# ICCO Web Services API

As of Version 11.0.230

November 4, 2025

Please contact Support to make sure this feature has been installed before continuing.

ICCO Web Services API (aka CreditSoft API or CollectPlus API) is a component you can install on your server that allows remote access to specific parts of your data.

CAUTION! As with any component that is exposed to the internet, you are responsible for securing access to it by configuring a firewall, antivirus, and intrusion prevention software. You also need to regularly review the server and application logs. In addition, CreditSoft and CollectPlus have a Web Security Log that will log errors and suspicious activity occurring on the web portal and the API.

The ICCO Web Services API has a SOAP Service and a REST service. They are virtually identical, with REST being a more programmer-friendly version. Any call available on SOAP will become available on REST, unless specific security requirements prevent us from using REST. If you find a call you need is not available you can request it and it will be added to the next version at no cost.

SOAP URL: https://yoursitehere/WSICCOCore.svc

REST URL: https://yoursitehere/WSICCOCoreREST.svc

A simple tester is in: https://yoursitehere/WSTester.html

# Security

Every web service call you make to the API requires you to always submit the API username and password in the Authorization header separated by a : colon character and Base64 encoded (using the btoa function in java for example). That is your first line of defense.

In addition, some calls also require a token, which is a temporary key that is obtained using a CreditSoft/CollectPlus web user's credentials. Those tokens require the internet username and password of an application user from Settings | Security | Users, different from the API User. This allows you to identify who is making the changes to the records in the audit log. This is your send line of defense.

## API Users

You should define at least one API user for EACH remote vendor or application, so you can easily grant, revoke, and define the specific access that vendor or application has to your API.

REST calls are allowed per call by system administrators per API user in Settings. REST calls will have "REST Available" next to them in the documentation.

# INTERNALONLY calls

Some API calls have an [INTERNALONLY] tag after them. For example GetMobileAPIToken has two versions:

GetMobileAPIToken[REST] and GetMobileAPIToken[REST][INTERNALONLY]

The INTERNALONLY tag means that this call can only be called from an internal server, defined in the Internal Server IP Addresses in the API Preferences.

If you add the INTERNALONLY call to the allowed calls and don't add the other one, then that call will only be available to an internal API server and not from any other IP addresses. This is useful if you want some calls to only be available to your own middle layer servers.

# GET vs POST Methods

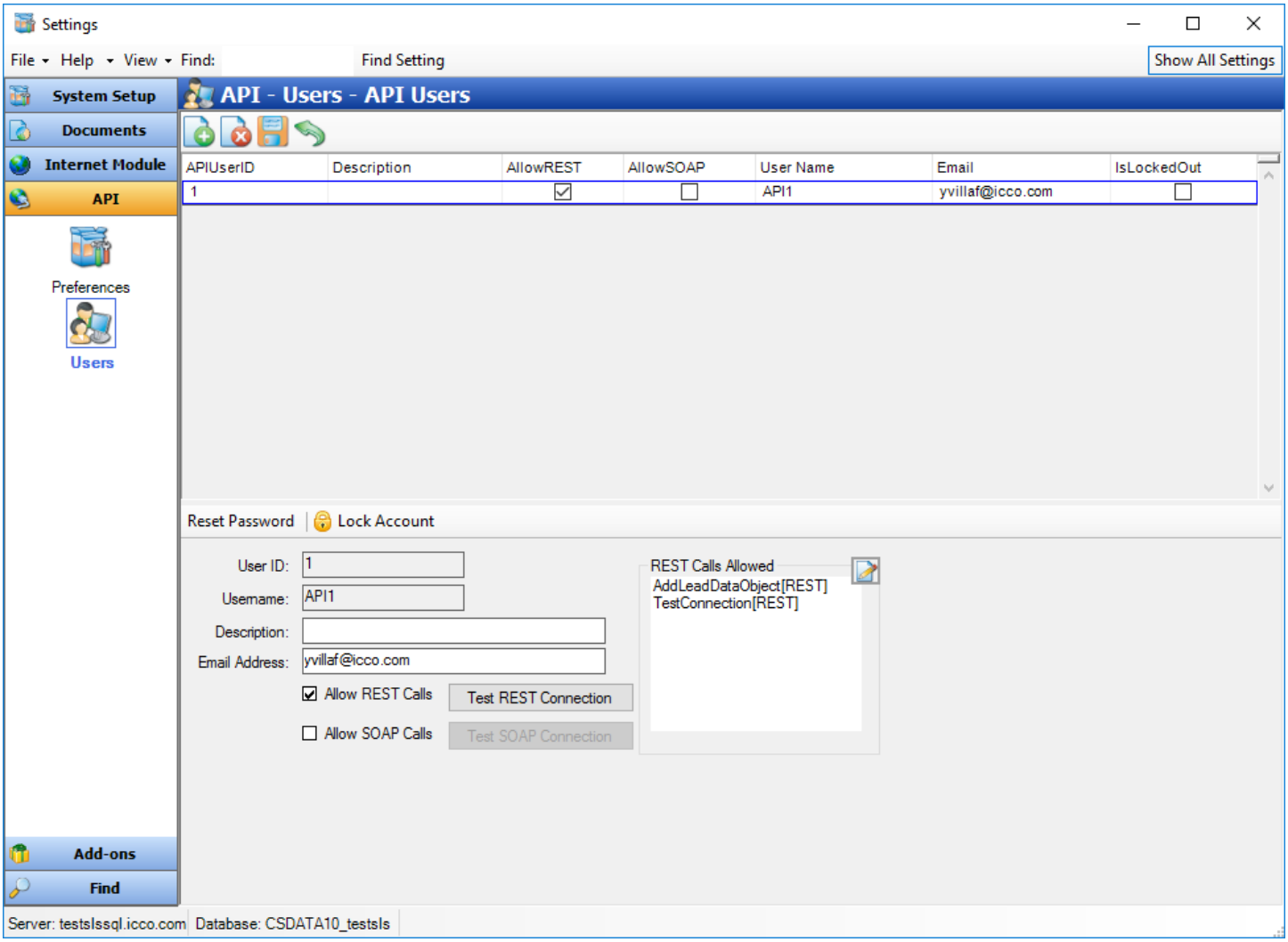
All calls are POST calls except About, TestConnection, and catalog calls.

If you use the wrong method you will get a 405 Method Not Allowed response.

# REST Calls and calls ending in J (JSON Object response)

All REST calls have two versions, the standard one which returns a serialized string in JSON, and one with the same command name but with a “J” at the end which returns the JSON object directly.

For example: you can call either LeadDataObject_Get or LeadDataObject_GetJ. They both will do the same thing but the first one will return the response serialized in a string, whereas the second one will return the JSON object directly (which is easier to view and interact with when using Postman for example).




SOAP calls are all allowed once the API user is tagged to Allow SOAP Calls.

## Tokens and token users

Some calls, for example GetDocumentsSent require that you pass a token, and not just your API User and the Lead or Client ID.

There are two kinds of tokens: a LeadAPIToken and a standard APIToken. The LeadToken is an older version of the token that only works on calls to lead objects. The newer APIToken allows calls that target lead, client, coapp, and other objects.

A token is a temporary key that has to be requested using the legacy User_GetLeadAPIToken or the newer calls: User_GetAPIToken or GetMobileAPIToken.

### *Working with the legacy LeadAPIToken*

#### Requesting a LeadAPIToken

The User_GetLeadAPIToken requires you pass a UserTokenDataObject with the following properties:

```
<DataContract()>
Public Class UserTokenDataObject
	<DataMember()> Public UserName As String 'web username from aspnet_users/Ezy_Users
	<DataMember()> Public Password As String 'web password from aspnet_users/Ezy_Users
	<DataMember()> Public LeadID As Nullable(Of Integer) 'LeadID that the user wants to modify

End Class
```

### *Working with the new APIToken*

#### Requesting an APIToken

In contrast, the newer WebUserTokenDataObject has the following properties:

```
<DataContract()>
Public Class WebUserTokenDataObject
	<DataMember()> Public UserName As String 'web username from aspnet_users/Ezy_Users
	<DataMember()> Public Password As String 'web password from aspnet_users/Ezy_Users
	<DataMember()> Public CSRecordID As Nullable(Of Integer) 'CSRecordID that the user wants to
modify
	<DataMember()> Public CSRecordType As String 'CSRecordType that the user wants to modify

End Class
```

Here's sample code in JQuery to get an API token using the newer method:

```
$('#btnUser_GetAPIToken').click(function () {

	var JSONObjectUser = {
		"objUser":
			{
			"UserName": $("#TokenUserName").val(),
			"Password": $("#TokenPassword").val(),
			"CSRecordID": $("#CSRecordID").val(),
			"CSRecordType": $("#CSRecordType").val()
			}
	};

	$.ajax({
		type: 'POST',
		url: URL + '/User_GetAPIToken',

		headers: { 'Authorization': btoa('apiuser1:myapipassword') },
```

objUser has the structure of WebUserTokenDataObject

Username and internet password of a user in Settings Security Users NOT THE API USER

API User username and password from Settings API Users

```
                        data: JSON.stringify(JSONObjectUser),
                        dataType: 'json',

                        contentType: 'application/json; charset=utf-8',

                        success: function (r) {

                                var responsetoken = JSON.parse(r.d);
                                $("#Token").val(responsetoken.Token.toString());

                        },
                        error: function (e) {
                               $("#div1").html('ERROR');
                        }
                });

        });
```

The response will be a TokenDataObject which includes the following properties:

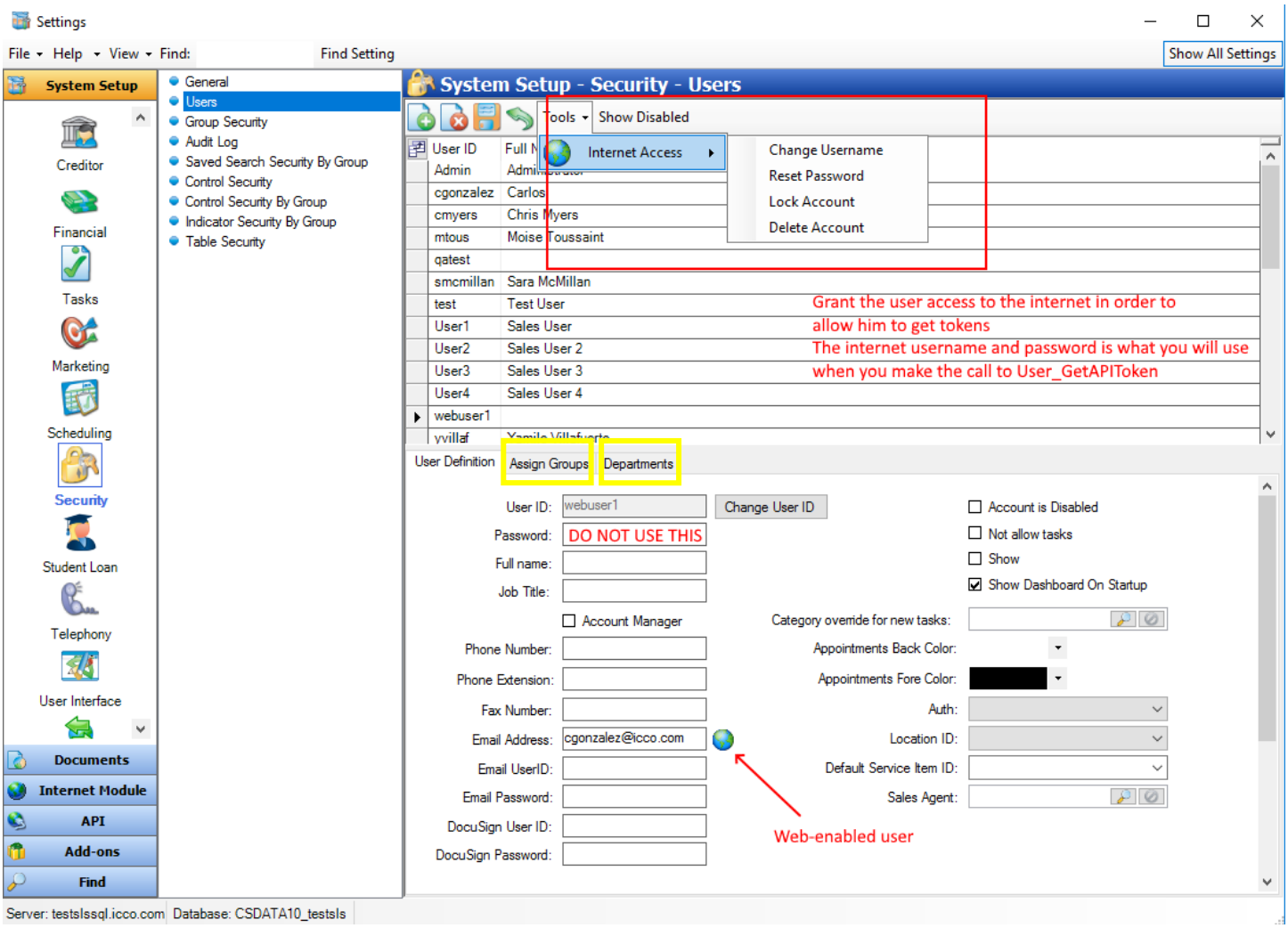
```
<DataContract()>
Public Class TokenDataObject
       <DataMember()> Public UserName As String ' This one is the web user linked to
aspnet_users/ezy_users, not the API user
       <DataMember()> Public CSRecordID As Integer
       <DataMember()> Public CSRecordType As String
       <DataMember()> Public Token As String
       <DataMember()> Public ResponseErrorDescription As String

End Class
```

For more information about the TokenDataObject click here

**IMPORTANT: Notice we not only pass the API user and password, but we also need to pass the internet UserName and Password of a standard user in CreditSoft/CollectPlus that has been web-enabled by going to Tools | Internet Access. (See screenshot below)**

API Users are only used for the authorization header of the API call. Tokens require you send the username and internet password of a standard user from CreditSoft/CollectPlus so it can tag the operation as that user in the audit logs.

In addition, we pass the CSRecordID and CSRecordType of the record we want. The CSRecordType can be: Lead, Client, LeadCoApp, ClientCoApp




Screen above showing a web-enabled user. Also, make sure the user is assigned to at least ONE GROUP and ONE DEPARTMENT.

More info about tokens here

## Date and datetime formats

When passing a date via JSON you must use the standard JSON date format like this: "/Date(700000+240)/", where the first number (700000 in the example provided) is the number of milliseconds in the server time zone starting at midnight, January 1, 1970. The number may be negative to represent earlier times. The part that consists of "+240" in the example is the offset in minutes. In this case it's 240 minutes (4 hours or GMT-0400) which is the offset for Eastern Daylight Time.

For example:

Using "/Date(0)/" would result in this date on the server 1970-01-01 00:00:00

Using "/Date(200000)/" would result in this date on the server 1970-01-01 00:03:20

This is equal to 1970-01-01 plus 200 seconds (200 seconds equals 3 minutes and 20 seconds)

You can also use a helper function like `convertToWCFFormatFromStr`. See this:

```
// Converts a date from a string like "2012-04-13 22:00:15" to a format that JSON understands
function convertToWCFFormatFromStr(date){
        let _date = new Date(date);
```

```
        if (date.indexOf(' ') >= 0) {
                return "\/Date(" + _date.getTime() + "+" + _date.getTimezoneOffset() + ")\/";
        }
        else {
                // If we don't pass the time, then JS already performs the conversion so the time results in 00:00:00 on
the server
                return "\/Date(" + _date.getTime() + ")\/";
        }
}

var JSONObjectIssue = {
                "LeadID": $("#LeadID").val(),
                "IssueSummary": "Test from API REST",
                "IssueCategoryID": 2
                ,
                // This works, but it's hard to read (0 equals 1970-01-01). See the document ICCOWebServices11:
                //"DateCreated": "\/Date(0)\/"
                // This does not work, as we don't have that serializer
                //"DateCreated": "2012-04-23T18:25:43.511Z"
                // You can use this helper function to make it easier:
                "DateCreated": convertToWCFFormatFromStr("2012-04-23 22:10:13")
        };
```

Using convertToWCFFormatFromStr("2012-04-23 22:10:13") would result in the same date and time on the server (as long as the client browser is in the same time zone as the server. If the browser is in a different time zone then the time will be converted. This is done automatically as we pass the timezone offset when you call the function with a date and time.

So if the browser is in Central Time and the server is in Eastern Time, and we pass convertToWCFFormatFromStr("2012-04-23 22:10:13") then the resulting date and time on the server will be 2012-04-23 23:10:13. Because 10PM in Central Time (what we passed) is 11PM in Eastern Time (the server time).

Keep in mind that when using the / character in JSON you must escape it first with a \ character.

So in reality, if you wanted the final string to be "/Date(0)/" then you would actually use "\/Date(0)\/" in the code. See the code example above.

# Troubleshooting

## User account not found

If you get his response:

```
{
        "d": "{\"UserName\":null,\"CSRecordID\":0,\"CSRecordType\":null,\"Token\":null,\"ResponseErrorDescription\":\"User account not found\"}"
}
```

Then it means the username and password you passed are not valid. Make sure the username and password work by logging in as that web user on the web portal. See the screenshot above where the user is web-enabled.

## Response Code 415 – Unsupported Media Type

HTTPError: Response code 415 (Cannot process the message because the content type 'application/json' was not the expected type 'application/soap+xml; charset=utf-8'.)

The request failed with status code: UnsupportedMediaType

This is because you are trying to use a SOAP/XML against the REST API or a JSON request against the SOAP API.

Keep in mind the URLs for SOAP and REST are different.

The one for SOAP (using XML request) is WSICCOCore.svc

and the one for REST (using JSON request) is WSICCOCoreREST.svc

### Access is denied / 401 Unauthorized

You will get an "Unauthorized" response when
The API Username does not exist
The API Password does not match the one for that API Username
The API Username does not have permission to have REST call
The call is INTERNALONLY and you call it from a server not in the list of internal servers

The permissions are set via Settings | API | API Users. You must check "Allow REST Calls" and choose the calls you want that API User to make by adding them via the multi-select control called REST Calls Allowed.

Unauthorized calls will show in the Web Security Log if the API Username exists.

### 405 Method Not Allowed

You used a GET method instead of POST or vice-versa

### Invalid Lead Token

You will get this in the `ResponseErrorDescription` when any of these occur:

- The token element is missing
- The token element exists but the Token property is empty
- The LeadID property is 0 or empty
- The Token property does not contain a Lead Token (contains a different token)
- The Token property is for a different LeadID than the LeadID element you passed in the token object

### An unsecured or incorrectly secured fault was received from the other party. See the inner FaultException for the fault code and detail.An error occurred when processing the security tokens in the message.

This means the API Username or API Password you are passing are either bad or the account is locked.

### Sales Force ID Not Found

This means that the SalesForceID passed is not valid or has been added recently. If you know it is valid, please RECYCLE the application pool of the API so it reloads the Settings tables, as SalesForce is a cached table.

## Common Postman Errors

### Mixed content error: cannot send request.  The request has been blocked because it requested an insecure HTTP resource.

If you use the Postman Desktop Agent you will get this one: Error: getaddrinfo ENOTFOUND {{base_url}}

In the Console you will notice that the variable is not being replaced: You will see something like:

http://{{base_url}}/About

This means that the Environment variables are not getting replaced. Make sure the Environment is ACTIVE. Click on Environments and check the box to the right of the environment name.

## Testing Framework

There are several ways to test calling the API.

### WSTester.html - AJAX

The API site has a built-in tester form. The file is WSTester.html in the root directory.

That tester uses AJAX with Javascript to test the REST calls and displays the responses on that same page.

### WSCreditSoftTester.exe

ICCO provides a Windows application called WSTester.exe. The application allows testing SOAP calls.

### Postman

Postman documentation is here.

https://documenter.getpostman.com/view/31280382/2s9YeAAuTh

## How to create your postman project

### Create your postman account and login

### Create an environment first and add the following variables:

base_url

This will be the base URL ending in WSICCOCoreREST.svc

Apiusername

This will be the API user you created in API Users in Settings

apipassword

### Create a collection and configure authorization

Create a collection and then click on the collection and the Authorization tab

Select Basic Authentication

For username enter: {{apiusename}}

For password enter: {{apipassword}}

### Add a request called TestConnection to the collection

Add a request and call it TestConnection

The type of the Request is GET

Notice that in the Authorization tab of the request it should default to “Inherit auth from parent”

In the Get type this: {{base_url}}/TestConnection

Click Send

After sending, you should get a 200 OK response.

In the body you should get something like this:

{

"d": "\"This is the WSICCOCoreREST service on Monday, October 28, 2024 at 11:41:45 AM\\r\\nConnected to version: 0\\r\\nSQL Server: SQL2019INT\\r\\nDatabase: CSDATA11_TEST\""

}

## Add a request called User_GetAPIToken

This is an example of a request where you will need to pass a JSON body with the parameters.

Click the collection and we are going to add additional variables that we will need for the User_GetAPIToken call.

Add the following variables:

WebUserName

The internet account username of a webuser (internal user from Security | Users – from the ezy_Users table)

WebUserPassword

The password of the webuser above

CSRecordID

A lead or client ID

CSRecordType

Lead or Client

Add a request and call it User_GetAPIToken

The type of the request is POST

In the Post type this: {{base_url}}/User_GetAPITokenJ

We are going to leave the Params tab EMPTY. The parameters will be going in a JSON object in the Body tab instead.

In the body, select type Raw JSON and type this:

{

"objUser": {

"UserName": "*{{WebUserName}}*",

"Password": "*{{WebUserPassword}}*",

"CSRecordID": *{{CSRecordID}}*,

"CSRecordType": "*{{CSRecordType}}*"

}

}

Click Send

You should get something like this:


{

"d": {

"__type": "TokenDataObject:#ICCOCoreAPI",

"CSRecordID": 10601,

"CSRecordType": "Lead",

"Remarks": **null**,

"ResponseErrorDescription": **null**,

"Token":
"A4GOC2nhCzZQa8ZEcqx7Lrc8TVCBIU8BVLtAYLoXKaLTRjjvFYbEQyBy7m2B1Tqm7m0CdvSybCpnvrUK12mDea488lm2z8b+6kDUueinUF9sLbCNqD5StkEwBIFFV9wvvHyHCfT+PTgXGgHA7TsQ5f1NgSJk/qgPsKZkWn4/GGVsKope0ZxohpQ56b6h9bZHzN4YSHmwbM4YzmYsFFZi79uydiz4NrvNNC8IIa88EQG+MUJNtowHH0r4lA7LR7sYGBtn4vsqvOpD3I4//gQ==",

"UserName": "yvillafweb"

}

}


# Development Framework

We recommend you use Visual Studio 2017 or Visual Studio Code.

When using Visual Studio for developing the integration please be sure to add the reference first by right clicking on the project and selecting "Add Service Reference". You will need to temporarily allow metadata on the config file of the web service site in order to add the reference.

# Quick Links to Topics


Lead Simple Calls

# Commonly Used Objects

## LeadClientDataObject

| FieldName | DataType |
|---|---|
| ClientID | int - only used for updating existing Lead |
| Salutation | Mr., Mrs., Ms. |
| FirstName | nvarchar(50) |
| LastName | nvarchar(50) |
| MiddleInitial | nvarchar(50) |
| Phone | nvarchar(50) send 10 digits without formatting |
| WorkPhone | nvarchar(50) send 10 digits without formatting |
| CellPhone | nvarchar(50) send 10 digits without formatting |
| DateEntered | smalldatetime |
| User | UserDataObject<br>This is the internet account username and password for this lead |
| Email | nvarchar(50) |
| DOB | smalldatetime |
| SSN | nvarchar(14) send 9 digits without formatting |
| Address1 | nvarchar(50) |
| Address2 | nvarchar(50) |
| City | nvarchar(50) |
| State | nvarchar(30) |
| Zip | nvarchar(15) |
| UserDefined1 | nvarchar(50) |
| UserDefined2 | nvarchar(50) - See Setup for Formatting |
| UserDefined3 | nvarchar(50) - See Setup for Formatting |
| UserDefined4 | nvarchar(50) - See Setup for Formatting |
| UserDefined5 | nvarchar(50) - See Setup for Formatting |
| UserDefined6 | nvarchar(60) |
| UserDefined7 | nvarchar(50) - See Setup for Formatting |
| LocationID | integer - From Locations Table |
| EnteredBy | nvarchar(20) - UserID from Ezy_Users Table |
| Counselor_UserID | nvarchar(20) - UserID from Ezy_Users Table |
| HomePhoneTime | nvarchar(10) - From PhoneTime Table |
| BankruptcyAttorney | nvarchar(250) - AttorneyID from BankruptcyAttorney Table |

| | |
|---|---|
| MaritalStatus | int - From MaritalStatus Table |
| RaceID | int - From Race Table |
| EthnicityID | int - From Ethnicity Table |
| NumberofParents | int |
| NumberofChildren | int |
| RentOwn | RENT/OWN |
| Working | True/False |
| CollectionCL | True/False |
| AttorneyCL | True/False |
| Bankruptcy | True/False |
| Notes | nttext |
| ContactMethodID | int - From ContactMethod Table |
| ReasonID | int - From CallReason Table |
| Referral | nvarchar(30) - From ReferralSource Table |
| AdvertisingID | int - From Advertising Table |
| Employer | nvarchar(25) |
| YearsAtJob | float |
| Occupation | nvarchar(250) |
| BankruptcySessionAmount | money |
| BankruptcySessionPaymentType | int - From BankruptcySessionPaymentTypes Table |
| BankruptcySessionPaidStatus | int - From BankruptcySessionPaidStatuses Table |
| GrossIncome | money |
| ProposalComments | nvarchar(146) |
| CreditorComments | nvarchar(146) |
| Language | int - From Languages Table |
| PaymentDay | smallint - 1-28 |
| PINNumber | nvarchar(8) |
| DriverLicense | nvarchar(50) |
| DriverLicenseState | nvarchar(30) |
| MaidenName | nvarchar(100) |
| GrossIncome | money |
| OTRDetail | nvarchar(40) |
| Year | int |
| DMPDebtRangeID | int - From DMPDebtRange Table |
| TaxFilingJointly | int - 0/1 |
| NumberofChildren | int |
| NumberofAccounts | int |

| | |
|---|---|
| BillStatus | nvarchar(20) - From BillStatus Table |
| CurrentMonthlyPayments | money |
| DebtorRepaymentPlan | nvarchar(20) - From DebtorRepaymentPlan Table |
| EstimatedMonthlyPayment | money |
| TypeCall | nvarchar(50) - From TypeCall Table |
| PSUserDefined1 | decimal** |
| PSUserDefined2 | decimal** |
| PSUserDefined3 | decimal** |
| PSUserDefined4 | decimal** |
| PSUserDefined5 | decimal** |
| PSUserDefined6 | decimal** |
| PSUserDefined7 | decimal** |
| PSUserDefined8 | decimal** |
| PSUserDefined9 | decimal** |
| UDFList | an array of IDs and Values following the structure of WSGenericResult |
| MilitaryStatusID | int - From MilitaryStatus Table |
| EduLevelID | int - From EduLevel Table |
| LeadAdminStatus | nvarchar(20) - From LeadAdminStatus Table |
| PublicService | True/False |
| TotDebt | money |
| FeeAsOfDate | Datetime |
| HouseNumber | nvarchar(10) (for Equifax) |
| Quadrant | nvarchar(2) (for Equifax) |
| StreetName | nvarchar(50) (for Equifax) |
| StreetType | nvarchar(20) (for Equifax) |
| ApartmentNumber | nvarchar(10) (for Equifax) |
| PostQuadrant | nvarchar(2) (for Equifax) |
| UnitDesignator | nvarchar(20) (for Equifax) |
| UseACH | True/False |
| NewsLetterOptIn | True/False |
| OccupationCategoryID | int - From OccupationCategory Table |
| ReferralPermission | True/False |
| FamilySize | Int |
| HomePhoneSMSEnabled | Boolean – nullable |
| WorkPhoneSMSEnabled | Boolean – nullable |
| MobilePhoneSMSEnabled | Boolean – nullable |
| HouseholdHead | Boolean – True/False |

| | |
|---|---|
| EmploymentStatus | Nvarchar (50) – must be a choice in the NFCC_EmploymentStatus table |

** In order to use PSUserDefined# fields the setting found in Internet Module > Preferences called "API_EnablePSUserDefinedFields" must be set to 1

PSUserDefinedField_GetDataType can be used to retrieve the datatype required for each field.

## WSGenericResult object

```
<DataContract()> _
Public Class WSGenericResult
	<DataMember()> Public ID As String
	<DataMember()> Public Value As String
	<DataMember()> Public ResponseErrorDescription As String

End Class
```

### WSResponse object

```
<DataContract()> _
Public Class WSResponse
	<DataMember()> _
	Public RecordID As Integer = 0

	<DataMember()> _
	Public ErrorDescription As String = String.Empty

	Public Sub New(ByVal Value As Integer)
		RecordID = Value
	End Sub

	Public Sub New(ByVal Message As String)
		ErrorDescription = Message
	End Sub

End Class
```

# Connection Test Calls

To test connectivity to the web services API site you can use the following calls

### About (REST Available)

This is the simplest call available. It returns a string with the version number of the API.

It does not require you send an Authorization header

```
  <OperationContract()>
  Function About() As String
```

### TestConnection (REST ONLY)

This one does require the Authorization header with the API user and password.

You will need to configure the call to be allowed for that API user in Settings

```
	<OperationContract()>
	Public Function TestConnection() As String
```

# Lead Related Calls

All the following Web Calls will return a WSResponse to indicate if the data was saved successfully or not. On success a RecordID will be returned. On error, the RecordID will be 0 and an error message will be returned.

### AddLead

Inserts Lead into the database using the following parameters and returns the LeadID.

```
  <OperationContract()>
```

```
Function AddLead(ByVal FirstName As String,
                 ByVal LastName As String,
                 ByVal MiddleInitial As String,
                 ByVal Phone As String,
                 ByVal Fax As String,
                 ByVal [Date] As String,
                 ByVal Email As String,
                 ByVal WorkPhone As String,
                 ByVal CellPhone As String,
                 ByVal SSN As String,
                 ByVal Street As String,
                 ByVal City As String,
                 ByVal State As String,
                 ByVal Zip As String,
                 ByVal UserDefined1 As String,
                 ByVal PreferredComunication As String,
                 ByVal Gender As String,
                 ByVal StartDate As Nullable(Of Date),
                 ByVal ProgramScheduleID As Nullable(Of Integer),
                 ByVal SalesForceID As Nullable(Of Integer),
                 ByVal LocationID As Nullable(Of Integer),
                 ByVal Counselor_UserID As String,
                 ByVal DOB As Nullable(Of Date)) As WSResponse
```

| **FieldName** | **DataType** |
|---|---|
| FirstName | nvarchar(50) |
| LastName | nvarchar(50) |
| MiddleName | nvarchar(50) |
| HomePhone | nvarchar(50) send 10 digits without formatting |
| Fax | nvarchar(50) send 10 digits without formatting |
| DateEntered | smalldatetime |
| Email | nvarchar(50) |
| WorkPhone | nvarchar(50) send 10 digits without formatting |
| Mobile | nvarchar(50) send 10 digits without formatting |
| SSN | nvarchar(14) send 9 digits without formatting |
| Address1 | nvarchar(50) |
| City | nvarchar(50) |
| State | nvarchar(30) |
| Zip | nvarchar(15) |
| UserDefined1 | nvarchar(50) |
| PreferredCommunication | Email,Fax,Mail |
| Gender | M, F |
| StartDate | smalldatetime |

| | |
|---|---|
| ProgramScheduleID | integer - From ProgramSchedules Table |
| SalesForceID | integer - From SalesForce Table |
| LocationID | integer - From Locations Table |
| Counselor_UserID | nvarchar(20) - UserID from Ezy_Users Table |
| DOB | smalldatetime |

## AddLeadDataObject (REST Available)

Inserts Lead into the database using the following dataobject with fields provided and returns the LeadID.

Here is the link to the LeadClientDataObject

```
<OperationContract()>
Function AddLeadDataObject(ByVal NewLead As LeadClientDataObject) As WSResponse
```

Below is sample code in JQuery (Many PHP and JQuery examples are provided in the package).

Notice the JSON object is defined using the fields available in the LeadClientDataObject

Notice we pass the Authorization header

And finally notice the response, being of type WSResponse, has a RecordID property which contains the ID of the lead we just added.

```
$('#btnAddLeadDataObject').click(function () {

  var JSONObjectLead = { "NewLead": { "FirstName": "Sally", "LastName": "Brown", "Email":
"myemail@test.com" } };

  $.ajax({
    type: 'POST',
    url: URL + '/AddLeadDataObject',

    headers: { 'Authorization': btoa('apiuser1:myapipassword') },

    data: JSON.stringify(JSONObjectLead),
    dataType: 'json',
    contentType: 'application/json; charset=utf-8',
                          success: function (r) {

                                  var response = JSON.parse(r.d);
                                  $("#LeadID").val(response.RecordID.toString());

                          },
    error: function (e) {
      $("#div1").html('ERROR');
                          }
  });
});
```

## AddLeadCoApp (REST Available)

Adds CoApp to existing Lead using the following parameters and returns the CoappID

```
<OperationContract()>
Function AddLeadCoApp(ByVal LeadID As Integer, ByVal CoAppFirstName As String,
```

```
ByVal CoAppLastName As String,
ByVal CoAppSSN As String, ByVal CoAppEmail As String,
ByVal IsSpouse As Boolean,
ByVal RelationShip As String,
ByVal CoDOB As Nullable(Of Date),
ByVal CoGender As String,
Optional ByVal CoAppEmployed As Boolean = False,
Optional ByVal CoAppEmployer As String = Nothing,
Optional ByVal CoAppHomePhone As String = Nothing,
Optional ByVal CoAppWorkPhone As String = Nothing,
Optional ByVal CoAppOtherPhone As String = Nothing) As WSResponse
```

| FieldName | DataType |
|---|---|
| LeadID | int - must exist in database |
| CoAppFirst | nvarchar(50) |
| CoAppLast | nvarchar(50) |
| CoAppSSN | nvarchar(14) send 9 digits without formatting |
| CoAppEmail | nvarchar(50) |
| IsSpouse | True/False |
| Relationship | nvarchar(20) - From RelationshipTypes Table |
| CoDOB | smalldatetime |
| CoGender | M,F |
| CoappEmployed | True/False |
| CoAppEmployer | nvarchar(30) |
| CoAppHomePhone | nvarchar(50) send 10 digits without formatting |
| CoAppWorkPhone | nvarchar(50) send 10 digits without formatting |
| CoAppOtherPhone | nvarchar(50) send 10 digits without formatting |

## AddLeadContactDataObject

Adds Contact to existing Lead using the following parameters and returns the ClientContactID

```
<OperationContract()>
Function AddLeadContactDataObject(ByVal NewLeadContact As ClientContactDataObject) As WSResponse
```

| FieldName | DataType |
|---|---|
| LeadID | int - must exist in database |

| | |
|---|---|
| ContactFirst | nvarchar(30) |
| ContactLast | nvarchar(30) |
| ContactMiddleName | nvarchar(1) |
| ContactSSN | nvarchar(14) send 9 digits without formatting |
| ContactDOB | smalldatetime |
| ContactAge | smallint |
| ContactSex | M,F |
| ContactOccupation | nvarchar(20) |
| CoAppEmployer | nvarchar(30) |
| ContactNetIncome | money |
| ContactHomePhone | nvarchar(50) send 10 digits without formatting |
| ContactWorkPhone | nvarchar(50) send 10 digits without formatting |
| ContactOtherPhone | nvarchar(50) send 10 digits without formatting |
| ContactAddress1 | nvarchar(100) |
| ContactAddress2 | nvarchar(50) |
| ContactCity | nvarchar(50) |
| ContactState | nvarchar(30) |
| ContactZIP | nvarchar(15) |
| Relationship | nvarchar(20) - From RelationshipTypes Table |
| ContactEmail | nvarchar(50) |
| ContactMarital | nvarchar(10) |
| ContactYearsAtJob | int |
| ContactNotes | ntext |
| ContactDepartment | nvarchar(50) |
| ContactTitle | nvarchar(50) |
| ContactHomePhoneCommunicationPreferred | int - 0/1 |
| ContactHomePhoneDoNotUse | int - 0/1 |
| ContactWorkPhoneCommunicationPreferred | int - 0/1 |
| ContactWorkPhoneDoNotUse | int - 0/1 |
| ContactOtherPhoneCommunicationPreferred | int - 0/1 |
| ContactOtherPhoneDoNotUse | int - 0/1 |
| ContactHomePhoneTime | nvarchar(10) - From PhoneTime Table |
| ContactWorkPhoneTime | nvarchar(10) - From PhoneTime Table |
| ContactOtherPhoneTime | nvarchar(10) - From PhoneTime Table |
| ContactHomePhoneSMSEnabled | int - 0/1 |

| | |
|---|---|
| ContactWorkPhoneSMSEnabled | int - 0/1 |
| ContactOtherPhoneSMSEnabled | int - 0/1 |
| ContactHomePhoneType | nvarchar(10) - From PhoneType Table |
| ContactWorkPhoneType | nvarchar(10) - From PhoneType Table |
| ContactOtherPhoneType | nvarchar(10) - From PhoneType Table |
| ContactAddress1CommunicationDoNotUse | int - 0/1 |
| ContactAddress1CommunicationPreferred | int - 0/1 |
| ContactEmailCommunicationPreferred | int - 0/1 |
| ContactEmailCommunicationDoNotUse | int - 0/1 |
| IsDefaultContact | int - 0/1 |
| ContactUserDefined1 | nvarchar(50) - See Setup for Formatting |
| ContactUserDefined2 | nvarchar(50) - See Setup for Formatting |
| ContactUserDefined3 | nvarchar(50) - See Setup for Formatting |

## AddLeadBankAccount (REST Available)

Adds Lead Bank Account to existing Lead using the following parameters and returns the LeadACHBankAccountID

```
<OperationContract()>
Function AddLeadBankAccount(ByVal LeadID As Integer,
                            ByVal BankAccountType As String,
                            ByVal BankName As String,
                            ByVal BankRoutingNumber As String,
                            ByVal BankAccountNumber As String) As WSResponse
```

| FieldName | DataType |
|---|---|
| LeadID | int - must exist in database |
| AccType | nvarchar(10) Checking/Savings |
| BankName | nvarchar(30) - Leave empty to auto populate from CreditSoft Federal ACH Directory Info |
| ABA | nvarchar(9) |
| BankAccountNumber | nvarchar(20) |

## AddLeadAccount (REST Available)

Adds Lead Account to existing Lead using the following parameters and returns the ClientCredID. Before adding accounts an Insert/Default CreditorID must be specified in Settings. A quick note will be added for the account with the Creditor Name and Address Info.

```
<OperationContract()>
Function AddLeadAccount(ByVal LeadID As Integer,
                ByVal OriginalDebt As Nullable(Of Decimal),
                ByVal OriginalAPR As Nullable(Of Decimal),
                ByVal OriginalMonthly As Nullable(Of Decimal),
```

```
                    ByVal DefaultMonthlyPayment As Nullable(Of Decimal),
                    ByVal CreditorName As String,
                    ByVal CreditorAccountNumber As String,
                    ByVal CreditorStreet As String,
                    ByVal CreditorState As String,
                    ByVal CreditorZip As String,
                    ByVal CreditorPhone As String,
                    ByVal CreditorPhoneExt As String,
                    ByVal DelinquencyStatus As String,
                    ByVal CreditorID As Nullable(Of Integer),
                    ByVal CreditorCity As String,
                    Optional ByVal CallToAction As Boolean = False,
                    Optional ByVal HardShip As Boolean = False,
                    Optional ByVal MinimumAcceptedByCreditor As Nullable(Of Decimal) = Nothing,
                    Optional ByVal AccountStatus As String = Nothing
    ) As WSResponse
```

| FieldName | DataType |
|---|---|
| LeadID | int - must exist in database |
| OrigDebt | money |
| InitialAPR | float send as decimal |
| OrigMonthly | money |
| DefMonthlyPayment | money |
| CreditorName | nvarchar(50) |
| AccountNumber | nvarchar(50) |
| CreditorStreet | nvarchar(50) |
| CreditorState | nvarchar(30) |
| CreditorZip | nvarchar(15) |
| CreditorPhone | nvarchar(50) |
| CreditorExt | nvarchar(10) |
| DelinquencyStatus | nvarchar(50) as specified in the Settings |
| CreditorID | int - must exist in database. Pass nothing to use the Generic Default CreditorID. |
| CreditorCity | nvarchar(50) |
| CallToAction | True/False |
| HardShip | True/False |
| MinimumAcceptedByCreditor | money |
| AccountStatus | nvarchar(20) - must exist in the database - Pass nothing to use the Default Account Status |

## AddLeadBudget (REST Available)

Adds Lead Budget Item to existing Lead using the following parameters and returns the BudgetScenarioDetailID. Categories\SubCategories must be predefined in Settings

```
    <OperationContract()>
    Function AddLeadBudget(ByVal LeadID As Integer,
                           ByVal BudgetCategory As String,
                           ByVal BudgetSubcategory As String,
```

```
                                ByVal BudgetAmount As Decimal) As WSResponse
```

| FieldName | DataType |
|---|---|
| LeadID | int - must exist in database |
| Category | nvarchar(50) |
| SubCategory | nvarchar(50) |
| Amount | money |

## AddLeadIncome (REST Available)

Adds Lead Income Item to existing Lead using the following parameters and returns the IncomeScenarioD. Categories must be predefined in Settings

```
        <OperationContract()>
        Function AddLeadIncome(ByVal LeadID As Integer,
               ByVal IncomeCategory As String,
               ByVal IncomeAmount As Decimal) As WSResponse
```

| FieldName | DataType |
|---|---|
| LeadID | int - must exist in database |
| IncomeName | nvarchar(50) |
| IncomeValue | money |

## AddLeadAsset (REST Available)

Adds Lead Income Item to existing Lead using the following parameters and returns the AssetScenarioD. Categories must be predefined in Settings

```
        <OperationContract()>
        Function AddLeadAsset(ByVal LeadID As Integer,
                              ByVal AssetName As String,
                              ByVal AssetValue As Decimal) As WSResponse
```

| FieldName | DataType |
|---|---|
| LeadID | int - must exist in database |
| AssetName | nvarchar(50) |
| AssetValue | money |

## AddLeadLiability (REST Available)

Adds Lead Income Item to existing Lead using the following parameters and returns the LiabilityScenarioD. Categories must be predefined in Settings

```
        <OperationContract()>
        Function AddLeadLiability(ByVal LeadID As Integer,
                              ByVal LiabilityName As String,
```

```
                                ByVal LiabilityValue As Decimal) As WSResponse
```

| FieldName | DataType |
|---|---|
| LeadID | int - must exist in database |
| LiabilityName | nvarchar(50) |
| LiabilityValue | money |

## MarkOriginalBudgetScenarioAsActual (REST Available)

Marks the scenario as the Actual Budget Scenario - this should be called after all Expenses, Income, Assets, and Liabilities have been sent. Will return the BudgetScenarioMasterID.

```
<OperationContract()>
Function MarkOriginalBudgetScenarioAsActual(ByVal LeadID As Integer) As WSResponse
```

| FieldName | DataType |
|---|---|
| LeadID | int - must exist in database |

## AddSalesForce

Adds Sales Agents to CreditSoft using the following parameters and returns the SalesForceID. The WorksFor ID would be already in the database and the SalesmanType will be automatically set based on the WorksFor.

```
<OperationContract()>
Function AddSalesForce(ByVal FirstName As String,
                       ByVal LastName As String,
                       ByVal Phone As String,
                       ByVal PhoneExtension As String,
                       ByVal Email As String,
                       ByVal UserDefined1 As String,
                       ByVal WorksFor As Nullable(Of Integer)) As WSResponse
```

| FieldName | DataType |
|---|---|
| FirstName | nvarchar(50) |
| LastName | nvarchar(50) |
| Phone | nvarchar(25) |
| PhoneExtension | nvarchar(10) |
| Email | nvarchar(50) |
| UserDefined1 | nvarchar(15) |
| WorksFor | int - must exist in database currently |

## AddLeadIssue (REST Available)

Creates a Task for the Lead for the specified category. If no categoryid is passed the Default Category will be used. The User and Department Assignment will follow the Task Category Setup.

```
<OperationContract()>
```

```
    Function AddLeadIssue(ByVal LeadID As Integer, ByVal IssueSummary As String, ByVal
IssueCategoryID As Nullable(Of Integer), Optional ByVal DateCreated As Nullable(Of Date) = Nothing,
Optional ByVal Status As String = Nothing) As WSResponse
```

| FieldName | DataType |
|---|---|
| LeadID | int - must exist in database |
| IssueSummary | nvarchar(2000) |
| IssueCategoryID | int - from IssueCategories Table |
| DateCreated | datetime |
| Status | nvarchar(20) - must exist in the database - Pass nothing to use the Default Task Status |

## AddIssueNote

Adds a Task Note to a Task and attaches a Document if specified.

```
  <OperationContract()>
  Function AddIssueNote(ByVal IssueID As Integer, ByVal NoteDate As Date, ByVal Notes As String,
Optional ByVal DocumentName As String = Nothing, Optional ByVal Document As Byte() = Nothing, Optional
DocumentType As String = Nothing) As WSResponse
```

| FieldName | DataType |
|---|---|
| IssueID | int - must exist in database (see AddLeadIssue call) |
| NoteDate | smalldatetime |
| Notes | ntext |
| DocumentName | nvarchar(1024) - Name of Actual file including the extension |
| Document | This will be a byte array (Byte()) that will contain the contents of the file to be attached to the Task. |

## AttachDocumentToIssue (REST Available)

Adds a Task Note to a Task and attaches the Document specified.

```
  <OperationContract()>
  Function AttachDocumentToIssue(ByVal IssueID As Integer, ByVal DocumentName As String, ByVal Document
As Byte(), Optional DocumentType As String = Nothing) As WSResponse
```

| FieldName | DataType |
|---|---|
| IssueID | int - must exist in database (see AddLeadIssue call) |
| DocumentName | nvarchar(1024) - Name of Actual file including the extension |

| | |
|---|---|
| Document | This will be a byte array (Byte()) that will contain the contents of the file to be attached to the Task. |

## AddValidatedGlobalAccountToLead

Adds a processing account to the lead that was previously assigned, enrolled and validated at Global. The account will be added and then re-exported to Global to update the clientid.

```
        <OperationContract()>
        Function AddValidatedGlobalAccountToLead(ByVal _
 LeadClientID As Integer,
 ByVal ABA As String,
 ByVal AccountNumber As String,
 ByVal DateCreated As DateTime,
 ByVal DateAssigned As DateTime,
 ByVal PIN As String,
 ByVal PolicyGroupID As Integer) As WSResponse
```

| FieldName | DataType |
|---|---|
| LeadClientID | int - must exist in database |
| ABA | nvarchar(9) - ABA for the Global Acct |
| AccountNumber | nvarchar(20) - Global Account # |
| DateCreated | datetime - Date Account Created for assignment by Global |
| DateAssigned | datetime - Date Account Assigned by vendor |
| PIN | nvarchar(4) - PIN for the Global Acct |
| PolicyGroupID | int - Policy Group with Global for the client |

## AddLeadCreditCard (REST Available)

Adds a credit card for processing credit card payments using the following dataobject with fields provided to an existing lead.

```
        <OperationContract()>
        Function AddLeadCreditCard(ByVal NewLeadCC As DebtorCCDataObject) As WSResponse
```

| FieldName | DataType |
|---|---|
| LeadClientID | int - must exist in database |
| CCFirstName | nvarchar(50) - Don't specify in object to use the Lead's Info |
| CCLastName | nvarchar(50) - Don't specify in object to use the Lead's Info |
| CCAddress1 | nvarchar(50) - Don't specify in object to use the Lead's Info |

| | |
|---|---|
| CCAddress2 | nvarchar(50) - Don't specify in object to use the Lead's Info |
| CCCity | nvarchar(50) - Don't specify in object to use the Lead's Info |
| CCState | nvarchar(30) - Don't specify in object to use the Lead's Info |
| CCZip | nvarchar(15) - Don't specify in object to use the Lead's Info |
| CCNumber | nvarchar(100) |
| CCExpirationMonth | int |
| CCExpirationYear | int |
| CCType | int - 0 Visa, 2 Mastercard, 3 Amex, 4 Discover |
| CCVerification | nvarchar(10) |
| CCAccountName | nvarchar(20) - Alias used in CreditSoft to refer to the account for scheduling purposes - Don't specify in object to use the database default value |

## AttachDocumentToLead (REST Available)

Adds a Task Note to a Lead and attaches the Document specified.

```
<OperationContract()>
        Public Function AttachDocumentToLead(ByVal LeadID As Integer, ByVal DocumentName As String,
        ByVal Document As Byte(), Optional DocumentType As String = Nothing) As String
```

| **FieldName** | **DataType** |
|---|---|
| LeadID | int - must exist in database |
| DocumentName | String Name of Actual file including the extension |
| Document | String This will be a byte array (Byte()) that will contain the contents of the file to be attached to the Task. |
| DocumentType | String Refers to the DocumentTypes enum – can be null |

# Lead Calls with API Token

Some calls require that you pass either a LeadAPIToken or a new standard APIToken, as mentioned in the section at the top called Tokens and token users.

## OBTAINING TOKENS

Below are the calls you can use to retrieve the different types of tokens

To obtain a Lead API Token use User_GetLeadAPIToken.

### User_GetLeadAPIToken

Returns a LeadTokenDataObject  for the UserTokenDataObject specified

***INPUT: UserTokenDataObject***

***OUTPUT: LeadTokenDataObject***

To get a new standard API Token use User_GetAPIToken.

### User_GetAPIToken

Returns a standard TokenDataObject for the WebUserTokenDataObject specified. The token is valid only for the username passed, and expires based on the Token Expiration in Minutes defined in Settings.

```
<OperationContract()>
Function User_GetAPIToken(ByVal objUser As WebUserTokenDataObject) As TokenDataObject
```

***INPUT: WebUserTokenDataObject***

Example JSON body with Postman variable bookmarks:

```
{
  "objUser": {
    "UserName": "{{WebUserName}}",
    "Password": "{{WebUserPassword}}",
    "CSRecordID": {{CSRecordID}},
    "CSRecordType": "{{CSRecordType}}"
  }
}
```

***OUTPUT: TokenDataObject***

Example JSON result:


```
{
   "d": {
      "__type": "TokenDataObject:#ICCOCoreAPI",
      "CSRecordID": 10601,
      "CSRecordType": "Lead",
      "Remarks": null,
      "ResponseErrorDescription": null,
      "Token":
"A4GOC2nhCzZa8ZEcqx7Lrc8TVCBIU8BVLtAYLoXKaIoTjZoEEm5pbwB487anebwWhLMrMRe0JwfuaxvJRTydQAEDg+b1YNa
W1ShF9GBuaNSEtise7B65Vmn4wXuTbVRIWViicYzFTf7+kR+3lO6AQ1RAovj6gQIWQkmB4OpzHIDcHp7/JxkP5mxQaOzi+4g
Y4TtelZAPyjrHJ2wm9FIg4YcpSmqiQDYjrTluwPP9D/E8Y97KNpI9RSbf92wmg1jfTftYuBfT58nHs4or2jw==",
      "UserName": "testleaduser"
   }
}
```


Another way to get a token for a lead is to know the internet username and password of the lead. This call should be used only by the lead (from a mobile app for example where the lead is entering their own credentials). You should exercise extreme caution if you ever cache or store a user's credentials.

### Lead_GetAPIToken

Returns the LeadTokenDataObject  for the UserDataObject specified

***INPUT: UserDataObject with lead's internet credentials***

***OUTPUT: LeadTokenDataObject***

# Objects used for authentication and tokens

Below are all the input/output objects used on the calls above to obtain tokens.

### Objects used to request and obtain tokens

#### *UserDataObject*

| FieldName | DataType |
|---|---|
| Username | nvarchar(256) |
| Password | nvarchar(128) |

```
<DataContract()> _
Public Class UserDataObject
	<DataMember()> Public UserName As String
	<DataMember()> Public Password As String

End Class
```

### *UserTokenDataObject*

Username and Password refer to the internet username and password of a web-enabled user - see User Setup in Settings

| FieldName | DataType |
|---|---|
| Username | nvarchar(256) |
| Password | nvarchar(128) |
| LeadID | Integer |

```
<DataContract()>
Public Class UserTokenDataObject
	<DataMember()> Public UserName As String 'web username from aspnet_users/Ezy_Users, NOT API
	<DataMember()> Public Password As String 'web password from aspnet_users/Ezy_Users
	<DataMember()> Public LeadID As Nullable(Of Integer) 'LeadID that the user wants to modify

End Class
```

### *WebUserTokenDataObject*

```
<DataContract()>
Public Class WebUserTokenDataObject
	<DataMember()> Public UserName As String 'web username from aspnet_users/Ezy_Users, NOT API
	<DataMember()> Public Password As String 'web password from aspnet_users/Ezy_Users
	<DataMember()> Public CSRecordID As Nullable(Of Integer) 'CSRecordID that the user wants to
modify
	<DataMember()> Public CSRecordType As String 'CSRecordType that the user wants to modify

End Class
```

## Token objects obtained after requesting them

### *LeadTokenDataObject*

```
<DataContract()> _
Public Class LeadTokenDataObject
	<DataMember()> Public LeadID As Integer
	<DataMember()> Public Token As String
	<DataMember()> Public ResponseErrorDescription As String

End Class
```

### *TokenDataObject*

```
<DataContract()>
Public Class TokenDataObject
```

```
        <DataMember()> Public UserName As String ' This one is the web user linked to
aspnet_users/ezy_users, not the API user
        <DataMember()> Public CSRecordID As Integer
        <DataMember()> Public CSRecordType As String
        <DataMember()> Public Token As String
        <DataMember()> Public ResponseErrorDescription As String


        Public Sub New()
        End Sub
End Class
```

Example of a TokenDataObject in JSON returned by User_GetAPITokenJ


```
{
  "UserName": "yvillafweb",
  "CSRecordID": 10589,
  "CSRecordType": "Lead",
  "Token":
"A4GOC2nhCzZQa8ZEcqx7Lrc8TVCBIU8BVLtAYoXKaLYa8Ajez3PDMFYq1kPGUkvVFqUiY55b7Biu5sqP9tmgXEN64whs
SAGbqX/wGBI/hOm00FOhhgRXwZVZuA81Sk6yLOCABiQsGB237/isa/9M18Zvn7JUWxFuhvymyeRouadX99rd7EW4VpwtA
cvlMaMLpOW0LTq4EUVSTf5pr39PB2MlIn/FTJKgsS7ctQsR9UtjvG+cUlH+pi46YIBq3tS/YhBlQAb1QM0fYAi+n3g=="
,
  "ResponseErrorDescription": null
}
```


When passing the token to an API (for example RequestCreditReportXML3) you would use it like this:


```
{ “Token” :
        {
          "UserName": "yvillafweb",
          "CSRecordID": 10589,
          "CSRecordType": "Lead",
          "Token":
        "A4GOC2nhCzZQa8ZEcqx7Lrc8TVCBIU8BVLAYLoXKaLYaAjez3PDMFYq1kPGUkvVFqUiY55b7Biu5sqP9tmgXEN
        64whsSAGbqX/wGBI/hOm00FOhhgRXwZVuA81Sk6yLOCBiQsGBJ237/isa/9M18Zvn7JUWxFuhvymyeRouadX99r
        d7EW4VpwtAcvlMaMLpOW0LTq4EUVSTf5pr39PB2MlIn/FTJKgsS7cQR9UtjvG+cUlH+mpi46YIBq3tS/YhBlQAb
        1QM0fYAi+n3g=="
        }
}
```


Another example, using it in GetStoredDocumentList after obtaining the token from GetMobileAPIToken:


```
{ "Token":

  {

    "__type": "TokenDataObject:#ICCOCoreAPI",

    "CSRecordID": 10559,

    "CSRecordType": "Client",

    "Remarks": null,

    "ResponseErrorDescription": null,
```

"Token":
"A4GOC2nhCzZQa8ZEcqx7Lrc8TVCBIUBVLtAYLoXKJEtoO1Q2wyda5PF+rHIHI7xdi61CNqE7ZuDFOGVg8g/W7wV4UzAEHwZ0Nd0d1gQHnsDwLk1Z+6c0R7XY5WNdKqzRi0g9GlaNIDRQbyhtVD7H0L0j0T/g76j0/f/I7GNJXsT27moWbZ3/drpjzsW1Sg6Gu7NRFiDF4dPQQYxJjMuMZM81qFfhndpNTK0HpwRjtjvZ5UslTMrRBV03f3kk+pUqRoOUmsmr0wcR+SNA==",

"UserName": "Rodrigo"

}

}

# TOKENIZED CALLS

NOTE: All the Web Calls will return a **ResponseErrorDescription** property to indicate if the data was saved successfully or not. If the ResponseErrorDescription property is empty then the call succeeded.

## LeadDataObject_Set, LeadDataObject_Get, LeadDataObject_Set2 & LeadDataObject_Get2

The LeadDataObject_Set and LeadDataObject_Get calls are for a lead updating itself and uses the LeadTokenDataObject.

The LeadDataObject_Set2 and LeadDataObject_Get2 calls are for a web user updating a lead and uses the TokenDataObject.

The LeadDataObject_Get will retrieve the lead specified in the LeadToken.LeadID

LeadDataObject_Get2 will retrieve the lead specified in the Token.CSRecordID (Token.CSRecordType must be “Lead”)

When adding or updating a lead, we do not use the LeadToken.LeadID or Token.CSRecordID/CSRecordType. We only look at the LeadClientDataObject.ClientID property to identify which lead to add/update.

The LeadDataObject_Set2 will ADD a new lead if the LeadClientDataObject object has a ClientID of 0 or negative.

The LeadDataObject_Set2 will UPDATE an existing lead if the LeadClientDataObject object has a ClientID greater than 0. It will update the lead based on the LeadClientDataObject.ClientID property, not the Token.CSRecordID.

Best practices:

Before calling LeadDataObject_Set or LeadDataObject_Set2 to UPDATE, you should first call LeadDataObject_Get or LeadDataObject_Get2 to retrieve a current version of the lead object. Then modify the field(s) and pass that object to LeadDataObject_Set/LeadDataObject_Set2 to update the database.

## LeadDataObject_Set [REST Available only with the J version]

Adds or updates a Lead. To create a new Lead the First Name, Last Name, Email, Username, and Password are all required.

```
<OperationContract()>
Function LeadDataObject_Set(ByVal LeadToken As LeadTokenDataObject, ByVal Lead As
LeadClientDataObject) As LeadTokenDataObject
```

***INPUT: LeadTokenDataObject, LeadClientDataObject***

***OUTPUT: LeadTokenDataObject***

The LeadTokenDataObject returned is the token the lead can use to update itself after the record has been added.

## LeadDataObject_Get [REST Available only with the J version]

Returns Lead information

```
<OperationContract()>
Function LeadDataObject_Get(ByVal LeadToken As LeadTokenDataObject) As LeadClientDataObject
```

***INPUT: LeadTokenDataObject***

***OUTPUT: LeadClientDataObject***

## LeadDataObject_Set2 [REST Available only with the J version]

A web user adds or updates a Lead.

To create a new Lead the First Name, Last Name, Email, Username are all required. The Password will be generated randomly and sent to the email associated with the lead.

```
<OperationContract()>
Function LeadDataObject_Set2(ByVal Token As TokenDataObject, ByVal Lead As LeadClientDataObject)
As WSResponse
```

***INPUT: TokenDataObject, LeadClientDataObject***

***OUTPUT: WSResponse***

## LeadDataObject_Get2 [REST Available only with the J version]

A web user requests Lead information

```
<OperationContract()>
Function LeadDataObject_Get2(ByVal Token As TokenDataObject) As LeadClientDataObject
```

***INPUT: TokenDataObject***

***OUTPUT: LeadClientDataObject***

## LeadQuickQuestion_Set

Updates the Bill Status and DMPReasons for the Lead. A valid LeadTokenDataObject is required

All DMPReasons should be sent, even when doing an update, as the service will delete any existing reasons.

```
<OperationContract()>
Function LeadQuickQuestions_Set(ByVal LeadToken As LeadTokenDataObject, ByVal LeadQuickQuestions
As LeadQuickQuestionsDataObject) As LeadQuickQuestionsDataObject
```

## LeadQuickQuestions_Get

Returns the data stored for the following dataobject. A valid LeadTokenDataObject is required for this call.

```
        <OperationContract()>
        Function LeadQuickQuestions_Get(ByVal LeadToken As LeadTokenDataObject) As
LeadQuickQuestionsDataObject
```

### *LeadQuickQuestionsDataObject*

| FieldName | DataType |
|---|---|
| DebtorBillStatus | nvarchar(20) - From DebtorBillStatus Catalog |
| DMPReason | Collection of LeadClientDMPReasonDataObject |

### *LeadClientDMPReasonDataObject*

| FieldName | DataType |
|---|---|
| DMPReason | nvarchar(50) - From DMPReasons Catalog |

## LeadAccount_Set (REST Available)

Adds/updates a collection of accounts to an existing Lead using the following LeadAccountDataObject. A valid LeadTokenDataObject is required. Before adding accounts a Insert/Default CreditorID must be specified in Settings.

```
        <OperationContract()>
        Function LeadAccount_Set(ByVal LeadToken As LeadTokenDataObject, ByVal LeadAccounts As
LeadAccountDataObject()) As LeadAccountDataObject()
```

## LeadAccount_GetList (REST Available)

Returns the data stored for the following dataobject. A valid LeadTokenDataObject is required for this call.

```
        <OperationContract()>
        Function LeadAccount_GetList(ByVal LeadToken As LeadTokenDataObject) As LeadAccountDataObject()
```

## DeleteLeadAccount_Set

Deletes the specified account. Returns a WSGenericResult.

```
        <OperationContract()>
        Function DeleteLeadAccount_Set(ByVal LeadToken As LeadTokenDataObject, ByVal ClientCredID As
        Integer) As WSGenericResult
```

### *LeadAccountDataObject*

| FieldName | DataType |
|---|---|
| ClientCredID | Int - Unique ID for the specific account - must be specified when sending updates |
| AccountNumber | nvarchar(50) |

| | |
|---|---|
| DebtType | nvarchar(50) - From DebtTypes Catalog |
| Balance | money |
| OrigMonthly | money |
| InitialAPR | float send as decimal |
| DelinquencyStatus | nvarchar(50) - From DelinquencyStatus Catalog |
| CreditorID | int - From CreditorsForEnrollment Catalog |
| CreditorName | nvarchar(50) -Should be nothing when passing a CreditorID or if no CreditorID is passed the Generic Creditor will be used and the name will be stored for user to see in CreditSoft |
| DefMonthlyPayment | money - to have the system calculate the payment pass nothing or 0. |
| CreditorStreet | nvarchar(50) |
| CreditorCity | nvarchar(50) |
| CreditorState | nvarchar(30) |
| CreditorZip | nvarchar(15) |
| CreditorPhone | nvarchar(50) |
| CreditorPhoneExt | nvarchar(10) |
| CallToAction | True/False |
| HardShip | True/False |
| MinimumAcceptedByCreditor | True/False |
| AccountStatus | nvarchar(20) - must exist in the database - Pass nothing to use the Default Account Status |
| UserDefined1 | nvarchar(50) |
| UserDefined2 | nvarchar(50) |
| UserDefined3 | nvarchar(50) |
| CoAppID | Int - should be the CoAppID for the account holder |
| JointAccount | True/False |
| CSPTier1 | True/False |
| CSPTier2 | True/False |
| CSPTier3 | True/False |

## EstimatedProgramAmountDue_Get

Returns the program payment for a specified Lead. A valid LeadTokenDataObject is required.

```
        <OperationContract()>
        Function EstimatedProgramAmountDue_Get(ByVal LeadToken As LeadTokenDataObject) As Nullable(Of
Decimal)
```

## RetrieveMyInfo_Set

Updates the DOB and SSN for a specified Lead and creates CoApp or Updates specified CoApp. When adding/updating a CoApp the First Name and Last Name will be required. A valid LeadTokenDataObject is required.

```
        <OperationContract()>
        Function RetrieveMyInfo_Set(ByVal LeadToken As LeadTokenDataObject, ByVal MyInformation As
MyInformationDataObject) As MyInformationDataObject
```

## RetrieveMyInfo_Get

Returns the data stored for the following dataobject. A valid LeadTokenDataObject is required for this call.

```
        <OperationContract()>
        Function RetrieveMyInfo_Get(ByVal LeadToken As LeadTokenDataObject) As MyInformationDataObject
```

### *MyInformationDataObject*

| FieldName | DataType |
|---|---|
| DOB | smalldatetime |
| SSN | nvarchar(14) send 9 digits without formatting |
| CoApp | CoAppsDataObject |

### *CoAppsDataObject*

| FieldName | DataType |
|---|---|
| CoAppID | Int - Unique ID for the specific coapp- must be specified when sending updates |
| CoAppFirst | nvarchar(50) |
| CoAppLast | nvarchar(50) |
| CoAppDOB | smalldatetime |
| CoAppSSN | nvarchar(14) send 9 digits without formatting |

## LeadCoApp_Delete

Requires a LeadToken and CoAppID and returns a WSGenericResult

It will only allow deletes if the coapp is not linked to any accounts, bank accounts, or credit reports.

```
        <OperationContract()>
        Function LeadCoApp_Delete(ByVal LeadToken As LeadTokenDataObject, ByVal CoAppID As Integer) As
WSGenericResult
```

## Lead_ccProgram_Get (REST Available)

Returns the before and after program comparison for a specified Lead. A valid LeadTokenDataObject is required. If either the before or after forecast will not payoff all debts then one of the following message will appear in the ResponseErrorDescription and no program comparison info will be returned

- Before program will never payoff
- Proposed program will never payoff
- None of the programs will ever payoff
- Unexpected exception has occurred

```
<OperationContract()>
Function Lead_ccProgram_Get(ByVal LeadToken As LeadTokenDataObject) As Lead_ccProgram_Get_Result
```

### *Lead_ccProgram_Get_Result*

| FieldName | DataType |
|---|---|
| Before_MonthlyPaymentTotal | Before DMP program Monthly Payment Total |
| Before_NumberOfMonths | Before DMP program - # of months to payoff |
| Before_TotalAmountPaid | Before DMP program - Total amount paid to debts to payoff |
| Program_MonthlyPaymentTotal | On DMP program Monthly Payment Total |
| Program_NumberOfMonths | On DMP program - # of months to payoff |
| Program_TotalAmountPaid | On DMP program - Total amount paid to debts to payoff |

## LeadDMPReason_Get (REST Available)

Returns the list of selected DMP Reasons for a Lead.. A valid LeadTokenDataObject is required for this call.

```
<OperationContract()>
Function LeadDMPReason_Get(ByVal LeadToken As LeadTokenDataObject) As
LeadClientDMPReasonDataObject()
```

## LeadDMPReason_Set (REST Available)

Updates a collection of DMP reasons for a Lead using the following LeadClientDMPReasonDataObject. A valid LeadTokenDataObject is required.

```
<OperationContract()>
Function LeadDMPReason_Set(ByVal LeadToken As LeadTokenDataObject, ByVal LeadDMPReasons As
LeadClientDMPReasonDataObject()) As LeadClientDMPReasonDataObject()
```

## LeadDMPReason_Delete

Deletes the DMP Reason specified. LeadClientDMPReasonID and LeadToken are required.

```
<OperationContract()>
Function LeadDMPReason_Delete(ByVal LeadToken As LeadTokenDataObject, ByVal
LeadClientDMPReasonID As Integer) As WSGenericResult
```

### *LeadClientDMPReasonDataObject*

| FieldName | DataType |
|---|---|
| DMPReason | From DMPReasons Catalog |

## GetLeadWebSignInLink

Returns the WSGenericResult dataobject which returns the LeadID as ID and the URL String as the Value. A valid UserDataObject is required which is the Lead's Username and Password. This would be only used by those who have their own landing marketing pages and then submit the lead and then want the Lead to continue within the Creditsoft Web Interface. The password expiration is reset to 3 days.

**** Security Warning - if using this in an email if another user receives the email other than the attended recipient then that user would now have access to their information.**

```
<OperationContract()>
Function GetLeadWebSignInLink(ByVal objUser As UserDataObject) As WSGenericResult
```

## LEAD ACH CALLS

The following ACH calls will use the maximum ACH Restriction specified in Security Groups as the rule for how close an ACH can be scheduled for

## LeadACHGroup_Get

Returns the list of Recurring ACH Setup for a Lead. A valid LeadTokenDataObject is required for this call.

```
<OperationContract()>
Function LeadACHGroup_Get(ByVal LeadToken As LeadTokenDataObject) As LeadACHGroupObject()
```

## LeadACHGroup_Set

Updates a collection of Recurring ACH Setup for a Lead using the following LeadACHGroupObject. A valid LeadTokenDataObject is required for this call.

```
<OperationContract()>
Function LeadACHGroup_Get(ByVal LeadToken As LeadTokenDataObject) As LeadACHGroupObject()
```

## LeadACHGroup_Delete

Deletes a specific Recurring ACH Setup for a Lead. A valid LeadTokenDataObject and a LeadACHGroupID is required for this call.

```
<OperationContract()>
Function LeadACHGroup_Delete(ByVal LeadToken As LeadTokenDataObject, ByVal LeadACHGroupID As
Integer) As WSGenericResult
```

### *LeadACHGroupObject*

| FieldName | DataType |
|---|---|
| LeadACHGroupID | Int - Unique ID (system generated) |

| | |
|---|---|
| ACHGroupID | nvarchar - From ACHGroups Table |
| Amount | money |
| ACHBankAccountID | Int - Identifies the bank account prevously added |
| StartDate | Datetime - date for first debit |
| ResponseErrorDescription | used to return any errors |

## LeadACHOneTime_Get

Returns the list of OneTime ACH Setup for a Lead. A valid LeadTokenDataObject is required for this call.

```
<OperationContract()>
Function LeadACHOneTime_Get(ByVal LeadToken As LeadTokenDataObject) As LeadACHOneTimeObject()
```

## LeadACHOneTime_Set

Updates a collection of OneTime ACH Setup for a Lead using the following LeadACHOneTimeObject. A valid LeadTokenDataObject is required for this call.

```
<OperationContract()>
Function LeadACHOneTime_Set(ByVal LeadToken As LeadTokenDataObject, ByVal LeadACHOneTime As
LeadACHOneTimeObject) As LeadACHOneTimeObject
```

## LeadACHOneTime_Delete

Deletes a specific OneTime ACH Setup for a Lead. A valid LeadTokenDataObject and a LeadACHOneTimeID is required for this call.

```
<OperationContract()>
Function LeadACHOneTime_Delete(ByVal LeadToken As LeadTokenDataObject, ByVal LeadACHOneTimeID As
Integer) As WSGenericResult
```

### *LeadACHOneTimeObject*

| **FieldName** | **DataType** |
|---|---|
| LeadACHOneTimeID | Int - Unique ID (system generated) |
| ACHGroupID | nvarchar - From ACHGroups Table |
| Amount | money |
| ACHBankAccountID | Int - Identifies the bank account prevously added |
| ScheduleDate | Datetime - date of debit |
| ResponseErrorDescription | used to return any errors |

## LeadACHScheduledTransactions_Get

Returns a list of the next 12 ACH transactions forcasted from the Recurring and OneTime Setup for a Lead. A valid LeadTokenDataObject and a LeadACHOneTimeID is required for this call.

```
<OperationContract()>
```

```
        Function LeadACHScheduledTransactions_Get(ByVal LeadToken As LeadTokenDataObject) As
LeadACHScheduledTransactionsObject()
```

### *LeadACHScheduledTransactionsObject*

| FieldName | DataType |
|---|---|
| DebitDate | Datetime |
| Amount | money |
| DebitType | nvarchar |
| EnteredBy | nvarchar |
| LeadACHOneTimeID | int |
| LeadACHGroupID | int |
| ACHBankAccountID | int |
| ResponseErrorDescription | used to return any errors |

## LeadBankAccounts_Getlist

Returns the bank accounts for Lead to be used for ACH added via AddLeadBankAccount. Requires a LeadToken and returns the BankAccountObject.

```
        <OperationContract()>
        Function LeadBankAccounts_GetList(ByVal LeadToken As LeadTokenDataObject) As BankAccountObject()
```

### *BankAccountObject*

| FieldName | DataType |
|---|---|
| ACHBankAccountID | Integer |
| BankAccountType | String |
| BankName | String |
| BankRoutingNumber | String |
| BankAccountNumber | String |
| ResponseErrorDescription | used to return any errors |

## LeadBankAccount_Delete

Will allow deleting a bank account as long as its not linked to a recurring or one time setup. Requires LeadToken and ACHBankAccountID. Retursn the WSGenericResult.

```
        <OperationContract()>
        Function LeadBankAccount_Delete(ByVal LeadToken As LeadTokenDataObject, ByVal ACHBankAccountID
As Integer) As WSGenericResult
```

## RequestCreditReportXML

Submits the credit report request, saves it to the database, and returns a WSGenericResult object with the XML as the value. A valid CBRRequestDataObject is required for this call. A valid ClientID would be used for a Client Credit Report

or LeadID for a Lead Credit Report. To also link the Coapp also pass a CoAppID. Minimum columns required are notated with an *. To make a request for a co-applicant, include the applicant as Debtor1 and the co-applicant as Debtor2. Minimum columns required when including Debtor2 are notated with **.

```
<OperationContract()>
Function RequestCreditReportXML(ByVal Request As ICCOCore.CBRRequestDataObject) As WSGenericResult
```

### *CBRRequestDataObject*

| FieldName | DataType |
|---|---|
| ClientID | int |
| LeadID | int |
| CoAppID | int |
| CreditReportAgencyCode | nvarchar(2) - only supporting "EQ" currently |
| Debtor1Alias1FirstName* | nvarchar |
| Debtor1Alias1MiddleName* | nvarchar |
| Debtor1Alias1LastName* | nvarchar |
| Debtor1Alias1NameSuffix | nvarchar |
| Debtor1SSN* | nvarchar(9) |
| Debtor1DOB | datetime |
| Debtor1Address1Current* | nvarchar - Street number |
| Debtor1Address2Current* | nvarchar - Street name |
| Debtor1Address3Current | nvarchar - Street Type |
| Debtor1CityCurrent* | nvarchar |
| Debtor1StateCurrent* | nvarchar |
| Debtor1ZipCurrent* | nvarchar |
| Debtor2Alias1FirstName** | nvarchar |
| Debtor2Alias1MiddleName** | nvarchar |
| Debtor2Alias1LastName** | nvarchar |
| Debtor2SSN** | nvarchar(9) |
| Debtor2DOB | datetime |

## RequestCreditReportXML3 (REST Available)

Uses the information in the Token to get a credit report for a debtor.

If pass Nothing to CreditReportAgencyCode it will use the preference CBRAgency, otherwise you can pass one of the two or five letter codes: TU = TransUnion, EQ= Equifax , EX=Experian, TUCAN= TransUnionCanada, EQCAN = Equifax Canada

Validates the token first. Then it validates that the debtor has all the required fields filled out in the database. Creates the CBRRequest with the debtor information from the database.

It then follows the same logic as RequestCreditReportXML in that it submits the credit report request, saves it to the database, and returns a WSGenericResult object. You can then query the CBRAccounts or any other CBR tables, or parse the XML document returned on your own.

```
<OperationContract()>
        Function RequestCreditReportXML3(Token As TokenDataObject, Optional CreditReportAgencyCode As
String) As WSGenericResult
```

In REST, you would pass the Token like this:

```
{ “Token” :
        {
          "UserName": "yvillafweb",
          "CSRecordID": 10589,
          "CSRecordType": "Lead",
          "Token":
        "A4GOC2nhCzZQa8ZEcqx7Lrc8VCBIU8BVLtAYLoXKaLYa8Ajez3PDMFYq1kPGUkvVFqUiY55b7Biu5sqP9tmgXE
        N64whsSAGbqX/wGBI/hOm00FOhhgRXwZVuA81Sk6yLOCABiQsGBJ237/isa/9M18Zvn7JUWxFuhvymyeRouadX9
        9rd7EW4VpwtAcvlMaMLpOW0LTq4EUVSTf5pr39PB2MlIn/FJKgsS7ctQsR9UtjvG+cUlH+mpi46YIBq3tS/YhBl
        QAb1QM0fYAi+n3g=="
        }
}
```

## LeadNONDMP_Get

Returns the list of NonDMP Accounts for a Lead. LeadToken is required.

```
        <OperationContract()>
        Function LeadNONDMP_Get(ByVal LeadToken As LeadTokenDataObject) As LeadNonDMPAccountDataObject()
```

## LeadNONDMP_Set

Adds a NonDMP Account for a Lead. LeadToken is required

```
        <OperationContract()>
        Function LeadNONDMP_Set(ByVal LeadToken As LeadTokenDataObject, ByVal LeadNonDMPAccounts As
LeadNonDMPAccountDataObject()) As LeadNonDMPAccountDataObject()
```

## LeadNONDMP_Delete

Deletes a NonDMP Account for a Lead. LeadToken and NDMPClientCredID is required.

```
        <OperationContract()>
        Function LeadNONDMP_Delete(ByVal LeadToken As LeadTokenDataObject, ByVal NDMPClientCredID As
Integer) As WSGenericResult
```

### *LeadNonDMPAccountDataObject*

| FieldName | DataType |
|---|---|
| NDMPClientCredID | Int - Unique ID (system generated) |
| CreditorID | Int - must be in Creditors Table |
| CreditorAlias | nvarchar(250) - must be in CreditorAlias Table and match the CreditorID |

| | |
|---|---|
| AccountNumber | nvarchar(50) |
| OrigDebt | money |
| OrigMonthly | money |
| OrigAPR | decimal(11,9) |
| Comments | nvarchar(255) |
| ResponseErrorDescription | used to return any errors |
| AccountRemoveReasonID | int - must be in RemoveReason Table |
| UserDefined1 | nvarchar(50) |
| UserDefined2 | nvarchar(50) |
| UserDefined3 | nvarchar(50) |

## LeadMoveAccountDMPorNonDMP

Moves a DMP account to NonDMP or NonDMP to DMP Account. Requires LeadToken and either ClientCredID or NDMPClientCredID. If ClientCredID is used then the AccountRemoveReasonID is required. That can be found in AccountRemoveReasons

```
        <OperationContract()>
        Function LeadMoveAccountDMPorNonDMP(ByVal LeadToken As LeadTokenDataObject, Optional ByVal
ClientCredID As Nullable(Of Integer) = Nothing, Optional ByVal NDMPClientCredID As Nullable(Of Integer)
= Nothing, Optional ByVal AccountRemoveReasonID As Nullable(Of Integer) = Nothing) As WSResponse
```

## LeadDMPProgramInfoGet

Returns the Monthly and Initial Fee Information for the Lead.. LeadToken is required

```
        <OperationContract()>
        Function LeadDMPProgramInfoGet(ByVal LeadToken As LeadTokenDataObject) As LeadDMPProgramInfoGet
```

### *LeadDMPProgramInfoGet*

| FieldName | DataType |
|---|---|
| ResponseErrorDescription | used to return any errors |
| InitialContributionBalance | money |
| InitialContributionPayment | money |
| MonthlyContributionPayment | money |
| FirstPaymentAmount | money - includes Creditor Payments, Monthly and Initial Fee |
| RecurringPaymentAmount | money - includes Creditor Payments and Monthly Fee |

## LeadClientAccomplish_Get

Returns the list of Goals that the Lead has given. LeadToken is required

```
  <OperationContract()>
  Function LeadClientAccomplish_Get(ByVal LeadToken As LeadTokenDataObject) As AccomplishObject()
```

## LeadClientAccomplish_Set

Set the list of Goals for the Lead. LeadToken is required.

```
  <OperationContract()>
  Function LeadClientAccomplish_Set(ByVal LeadToken As LeadTokenDataObject, ByVal LeadClientAccomplish
As AccomplishObject()) As AccomplishObject()
```

## LeadClientAccomplish_Delete

Deletes the Accomplish specified. LeadToken and DebtorAccomplishID are required.

```
  <OperationContract()>
  Function LeadClientAccomplish_Delete(ByVal LeadToken As LeadTokenDataObject, ByVal DebtorAccomplishID
As Integer) As WSGenericResult
```

### *AccomplishObject*

| FieldName | DataType |
|---|---|
| DebtorAccomplishID | Int - Unique ID (system generated) |
| Accomplish | nvarchar(50) - must be in Accomplish Table |
| ResponseErrorDescription | used to return any errors |

## LeadClientActionPlan_Set

Ability to set the Action Plan Issue and Goal Notes. LeadToken is required

## LeadClientActionPlan_Get

Ability to return the Action Plan Issue and Goal Notes. LeadToken is required

### *LeadClientActionPlanObject*

| FieldName | DataType |
|---|---|
| ClientIssueNotes | nvarchar(1000) |
| ClientGoalNotes | nvarchar(1000) |
| ResponseErrorDescription | used to return any errors |

**LeadCoAppDataObject_Get**

Ability to return all the Coapps linked to a Lead. LeadToken is required

## LeadCoAppDataObject_Set

Ability to add a CoApp to a Lead. LeadToken is required.

### *CoAppsDataObject*

| FieldName | DataType |
|---|---|
| CoAppID | Int - Unique ID (system generated) |
| CoAppFirst | nvarchar(50) - required |
| CoAppLast | nvarchar(50) - required |
| CoAppDOB | smalldatetime |
| CoAppSSN | nvarchar(14) |
| CoAppEmail | nvarchar(50) |
| IsSpouse | True/False |
| Relationship | nvarchar(50) - Must be in RelationshipTypes Table |
| CoGender | M/F |
| CoAppEmployed | True/False |
| CoAppEmployer | nvarchar(50) |
| CoAppHomePhone | nvarchar(50) |
| CoAppWorkPhone | nvarchar(50) |
| CoAppOtherPhone | nvarchar(25) |
| ResponseErrorDescription | used to return any errors |
| CoAppOccupationCategoryID | integer |
| CoAppActive | True/False - can only be set with the call LeadChangeCoAppActive |
| IsDefault | True/False |
| CoAppAddress1 | nvarchar(100) |
| CoAppAddress2 | nvarchar(100) |
| CoAppCity | nvarchar(50) |
| CoAppState | nvarchar(30) |
| CoAppZip | nvarchar(15) |
| CoAppEmploymentStatus | Nvarchar (50) – must be a choice in the NFCC_EmploymentStatus table |

### LeadChangeCoAppActive

Requires LeadToken, CoAppID, and NewActiveStatus (True/False). Returns WSGenericResult. A CoApp will not be able to be marked Inactive if they are linked to Accounts or Bank Accounts.

# Budget Scenarios

### ScenarioID parameter

When retrieving budget information, you can use the ScenarioID to pick a specific financial scenario. You can also use the following negative values to pick a particular type of scenario:

```
 0 = Look for Original, and if one doesn't exist, create one
-1 = Look for Actual
-2 = Look for Proposed
```

## AddLeadFinancialScenario

Add the Proposed or Original Scenario for the Budget. This will return the ScenarioID to be used in subsequent calls. LeadToken is required.

```
  <OperationContract()>
  Function AddLeadFinancialScenario(ByVal LeadToken As LeadTokenDataObject, ByVal
FinancialScenarioObject As FinancialScenarioObject) As WSResponse
```

### *FinancialScenarioObject*

| FieldName | DataType |
|---|---|
| ScenarioID | Int - Unique ID (system generated) |
| ScenarioName | nvarchar(255) |
| OriginalBudget | True/False |
| ProposedBudget | True/False |
| ResponseErrorDescription | used to return any errors |

## LeadIncomeScenario_Get

Returns Income details for a specific Financial Scenario. LeadToken, ScenarioID are required

```
       <OperationContract()>
       Function LeadIncomeScenario_Get(ByVal LeadToken As LeadTokenDataObject, ByVal ScenarioID As
Integer) As LeadIncomeDataObject()
```

## LeadIncomeScenario_Set

Add Income details for a specific Financial Scenario. LeadToken, ScenarioID are required and would be sent with a LeadIncomeDataObject

```
       <OperationContract()>
       Function LeadIncomeScenario_Set(ByVal LeadToken As LeadTokenDataObject, ByVal ScenarioID As
Integer, ByVal LeadIncomes As LeadIncomeDataObject()) As LeadIncomeDataObject()
```

## LeadBudgetScenario_Get

Returns Budget details for a specific Financial Scenario. LeadToken, ScenarioID are required

```
       <OperationContract()>
       Function LeadBudgetScenario_Get(ByVal LeadToken As LeadTokenDataObject, ByVal ScenarioID As
Integer) As LeadBudgetDataObject()
```

## LeadBudgetScenario_Set

Add Budget details for a specific Financial Scenario. LeadToken, ScenarioID are required and would be sent with a LeadBudgetDataObject

```
       <OperationContract()>
       Function LeadBudgetScenario_Set(ByVal LeadToken As LeadTokenDataObject, ByVal ScenarioID As
Integer, ByVal LeadBudget As LeadBudgetDataObject()) As LeadBudgetDataObject()
```

### LeadAssetsScenario_Get

Returns Asset details for a specific Financial Scenario. LeadToken, ScenarioID are required

```
        <OperationContract()>
        Function LeadAssetsScenario_Get(ByVal LeadToken As LeadTokenDataObject, ByVal ScenarioID As
Integer) As LeadClientAssetsDataObject()
```

### LeadAssetsScenario_Set

Add Asset details for a specific Financial Scenario. LeadToken, ScenarioID are required and would be sent with a LeadClientAssetsDataObject

```
        <OperationContract()>
        Function LeadAssetsScenario_Set(ByVal LeadToken As LeadTokenDataObject, ByVal ScenarioID As
Integer, ByVal LeadAssets As LeadClientAssetsDataObject()) As LeadClientAssetsDataObject()
```

### LeadLiabilitiesScenario_Get

Returns Liability details for a specific Financial Scenario. LeadToken, ScenarioID are required

```
        <OperationContract()>
        Function LeadLiabilitiesScenario_Get(ByVal LeadToken As LeadTokenDataObject, ByVal ScenarioID As
Integer) As LeadClientLiabilitiesDataObject()
```

### LeadLiabilitiesScenario_Set

Add Liability details for a specific Financial Scenario. LeadToken, ScenarioID are required and would be sent with a LeadClientLiabilitiesDataObject.

```
        <OperationContract()>
        Function LeadLiabilitiesScenario_Set(ByVal LeadToken As LeadTokenDataObject, ByVal ScenarioID As
Integer, ByVal LeadAssets As LeadClientLiabilitiesDataObject()) As LeadClientLiabilitiesDataObject()
```

### MarkBudgetScenarioAsActual

Will mark the specified ScenarioID as actual. This should be the last call when doing any Financial calls. It requires LeadToken and ScenarioID and WSReponse is returned.

```
        <OperationContract()>
        Function MarkBudgetScenarioAsActual(ByVal Leadtoken As LeadTokenDataObject, ByVal ScenarioID As
Integer) As WSResponse
```

## Budget Calls

### Income_Get (REST Only)

```
    <OperationContract()>
    Public Function Income_Get(ByVal Token As TokenDataObject, Optional ScenarioID As Integer = 0) As
String
```

Returns: Array of IncomeDataObject

### Expenses_Get (REST Only)

```
  <OperationContract()>
  Public Function Expenses_Get(ByVal Token As TokenDataObject, Optional ScenarioID As Integer = 0) As
String
```

Returns: Array of BudgetDataObject

### Assets_Get (REST Only)

```
  <OperationContract()>
  Public Function Assets_Get(ByVal Token As TokenDataObject, Optional ScenarioID As Integer = 0) As
String
```

Returns: Array of AssetsDataObject

### Liabilities_Get (REST Only)

```
  <OperationContract()>
  Public Function Liabilities_Get(ByVal Token As TokenDataObject, Optional ScenarioID As Integer = 0)
As String
```

Returns: Array of LiabilitiesDataObject

### FixedSpending_Get (REST Only)

```
  <OperationContract()>
  Public Function FixedSpending_Get(ByVal Token As TokenDataObject, Optional ScenarioID As Integer = 0)
As String
```

Returns: Array of BudgetDataObject

### DiscretionarySpending_Get (REST Only)

```
  <OperationContract()>
  Public Function DiscretionarySpending_Get(ByVal Token As TokenDataObject, Optional ScenarioID As
Integer = 0) As String
```

Returns: Array of BudgetDataObject

### LeadIncome_Set

Updates a collection of income entries in the Budget using the following LeadIncomeDataObject. A valid LeadTokenDataObject is required.

```
      <OperationContract()>
      Function LeadIncome_Set(ByVal LeadToken As LeadTokenDataObject, ByVal LeadIncomes As
LeadIncomeDataObject()) As LeadIncomeDataObject()
```

### LeadIncome_Get

Returns the data stored for the following dataobject. A valid LeadTokenDataObject is required for this call.

```
      <OperationContract()>
      Function LeadIncome_Get(ByVal LeadToken As LeadTokenDataObject) As LeadIncomeDataObject()
```

### *LeadIncomeDataObject*

| FieldName | DataType |
|---|---|
| IncomeName | From IncomeCategories Catalog |
| IncomeValue | money - CreditSoft Calculated - do not pass |
| Frequency | nvarchar(50) - From BudgetFrequencies Catalog |
| FrequencyAmount | money |
| Comments | nvarchar(255) |

## LeadAssets_Set (REST Available)

Updates a collection of asset entries in the Budget using the following LeadClientAssetsDataObject. A valid LeadTokenDataObject is required.

```
	<OperationContract()>
	Function LeadAssets_Set(ByVal LeadToken As LeadTokenDataObject, ByVal LeadAssets As
LeadClientAssetsDataObject()) As LeadClientAssetsDataObject()
```

## LeadAssets_Get (REST Available)

Returns the data stored for the following dataobject. A valid LeadTokenDataObject is required for this call.

```
	<OperationContract()>
	Function LeadAssets_Get(ByVal LeadToken As LeadTokenDataObject) As LeadClientAssetsDataObject()
```

### *LeadClientAssetsDataObject*

| FieldName | DataType |
|---|---|
| AssetName | From AssetCategories Catalog |
| AssetValue | money |

## LeadLiabilities_Set (REST Available)

Updates a collection of liabilities entries in the Budget using the following LeadClientLiabilitiesDataObject. A valid LeadTokenDataObject is required.

```
	<OperationContract()>
	Function LeadLiabilities_Set(ByVal LeadToken As LeadTokenDataObject, ByVal LeadLiabilitiesList
As LeadClientLiabilitiesDataObject()) As LeadClientLiabilitiesDataObject()
```

## LeadLiabilities_Get (REST Available)

Returns the data stored for the following dataobject. A valid LeadTokenDataObject is required for this call.

```
	<OperationContract()>
	Function LeadLiabilities_Get(ByVal LeadToken As LeadTokenDataObject) As
LeadClientLiabilitiesDataObject()
```

### *LeadClientLiabilitiesDataObject*

| FieldName | DataType |
|---|---|
| LiabilityName | From LiabilityCategories Catalog |
| LiabilityValue | money |

### ExpenseCategories_Set

Updates a collection of expense entries in the Budget using the following LeadBudgetDataObject. A valid LeadTokenDataObject is required.

```
        <OperationContract()>
        Function ExpenseCategories_Set(ByVal LeadToken As LeadTokenDataObject, ByVal LeadBudget As
LeadBudgetDataObject()) As LeadBudgetDataObject()
```

### FixedSpending_Get

Returns the data stored for the following dataobject for all categories marked as Fixed Spending. A valid LeadTokenDataObject is required for this call.

```
        <OperationContract()>
        Function FixedSpending_Get(ByVal LeadToken As LeadTokenDataObject) As LeadBudgetDataObject()
```

### DiscretionarySpending_Get

Returns the data stored for the following dataobject for all categories marked as Discretionary Spending. A valid LeadTokenDataObject is required for this call.

```
        <OperationContract()>
        Function DiscretionarySpending_Get(ByVal LeadToken As LeadTokenDataObject) As
LeadBudgetDataObject()
```

#### *LeadBudgetDataObject*

| FieldName | DataType |
|---|---|
| Category | From FixedSpendingCategories or DiscretionarySpendingCategories Catalog |
| Subcategory | From FixedSpendingCategories or DiscretionarySpendingCategories Catalog |
| BudgetAmount | money - CreditSoft Calculated - do not pass |
| Frequency | nvarchar(50) - From BudgetFrequencies Catalog |
| FrequencyAmount | money |
| Comments | nvarchar(255) |

# Other Lead Calls

## LeadSetHardshipLevel

Specifies the CSP Hardship Level for the Lead. This call should be done after all the accounts are added. This call only works with CSP configured on the database and will update the account payments and APR to the appropriate CSP Tier 1,2,3 or Call to Action/Hardship based on the creditor configuration. LeadToken and HardshipLevel (F,G, H) are required.

```
        <OperationContract()>
        Function LeadSetHardshipLevel(ByVal LeadToken As LeadTokenDataObject, ByVal HardshipLevel As
String) As WSResponse
```

## LeadSetSSN

Requires LeadToken, SSN, with an optional CoAPPID. Returns the WSResponse Object. This call confirms that the SSN doesn't already exist for a Lead, Client, or CoAPP before updating the SSN.

```
        <OperationContract()>
        Function LeadSetSSN(ByVal LeadToken As LeadTokenDataObject, ByVal SSN As String, Optional ByVal
CoAppID As Nullable(Of Integer) = Nothing) As WSResponse
```

## AddLeadNote

Requires LeadToken and Notes object. Returns WsResponse with the RecordID

```
        <OperationContract()>
        Function AddLeadNote(ByVal LeadToken As LeadTokenDataObject, ByVal NoteInfo As Notes) As
WSResponse
```

### *NotesObject*

| FieldName | DataType |
|---|---|
| UserID | nvarchar(20) |
| NoteDate | Datetime |
| Note | nvarchar(max) |

## ConvertLeadToClient (REST Available)

Allows the Lead to be converted to a Client as long as the Lead Conversion Requirement is met. Requires LeadToken and returns WSResponse.

```
        <OperationContract()>
        Function ConvertLeadToClient(ByVal LeadToken As LeadTokenDataObject) As WSResponse
```

## ConvertLeadtoClientValidation (REST Available)

Allows to check if the lead can be converted by checking the Lead Conversion Requirement. Requires LeadToken and returns WSGenericResult

```
        <OperationContract()>
        Function ConvertLeadToClientValidation(ByVal LeadToken As LeadTokenDataObject) As
WSGenericResult
```

## LeadWaiveOrLowerFee

Requires a LeadToken and a OverrideFeeObject (all fields are required). Returns WSReponse with the ClientCredID for the unique Fee as the RecordID when successful.

```
	<OperationContract()>
	Function LeadWaiveOrLowerFee(ByVal LeadToken As LeadTokenDataObject, ByVal FeeObject As
OverrideFeeObject) As WSResponse
```

### *OverrideFeeObject*

| FieldName | DataType |
|---|---|
| FeeCreditorID | Int - Default Creditor Fee CreditorID |
| NewFeeAmount | money |
| ChangeFeeReasonID | Int - See ChangeFeeReason Table |
| AccountStatus | nvarchar(20) - See AccountStatus Table |
| FeeChangeDate | Datetime |
| FeeChangeUser | nvarchar(20) |
| ResponseErrorDescription | used to return any errors |

## ChangeLeadStatus

Requires a LeadToken, NewStatus, Note, and UserID. Returns WSGenericResult. NewStatus must be a valid LeadStatus but not CONVERTED. UserID must be a valid User in Settings.

```
<OperationContract()>
	Function ChangeLeadStatus(ByVal LeadToken As LeadTokenDataObject, ByVal NewStatus As String,
ByVal Note As String, ByVal UserID As String) As WSGenericResult
```

## GetDebtorHardshipScenarios

This can be used for Leads or Clients and requires a Token (note NOT a LeadToken) and will return an array of AccountHardshipObject

```
<OperationContract()>
	Function GetDebtorHardshipScenarios(ByVal Token As TokenDataObject) As AccountHardshipObject()
```

### *AccountHardshipObject*

| FieldName | DataType |
|---|---|
| DebtorID | Int |
| DebtorType | Lead or Client |
| LeadClientCredID | Int – If returning Lead Account |
| ClientCredID | Int – If returning Client Account |

| | |
|---|---|
| Balance | Decimal |
| IsChargedOff | Boolean |
| StandardAmount | Decimal - DMP Payment |
| StandardAPR | Decimal – DMP APR |
| Tier1Amount | Decimal - Tier 1 Payment |
| Tier1APR | Decimal – Tier 1 APR |
| Tier1Allowed | Boolean – whether or not the creditor allows Tier1 |
| Tier2Amount | Decimal - Tier 2 Payment |
| Tier2APR | Decimal – Tier 2 APR |
| Tier2Allowed | Boolean – whether or not the creditor allows CTA or CSP Tier 2 |
| Tier3Amount | Decimal - Tier 3 Payment |
| Tier3APR | Decimal – Tier 3 APR |
| Tier3Allowed | Boolean – whether or not the creditor allows CTH or CSP Tier 3 |
| LTFBAAmount | Decimal - LTFB A Payment |
| LTFBANewBalance | Decimal - LTFB A Reduced Balance |
| LTFBAAllowed | Boolean – whether or not the creditor allows LTFB A |
| LTFBCAmount | Decimal - LTFB C Payment |
| LTFBCNewBalance | Decimal - LTFB C Reduced Balance |
| LTFBCAllowed | Boolean – whether or not the creditor allows LTFB C |
| ExtendedTermAmount | Decimal - Extended Payment |
| ExtendedTermAPR | Decimal – Extended APR |
| ExtendedTermAllowed | Boolean – whether or not the creditor allows Extended |
| ResponseErrorDescription | used to return any errors |

# Finding Creditors

## FindCreditors (REST Available)

Returns a list of Creditors that match the account number and creditor name passed. Columns returned: CreditorID, CreditorAlias, MinimumPercent, PercentOf, DefaultSettlementPercentage

```
        <OperationContract()>
        Function FindCreditors(ByVal AccountNumber As String,
         ByVal CreditorName As String) As WSCreditorResult()
```

| FieldName | DataType |
|---|---|
| AccountNumber | nvarchar(50) |
| CreditorName | nvarchar(50) |

***OUTPUT: Array of WSCreditorResult***

***WSCreditorResult***

```
<DataContract()> _
Public Class WSCreditorResult
        <DataMember()> _
        Public CreditorID As Integer = -1

        <DataMember()> _
        Public CreditorAlias As String = String.Empty

        <DataMember()> _
        Public MinimumPercent As Nullable(Of Decimal)

        <DataMember()> _
        Public PercentOf As Integer

        <DataMember()> _
        Public DefaultSettlementPercentage As Nullable(Of Decimal)

End Class
```

## FindAccountConcessions

Returns the first CreditorID that match the account number and creditor name passed, if no Creditor matches it will pass the Generic CreditorID specified in the Administration Console.

```
        <OperationContract()>
        Function FindAccountConcessions(ByVal AccountNumber As String,
         ByVal CreditorName As String, ByVal Balance As Decimal) As WSAccountConcessionsResult
```

Columns returned: CreditorID, MonthlyPayment, Reduced APR, CreditorAlias

| FieldName | DataType |
|---|---|
| AccountNumber | nvarchar(50) |

| | |
|---|---|
| CreditorName | nvarchar(50) |
| Balance | money |

***INPUT: AccountNumber, CreditorName, Balance***

***OUTPUT: WSAccountConcessionsResult***

***WSAccountConcessionsResult***

```
<DataContract()> _
Public Class WSAccountConcessionsResult
	<DataMember()> _
	Public CreditorID As Integer = -1

	<DataMember()> _
	Public CalculatedPayment As Nullable(Of Decimal)

	<DataMember()> _
	Public ReducedAPR As Nullable(Of Decimal)

	<DataMember()> Public ErrorDescription As String = String.Empty

	<DataMember()>
	Public CreditorAlias As String = String.Empty

	Public Sub New()

	End Sub
End Class
```

## AddCreditor

```
	<OperationContract()>
	Function AddCreditor(ByVal WebUserInfo As UserDataObject, ByVal CreditorInfo As CreditorObject)
As WSResponse
```

INPUT: WebUserInfo as UserDataObject, CreditorInfo as CreditorObject

OUTPUT: WSResponse with the new CreditorID as the RecordID

***CreditorObject***

| **FieldName** | **DataType** |
|---|---|
| Name | nvarchar(250) |
| Address1 | nvarchar(255) |
| Address2 | nvarchar(100) |
| City | nvarchar(50) |
| State | nvarchar(30) |
| Zip | nvarchar(15) |
| Phone | nvarchar(25) |
| CreditorType | nvarchar(20) - See CreditorTypes Table |

# Lead Web Services Catalogs

In order to populate some fields within the Lead Web Service a developer must map their configuration choices with the configuration options from Settings. The following calls can be used to return the list of configuration settings.

The following calls will return the WSGenericResult dataobject which returns the columns ID and Value. The ID must be used when submitting information in web service calls.

- DebtorBillStatus_GetList
- DMPReasons_GetList
- DebtTypes_GetList
- DelinquencyStatus_GetList
- IncomeCategories_GetList - categories marked Show for Enrollment
- AssetCategories_GetList - categories marked Show for Enrollment
- BudgetFrequencies_GetList - categories marked Show for Enrollment
- LiabilityCategories_GetList - categories marked Show for Enrollment
- DMPDebtRange_GetList
- CreditorsForEnrollment_GetList - external creditors marked Use For Enrollment, Active, Use for Internet - limited to 15000 creditors
- AccomplishCategories_GetList

The following calls will return the CategoriesDataObject dataobject which returns the columns Category, SubCategory, and SubCategoryDescription. The Category and SubCategory must be used when submitting information in web service calls.

- FixedSpendingCategories_GetList - categories marked Show for Enrollment and Category Type = 'F'
- DiscretionarySpendingCategories_GetList - - categories marked Show for Enrollment and Category Type = 'D'

# Client Calls

## FindClients (REST Available)

Returns a list of clients for the specified Sales Agent. The SalesForceID is required by default per database. To make the SalesForceID not required please contact Training.

```
  <OperationContract()>
       Function FindClients(ByVal SSN As String, ByVal SalesForceID As Nullable(Of Integer)) As
WSClientSearchResult()
```

| Input Parameters | DataType |
|---|---|
| SSN | nvarchar(14) |
| SalesForceID | integer - From SalesForce Table |
| | |
| **Columns in Result List** | **DataType** |
| ClientID | int |
| FirstName | nvarchar(50) |
| LastName | nvarchar(50) |

### *OUTPUT: Array of WSClientStatusResult*

### *WSClientStatusResult*

```
<DataContract()> _
Public Class WSClientStatusResult
       <DataMember()> _
       Public ClientID As Integer = -1

       <DataMember()> _
       Public ClientStatus As String = String.Empty

       <DataMember()> _
       Public CloseReason As String = String.Empty

End Class
```

## FindClientStatus (REST Available)

Returns a list of clients and their current status for the specified Sales Agent. The SalesForceID is required by default per database. To make the SalesForceID not required please contact Training.

```
       <OperationContract()>
       Function FindClientStatus(ByVal ClientID As Nullable(Of Integer), ByVal SalesForceID As
Nullable(Of Integer)) As WSClientStatusResult()
```

| Input Parameters | DataType |
|---|---|
| ClientID | int |
| SalesForceID | integer - From SalesForce Table |

| | |
|---|---|
| **Columns in Result List** | **DataType** |
| ClientID | int |
| ClientStatus | nvarchar(20) |
| CloseReason | nvarchar(5) |

### *OUTPUT: Array of WSClientStatusResult*

### *WSClientStatusResult*

```
<DataContract()> _
Public Class WSClientStatusResult
	<DataMember()> _
	Public ClientID As Integer = -1

	<DataMember()> _
	Public ClientStatus As String = String.Empty

	<DataMember()> _
	Public CloseReason As String = String.Empty

	Public Sub New()

	End Sub
End Class
```

## FindReceiptHistory

Returns a list of receipts for a client for the date range specified. Leave ClientID as nothing to return all clients for the specified Sales Agent. The SalesForceID is required by default per database. To make the SalesForceID not required please contact Training.

```
	<OperationContract()>
	Function FindReceiptHistory(ByVal ClientID As Nullable(Of Integer),

	ByVal SalesForceID As Nullable(Of Integer),

	ByVal StartDate As Nullable(Of Date),

	ByVal EndDate As Nullable(Of Date)) As WSClientReceiptHistoryResult()
```

### *INPUT:*

| | |
|---|---|
| **Input Parameters** | **DataType** |
| ClientID | int |
| SalesForceID | integer - From SalesForce Table |
| StartDate | DateTime (nullable) |
| EndDate | DateTime (nullable) |
| | |
| **Columns in Result List** | **DataType** |
| ClientID | int |
| DateReceived | Datetime |

| | |
|---|---|
| ReceiptAmount | Currency |
| PaymentType | nvarchar(10) |
| NSF | True/False |

### *OUTPUT: Array of WSClientReceiptHistoryResult*

### *WSClientReceiptHistoryResult*

```
<DataContract()> _
Public Class WSClientReceiptHistoryResult
	<DataMember()> _
	Public ClientID As Integer = -1

	<DataMember()> _
	Public DateReceived As Nullable(Of Date)

	<DataMember()> _
	Public ReceiptAmount As Nullable(Of Decimal)

	<DataMember()> _
	Public PaymentType As String = String.Empty

	<DataMember()> _
	Public NSF As Nullable(Of Boolean)

	Public Sub New()

	End Sub
End Class
```

## AttachDocumentToClient (REST Available)

Adds a Task Note to a Client and attaches the Document specified.

```
<OperationContract()>
	Public Function AttachDocumentToClient(ByVal ClientID As Integer, ByVal DocumentName As String,
	ByVal Document As Byte(), Optional DocumentType As String = Nothing) As String
```

| FieldName | DataType |
|---|---|
| ClientID | int - must exist in database |
| DocumentName | String Name of Actual file including the extension |
| Document | String This will be a byte array (Byte()) that will contain the contents of the file to be attached to the Task. |
| DocumentType | String Refers to the DocumentTypes enum – can be null |

### AddClientIssue (REST Available)

Creates a Task for the Client for the specified category. If no categoryid is passed the Default Category will be used. The User and Department Assignment will follow the Task Category Setup.

```
	<OperationContract()>
	Function AddClientIssue(ByVal ClientID As Integer, ByVal IssueSummary As String, ByVal
IssueCategoryID As Nullable(Of Integer), Optional ByVal DateCreated As Nullable(Of Date) = Nothing,
Optional ByVal Status As String = Nothing) As WSResponse
```

| FieldName | DataType |
|---|---|
| ClientID | int - must exist in database |
| IssueSummary | nvarchar(2000) |
| IssueCategoryID | int - from IssueCategories Table |
| DateCreated | datetime |
| Status | nvarchar(20) - must exist in the database - Pass nothing to use the Default Task Status |

# User Related Calls

### ValidateDebtorUserLogin

```
	<OperationContract()>
	Function ValidateDebtorUserLogin(ByVal UserInfo As UserDataObject) As UserValidationObject
```

INPUT: UserInfo as UserDataObject

OUTPUT: UserValidationObject

| FieldName | DataType |
|---|---|
| DebtorID | Integer of found Lead or ClientID |
| DebtorType | string - Lead or Client |
| Status | string (Invalid, Reset, Locked, Closed, Valid) |
| ResponseErrorDescription | string |
| Email | string - Lead or Client email address |

#### *UserValidationObject*

```
<DataContract()>
Public Class UserValidationObject
	<DataMember()> Public DebtorID As Integer
	<DataMember()> Public DebtorType As String
	<DataMember()> Public Status As String
	<DataMember()> Public ResponseErrorDescription As String
```

```
        <DataMember()> Public Email As String

        Public Sub New()
        End Sub
End Class
```

## DebtorChangePassword

This will allow for changing the password as long as you have a valid Username and Password.

```
        <OperationContract()>
        Function DebtorChangePassword(ByVal UserInfo As UserDataObject, ByVal NewPassword As String) As
WSGenericResult
```

INPUT: UserInfo as UserDataObject, NewPassword

OUTPUT: WSGenericResult.

## DebtorResetPassword

```
        <OperationContract()>
        Function DebtorResetPassword(ByVal WebUserInfo As UserDataObject, ByVal DebtorUserInfo As
UserDataObject) As WSGenericResult
```

INPUT: Web User UserDataObject, Debtor User UserDataObject

OUTPUT: WSGenericResult

The Web User would be a user login created in Settings > Users. This is the same as the user that companies use for generating a Token for a Lead. The Debtor User UserDataObject would then contain the username and the new password for the Lead/Client.

## DebtorRetrieveWebByPassLogin

```
        <OperationContract()>
        Function DebtorRetrieveWebByPassLogin(ByVal WebUserInfo As UserDataObject, ByVal IPAddress As
String) As WSGenericResult
```

INPUT: Debtor User UserDataObject,,IPAddress as String

OUTPUT: WSGenericResult with the URL that could be used to redirect from one site to another.

# Document Info Related Calls

## GenerateDocumentSetandSendViaSignatureProvider (REST Available)

Processes a document set and sends it to the Signature Provider specified in Setup. Only SQL Report or Express Word documents that generate as PDF can be used with this call.

```
	<OperationContract()>
	Function GenerateDocumentSetandSendViaSignatureProvider(ByVal LeadToken As LeadTokenDataObject,
ByVal DocumentSetID As Integer) As WSGenericResult
```

INPUT: LeadToken as LeadTokenDataObject, DocumentSetID

OUTPUT: WSGenericResult

## GenerateDocumentSetandSendViaSignatureProvider2 (REST Available)

Processes a document set and sends it to the Signature Provider specified in Setup. Only SQL Report or Express Word documents that generate as PDF can be used with this call.

```
	<OperationContract()>
	Function GenerateDocumentSetandSendViaSignatureProvider2(ByVal Token As TokenDataObject, ByVal
DocumentSetID As Integer) As WSGenericResult
```

INPUT: Token as TokenDataObject, DocumentSetID

OUTPUT: WSGenericResult

## GenerateDocumentSetandSend (REST Available)

Processes a document set and sends it to the lead or client. The document set must have a Default Send Method set to Email or SMS. If using Email, the lead or client must have an email address. If using SMS, the lead or client must have a default SMS phone number that is SMS Enabled. We use the scalar function GetDebtorDefaultSMSPhone to retrieve it.

```
	<OperationContract()>
	Function GenerateDocumentSetandSend(ByVal Token As TokenDataObject, ByVal DocumentSetID As
Integer) As WSGenericResult
```

INPUT: Token as TokenDataObject, DocumentSetID

OUTPUT: WSGenericResult

## GenerateDocumentSetOnly (REST Available)

Processes a document set and generates and stores the documents.  Only SQL Report documents that generate as PDF can be used with this call.

```
	<OperationContract()>
	Function GenerateDocumentSetOnly(ByVal LeadToken As LeadTokenDataObject, ByVal DocumentSetID As
Integer) As DocumentObject()
```

INPUT: LeadTokenDataObject,  DocumentSetID

OUTPUT: DocumentObject

***DocumentObject***

| FieldName | DataType |
|---|---|
| DocumentName | String |
| DocumentID | Integer |
| DocumentHistoryID | Integer |
| ResponseErrorDescription | used to return any errors |
| DocumentSetName | String |
| DateSent | Datetime |
| eSignDocumentID | String - unique ID at Signature Provider |
| eSignStatus | String - status at Signature Provider |
| eSignProvider | String |
| eSignSigningURL | String<br>Only returned by GetDocumentsSent<br>Only for SignNow and ICCO Sign |

## GenerateDocumentSetOnlyDebtor

Processes a document set and generates and stores the documents. Only SQL Report documents that generate as PDF can be used with this call. This can be used for Leads or Clients

```
<OperationContract()>
  Function GenerateDocumentSetOnlyDebtor(ByVal Token As TokenDataObject, ByVal DocumentSetID As
Integer) As DocumentObject()
```

INPUT: TokenDataObject, DocumentSetID

OUTPUT: DocumentObject

## LeadGetDocument

```
      <OperationContract()>
      Function LeadGetDocument(ByVal LeadToken As LeadTokenDataObject, ByVal DocumentHistoryID As
Integer) As DocumentFileInfoObject
```

INPUT: LeadTokenDataObject,DocumentHistoryID for the document that is desired

OUTPUT: DocumentFileInfoObject

### *DocumentFileInfoObject*

| FieldName | DataType |
|---|---|
| FileName | String |
| FileExtension | String |
| FileContent | Bytes |
| DocumentHistoryID | Integer |
| ResponseErrorDescription | used to return any errors |

## DebtorGetDocument

Can be used for Leads or Clients

```
<OperationContract()>
  Function DebtorGetDocument(ByVal Token As TokenDataObject, ByVal DocumentHistoryID As Integer) As
DocumentFileInfoObject
```

INPUT: TokenDataObject,DocumentHistoryID for the document that is desired

OUTPUT: DocumentFileInfoObject

### LeadGetDocumentsSent (REST Available)

```
        <OperationContract()>
        Function LeadGetDocumentsSent(ByVal LeadToken As LeadTokenDataObject) As DocumentObject()
```

INPUT: LeadToken as LeadTokenDataObject.

OUTPUT: Array of DocumentObject.

### LeadCancelESignDocument

```
        <OperationContract()>
        Function LeadCancelESignDocument(ByVal LeadToken As LeadTokenDataObject, ByVal eSignDocumentID
As String, ByVal CancelReason As String) As WSGenericResult
```

INPUT: LeadToken as LeadTokenDataObject, eSignDocumentID as String, CancelReason as String

OUTPUT: WSGenericResult

### GetDocumentsSent (REST Available)

Returns the documents sent to the lead or client

```
        <OperationContract()>
        Function GetDocumentsSent(ByVal Token As TokenDataObject) As DocumentObject()
```

***INPUT: TokenDataObject***

***OUTPUT: Array of DocumentObject***

### GenerateAndSendSimpleEmail (REST Available)

```
        <OperationContract()>
        Function GenerateAndSendSimpleEmail(ByVal LeadToken As LeadTokenDataObject, ByVal DocumentSetID
As Integer) As WSGenericResult
```

# Saved Search Calls

### GetDataFromSavedSearch

The API Call GetDataFromSavedSearch allows you to execute and retrieve the results of a Saved Search. Only saved searches added through the “Enabled Saved Search Permissions” screen can be used.

### How to enable a saved search call:

1. Navigate to Settings -> API -> Preferences.
2. Click Enable Saved Search API Permissions.
3. Double click to add a Saved Search API call.
4. Navigate to Settings -> API -> API Users.
5. Click the Rest Calls Allowed button.

6. Look for the Saved Search API call that corresponds to the SearchID you want and check the box (e.g. GetDataFromSavedSearch_123 if 123 is your Saved Search ID)
7. In addition, you have to always add permission to the generic GetDataFromSavedSearch call.

## How to delete a saved search call:

1. Navigate to Settings -> API -> Preferences.
2. Click Delete Saved Search API Permissions.
3. Double click to delete a Saved Search API call.

```
<OperationContract()>
        Public Function GetDataFromSavedSearch(ByVal JSONObject As GetDataFromSavedSearchDataObject) As
String
```

INPUT: JSONObject as GetDataFromSavedSearchDataObject

OUTPUT: String

### *GetDataFromSavedSearchDataObject*

| FieldName | DataType | Example |
|---|---|---|
| SavedSearchID | Integer | 123 |
| JSONParam | String.<br>Used to pass the {criteria.jp_xyz} bookmarks with their values. | {“jp_debtorid”:”1234”,”jp_userid”:”Smith”} |

The Saved Search SQL can contain bookmarks that will be replaced from the JSONParam list. The bookmarks in the Saved Search SQL must be in the format **{criteria.**jp_xyz**}** where jp_xyz is the bookmark defined in the JSONParam.

So for example the JSON parameter called “jp_debtorid” would need to have a {criteria.jp_debtorid} bookmark in the SQL.

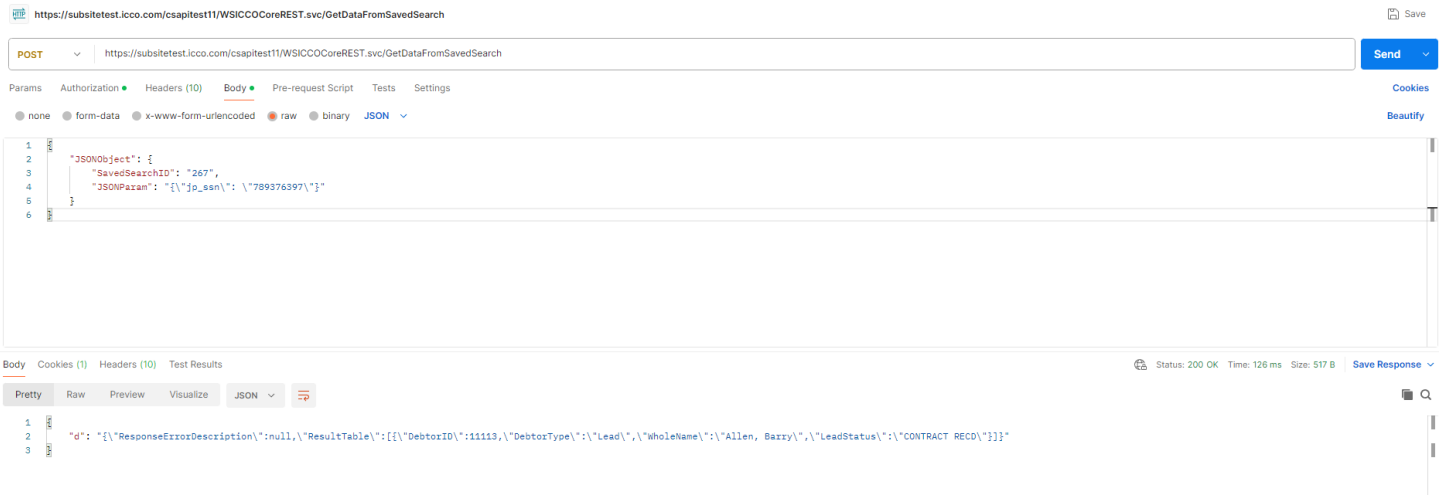
## Example: Finding leads by SSN

Saved search SQL:

```
select
   lc.ClientID as DebtorID,
   'Lead' as DebtorType,
   lc.WholeName,
   lc.LeadStatus
 from LeadClient lc
 WHERE
   dbo.EmptyToNull(CAST(DECRYPTBYKEYAUTOCERT(CERT_ID('ICCO_CERTIFICATE'),NULL,lc.SSN_Encrypted)  AS NVARCHAR(MAX))) =
'{criteria.jp_ssn}'
```

Notice that we surround the bookmark with **single quotes**, since it’s a string.

API Call:

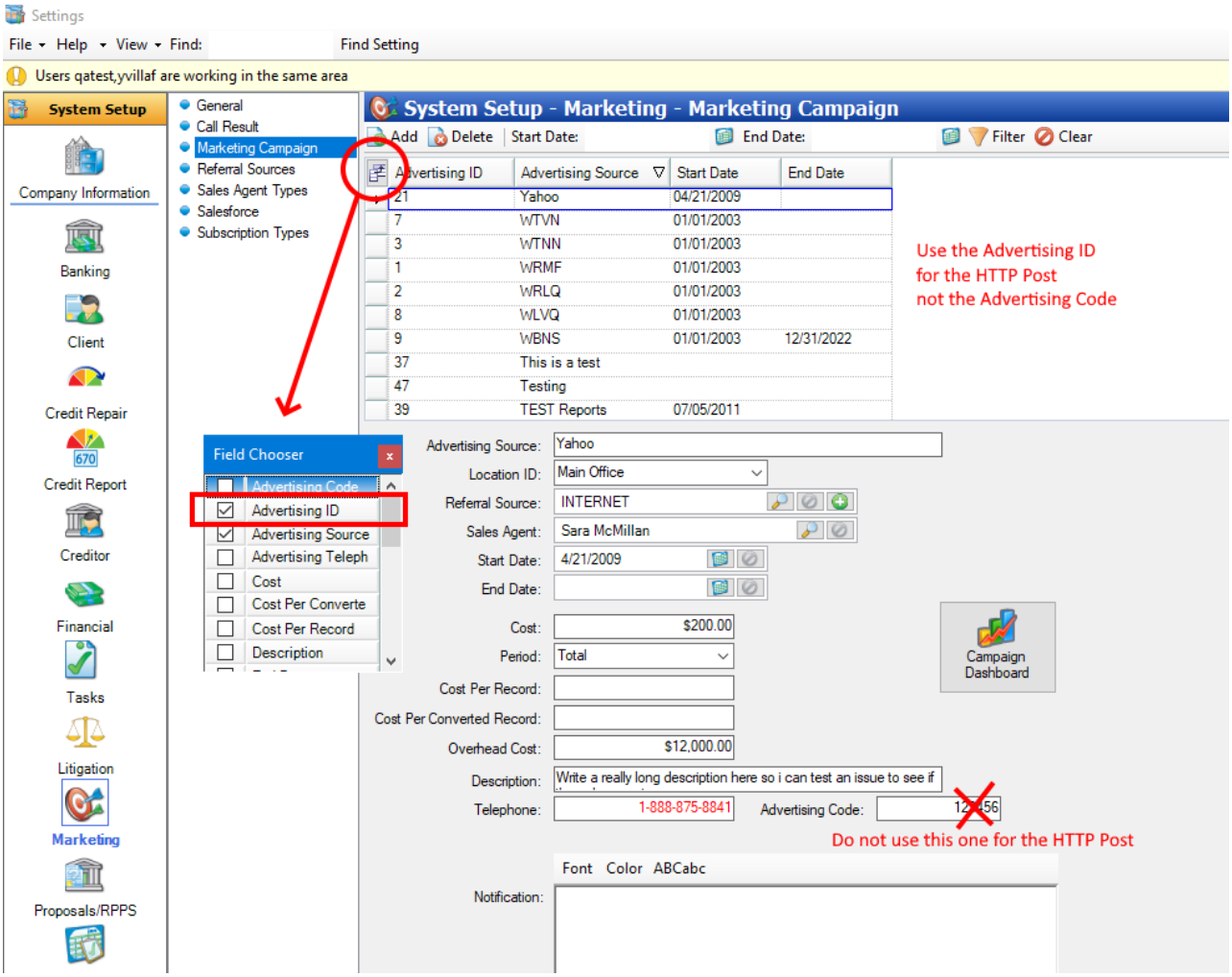



# Lead HTTP Post

An HTTP Post can be submitted to the API to create a Lead in the CreditSoft Database. On success, the API will send the Lead ID assigned in the HTTP response. If an error occurs and the lead cannot be saved into CreditSoft, a message will be returned in the HTTP response.

## Fields Sent via HTTP Post

- FirstName
- LastName
- Address1
- Address2
- City
- State
- Zip
- Email
- HomePhone
- WorkPhone
- HomePhoneTime
- DebtAmount
- BillStatus
- AdvertisingID (Must be the Advertising ID from the advertising table)




## Post Examples

The web page that will process the HTTP POSTS is API_AddLead.aspx. There will be two ways of sending information to this page: Query String and Create HTTP POST request.

## Add Lead using Query String

https://<ICCOCOREAPISITE>/API_AddLead.aspx?firstname=John&lastname=Smith&address1=1600%20pennsylvania%20ave&address2=&city=Jonestown&state=FL&zip=99999&email=john@smith.com&homephone=9995551212&workphone=8995551212&homephonetime=Anytime&debtamount=50000&billstatus=skipping&advertisingid=21

## Add Lead using HTTP Post

<form name="API_AddLead" id="API_AddLead" method="POST" action="https://< ICCOCOREAPISITE >/API_AddLead.aspx " target="cnfm">

<input type="hidden" name="__VIEWSTATE" value="" />

<input name="FirstName" type="hidden" id="FirstName" value="John" />

<input name="lastname" type="hidden" id="lastname" value="Smith" />

<input name="address1" type="hidden" id="address1" value="1600 pennsylvania ave" />

<input name="address2" type="hidden" id="address2" value="" />

<input name="city" type="hidden" id="city" value="Jonestown" />

<input name="state" type="hidden" id="state" value="FL" />

<input name="zip" type="hidden" id="zip" value="99999" />

<input name="email" type="hidden" id="email" value="john@smith.com" />

<input name="homephone" type="hidden" id="homephone" value="9995551212" />

<input name="workphone" type="hidden" id="workphone" value="8995551212" />

<input name="homephonetime" type="hidden" id=" homephonetime " value="1300" />

<input name="debtamount" type="hidden" id="debtamount" value="50000" />

<input name="billstatus" type="hidden" id=" billstatus " value=" oneday " />

<input name="advertisingid" type="hidden" id="advertisingid" value="21" />

</form>

**Responses**

**On Success**

RESULT=SUCCESS<br>
LEADID=[CreditSoftLeadID]

**On Failure**

RESULT=FAILURE<BR>
ERROR=[ErrorMessage]

The possible error messages are:

Invalid Advertising ID
First Name is required
Last Name is required
Debt Amount must be numeric
Unknown error ocurred.  Error Details: [ErrorDetails]

# Outlook Add-In Objects

## OutlookAddInServiceGeneric_Result

Used by the most api calls from Outlook Add-In as return class it contains the responseErrorDescription and the data

| FieldName | DataType |
|---|---|
| responseErrorDescription | string |
| data | object |

## OutlookAddInService_Result

Used by the ProcessMailItem call as data in OutlookAddInServiceGeneric_Result

| FieldName | DataType |
|---|---|
| ViewTaskURL | string |
| ViewLeadURL | string |
| ViewClientURL | string |
| ViewCreditorURL | string |
| UserID | string |
| Product | string |
| lblFrom | string |
| txtFrom | string |
| IssueID | int? |
| lblCompany | string |
| txtCustomerID | string |
| txtCustomer | string |
| txtStatus | string |
| txtCustomerIDTag | string |
| txtStatusForeColor | string |
| txtStatusBackColor | string |
| txtStatusForeColorHtml | string |
| txtStatusBackColorHtml | string |
| tsbRecipientDetails | string |
| tsbViewIssues | string |

| | |
|---|---|
| txtCustomerInactiveVisible | bool |
| txtContact | string |
| txtContactID | string |
| txtContactIDTag | string |
| txtContactInactiveVisible | bool |
| txtIssue | string |
| tsmiOther_UntagEmailVisible | bool |
| tsmiOther_UntagConversationVisible | bool |
| tsbAddToIssueEnabled | bool |
| tsbAddToIssueIEnabled | bool |
| tsbIssueDetailEnabled | bool |
| tsbCreateIssueEnabled | bool |
| tsbCreateIssueIVisible | bool |
| tsbIssueDetailsVisible | bool |
| tsbCreateAdditionalIssueIEnabled | bool |
| txtIssueStatus | string |
| txtIssueStatusTag | string |
| txtIssueCategory | string |
| txtAssignedUserID | string |
| txtIssueSummary | string |
| IssueRecordName | string |
| txtIssueStatusForeColor | string |
| txtIssueStatusBackColor | string |
| txtIssueStatusForeColorHtml | string |
| txtIssueStatusBackColorHtml | string |
| txtIssueClientAccountInfo | string |
| tsbTagOnlyEnabled | bool |
| tsbIssueDetailsEnabled | bool |
| btnChooseObjectVisible | bool |

| | |
|---|---|
| txtFromTag | string |
| GetItemsFromEmail | DataTable |
| ChooseCustomerTable | DataTable |
| txtCustom | string |
| txtCustomVisible | bool |
| isItemTaggedWithIssueID | bool |
| taskString | string |
| IssueRecordType | string |
| IssueRecordID | string |

# Outlook Add-In Calls

## AddIssueNoteToIssue (Rest Available)

Create a file of type .MSG with the mail and attach that file as a note to the issueId

```
<OperationContract()>
Public Async Function AddIssueNoteToIssue(Token As TokenDataObject, MessageID As String, MessageDateUTC As DateTime,
IssueID As Integer?, RecordID As Integer?, RecordType As String, ContactID As Integer?, ContactType As String, SenderName As
String, Subject As String, Body As String, EmailFrom As String, EmailTo As String) As Threading.Tasks.Task(Of String)
```

Returns a OutlookAddInServiceGeneric_Result, data is the created IssueNoteID

## AddIssueWithMessage (Rest Available)

Create a new issue into the database and create a file of type MSG with the mail and attach that file as a note to that issueID

```
<OperationContract()>
Public Async Function AddIssueWithMessage(Token As TokenDataObject, MessageID As String, MessageDateUTC As DateTime,
RecordID As Integer?, RecordType As String, ContactID As Integer?, ContactIDTag As String, IssueCategoryID As Integer,
SenderName As String, Subject As String, Body As String, EmailFrom As String, EmailTo As String) As Threading.Tasks.Task(Of
String)
```

Returns a OutlookAddInServiceGeneric_Result, data is the created IssueID

## CopyMessageAppendIssueToSubject (Rest Available)

Create a copy of the message and append the issue id at the end of the subject example: Issue with DQ to Issue with DQ [Task #12345]

```
<OperationContract()>
Public Async Function CopyMessageAppendIssueToSubject(Token As TokenDataObject, MessageID As String, MessageDateUTC
As DateTime, IssueID As Integer) As Threading.Tasks.Task(Of String)
```

Returns a OutlookAddInServiceGeneric_Result, data is a Boolean that indicates if the email has been created

## CreateActions (Rest Available)

Retrieve the data form the stored procedure GetEmailChangeCandidates (Email WorkFlow rules)

```
<OperationContract()>
Public Function CreateActions(Token As TokenDataObject, IssueID As Integer?, ClientID As Integer?, LeadID As Integer?,
CreditorID As Integer?, UserID As String, EmailSubject As String,
EmailBody As String, EmailSender As String, EmailTo As String, EmailCC As String, EmailSentOn As DateTime) As String
```

Returns a OutlookAddInServiceGeneric_Result, data is a list with the EmailChangeMasterID and Description

| FieldName | DataType |
|---|---|
| EmailChangeMasterID | int |
| Description | string |

## ExecuteEmailChange (Rest Available)

Run the stored procedure ExecuteEmailChange that executes the selected EmailChangeMasterID

```
<OperationContract()>
Public Function ExecuteEmailChange(Token As TokenDataObject, ECMID As Integer?, IssueID As Integer?, ClientID As Integer?,
LeadID As Integer?, CreditorID As Integer?, UserID As String, EmailSubject As String, EmailBody As String, EmailSender As String,
EmailTo As String, EmailCC As String, EmailSentOn As DateTime?) As String
```

Returns a OutlookAddInServiceGeneric_Result, data is the OutputText if the values is in the update statement form the email workflow selected

## GetActiveRecords (Rest Available)

Retrieve a list of the active CSRecordType (Lead,Client,Creditor).

```
<OperationContract()>
Public Function GetActiveRecords(Token As TokenDataObject, RecordType As String, Filter As String) As String
```

Returns a OutlookAddInServiceGeneric_Result, data is a list with the following fields

For clients and leads returns

| FieldName | DataType |
|---|---|
| ClientID/LeadID | int |
| WholeName | string |
| SSN | string |

For Creditors returns

| FieldName | DataType |
|---|---|
| CreditorID | int |
| Creditor | string |

| | |
|---|---|
| Address1 | string |
| City | string |
| State | string |
| Zip | string |

### GetIssueCategories (Rest Available)

Retrieve a list of the active Issue Categories

```
<OperationContract()>
Public Function GetIssueCategories(Token As TokenDataObject) As String
```

Returns a OutlookAddInServiceGeneric_Result, data is a list with the following fields

| FieldName | DataType |
|---|---|
| CategoryID | int |
| Category | string |

### GetRecentIssues (Rest Available)

Retrieve a list with the recent issues

```
<OperationContract()>
Public Function GetRecentIssues(Token As TokenDataObject, UserID As String, RecordID As Integer?, RecordType As String) As String
```

Returns a OutlookAddInServiceGeneric_Result, data is a list with the recent issues with the following fields

| FieldName | DataType |
|---|---|
| IssueID | int |
| Customer | string |
| Summary | string |
| LastNote | string |
| Category | string |
| Updated | Datetime |

### LinkRecipientToDBObject (Rest Available)

Link the email as Coapp or ClientContact to an Client lead or Creditor

```
<OperationContract()>
Public Function LinkRecipientToDBObject(Token As TokenDataObject, RecordType As String, RecordID As Integer, Email As String, Contact As String) As String
```

Returns a OutlookAddInServiceGeneric_Result, data is the WholeName of the Client/Lead or Name of the Creditor

### LoginOutlookAddIn (Rest Available)

Used to retrieve a TokenDataObject that is used in others api calls for Outlook add-in

```
<OperationContract()>
Public Function LoginOutlookAddIn(ByVal objUser As WebUserTokenDataObject) As String
```

Returns a TokenDataObject

### ProcessMailItem (Rest Available)

Retrieve all the data needed to show the info in the Outlook Add-in window

```
<OperationContract()>
Public Function ProcessMailItem(Token As TokenDataObject, Optional EmailSubject As String = Nothing, Optional EmailBody As String = Nothing, Optional EmailSender As String = Nothing, Optional EmailSenderName As String = Nothing, Optional EmailTo As String = Nothing, Optional EmailCC As String = Nothing, Optional EmailSentOn As Nullable(Of DateTime) = Nothing, Optional ReloadClientInfo As Boolean = True, Optional ReloadTaskInfo As Boolean = True) As String
```

Returns a OutlookAddInServiceGeneric_Result, data is an OutlookAddInService_Result

### RemoveIssueIDTagfromItem (Rest Available)

Create a copy of the message and deletes the issue id at the end of the subject example: Issue with DQ [Task #12345] to Issue with DQ

```
<OperationContract()>
Public Async Function RemoveIssueIDTagfromItem(Token As TokenDataObject, MessageID As String, MessageDateUTC As DateTime, IssueID As Integer, Subject As String) As Threading.Tasks.Task(Of String)
```

Returns a OutlookAddInServiceGeneric_Result, data is a Boolean that indicates if the email has been created

## Mobile App Calls

### GetMobileAPIToken (Rest Only)

Returns a token that you can use to make other API calls. The IP address should be the one from the client's device

```
<OperationContract()>
       Public Function GetMobileAPITokenJ(UserName As String, Password As String, ByVal IPAddress As String) As TokenDataObject
```

UserName: This is the username from the internet account of the lead or client (same as the one they use to login to the web portal).

Password: This is the password of the lead or client (same as the one they use to login to the web portal)

IP Address: This is the mobile device IP address

## LinkAccountInitiate (Rest Only)

This call allows you to start the process of either:

a) create a username for a lead, client or co-applicant
   or
b) find the username and reset the password of a lead, client or co-applicant

Example JSON Body:

{

"emailorphone" : "test@test.com",

"IPAddress" : "12.12.12.12"

}

Parameter: emailorphone

The email or hone of a lead, client or co-applicant and returns a LinkID value (a GUID) that you can then pass to the LinkAccountComplete.

You will get “User account not found” if the email address or phone is not found or there are multiple matches.

If an Email address is provided, it sends a verification code to the email. If a phone number is provided, it sends a verification code to the phone.

Parameter: IPAddress

The forwarded IP address of the client’s device (since this API call is made by an internal server).

Returns:

A JSON object where the Value contains the LinkID.

Example of a return:


```
{
    "d": {
        "__type": "WSGenericResult:#ICCOCoreAPI",
        "ID": null,
        "ResponseErrorDescription": null,
        "Value": "bb06ccee-b1b3-412e-ad19-28d7c56e2c93"
```

```
    }
}


<OperationContract()>
       Public Function LinkAccountInitiate(emailorphone As String, IPAddress As String) As
WSGenericResult
```

## LinkAccountComplete (Rest Only)

This is used both for LinkAccount/Registration and also for ForgotPassword functionality on a mobile app.

Example JSON body:

```
{
   "LinkID": "bb06ccee-b1b3-412e-ad19-28d7c56e2c93",
   "Code" : "404026",
   "NewPassword" : "doohickey",
   "IPAddress" : "12.12.12.12"
}
```

This call requires that you pass the following:

Parameter: LinkID

The LinkID you received from the LinkAccountInitiate.

Parameter: Code

The code received by the lead, client or co-applicant after you called LinkAccountInitiate.

Parameter: NewPassword

The new password for the account.

Returns:

A TokenDataObject that you can use for any additional API calls.

Example of a return:

```
{
   "d": {
      "__type": "TokenDataObject:#ICCOCoreAPI",
      "CSRecordID": 10693,
      "CSRecordType": "Client",
      "Remarks": null,
```

"ResponseErrorDescription": **null**,

"Token":
"A4GOC2nhCzZQa8ZEcqx7Lrc8TVCIU8BVLtAYLoXKaImLFs7hgWqG8XPMOZnWF/BW17MLtwwZFoT6ZpeGiXdxTMg/YAC+KH+0KAj9xcZlTgAr3xclrW/SixGHq85+lA7j2FnRS7CP0hsrGQuQmlIva6NZXwy7u788SGhRuTNgsDeykftYZcqoT7iIAwiIFQeZD9ww4GAfbMp523cLO4Ybj2/bmMXpEb+1qqJ6pzNpvgWbnquqvkXJR3nC7dL/0F3EobjZ7O6kUgCFtYQvQWw==",

"UserName": "test@test.com"

}

```
<OperationContract()>
	Public Function LinkAccountComplete(LinkID As String, Code As String, NewPassword As String,
IPAddress As String) As TokenDataObject
```

## GetStoredDocumentList (Rest Only)

List of documents that a user can either download or sign in the custom mobile app

```
<OperationContract()>
	Public Function GetStoredDocumentListJ(ByVal Token As TokenDataObject) As
WSStoredDocumentListResult()
```

### *WSStoredDocumentListResult*

| FieldName | DataType |
|---|---|
| FileID | Integer<br>The IssueNoteID or DocumentHistoryID depending on the FileRepositoryType |
| FileRepositoryType | String<br>IssueNotes or DocumentHistory |
| VirtualFullFileName | String<br>This is the name you will need to pass to retrieve the actual document bytes |
| DisplayName | String<br>The friendly name of the document |
| IssueID | Integer<br>If the document is linked to an issue this will return the IssueID |
| SignedStatus | String<br>For ICCOSign:<br>-1 = Error/Not found/Not applicable<br>0 = Still pending for signature or rejection |

| | |
|---|---|
| | +1 = Rejected less than 10 minutes ago<br>+2 = Rejected more than 10 minutes ago<br>+3 = Signed less than 10 minutes ago<br>+4 = Signed more than 10 minutes ago<br><br>For any other Electronic Signature:<br>0 = Still pending for signature or rejection<br>1 = Signed |
| SignatureRequestGUID | String<br>If this is a signable document via ICCOSign it will return the Signature Request GUID |
| SignatureURL | String<br>If this is a signable document via ICCOSign it will return the link to view or sign the document |
| ResponseErrorDescription | used to return any errors |

## GetStoredDocument (Rest Only)

Returns the document including the document bytes

```
        <OperationContract()>
        Public Function GetStoredDocumentJ(ByVal Token As TokenDataObject, VirtualFullFileName As
String) As WSStoredDocumentResult
```

### *WSStoredDocumentResult*

| FieldName | DataType |
|---|---|
| DisplayName | String |
| DocumentBytes | Bytes<br>The bytes of the document. The VirtualFullFileName will tell you which type of document it is based on the extension (.PDF, .TXT, etc…) |
| ResponseErrorDescription | used to return any errors |

## GetSingleUseLoginURL (Rest Only)

Returns a single-use login URL that expires in 30 secs or after the URL is visited. The number of seconds can be modified in API Settings under Single Use Login URL expiration in seconds.

The SignatureGUID parameter is optional. If passed the URL will be the one to sign the document. If not passed the URL will be the default landing page after auto login.

```
<OperationContract()>
       Public Function GetSingleUseLoginURLJ(ByVal Token As TokenDataObject, IPAddress As String,
Optional SignatureRequestGUID As String = Nothing, Optional AllowLoginFromAnyIP As Boolean = False) As
WSGenericResult
```

The AllowLoginFromAnyIP is also optional. You can pass true to allow login from an IP address different from the one you supplied. This could be useful if the link will be used in a device that is switching networks (rare).

Example of a JSON body for this call:


{ "Token":

  {

    "__type": "TokenDataObject:#ICCOCoreAPI",

    "CSRecordID": 10559,

    "CSRecordType": "Client",

    "Remarks": **null**,

    "ResponseErrorDescription": **null**,

    "Token":
"A4GOC2nhCzZQa8ZEcqx7Lrc8TVCBIUBVLtAYLoKaJEtoOQ2wyda5PF+rHIHI7xdi61CNqE7ZuDFOGVg8g/W7wV4UzAEHwZ0Nd0d1gQHnsDwLk1Z+6c0R7XY5WNdKqRi0gGlwiaNIDRQbyhtVDH0L0j0T/g76j0/f/l7GNJXsT27moWbZ3/drpjzsW1Sg6Gu7NRFiDF4dPQQYxJjMuMZM81qFfhndpNTK0HpwRjtjvZ5UsRTMrhRBV03f3kkpUqRoOUmsmr0wcR+SNA==",

    "UserName": "Rodrigo"

  },

  "IPAddress":"12.0.0.5",

  "SignatureRequestGUID": **null**,

  "AllowLoginFromAnyIP": **true**

}