



Colonial Transfer, LLC

Specification for the Colonial Transfer Corporation Web Services API

Version 2.0

Contents

Change Log	4
Overview.....	8
REST Model	8
Operations	9
Pre-requisite to using the API.....	9
Request Headers	10
Response Codes.....	10
200 - OK	10
201 - Created.....	10
400 – Bad Request.....	10
404 – Not Found	10
500 – Internal Server Error	10
503 – Service Unavailable	10
Response Entity	11
Raw Data Types.....	11
Boolean	11
Datetime	11
Decimal	11
Encrypted String	11
Money.....	11
Number.....	11
String	11
Testing the API	12
Authentication.....	12
Request.....	12
Response Success	12
Response Fail.....	13
Create New Loan Draft iFrame and Create New Lease Draft iFrame.....	14
Request.....	14
Response Success	15
Response Fail.....	15
Create New Loan Draft Silent and Create New Lease Draft Silent.....	17

Request	17
Response Success	17
Response Fail.....	17
Variable You Can Add to the Request	18
Start Loan from Draft	20
Request	20
Response Success	20
Response Fail.....	20
Get Drafts	22
Request	22
Response Success	22
Response Fail.....	22
Get Contract PDF File.....	24
Request	24
Response Success	24
Response Fail.....	24
Get Bank Account Types	25
Request	25
Response Success	25
Cancel Draft.....	26
Request	26
Response Success	26
Response Fail.....	26
Calculate Loan.....	27
Request	27
Response Success	27
Response Fail.....	28
Create Vehicle Loan iFrame	29
Request	29
Create Vehicle Lease iFrame.....	30
Request	30
Create Mortgage iFrame.....	31
Request	31

Create Other Loan iFrame	32
Request	32
Create Vehicle Loan Silent	33
Request	33
Create Vehicle Lease Silent.....	34
Request	34
Create Mortgage Silent.....	35
Request	35
Create Other Loan Silent	36
Request	36
Error Codes	37
General 0 – 99	37
Loan 100 – 199	37
Customer 200 – 299.....	38
Co-signer 300 – 399.....	38

Change Log

Version 1.0

Released: 02/11/2016

- Created core document
- Primary use of the web service is for creating loans and retrieving loan drafts

Version 1.1

Released: 03/01/16

- updated Encrypted Strings to encode with 3DES. Added example C# code for enc/dec

Version 1.2

Released: 03/02/16

- added description for ban account types

- updated descriptions for Get Saved Drafts and Create New Loan
- cosmetic changes to document structure

Version 1.3

Released: 03/04/16

- Added Get Passphrase
- Corrected required and not required fields

Version 1.4

Released: 03/15/16

- Added Cancel Loan

Version 1.5

Released: 10/31/18

- Change security protocol to TLS 1.2
- Remove Passphrase from API
- Add enrollment fee and debit fee to API
- Allow override of our calculated payment monthly amount to be passed through API
- Create additional friendly messages for all errors and put a chart of all error messages into API doc

Version 1.6

Released: 12/04/18

- Added Create New Lease Silent
- Added Create New Lease

Version 1.7

Released: 09/06/21

- Second and Third initial debit date and amount fields are removed from loan creation process
- Customer and Cosigner address1 and address2 changed to just address

- Field **LoanOriginatorUserEmail** has been added to loan creation process in order to identify who is the originator for loan.

Version 1.8

Released 07/18/22

- “Pre-requisite to use the API” has been added
- Adding Vendors authentication to begin a session with the API
- Changed the logo, business information and new urls to represent Colonial Transfer
- “Create New Loan – Silent Mode” – Removed
- “Create New Lease – Silent Mode” – Removed
- “Cancel Loan” – Removed
- “Get Banks” – Removed

Version 1.9

Released 10/07/22

- The encryption algorithm has been changed from TDES to AES
- Added additional methods and refactored the existing methods such as silent vs iframe mode for loan and lease

Version 1.9.2

Released 01/13/22

- Added option to send a **base64 string of pdf file** in the **Create New Loan Draft Silent** and **Create New Lease Draft Silent** method
- Added a new method called **Calculate Loan** which can be used to receive loan parameters and calculate loan schedule with benefits.

Version 1.9.4

Released 02/10/23

- New fields **LoanNote** and **LoanSalesManager** has been added to **Create New Loan Draft iFrame** and **Create New Lease Draft iFrame** endpoints

- New fields **LoanNote** has been added to **Create New Loan Draft Silent** and **Create New Lease Draft Silent**

Version 1.9.5

Released 09/10/24

- Phone number format simplification. Added support of any valid phone number format.

Version 2.0

Released 04/13/25

The V2 API methods are now available. In particular:

- Method that allows to calculate and return loan numbers
- Methods that allow to directly load vehicle loan, vehicle lease, mortgage or other loan iframe.
- Methods that allow to silently create vehicle loan, lease, mortgage or other loan.

Overview

Colonial Transfer Corporation (CTC) web services API provides a way for Menu Software Vendors (Client) to make authenticated calls against Colonial Transfer Corporation from existing applications or third-party tools.

The overall goal of this Web Service is to provide an API for clients to create, save and retrieve loans from the CTC Biweekly Payment System.

REST Model

Clients make standard HTTPS requests and should always validate the certificate of the CTC endpoint with which the client is communicating.

- **Testing Endpoint:** <https://sandbox.eagle.api.colonialtransfer.com/>
- **Production Endpoint:** <https://eagle.api.colonialtransfer.com/>

Operations

API requests are standard HTTPS requests against Colonial Transfer Corporation resources. In general:

- **GET** queries for a list of resources or the details of a specific resource.
- **POST** creates a new resource instance and will provide a resource instance in the response body.

Pre-requisite to using the API

Before you can make API calls to the CTC web service, there are few steps that must be completed:

1. Your organization must be registered in Colonial Transfer CRM system as a Vendor
2. Your organization must have permission to make calls to the Web API
3. Your organization must have a unique Vendor ID and Vendor password for API authentication and token generation
4. You must provide a valid email that will allow CTC to identify the user you are creating the loan for. This email must be associated with an **Active User** in the Colonial Transfer CRM that must be connected to **one or more Dealerships**. If the user has more than one Dealership connected to them, we will provide a dealership dropdown list in the iframe you present to them so they can pick the Dealership they intend to add the loan for.

Request Headers

When a client starts a new authentication request, Colonial Transfer Web Service returns a token to be used as the **AccessToken** for all future requests. The **AccessToken** expires in 24-hours.

The authentication values are provided as HTTP headers:

- Header: "**Authorization**" - "**Bearer [AccessToken]**"
- Header: "**OriginatorUserEmail**" - The email address of an active user in the CRM (i.e.: a Finance Manager) who is connected to at least one dealership.

Authorization is required.

For example:

```
-H "Authorization: Bearer [AccessToken]"  
-H "OriginatorUserEmail: joe@doe.com"
```

Response Codes

Colonial Transfer Web Service responds with standard HTTP response codes appropriate to the result of the request.

200 - OK

The request was successful and details about the response can be found in the body of the response.

201 - Created

The requested POST operation was successful, and an object was created in the system.

400 – Bad Request

Your request was improperly formatted. You should verify that your request conforms to this specification and re-issue the request in a properly formatted manner.

404 – Not Found

The requested resource does not exist.

500 – Internal Server Error

Internal server error. Please retry again. If this fails again contact support at 1-866-882-3110.

503 – Service Unavailable

Service is unavailable

Response Entity

All GET methods respond with the JSON of the resource(s) being requested, POST methods may respond with a 201 CREATED response code along with a base64 encoded contract depending on whether the creation completed immediately or is an asynchronous operation. If the resource was created immediately, Colonial Transfer Web Service will provide a JSON entity that includes the new resource's unique ID.

Raw Data Types

Boolean

Represented by the text "true" or "false"

Datetime

MMM-DD-YYYY

Most standard options will work but we recommend you stick with the format above.

example: Jan 21, 2016

**Important! The system detects Month/Day/Year and not Day/Month/Year
So 05/06/2022 equals May 6, 2022 and NOT June 5, 2022**

To reiterate, in order to keep debugging easy for both ends, please use our recommended format of (MMM-DD-YYYY)

Decimal

Numbers can have decimal points where appropriate

Encrypted String

All encrypted strings are encrypted using AES encryption algorithm.

Money

All amounts must have a two-digit decimal value. For example, 45.40

Number

Any kind of raw number value without a decimal place.

String

A string is any text from the Unicode spectrum

Testing the API

You can find an example of the headers, JSON, and responses of the API calls and returns by going to the following URL. By filling in your actual credentials, you have a test environment with access to all the methods made available to your account.

TEST API: <https://sandbox.eagle.api-tester.colonialtransfer.com>

Also note the documentation for using the API can be downloaded here as well.

Authentication

To start working with Colonial Transfer web services API, the client should provide their Vendor username and password to receive a unique Access Token. This Access Token will be valid for 24 hours.

Request

POST: /auth

Request Body:

```
{
  "Username": "username",
  "Password": "Password"
}
```

Response Success

HttpCode: 200 OK

Response Content:

```
{
  "IsSuccess": true,
  "Data": {
    "AccessToken": "[Access Token string]",
    "Type": "bearer",
    "ExpirationDate": "[Date]"
  }
}
```

Response Fail

HttpCode: 401 Unauthorized

Response Content:

```
{
  "IsSuccess": false,
  "Data": {
    "Errors": [
      {
        "Code": "[Error Code 1]",
        "ErrorMessage": "[Error Message 1]"
      }
    ]
  }
}
```

All API methods will return 401 Unauthorized Access if no Access token is provided. A 404 Not Found will be returned if caller tries to access a method that doesn't exist. Each API call should contain the access token in the request header to access API methods. (See [Request Headers](#) for more information)

Create New Loan Draft iFrame and Create New Lease Draft iFrame

This method is used to load an i-frame that contains the CTC loan creation wizard form. The wizard form **optionally** can be sent with loan/lease details that may already be known during your request call or you can show the wizard with a blank form by sending no information when making the request.

To see a full list of all properties that can be sent with the initial request see this link here:

See [Variable You Can Add to the Request](#)

Note: All valid phone number formats are supported, including dash, space, leading +1 or leading 1 characters.

Request

POST: `/loan/start`

or

POST: `/lease/start`

Request Body, same for both endpoints:

```
{
  "CoSignerAddress": "4444 Park Avenue",
  "CoSignerBirthDate": "Nov 1, 1975",
  "CoSignerCity": "New York",
  "CoSignerEmail": "jane@doe.com",
  "CoSignerExists": "true",
  "CoSignerFirstname": "Jane",
  "CoSignerLastname": "Doe",
  "CoSignerMobile": "561-555-7689",
  "CoSignerPhone": "330 555 8831",
  "CoSignerSSN": "11111111",
  "CoSignerState": "NY",
  "CoSignerWorkPhone": "+1 916-555-2284",
  "CoSignerZip": "11124",
  "CustomerAddress": "3333 Park Avenue",
  "CustomerBankAccountNumber": "111111111",
  "CustomerBankAccountTypeCode": "1",
  "CustomerBankName": "Bank of America",
  "CustomerBankRoutingNumber": "111111111",
  "CustomerBirthDate": "Dec 1, 1990",
  "CustomerCity": "New York",
  "CustomerEmail": "joe@doe.com",
  "CustomerFirstname": "Joe",
  "CustomerLastname": "Doe",
  "CustomerMobile": "334-555-6951",
  "CustomerPhone": "928 555 7874",
  "CustomerSSN": "44444444",
  "CustomerState": "NY",
  "CustomerWorkPhone": "12565554461",
  "CustomerZip": "11123",
  "LoanAmount": "20000",
}
```

```
"LoanBankAccountNumber": "111111111",
"LoanBankName": "Bank of America",
"LoanBankPaymentDueDate": "Dec 1, 2016",
"LoanBankRoutingNumber": "111111111",
"LoanBoost": "100",
"LoanContractBase64": "Loan contract pdf valid base64 string goes here",
"LoanFirstInitialDebitAmount": "500.00",
"LoanFirstInitialDebitDate": "Dec 1, 2016",
"LoanEnrollmentTypeCode": "2",
"LoanInterestRate": "5.9",
"LoanMonthlyPaymentAmount": "500.00",
"LoanNote": "My wonderful note goes here",
"LoanRecurringDebitStartDate": "Dec 1, 2016",
"LoanSalesManager": "Sally Doe",
"LoanTerm": "48",
"LoanVehicleModel": "M3",
"LoanVehicleYear": "2003",
"LoanVehicleMake": "BMW",
"LoanVehicleVin": "WBSBLKJHWQJDKAJSDHK8754DE",
"ReturnUrl": "https://your-return-url-goes-here"
}
```

Response Success

HttpCode: 200 OK

Response Content

```
{
  "IsSuccess": true,
  "Data": {
    "iFrameURL": "[valid https url]"
  }
}
```

Response Fail

HttpCode: 400 Bad Request

Response Content

```
{
  "IsSuccess": false,
  "Data": {
    "Errors": [
      {
        "Code": "[Error Code 1]",
        "ErrorMessage": "[Error Message 1]"
      },
      {
        "Code": "[Error Code 2]",
        "ErrorMessage": "[Error Message 2]"
      },
      "...
    ]
  }
}
```

Property	Description
ReturnUrl	Url, where client will be redirected after loan creation process is finished * Required

iFrameURL is the url you will need to present the CTC wizard inside an iFrame inside your Menu Application.

Create New Loan Draft Silent and Create New Lease Draft Silent

Request

POST: `/loan/create`

or

POST: `/lease/create`

Request Body: is the same for “**Create New Loan Draft iFrame**” and “**Create New Lease Draft iFrame**”

Response Success

HttpCode: 201 Created

Response Content

```
{
  "IsSuccess": true,
  "Data": {
    "ContractPdf": "[base64 string of pdf file]",
    "LoanID": "FD586F6CA1C3"
  }
}
```

Response Fail

HttpCode: 400 Bad Request

Response Content

```
{
  "IsSuccess": false,
  "Data": {
    "Errors": [
      {
        "Code": "[Error Code 1]",
        "ErrorMessage": "[Error Message 1]"
      },
      {
        "Code": "[Error Code 2]",
        "ErrorMessage": "[Error Message 2]"
      },
      "... "
    ]
  }
}
```

Variable You Can Add to the Request

Below is a brief description for the properties that are **optionally** provided as JSON with the API call.

IMPORTANT! None of these items are required to load the iframe up but the items listed as Required will be needed to complete the form before the Finance Manager can submit.

Property	Description	(R)quired or (O)ptional
CoSignerAddress	Cosigner address	O
CoSignerBirthDate	Cosigner date of birth	O
CoSignerCity	Cosigner city	O
CoSignerEmail	Cosigner email	R If cosigner is true
CoSignerExists	Whether cosigner exists or not	true or false
CoSignerFirstname	Cosigner first name	R If cosigner is true
CoSignerLastname	Cosigner last name	R If cosigner is true
CoSignerMobile	Cosigner mobile phone	O
CoSignerPhone	Cosigner home phone	O
CoSignerSSN	Cosigner ssn	R If cosigner is true
CoSignerState	Cosigner state	O
CoSignerWorkPhone	Cosigner work phone	O
CoSignerZip	Cosigner zip code	O
CustomerAddress	Customer address	R
CustomerBankAccountNumber	Customer bank account number	R
CustomerBankAccountTypeCode	Customer bank account type code	R
CustomerBankName	Customer bank name	R
CustomerBankRoutingNumber	Customer bank routing number	R
CustomerBirthDate	Customer date of birth	R
CustomerCity	Customer city	R
CustomerEmail	Customer email	R
CustomerFirstname	Customer first name	R
CustomerLastname	Customer last name	R
CustomerMobile	Customer mobile phone	R
CustomerPhone	Customer home phone	O
CustomerSSN	Customer SSN	R
CustomerState	Customer state	R
CustomerWorkPhone	Customer work phone	O
CustomerZip	Customer zip code	R
LoanAmount	Loan amount	R
LoanBankAccountNumber	Loan bank account number	R
LoanBankName	Loan bank name	O
LoanBankPaymentDueDate	Loan bank payment due date	R
LoanBankRoutingNumber	Loan bank routing number	R
LoanBoost	Loan boost payment amount	O
LoanContractBase64	Loan contract pdf base64 string	O
LoanEnrollmentTypeCode	Debit Period, 2 – Biweekly, 4 Weekly	R
LoanFirstInitialDebitAmount	Loan first initial debit amount	R if date is set
LoanFirstInitialDebitDate	Loan first initial debit date	O

LoanInterestRate	Loan interest rate (APR) in percent	R
LoanMonthlyPaymentAmount	Loan monthly payment amount. If not sent then system will calculate this amount.	O
LoanNote	A note for the loan	O
LoanRecurringDebitStartDate	Biweekly payment start date. A date when first biweekly transaction should occur.	O
LoanSalesManager	Sales manager's first and last name	O
LoanTerm	Loan term (in months)	
LoanVehicleMake	Vehicle make	R
LoanVehicleModel	Vehicle model	R
LoanVehicleYear	Vehicle year	R
LoanVehicleVin	Vehicle VIN Number	O

Start Loan from Draft

This method allows the client to load the Colonial Transfer Corporation loan creation wizard iFrame that was previously saved. All values will be loaded into the wizard that was saved from a previous session.

Request

POST: /loan/start/draft/[Draft ID]

Request Body

```
{
  "ReturnUrl" " https://your-return-url-goes-here"
}
```

Property	Description	
ReturnUrl	Url, where client will be redirected after loan creation process is finished	* Required

iFrameURL is the url you will need to present the CTC wizard inside an iFrame.

Response Success

HttpCode: 200 OK

Response Content

```
{
  "IsSuccess": true,
  "Data": {
    "iFrameURL": "[valid https url]"
  }
}
```

Response Fail

HttpCode: 400 Bad Request

Json:

```
{
  "IsSuccess": false,
  "Data": {
    "Errors": [
      {
        "Code": "[Error Code 1]",
        "ErrorMessage": "[Error Message 1]"
      },
      {
        "Code": "[Error Code 2]",
        "ErrorMessage": "[Error Message 2]"
      },
      "...
    ]
  }
}
```

```
}  
}
```

At the end of this process a newly created loan ID will be added to the query string of returnUrl
Example: [http://return-url/?LoanID=\[Loan ID\]](http://return-url/?LoanID=[Loan ID])

Get Drafts

This method is used to retrieve list of all drafts that have been saved by client. These drafts are only available to the client if they did not submit to Eagle for review.

Request

GET: /loan/get-drafts

Response Success

HttpCode: 200 OK

Response Content

```
{
  "IsSuccess": true,
  "Data": {
    "Drafts": [
      {
        "DraftID": "[uniq id of the draft]",
        "Caption": "[Draft Caption]",
        "Date": "[date saved]"
      },
      {
        "DraftID": " 7F1711636B99",
        "Caption": "[Draft Caption]",
        "Date": "Dec 1, 2016 05:34:20 PM"
      },
      {
        "DraftID": " EB7E2567E1ED",
        "Caption": "[Draft Caption]",
        "Date": "Dec 1, 2016 07:44:33 PM"
      },
      "... "
    ]
  }
}
```

Response Fail

HttpCode: 400 Bad Request

Response Content

```
{
  "IsSuccess": false,
  "Data": {
    "Errors": [
      {
        "Code": "[Error Code 1]",
        "ErrorMessage": "[Error Message 1]"
      },
      {
        "Code": "[Error Code 2]",
        "ErrorMessage": "[Error Message 2]"
      }
    ]
  }
}
```

```
    },  
    "..."  
  ]  
}
```

Get Contract PDF File

This method is used to get a base64 encoded string of the pdf contract from a created loan.

Request

GET: /loan/get-contract/[Draft ID]

Response Success

HttpCode: 200 OK

Response Content

```
{
  "IsSuccess": true,
  "Data": {
    "ContractPdf": "[base64 string of pdf file]"
  }
}
```

Response Fail

HttpCode: 400 Bad Request

Json:

```
{
  "IsSuccess": false,
  "Data": {
    "Errors": [
      {
        "Code": "[Error Code 1]",
        "ErrorMessage": "[Error Message 1]"
      },
      {
        "Code": "[Error Code 2]",
        "ErrorMessage": "[Error Message 2]"
      },
      "...
    ]
  }
}
```

Get Bank Account Types

This method is used to get a list of all valid bank account types in the Colonial Transfer Corporation Biweekly Payment System. You use the bank account id when creating a new account into the system.

Request

GET: /dictionary/get-bank-account-types/

Response Success

HttpCode: 200 OK

Response Content

```
{
  "IsSuccess": true,
  "Data": {
    "AccountTypes": [
      {
        "TypeCode": "1",
        "Type": "Checking"
      },
      {
        "TypeCode": "2",
        "Type": "Saving"
      },
      "... "
    ]
  }
}
```

Cancel Draft

This method is used to cancel a personal draft. This will permanently delete the draft. Please note this is not going to delete a draft that was submitted to Eagle for review. In this case, the finance manager must call support directly.

Request

PUT: /loan/cancel/

Json

```
{
  "LoanID": "[Loan ID]"
}
```

Property		
LoanID	ID of the loan to cancel	* Required

Response Success

HttpCode: 200 OK

Response Content

```
{
  "IsSuccess": true
}
```

Response Fail

HttpCode: 404 Not Found

Response Content

```
{
  "IsSuccess": false,
  "Data": {
    "Errors": [
      {
        "Code": "[Error Code 1]",
        "ErrorMessage": "[Error Message 1]"
      },
      {
        "Code": "[Error Code 2]",
        "ErrorMessage": "[Error Message 2]"
      },
      "... "
    ]
  }
}
```

Calculate Loan

This method is used to receive loan parameters and calculate loan schedule with benefits.

Request

POST: /v2/loan/calculate/

Json

```
{
  "LoanAmount": "32000",
  "LoanBankPaymentDateStart": "Feb 25, 2023",
  "LoanBoostAmount": "0",
  "LoanDebitFrequency": "biweekly | weekly | monthly",
  "LoanFirstInitialDebitDate": "Jan 25, 2023",
  "LoanFirstInitialDebitPart": "half | full",
  "LoanInterestRate": "5.9",
  "LoanMonthlyPaymentAmount": "530.27",
  "LoanRecurringDebitDateStart": "Feb 8, 2023",
  "LoanTerm": "72"
}
```

Property		
LoanAmount	Loan total amount	* Required
LoanBankPaymentDateStart	Loan bank payment date first	* Required
LoanBoostAmount	Loan boost amount	Optional
LoanDebitFrequency	The frequency of debit. Accepts three values only: biweekly , weekly or monthly	* Required
LoanFirstInitialDebitDate	The date to debit customer for a first time	* Required
LoanFirstInitialDebitPart	Is first debit supposed to be full amount or half. Accepts two values only: half or full	* Required
LoanInterestRate	Loan interest rate	* Required
LoanMonthlyPaymentAmount	Loan monthly bank payment amount	Optional
LoanRecurringDebitDateStart	Date when recurring debits should start	* Required
LoanTerm	Loan Term	* Required

Response Success

HttpCode: 200 OK

Response Content

```
{
  "IsSuccess": true,
  "Data": {
    "LoanMonthlyPaymentAmount": 530.27,
    "LoanBoostSuggestion": "Round up to $270.00 by adding $2.92",
    "InterestSavings": 543.72,
    "TermReduction": -6,
    "AccelerationBenefit": -3181.62,
    "DeferredProgramFees": 399.0000,
  }
}
```

```
"TotalConvenienceFees": 278.85,
"NetSavings": -134.13,
"LoanCompareSchedule": [
  {
    "RowNumber": 1,
    "BankBeginningBalance": 32000.0,
    "CompanyBeginningBalance": 32000.0
  },
  {
    "RowNumber": 2,
    "BankBeginningBalance": 31627.2333333333333333333333333333,
    "CompanyBeginningBalance": 31627.0633333333333333333333333333
  },
  ...(repeated pattern)
],
"LoanBankWithdrawalDates": [
  {
    "RowNumber": 1,
    "Date": "Jan 25, 2023",
    "DebitAmount": 532.2200
  },
  {
    "RowNumber": 2,
    "Date": "Feb 08, 2023",
    "DebitAmount": 267.0850
  },
  ...(repeated pattern)
]]}
```

Response Fail

HttpCode: 400 Bad Request

Response Content

```
{
  "IsSuccess": false,
  "Data": {
    "Errors": [
      {
        "Code": "[Error Code 1]",
        "ErrorMessage": "[Error Message 1]"
      },
      {
        "Code": "[Error Code 2]",
        "ErrorMessage": "[Error Message 2]"
      },
      ...
    ]
  }
}
```

Create Vehicle Loan iFrame

This method is used to load an embedded iframe wizard form, that allows you to create a vehicle loan.

Request

POST: /v2/loan/create/embed/auto-loan

This API call utilizes the same Request and Response JSON body as the previous loan creation process. You can address to the following sections:

- [Create New Loan Draft iFrame and Create New Lease Draft iFrame](#)
- [Create New Loan Draft Silent and Create New Lease Draft Silent](#)
- [Variable You Can Add to the Request](#)

Create Vehicle Lease iFrame

This method is used to load an embedded iframe wizard form, that allows you to create a vehicle lease.

Request

POST: /v2/loan/create/embed/auto-lease

This API call utilizes the same Request and Response JSON body as the previous loan creation process. You can address to the following sections:

- [Create New Loan Draft iFrame and Create New Lease Draft iFrame](#)
- [Create New Loan Draft Silent and Create New Lease Draft Silent](#)
- [Variable You Can Add to the Request](#)

Create Mortgage iFrame

This method is used to load an embedded iframe wizard form, that allows you to create mortgage.

Request

POST: /v2/loan/create/embed/auto-lease

This API call utilizes the same Request and Response JSON body as the previous loan creation process. You can address to the following sections:

- [Create New Loan Draft iFrame and Create New Lease Draft iFrame](#)
- [Create New Loan Draft Silent and Create New Lease Draft Silent](#)
- [Variable You Can Add to the Request](#)

Create Other Loan iFrame

This method is used to load an embedded iframe wizard form, that allows you to create other loans, besides no vehicle loan/lease and mortgage.

Request

POST: /v2/loan/create/embed/other-loan

This API call utilizes the same Request and Response JSON body as the previous loan creation process. You can address to the following sections:

- [Create New Loan Draft iFrame and Create New Lease Draft iFrame](#)
- [Create New Loan Draft Silent and Create New Lease Draft Silent](#)
- [Variable You Can Add to the Request](#)

Create Vehicle Loan Silent

This method is used to submit vehicle loan information silently, without loading iframe.

Request

POST: /v2/loan/create/silent/auto-loan

This API call utilizes the same Request and Response JSON body as the previous loan creation process. You can address to the following sections:

- [Create New Loan Draft iFrame and Create New Lease Draft iFrame](#)
- [Create New Loan Draft Silent and Create New Lease Draft Silent](#)
- [Variable You Can Add to the Request](#)

Create Vehicle Lease Silent

This method is used to submit vehicle lease information silently, without loading iframe.

Request

POST: /v2/loan/create/silent/auto-lease

This API call utilizes the same Request and Response JSON body as the previous loan creation process. You can address to the following sections:

- [Create New Loan Draft iFrame and Create New Lease Draft iFrame](#)
- [Create New Loan Draft Silent and Create New Lease Draft Silent](#)
- [Variable You Can Add to the Request](#)

Create Mortgage Silent

This method is used to submit mortgage information silently, without loading iframe.

Request

POST: /v2/loan/create/silent/mortgage

This API call utilizes the same Request and Response JSON body as the previous loan creation process. You can address to the following sections:

- [Create New Loan Draft iFrame and Create New Lease Draft iFrame](#)
- [Create New Loan Draft Silent and Create New Lease Draft Silent](#)
- [Variable You Can Add to the Request](#)

Create Other Loan Silent

This method is used to submit other loan information silently, without loading iframe.

Request

POST: /v2/loan/create/silent/other-loan

This API call utilizes the same Request and Response JSON body as the previous loan creation process. You can address to the following sections:

- [Create New Loan Draft iFrame and Create New Lease Draft iFrame](#)
- [Create New Loan Draft Silent and Create New Lease Draft Silent](#)
- [Variable You Can Add to the Request](#)

Error Codes

General 0 – 99

Error Code	Error Message
0	Sorry, something went wrong
1	Invalid username or password
2	Access token is invalid
3	Access token is expired
4	The resource you are looking for is not found
5	Loan originator email was not found in your dealership

Loan 100 – 199

Error Code	Error Message
100.01	Loan amount is required
100.02	Loan amount is invalid. Should be valid dollar amount, without commas, only one dot is allowed to identify cent amounts (example 10000.50)
101.01	Loan term is required
101.02	Loan term format is invalid, must be a valid digit, amount of months. (example 12)
102.01	Loan interest rate is required
102.02	Loan interest rate is invalid. Must be a valid annual percent amount (example: 5.35)
103.01	Loan monthly payment amount is required
103.02	Loan monthly payment amount format is invalid. Should be valid dollar amount, without commas, only one dot is allowed to identify cent amounts (example 100.50)
103.02	Loan monthly payment amount is invalid. Payment cannot exceed loan amount
104.01	Loan bank payment date is required
104.02	Loan bank payment date format is invalid. Valid date example: May 1, 2015
104.03	Loan bank payment date must be at least 12 business days from today
105.01	Loan recurring start date (biweekly payment start date) is required
105.02	Loan recurring start date (biweekly payment start date) format is invalid. Valid date example: May 1, 2015
105.03	Loan recurring start date (biweekly payment start date) must be at least 1 business day from today.
106.01	Loan boost amount is invalid. Should be valid dollar amount, without commas, only one dot is allowed to identify cent amounts (example 100.50)
107.01	Loan first initial debit date format is invalid. Valid date example: May 1, 2015
107.02	Loan first initial debit date must be at least 5 business days from now and 5 business days before loan bank payment date
108.01	Loan second initial debit date format is invalid. Valid date example: May 1, 2015
108.02	Loan second initial debit date must be at least 5 business days from now and 5 business days before loan bank payment date
109.01	Loan third initial debit date format is invalid. Valid date example: May 1, 2015
109.02	Loan third initial debit date must be at least 5 business days from now and 5 business days before loan bank payment date
110.01	Loan first initial debit amount is invalid. Should be valid dollar amount, without commas, only one dot is allowed to identify cent amounts (example 100.50)

111.01	Loan second initial debit amount is invalid. Should be valid dollar amount, without commas, only one dot is allowed to identify cent amounts (example 100.50)
112.01	Loan third initial debit amount is invalid. Should be valid dollar amount, without commas, only one dot is allowed to identify cent amounts (example 100.50)
113.01	Loan initial debit amounts cannot exceed loan total amount
114.01	Loan payment schedule has not created
115.01	There is not enough money for bank payment
116.01	Loan ending balance is not zero after last transaction
117.01	Total commission amount cannot be covered
118.01	Vehicle make is required
119.01	Vehicle model is required
120.01	Vehicle year is required
120.02	Vehicle year must be a valid year number. example: 2005
120.03	Vehicle year can't be less than 1900 or greater than 2100

Customer 200 – 299

Error Code	Error Message
200.01	Customer first name is required
201.01	Customer last name is required
202.01	Customer email is required
202.02	Customer email format is invalid. Valid format example someone@domain.com
202.03	Customer email is already in use
203.01	Customer social security number is required
203.02	Customer social security number is invalid. Must contain 9 digits only, example 111223333
204.01	Customer date of birth is required
204.02	Customer date of birth format is invalid. Valid date example: May 1, 2015
205.01	Customer mobile phone number is required
205.02	Customer mobile phone number is invalid. Must contain 10 digits only (example: 1234567890)
206.01	Customer home phone number is required
206.02	Customer home phone number is invalid. Must contain 10 digits only (example: 1234567890)
207.01	Customer work phone number is required
207.02	Customer work phone number is invalid. Must contain 10 digits only (example: 1234567890)
208.01	Customer zip is required
208.02	Customer zip is invalid. Must contain 5 digits only (example: 12345)
209.01	Customer state is required
209.02	Customer state is invalid. Must be valid US state abbreviation, case insensitive (example NY)
210.01	Customer city is required
211.01	Customer address 1 is required
212.01	Customer address 2 is required
213.01	Customer bank account type code is invalid. Must be a valid "TypeCode" from dictionary.
214.01	Customer bank account number is invalid. It must be 10-digit number

Co-signer 300 – 399

Error Code	Error Message
300.01	Cosigner first name is required
301.01	Cosigner last name is required
302.01	Cosigner email is required
302.02	Cosigner email format is invalid. Valid format example someone@domain.com
302.03	Cosigner email is already in use
303.01	Cosigner social security number is required
303.02	Cosigner social security number is invalid. Must contain 9 digits only, example 111223333
304.01	Cosigner date of birth is required
304.02	Cosigner date of birth format is invalid. Valid date example: May 1, 2015
305.01	Cosigner mobile phone number is required
305.02	Cosigner mobile phone number is invalid. Must contain 10 digits only (example: 1234567890)
306.01	Cosigner home phone number is required
306.02	Cosigner home phone number is invalid. Must contain 10 digits only (example: 1234567890)
307.01	Cosigner work phone number is required
307.02	Cosigner work phone number is invalid. Must contain 10 digits only (example: 1234567890)
308.01	Cosigner zip is required
308.02	Cosigner zip is invalid. Must contain 5 digits only (example: 12345)
309.01	Cosigner state is required
309.02	Cosigner state is invalid. Must be valid US state abbreviation, case insensitive (example NY)
310.01	Cosigner city is required
311.01	Cosigner address 1 is required
312.01	Cosigner address 2 is required