservice10/api/PartyCommercePreferences/253
7794		200	HTTP	Tunnel to	tenant1.i10:443
7795	15:54:28.503	200	HTTPS	tenant1.i10	/imisservice10/api/Party/253
7796		200	HTTP	Tunnel to	tenant1.i10:443
7797	15:54:28.544	200	HTTPS	tenant1.i10	/imisservice10/api/User/253
7798		200	HTTP	Tunnel to	tenant1.i10:443
7799	15:54:28.582	200	HTTPS	tenant1.i10	/imisservice10/api/User/253
7800		200	HTTP	Tunnel to	tenant1.i10:443
7801	15:54:28.608	200	HTTPS	tenant1.i10	/imisservice10/api/PaymentMethodSet?Name=Staff&ContextId=
7802		200	HTTP	Tunnel to	tenant1.i10:443
7803	15:54:29.367	200	HTTPS	tenant1.i10	/imisservice10/api/ComboOrder/_validate
7804		200	HTTP	Tunnel to	tenant1.i10:443
7805	15:54:34.043	201	HTTPS	tenant1.i10	/imisservice10/api/ComboOrder
7808		200	HTTP	Tunnel to	tenant1.i10:443
7809	15:54:34.104	200	HTTPS	tenant1.i10	/imisservice10/api/User/253

To get the JSON used out go to Inspectors > Raw and copy and paste

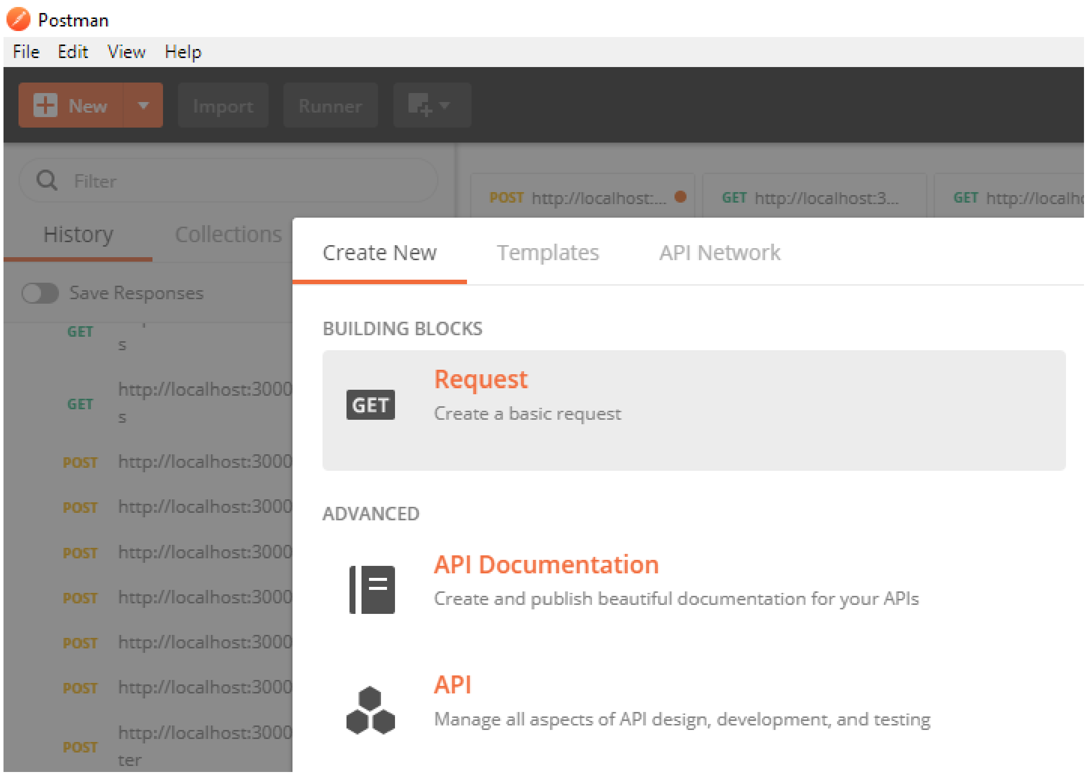


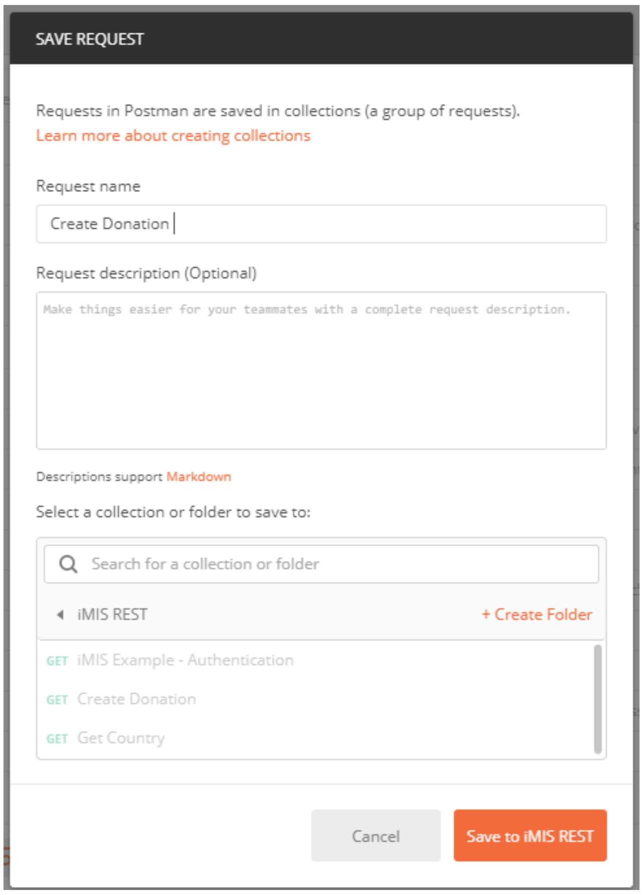
```
POST https://tenant1.i10/imisservice10/api/ComboOrder/_validate HTTP/1.1
Connection: keep-alive
Authorization: Bearer 9Gn2_ncD5iKfqjqfwCV6N3VFNYdHrI8CFlNnn3_mWnV5wtF5ehidRRvgEGuIsjOZgYaQnGqdA557spumajS21j
Accept: application/json
Accept-Encoding: gzip, deflate, br
Accept-Language: en, en-US; q=0.5
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:70.0) Gecko/20100101 Firefox/70.0
Origin: https://tenant1.i10
Content-Type: application/json; charset=utf-8
Host: tenant1.i10
Content-Length: 20002
Expect: 100-continue

{"$type":"Asi.Soa.Commerce.DataContracts.ComboOrderData, Asi.Contracts","Currency":{"$type":"Asi.Soa.Core.Da
Asi.Contracts","Reason":"ship"}]},"Email":"johndmarlowe@mailinator.com","FullAddressId":"166","Note":"P.S.I.
```

Use Postman to test the REST calls

Next in Postman create a new request




SAVE REQUEST

Requests in Postman are saved in collections (a group of requests).
Learn more about creating collections

Request name: Create Donation

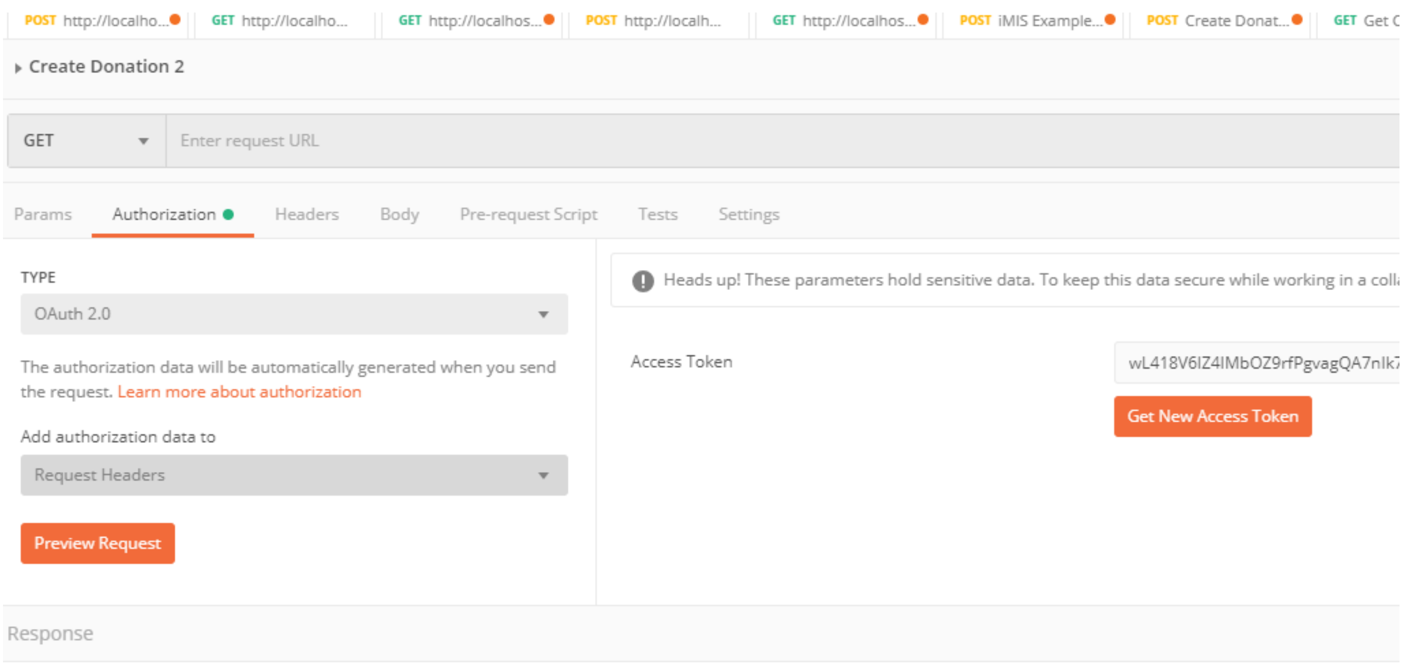
Request description (Optional)

Make things easier for your teammates with a complete request description.

Descriptions support Markdown

Select a collection or folder to save to:

Click on Authorization and select OAuth 2.0 and then click on the **Get New Access Token** button



POST http://localho... GET http://localho... GET http://localhos... POST http://localh... GET http://localhos... POST iMIS Example... POST Create Donat... GET Get C

Create Donation 2

GET Enter request URL

Params Authorization Headers Body Pre-request Script Tests Settings

TYPE
OAuth 2.0

The authorization data will be automatically generated when you send the request. Learn more about authorization

Add authorization data to
Request Headers

Preview Request

Heads up! These parameters hold sensitive data. To keep this data secure while working in a coll

Access Token wL418V6IZ4IMbOZ9rfPgvagQA7nlk7

Get New Access Token

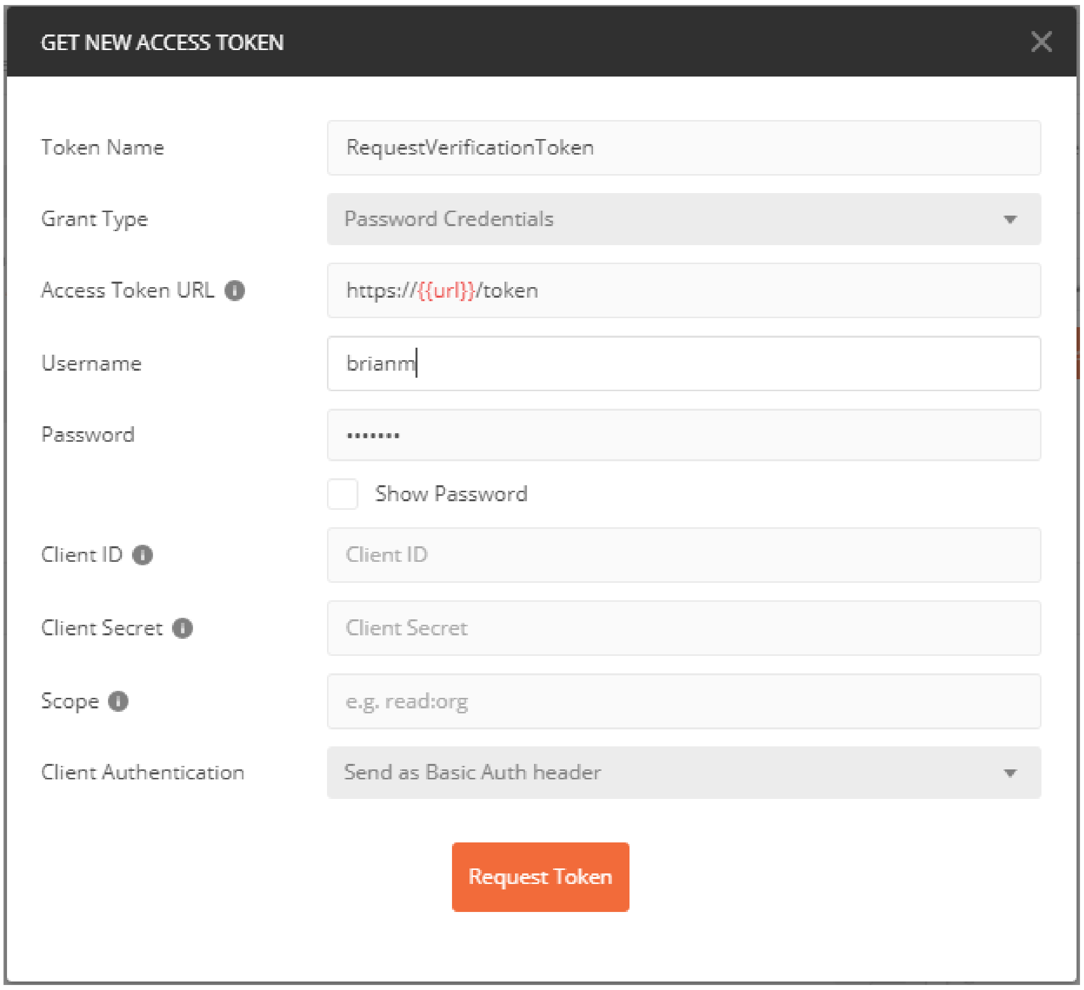
Response

Now set the token name to **RequestVerificationToken**

Set the Grant Type to **Password credentials**

Set the token URL to: **https://{imisurl}/token**

And click the **Request Token** button – here is the screen

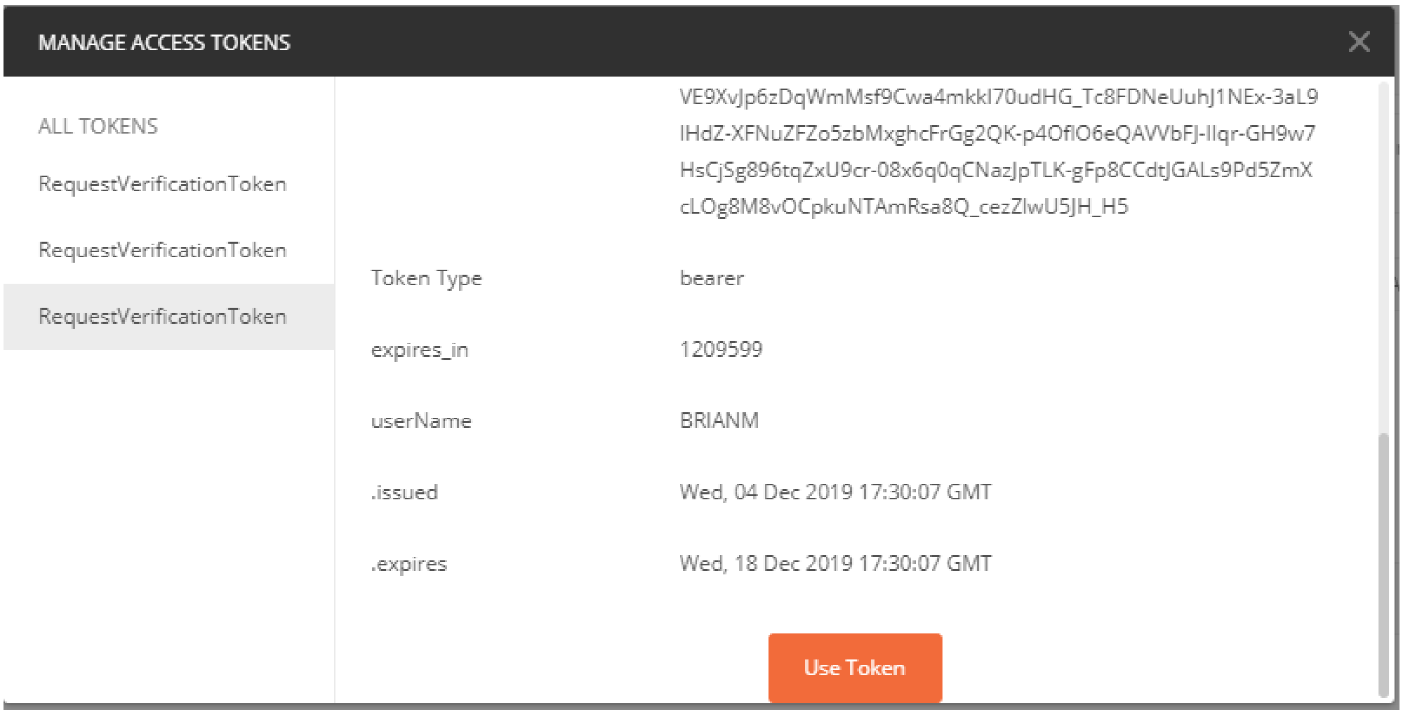


GET NEW ACCESS TOKEN

Token Name	RequestVerificationToken
Grant Type	Password Credentials
Access Token URL	https://{{url}}/token
Username	brianm
Password	•••••••
	Show Password
Client ID	Client ID
Client Secret	Client Secret
Scope	e.g. read:org
Client Authentication	Send as Basic Auth header

Request Token

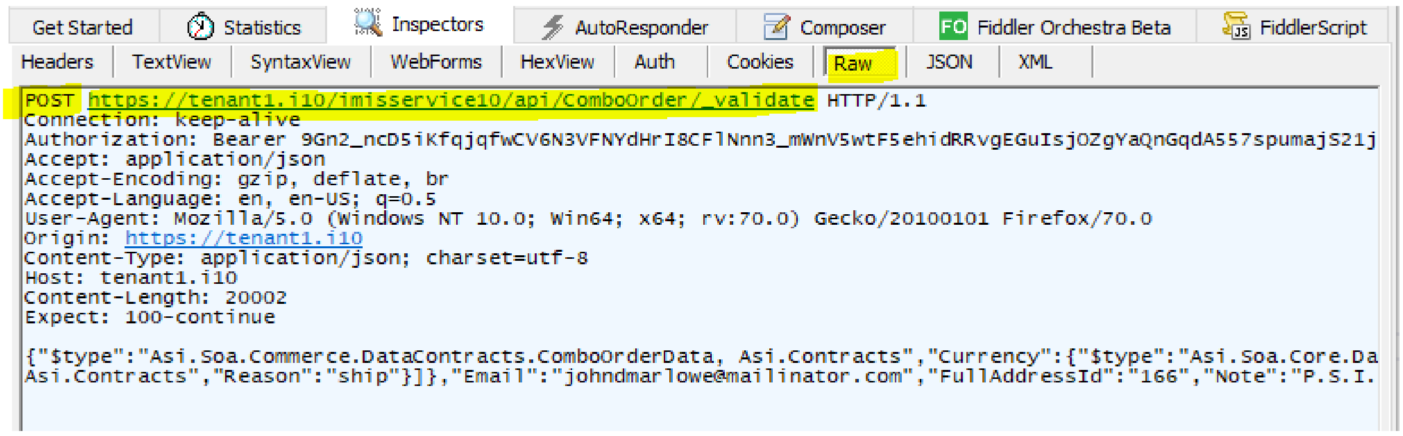
Scroll down and click “**Use Token**”



	VE9XvJp6zDqWmMsf9Cwa4mkkI70udHG_Tc8FDNeUuhJ1NEx-3aL9 lHdZ-XFNuZFZo5zbMxghcFrGg2QK-p4OflO6eQAVVbFJ-Ilqr-GH9w7 HsCjSg896tqZxU9cr-08x6q0qCNazJpTLK-gFp8CCdtJGALs9Pd5ZmX cLOg8M8vOCpkuNTAmRsa8Q_cezZlwU5JH_H5
Token Type	bearer
expires_in	1209599
userName	BRIANM
.issued	Wed, 04 Dec 2019 17:30:07 GMT
.expires	Wed, 18 Dec 2019 17:30:07 GMT

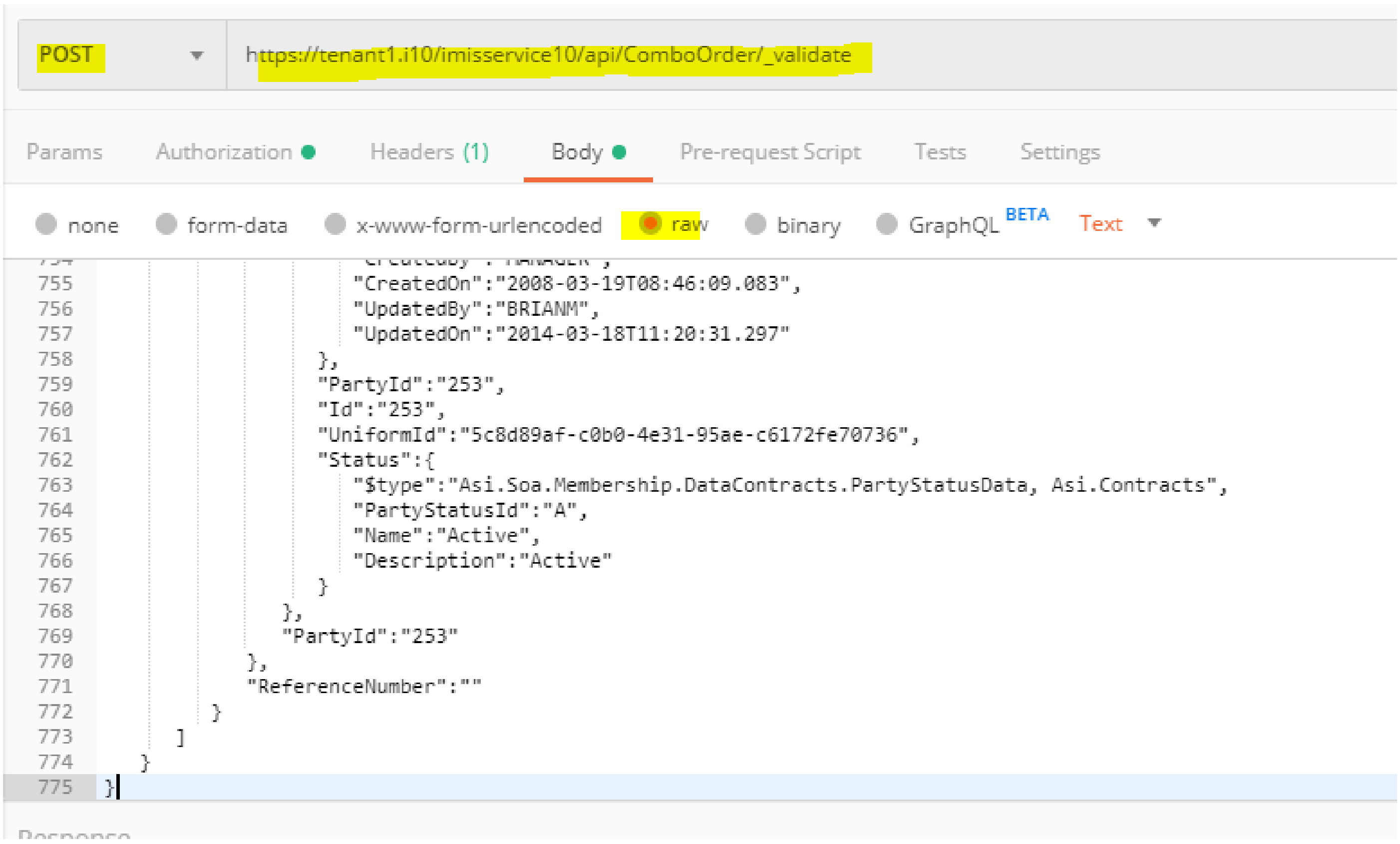
Once you have a token new Postman Requests can use an existing token, so long as the type is OAuth 2.0.

Now set the values in Postman based on what we saw in Fiddler



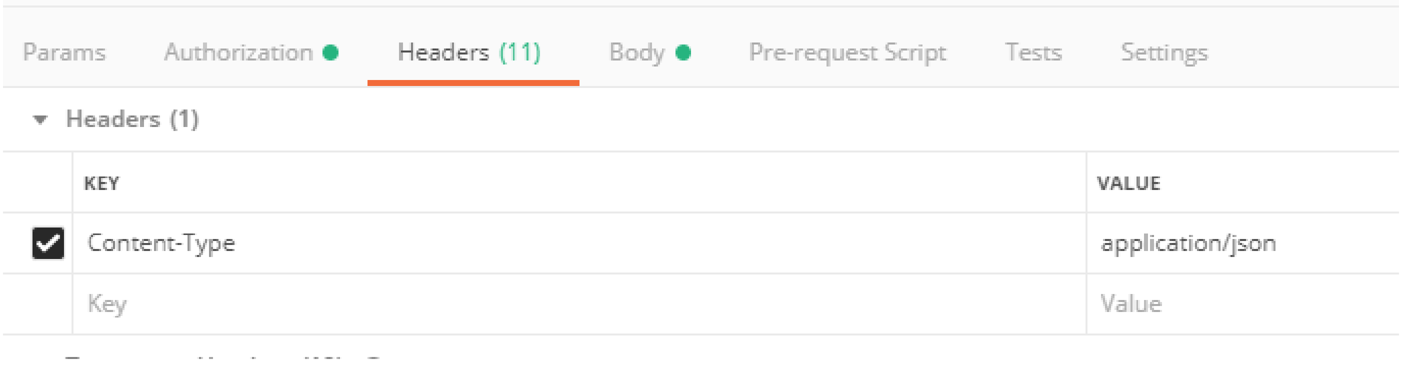
```
POST https://tenant1.i10/imisservice10/api/ComboOrder/_validate HTTP/1.1
Connection: keep-alive
Authorization: Bearer 9Gn2_ncD5iKfqjqfwCV6N3VFNYdHrI8CFlNnn3_mWnV5wtF5ehidRRvgEGuIsjOZgYaQnGqdA557spumajS21j
Accept: application/json
Accept-Encoding: gzip, deflate, br
Accept-Language: en, en-US; q=0.5
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:70.0) Gecko/20100101 Firefox/70.0
Origin: https://tenant1.i10
Content-Type: application/json; charset=utf-8
Host: tenant1.i10
Content-Length: 20002
Expect: 100-continue

{"$type":"Asi.Soa.Commerce.DataContracts.ComboOrderData, Asi.Contracts","Currency":{"$type":"Asi.Soa.Core.Da
Asi.Contracts","Reason":"ship"}]},"Email":"johndmarlowe@mailinator.com","FullAddressId":"166","Note":"P.S.I.
```


POST https://tenant1.i10/imisservice10/api/ComboOrder/_validate

```
                    "CreatedOn":"2008-03-19T08:46:09.083",
                    "UpdatedBy":"BRIANM",
                    "UpdatedOn":"2014-03-18T11:20:31.297"
                 },
                 "PartyId":"253",
                 "Id":"253",
                 "UniformId":"5c8d89af-c0b0-4e31-95ae-c6172fe70736",
                 "Status":{
                    "$type":"Asi.Soa.Membership.DataContracts.PartyStatusData, Asi.Contracts",
                    "PartyStatusId":"A",
                    "Name":"Active",
                    "Description":"Active"
                 }
              },
              "PartyId":"253"
           },
           "ReferenceNumber":""
        }
     ]
   }
}
```

Also under Headers set the content type



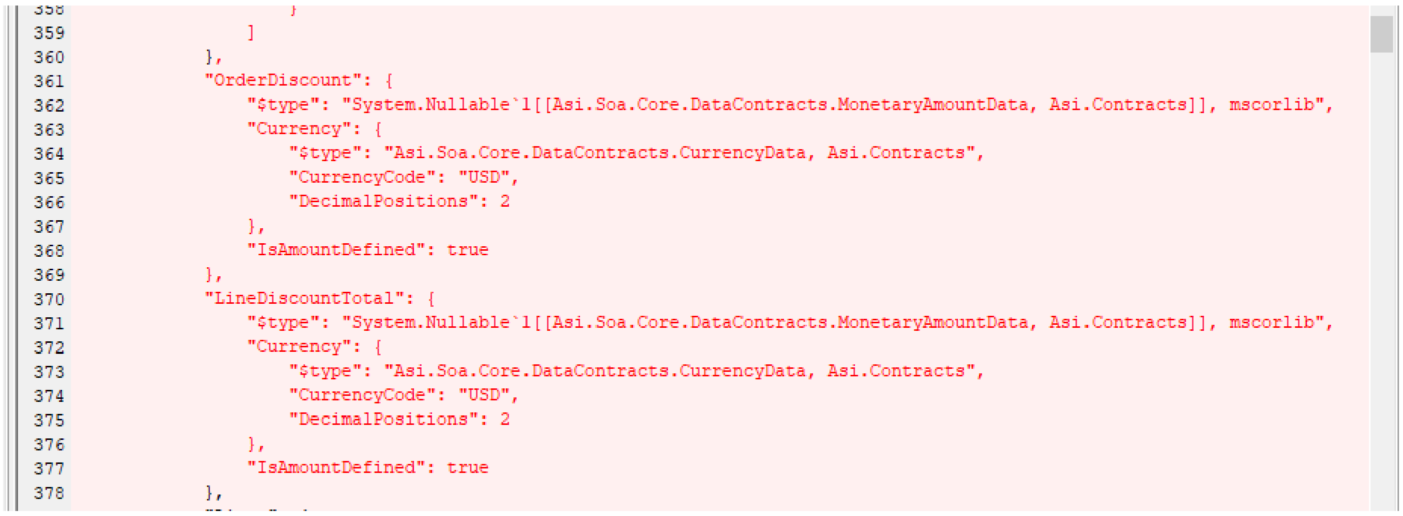
	KEY	VALUE
✓	Content-Type	application/json
	Key	Value

That should allow you to successfully submit the donation to be validated.

In this example we are validating the Cart containing a donation prior to submitting it – so check the IsValid flag on the return value and if it is true then submit the donation to

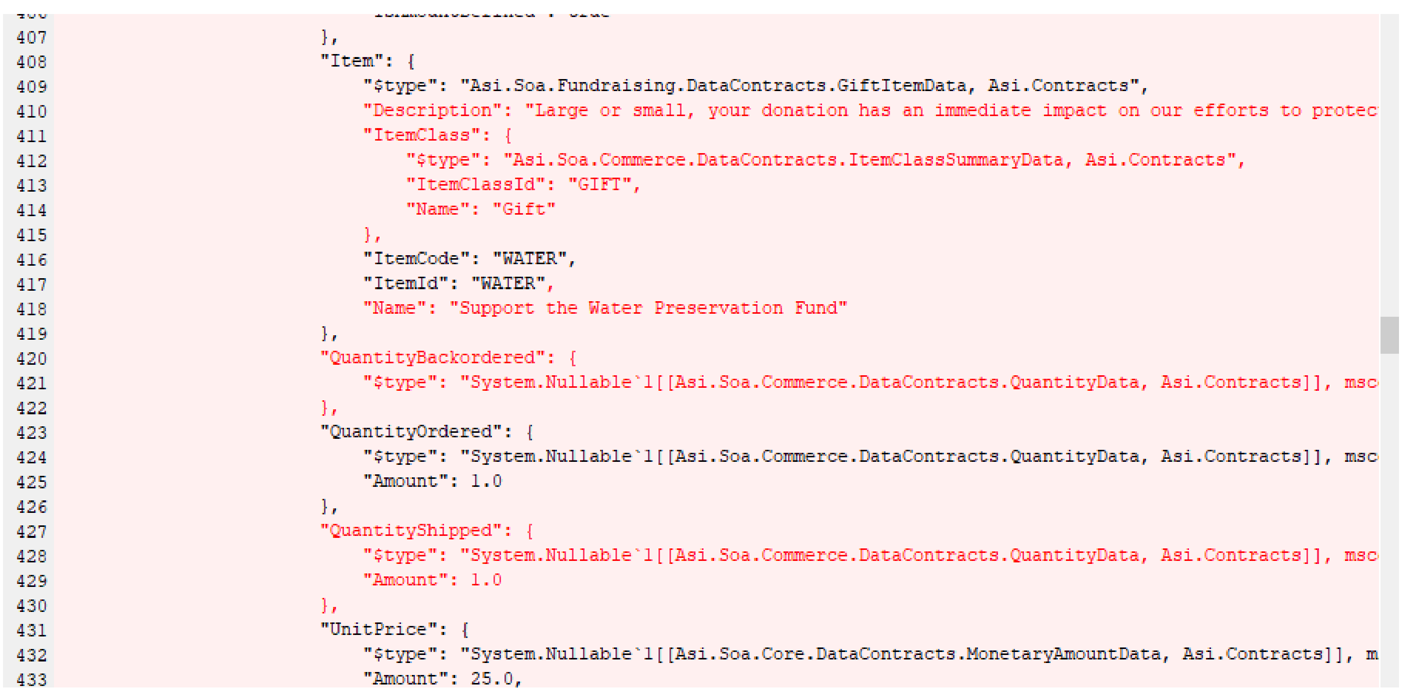
https://tenant1.i10/imisservice10/api/ComboOrder

A note on the json returned from the validate – it generally contains the original information plus a few additional details calculated about the order – for example the OrderDiscount and LineDiscountTotals



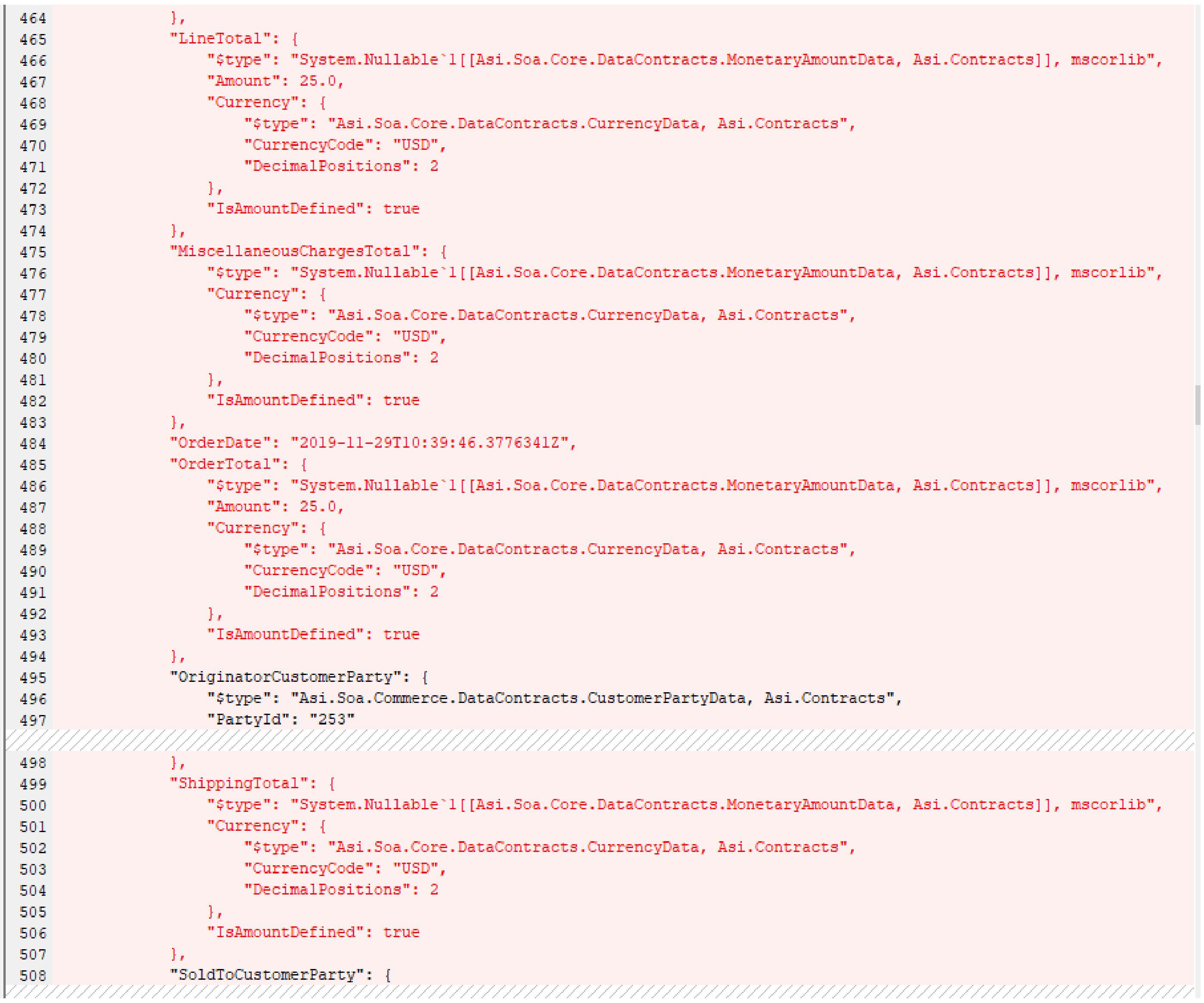
```
                    }
                ]
            },
            "OrderDiscount": {
                "$type": "System.Nullable`1[[Asi.Soa.Core.DataContracts.MonetaryAmountData, Asi.Contracts]], mscorlib",
                "Currency": {
                    "$type": "Asi.Soa.Core.DataContracts.CurrencyData, Asi.Contracts",
                    "CurrencyCode": "USD",
                    "DecimalPositions": 2
                },
                "IsAmountDefined": true
            },
            "LineDiscountTotal": {
                "$type": "System.Nullable`1[[Asi.Soa.Core.DataContracts.MonetaryAmountData, Asi.Contracts]], mscorlib",
                "Currency": {
                    "$type": "Asi.Soa.Core.DataContracts.CurrencyData, Asi.Contracts",
                    "CurrencyCode": "USD",
                    "DecimalPositions": 2
                },
                "IsAmountDefined": true
            },
```

The items in the order contain additional details



```
                        },
                        "Item": {
                            "$type": "Asi.Soa.Fundraising.DataContracts.GiftItemData, Asi.Contracts",
                            "Description": "Large or small, your donation has an immediate impact on our efforts to protec
                            "ItemClass": {
                                "$type": "Asi.Soa.Commerce.DataContracts.ItemClassSummaryData, Asi.Contracts",
                                "ItemClassId": "GIFT",
                                "Name": "Gift"
                            },
                            "ItemCode": "WATER",
                            "ItemId": "WATER",
                            "Name": "Support the Water Preservation Fund"
                        },
                        "QuantityBackordered": {
                            "$type": "System.Nullable`1[[Asi.Soa.Commerce.DataContracts.QuantityData, Asi.Contracts]], msc
                        },
                        "QuantityOrdered": {
                            "$type": "System.Nullable`1[[Asi.Soa.Commerce.DataContracts.QuantityData, Asi.Contracts]], msc
                            "Amount": 1.0
                        },
                        "QuantityShipped": {
                            "$type": "System.Nullable`1[[Asi.Soa.Commerce.DataContracts.QuantityData, Asi.Contracts]], msc
                            "Amount": 1.0
                        },
                        "UnitPrice": {
                            "$type": "System.Nullable`1[[Asi.Soa.Core.DataContracts.MonetaryAmountData, Asi.Contracts]], m
                            "Amount": 25.0,
```

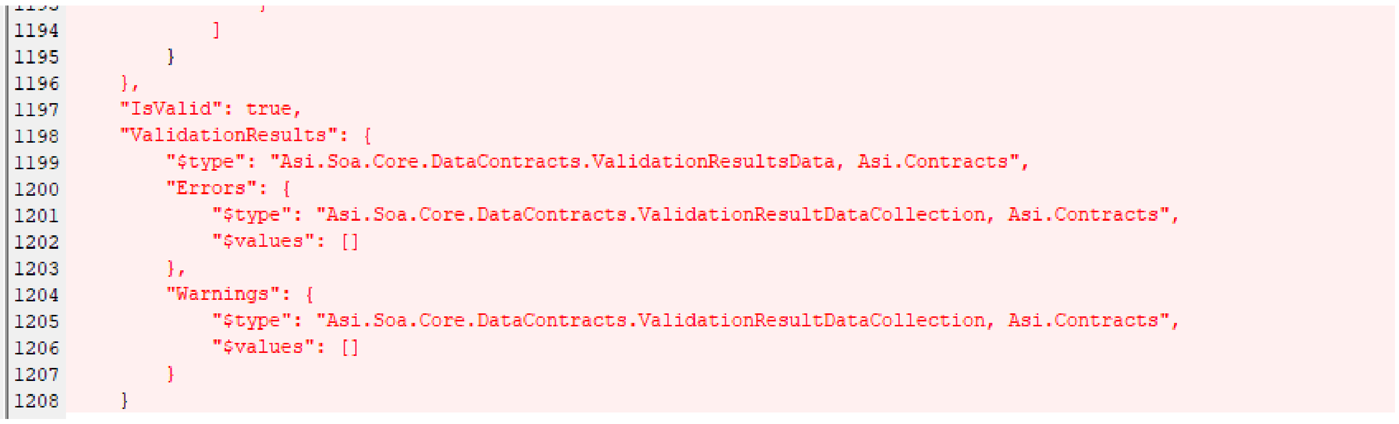
Line Totals, Miscellaneous Charges, Order Totals and Shipping Totals also appear



```
            },
            "LineTotal": {
                "$type": "System.Nullable`1[[Asi.Soa.Core.DataContracts.MonetaryAmountData, Asi.Contracts]], mscorlib",
                "Amount": 25.0,
                "Currency": {
                    "$type": "Asi.Soa.Core.DataContracts.CurrencyData, Asi.Contracts",
                    "CurrencyCode": "USD",
                    "DecimalPositions": 2
                },
                "IsAmountDefined": true
            },
            "MiscellaneousChargesTotal": {
                "$type": "System.Nullable`1[[Asi.Soa.Core.DataContracts.MonetaryAmountData, Asi.Contracts]], mscorlib",
                "Currency": {
                    "$type": "Asi.Soa.Core.DataContracts.CurrencyData, Asi.Contracts",
                    "CurrencyCode": "USD",
                    "DecimalPositions": 2
                },
                "IsAmountDefined": true
            },
            "OrderDate": "2019-11-29T10:39:46.3776341Z",
            "OrderTotal": {
                "$type": "System.Nullable`1[[Asi.Soa.Core.DataContracts.MonetaryAmountData, Asi.Contracts]], mscorlib",
                "Amount": 25.0,
                "Currency": {
                    "$type": "Asi.Soa.Core.DataContracts.CurrencyData, Asi.Contracts",
                    "CurrencyCode": "USD",
                    "DecimalPositions": 2
                },
                "IsAmountDefined": true
            },
            "OriginatorCustomerParty": {
                "$type": "Asi.Soa.Commerce.DataContracts.CustomerPartyData, Asi.Contracts",
                "PartyId": "253"

            },
            "ShippingTotal": {
                "$type": "System.Nullable`1[[Asi.Soa.Core.DataContracts.MonetaryAmountData, Asi.Contracts]], mscorlib",
                "Currency": {
                    "$type": "Asi.Soa.Core.DataContracts.CurrencyData, Asi.Contracts",
                    "CurrencyCode": "USD",
                    "DecimalPositions": 2
                },
                "IsAmountDefined": true
            },
            "SoldToCustomerParty": {
```

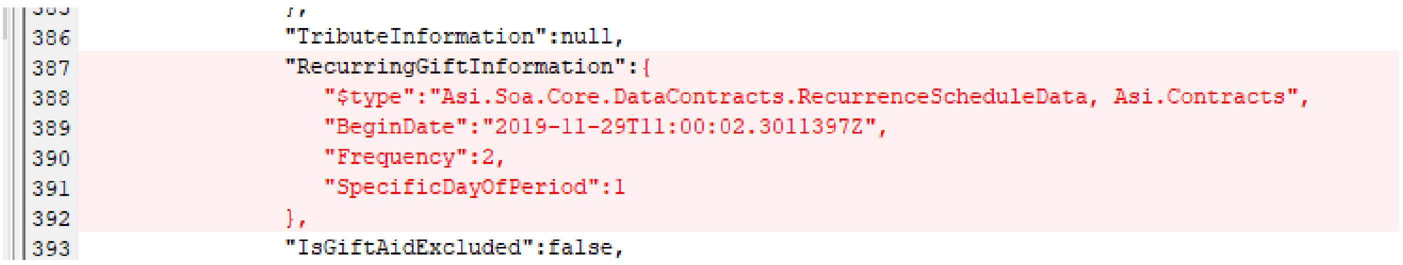
Most importantly for validation the IdValid flag is returned and if there are any errors or warnings these will appear here:



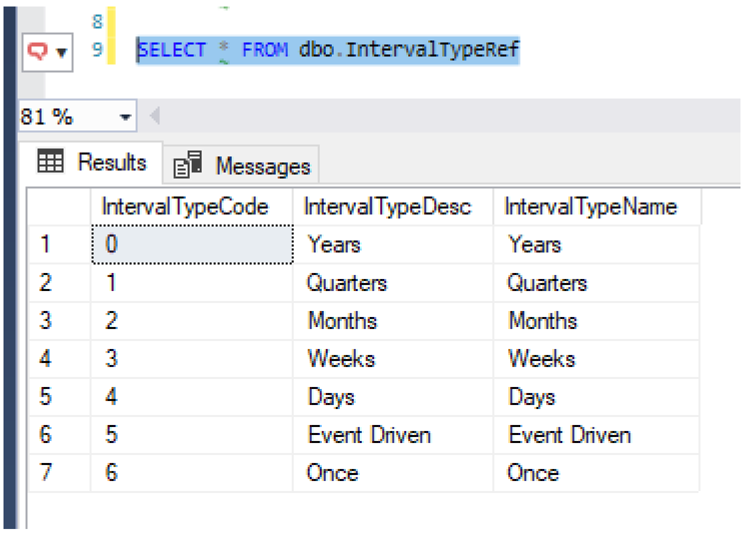
```
            ]
        }
    },
    "IsValid": true,
    "ValidationResults": {
        "$type": "Asi.Soa.Core.DataContracts.ValidationResultsData, Asi.Contracts",
        "Errors": {
            "$type": "Asi.Soa.Core.DataContracts.ValidationResultDataCollection, Asi.Contracts",
            "$values": []
        },
        "Warnings": {
            "$type": "Asi.Soa.Core.DataContracts.ValidationResultDataCollection, Asi.Contracts",
            "$values": []
        }
    }
```

Assuming the validation passes, submit the donation to iMIS , the return value again is similar to before with additional information such as additional charges, tax and total price, however there is no IsValid flag returned.

To submit a recurring donation the Json is nearly the same but includes RecurringGiftInformation, with the Frequency using the IntervalTypeRef table code.



```
                },
                "TributeInformation":null,
                "RecurringGiftInformation":{
                   "$type":"Asi.Soa.Core.DataContracts.RecurrenceScheduleData, Asi.Contracts",
                   "BeginDate":"2019-11-29T11:00:02.3011397Z",
                   "Frequency":2,
                   "SpecificDayOfPeriod":1
                },
                "IsGiftAidExcluded":false,
```



```
SELECT * FROM dbo.IntervalTypeRef
```

81 %

Results | Messages

	IntervalTypeCode	IntervalTypeDesc	IntervalTypeName
1	0	Years	Years
2	1	Quarters	Quarters
3	2	Months	Months
4	3	Weeks	Weeks
5	4	Days	Days
6	5	Event Driven	Event Driven
7	6	Once	Once

Note if you wish to tidy up the json you get from Fiddler, prior to using it in Postman there are plenty of online json formatters – for example https://jsonformatter.curiousconcept.com/