JCS API & LVI Platform API

Table of Contents

Introduction

This document covers the APIs which are provided for:

- LVI Platform communication with the Jafra JCS backend server running on the AS/400.
- Online Sponsoring & Client Direct client-side requests to the LVI Platform backend

Most of the request & response data, success and error codes from the JCS API are mirrored in the LVI API. The LVI API serves as a proxy between the client and the Jafra system, providing some unique additions and caching to help with scaling. Additional information on this can be found in the LVI Web Stack architecture documents.

Conventions

Where different, the LVI Platform URI (URI) will be listed separately from the corresponding JCS API URI (JCS URI). This is also the case for the HTTP verb/method used for each service.

Authenticate

Method: POST
URI: /api/clients/authenticate
JCS URI: /cgidev2/JCD05001P.pgm
Request:

```
{
  "email": "jsmith@gmail.com", // required
  "password": "some password"  // required
}
```

Success

Status Code: 200
Response:

```
{
  "clientId": <clientId>
}
```

Errors

Status Code: 401
Responses:

```
{
    "statusCode": 401,
    "errorCode": "invalidCredentials",
    "message": "Invalid Credentials"
}
```

Get Client

Method: GET
URI: /api/clients/{clientId}
JCS URI: /cgidev2/JCD05007P.pgm?clientId={clientId}
Info: Get client data
Request:

clientId - this is the unique Jafra client ID

Success

Status Code: 200
Response:

```
{
  "id": 1,
  "email": "jsmith@gmail.com",
  "firstName": "John",
  "lastName": "Smith",
  "consultantIds": [
    4657323, 4657324
  ],
  "language": "en_US",
  "phone": "555-555-4432",
  "dateOfBirth": "12/01/1978",
  "creditCards": [],
  "addresses": [],
  "notificationPreferences": {
    "sms": 0,
    "email": 1
  }
}
```

Errors

Status Code: 404
Info: The client was not found
Responses:

The client with the specified ***clientId*** was not found

```
{
    "statusCode": 404,
    "errorCode": "clientNotFound",
    "message": "Client not found"
}
```

Create Client

Method: POST
URI: /api/clients
JCS URI: /cgidev2/JCD05002P.pgm
Info: Create a new client record, which can be used to authenticate into Client Direct & future systems
Request:

```
{
  "email": "jsmith@gmail.com", // required
  "password": "some password", // required
  "firstName": "John",         // optional
  "lastName": "Smith",         // optional
  "phone": "555-555-4432",     // optional
  "dateOfBirth": "12/01/1978", // optional
  "consultantId": 4657323,     // optional
  "language": "en_US",         // optional
  "source": "fb",              // optional
  "notificationPreferences": {
    "sms": 0,
    "email": 1
  }
}
```

Success

Status Code: 201
Response:

```
{
  "clientId": <clientId>
}
```

Errors

Status Code: 400

Info: Specified data was invalid and should be rejected by the server
Responses:

Only respond with this when the optional consultant ID is provided and it isn't valid. Ignore instead if empty

```
{
    "statusCode": 400,
    "errorCode": "invalidConsultantId",
    "message": "Invalid consultant Id"
}
```

Invalid email format, bad characters, etc.

```
{
    "statusCode": 400,
    "errorCode": "invalidEmailAddress",
    "message": "Invalid email address"
}
```

Password that doesn't meet password requirements (length, variation, etc.)

```
{
    "statusCode": 400,
    "errorCode": "invalidPassword",
    "message": "Invalid password"
}
```

Status Code: 409
Responses:

There was a conflict creating an account with the specified email, because it is already in the system.

```
{
    "statusCode": 409,
    "errorCode": "emailAlreadyInUse",
    "message": "Email address already in use"
}
```

Get Consultant

Method: GET
URI: /api/consultants/{consultantId}
JCS URI: /cgidev2/JOS05007P.pgm?consultantId={consultantId}
Info: Get consultant data
Request:

consultantId - this is the unique Jafra consultant ID

Success

Status Code: 201
Response:

```
{
    "consultantId": <consultantId>
}
```

Errors

Status Code: 404
Info: The consultant was not found
Responses:

```
{
    "statusCode": 404,
    "errorCode": "consultantNotFound",
    "message": "Consultant not found"
}
```

Create Consultant

Method: POST
URI: /api/consultants
JCS URI: /cgidev2/JOS05002P.pgm
Info: This is similar to creating a Client, however a few more fields are required. Create a consultant record, which can be used to authenticate into Jafra Biz & eventually Client Direct/Online Ordering.

This request data is GPG encrypted with the public key on the client side in JavaScript and the entire encrypted packet is sent through the LVI Platform unaltered to the JCS API. The JCS API decrypts the data and processes the request.

Request:

```
{
  "email" : "example@example.com",// required
  "firstName": "John",            // required
  "lastName": "Smith",            // required
  "sponsorId": "555678",          // optional
  "language": "en_US",            // required
  "ssn": "553-21-1923",           // required
  "dateOfBirth": "12/01/1978",    // required
  "phone": "555-333-2222",        // required
  "notificationPreferences": {    // required
    "sms": 0,
    "email": 1
  },
  "billingAddress": {             // required
    "name": "Joe Smith",
    "address1": "1111 Test Ln",
    "address2": "",
    "city": "Corona",
    "state": "CA",
    "zip": "92880",
    "county": "Ventura",
    "country": "US",
    "phone": "555-333-2222"
  },
  "shippingAddress": {            // required
    "name": "Joe Smith",
    "address1": "1111 Test Ln",
    "address2": "",
    "city": "Corona",
    "county": "Ventura",
    "geocode": "000000000",
    "state": "CA",
    "zip": "92880",
    "country": "US",
    "phone": "555-333-2222"
  },
  "creditCard": {                  // required
    "name": "Joe Smith",
    "card": "4111111111111111",
    "expMonth": "12",
    "expYear": "2015",
    "cvv": "1111"
  },
  "agreementAccepted": "true",     // required
  "source": "fb",                  // required
  "total": 102.67,                 // required
  "products": [                    // required
    {
      "sku": "25386",
      "qty": 2
    },
    {
      "sku": "19193",
      "qty": 1,
      "kitSelections": {
        "NUPL19193": [{        // each kit group
          "sku": "12026",      // product sku
          "qty": 1             // quantity
        }, {
          "sku": "12029",      // product sku
          "qty": 1             // quantity
        }]
      }
    }
  ]
}
```

sponsorId - initially create as a house account when not specified, send information to CRT for contact and manual sponsor assignment.
source - can be one of {mb, pws}

Success

Status Code: 201
Response:

```
{
    "consultantId": <consultantId>
}
```

Errors

Status Code: 400
Info: Specified data was invalid and should be rejected by the server
Responses:

The SSN was not properly formatted or was otherwise invalid.

```
{
    "statusCode": 400,
    "errorCode": "invalidSSN",
    "message": "Invalid SSN"
}
```

Only respond with this when the optional sponsor ID is provided and it isn't valid. Ignore instead if empty

```
{
    "statusCode": 400,
    "errorCode": "invalidSponsorId",
    "message": "Invalid sponsor Id"
}
```

Invalid email format, bad characters, etc.

```
{
        "statusCode": 400,
        "errorCode": "invalidEmailAddress",
        "message": "Invalid email address"
    }
```

Password that doesn’t meet password requirements (length, variation, etc.)

```
    {
        "statusCode": 400,
        "errorCode": "invalidPassword",
        "message": "Invalid password"
    }
```

Status Code: 402
Responses:

The credit card specified is invalid.

```
    {
        "statusCode": 402,
        "errorCode": "InvalidCreditCard",
        "message": "Invalid Credit Card"
    }
```

The credit card was declined (e.g. Merchant rejected - Elavon)
402 “{
“”statusCode”“: 402,
“”errorCode”“: “”Declined”“,
“”message”“: “”Declined credit card””
}

Call center error.

```
    {
        "statusCode": 402,
        "errorCode": "CallAuthCenter",
        "message": "Call Auth Center"
    }
```

Status Code: 409
Responses:

There was a conflict creating an account with the specified email, because it is already in the system.

```
{
    "statusCode": 409,
    "errorCode": "emailAlreadyInUse",
    "message": "Email address already in use"
}
```

There was a conflict creating an account with the specified SSN, because it is already in the system.

```
{
    "statusCode": 409,
    "errorCode": "ssnAlreadyInUse",
    "message": "SSN already in use"
}
```

Lookup Consultant by SSN

Method: POST
URI: /api/ssn
JCS URI: /cgidev2/JOS05004P.pgm
Info: Determine if a consultant with an SSN exists
Request:

```
{
  "ssn": <ssn>         // required, encrypted
}
```

This request is GPG (asymmetrically) encrypted and sent over to JCS, where it is decrypted and processed.

Success

Status Code: 200

```
{
  "consultantId": <consultantId>
}
```

Errors

Status Code: 404
Responses:

A generic message that simply tells us the server didn’t like the data it received and the lead was not saved.

```
{
    "statusCode": 404,
    "errorCode": "ssnNotFound",
    "message": "SSN not found"
}
```

Create Lead

Method: POST
URI: /api/leads
JCS URI: /cgidev2/JOS05005P.pgm
Info: Created from basic info provided as new consultant is signing up
Request:

```
{
  "firstName": "John",          // required
  "lastName": "Smith",          // required
  "email": "jsmith@gmail.com", // required
  "phone": "555-234-1122",      // required
```

```
    "language": "en_US"           // required
    "type": free text (12 positions allowed) //Optional
  }
```

Success

Status Code: 204
Response:

```

```

JCS Status Code: 200
JCS Response:

```
  {
    "leadId": <leadId>
  }
```

Errors

Status Code: 400
Responses:

A generic message that simply tells us the server didn’t like the data it received and the lead was not saved.

```
  {
      "statusCode": 400,
      "errorCode": "invalidLeadData",
      "message": "Invalid lead data"
  }
```

Status Code: 409
Responses:

The lead has already been saved in JCS

```
{
    statusCode: 409,
    errorCode: "emailAlreadyInUse",
    message: "Email address already in use"
}
```

Get Addresses

Method: GET
URI: /api/clients/{clientId}/addresses
JCS URI: /cgidev2/JCD05005P.pgm?clientId={clientId}

Success

Status Code: 200
Response Body:

```
[{
  "id": 111,
  "name": "Joe Smith",
  "address1": "1111 Test Ln",
  "address2": "",
  "city": "Corona",
  "state": "CA",
  "zip": "92880",
  "country": "US",
  "phone": "555-333-2222",
  "geocode":"00000",
  "stateDescription":   "California",
  "county" : "Ventura"
}]
```

Errors

Status Code: 404
Response Body:

The client ID was not found

```
{
  "statusCode": 404,
  "errorCode": "clientNotFound",
  "message": "Client not found"
}
```

Addresses not found

```
{
  "statusCode": 404,
  "errorCode": "addressNotFound",
  "message": "Address not found"
}
```

Get Address

Info: Get a specific address for client by clientId and addressId
Method: GET
URI: /api/clients/{clientId}/addresses/{addressId}
JCS URI: /cgidev2/JCD05005P.pgm?clientId={clientId}&addressId={addressId}

Success

Status Code: 200
Response:

```
{
  "id": 111,
  "name": "Joe Smith",
  "address1": "1111 Test Ln",
  "address2": "",
  "city": "Corona",
  "state": "CA",
  "zip": "92880",
  "country": "US",
  "phone": "555-333-2222",
  "geocode":"00000",
  "stateDescription":   "California",
  "county" : "Ventura"
}
```

Errors

Status Code: 404
Response Body:

The client ID was not found

```
{
  "statusCode": 404,
  "errorCode": "clientNotFound",
  "message": "Client not found
}
```

The address ID was not found

```
{
  "statusCode": 404,
  "errorCode": "addressNotFound",
  "message": "Address not found
}
```

Create Address

Method: POST
URI: /api/clients/{clientId}/addresses
JCS URI: /cgidev2/JCD05005P.pgm?clientId={clientId}
Request:

```
{
  "name": "Joe",                // optional
  "address1": "1111 Test Ln", // required
  "address2": "",             // optional
  "city": "Corona",           // required
  "state": "CA",              // required
  "zip": "92880",             // required
  "country": "US",            // required
  "phone": "555-333-2222",     // optional
  "geocode":"00000",          // required
  "stateDescription": "California", // optional
  "county" : "Ventura", // required
}
```

Success

Status Code: 201
Response:

```
{
  "addressId": <addressId>
}
```

Errors

Status Code: 400
Response Body:

The address data was invalid or incomplete.

```
{
  "statusCode": 400,
  "errorCode": "addressError",
  "message": "Address not created"
}
```

Status Code: 404
Response Body:

The client ID was not found

```
{
  "statusCode": 404,
  "errorCode": "clientNotFound",
  "message": "Client not found
}
```

Update Address

Method: PUT
URI: /api/clients/{clientId}/addresses/{addressId}
JCS Method: POST
JCS URI: /cgidev2/JCD05005P.pgm?clientId={clientId}&addressId={addressId}
Request:

```
{
  "name": "Joe",                    // optional
  "address1": "1111 Test Ln",       // required
  "address2": "",                   // optional
  "city": "Corona",                 // required
  "state": "CA",                    // required
  "zip": "92880",                   // required
```

```
    "country": "US",                    // required
    "phone": "555-333-2222",            // optional
    "geocode":"00000",                  // required
    "stateDescription": "California", // optional
    "county" : "Ventura",               // required
  }
```

Success

Status Code: 204
Response Body:

```

```

JCS Status Code: 201
JCS Response Body:

```
  {
    "addressId": <addressId>
  }
```

Errors

Status Code: 404
Response Body:

The client ID was not found

```
  {
    "statusCode": 404,
    "errorCode": "clientNotFound",
    "message": "Client not found
  }
```

Status Code: 400
Response Body:

The address data was invalid or incomplete.

```
{
  "statusCode": 400,
  "errorCode": "addressError",
  "message": "Address not created"
}
```

Delete Address

Method: DELETE
URI: /api/clients/{clientId}/addresses/{addressId}
JCS URI: /cgidev2/JCD05005P.pgm?clientId={clientId}&addressId={addressId}

Success

Status Code: 204

Errors

Status Code: 404
Response Body:

The client ID was not found

```
{
  "statusCode": 404,
  "errorCode": "clientNotFound",
  "message": "Client not found
}
```

The address ID was not found

```
  {
    "statusCode": 404,
    "errorCode": "addressNotFound",
    "message": "Address not found
  }
```

Get Credit Cards

Method: GET
URI: /api/clients/{clientId}/creditCards
JCS URI: /cgidev2/JCD05008P.pgm?clientId={clientId}

Success

Status Code: 200
Response Body:

```
  [{
    "id": 123,
    "name": "Joe Smith",
    "lastFour": "1111",
    "cardType": "visa",
    "expMonth": "12",
    "expYear": "2015"
  }]
```

Errors

Status Code: 404
Response Body:

The client ID is not found

```
{
  "statusCode": 404,
  "errorCode": "clientNotFound",
  "message": "Client not found
}
```

Get Credit Card

Info: Get a specific creditCard for client by clientId
Method: GET
URI: /api/clients/{clientId}/creditCards/{creditCardId}
JCS URI: /cgidev2/JCD05008P.pgm?clientId={clientId}&cardId={cardId}

Success

Status Code: 200
Response:

```
{
  "id": 123,
  "name": "Joe Smith",
  "lastFour": "1111",
  "cardType": "visa",
  "expMonth": "12",
  "expYear": "2015"
}
```

Errors

Status Code: 404
Response Body:

The client ID was not found

```
{
  "statusCode": 404,
  "errorCode": "clientNotFound",
  "message": "Client not found
}
```

The creditCard ID was not found

```
{
  "statusCode": 404,
  "errorCode": "clientNotFound",
  "message": "Client not found
}
```

Create Credit Card

Method: POST
URI: /api/clients/{clientId}/creditCards
JCS URI: /cgidev2/JCD05008P.pgm?clientId={clientId}
Request:

```
{
  "name": "Joe Smith",                // required
  "card": "4111111111111111",        // required
```

```
    "expMonth": "12",                   // required
    "expYear": "2015",                  // required
    "cvv": "1111",                      // required
    "cardType": "visa"                  // optional
  }
```

Success

Status Code: 201
Response:

```
  {
    "creditCardId": <creditCardId>
  }
```

Errors

Status Code: 404
Response Body:

The client ID was not found

```
  {
    "statusCode": 404,
    "errorCode": "clientNotFound",
    "message": "Client not found
  }
```

Status Code: 400
Response Body:

```
  {
    "statusCode": 400,
    "errorCode": "creditCardError",
    "message": "Credit card not created"
  }
```

Update Credit Card

Method: PUT
URI: /api/clients/{clientId}/creditCards/{creditCardId}
JCS Method: POST
JCS URI: /cgidev2/JCD05008P.pgm?clientId={clientId}&cardId={cardId}
Request:

```
  {
    "name": "Joe Smith",                // required
    "card": "4111111111111111",        // required
    "expMonth": "12",                  // required
    "expYear": "2015",                 // required
    "cvv": "1111"                      // required
    "cardType": "visa"                 // optional
  }
```

Success

Status Code: 200
Response Body:

```
  {
    "id": 123,
    "name": "Joe Smith",
    "lastFour": "1111",
    "cardType": "visa",
    "expMonth": "12",
    "expYear": "2015"
  }
```

Errors

Status Code: 404
Response Body:

The client ID was not found

```
{
  "statusCode": 404,
  "errorCode": "clientNotFound",
  "message": "Client not found
}
```

The creditCard ID was not found

```
{
  "statusCode": 404,
  "errorCode": "creditCardNotFound",
  "message": "Credit card not found"
}
```

Delete Credit Card

Method: DELETE
URI: /api/clients/{clientId}/creditCards/{creditCardId}
JCS URI: /cgidev2/JCD05008P.pgm?clientId={clientId}&cardId={cardId}

Success

Status Code: 204

Errors

Status Code: 404
Response Body:

The client ID was not found

```
{
  "statusCode": 404,
  "errorCode": "clientNotFound",
  "message": "Client not found
}
```

The creditCard ID was not found

```
{
  "statusCode": 404,
  "errorCode": "creditCardNotFound",
  "message": "Credit card not found"
}
```

Create Order

Method: POST
URI: /api/orders
JCS URI: /cgidev2/JCD05020P.pgm
Info: Process a client order for the specified SKUs (including any kit selections)
Request:

```
{
  "firstName": "John",          // required
  "lastName": "Smith",          // required
  "clientId": 237654,           // required
  "consultantId": 11111,        // optional
  "language": "en_US",          // required
```

```
    "billingAddressId": 326754,  // required
    "shippingAddressId": 326755, // required
    "creditCardId": 74545,       // required
    "source": "fb",              // required
    "total": 102.67,             // required
    "products": [                // required
      {
        "sku": "25386",
        "qty": 2
      },
      {
        "sku": "19193",
        "qty": 1,
        "kitSelections": {
          "NUPL19193": [{        // each kit group
            "sku": "12026",      // product sku
            "qty": 1             // quantity
          }, {
            "sku": "12029",      // product sku
            "qty": 1             // quantity
          }]
        }
      }
    ]
  }
```

clientId - this field will be used to associate with a previously created client
products - list of products with optional configurations. TBD: include quantity or simply have multiple items?
creditCardId - the ID of the credit card to use, as loaded in the client data
consultantId - the ID of the consultant to attribute this order to
billingAddressId - the ID of the billing address to use, as loaded in the client data
shippingAddressId - the ID of the shipping address to use, as loaded in the client data
source - fb, pws
total - used to compare the total the user sees on the front-end with what JCS processes. The order should fail if the numbers do not match. This number is retrieved via the calculateTax method from JCS

Success

Status Code: 201
Response:

```
{
    "orderId": <orderId>
}
```

Errors

Status Code: 400
Info: Specified data was invalid and should be rejected by the server
Responses:

A client with clientId was not found.

```
{
    "statusCode": 400,
    "errorCode": "clientNotFound",
    "message": "Client not found"
}
```

A billing address with billingAddressId was not found.

```
{
    "statusCode": 400,
    "errorCode": "billingAddressNotFound",
    "message": "Billing Address not found"
}
```

A shipping address with shippingAddressId was not found.

```
{
    "statusCode": 400,
    "errorCode": "shippingAddressNotFound",
    "message": "Shipping Address not found"
}
```

A credit card with creditCardId was not found.

```
{
    "statusCode": 400,
    "errorCode": "creditCardNotFound",
    "message": "Credit Card not found"
}
```

A consultant with consultantId was not found.

```
{
    "statusCode": 400,
    "errorCode": "consultantNotFound",
    "message": "Consultant not found"
}
```

One of the products specified was not found.

```
{
    "statusCode": 400,
    "errorCode": "failedGetItems",
    "message": "Failed to get the Items"
}
```

One of the products specified is out of stock.

```
{
    "statusCode": 400,
    "errorCode": "errorValidateItemsInv",
    "message": "Error in Validate items and inventory"
}
```

There was an error while processing taxes, shipping & handling charges.

```
{
    "statusCode": 400,
    "errorCode": "errorTaxShipping",
    "message": "Error in Get Tax and Shipping and Handling"
}
```

The card specified by creditCardId was not found.

```
{
    "statusCode": 400,
    "errorCode": "errorNumberCard",
    "message": "Error in Get Number of card"
}
```

There was a merchant error while processing the order.

```
{
    "statusCode": 400,
    "errorCode": "requestElavonError",
    "message": "Error in Set variables to sending request to
Elavon"
}
```

There was an error with the payment method used. Invalid credit card data or unable to charge.

```
{
    "statusCode": 400,
    "errorCode": "applyPaymentError",
    "message": "Error in Apply Payment"
}
```

Calculate Tax & Shipping

Method: POST
URI : /api/shippingTax
JCS URI: /cgidev2/JCD05022P.pgm
Info: Calculate the tax and shipping
Request Body:

```
{
   "clientId":<clientId>,         //Required in CD OE
```

```
    "consultantId":<consultantId>, //Required
    "geocode" : <geocode>,          //Required
    "typeOrder": <typeOrder>,      //Required
    "source"   : <source>          //Required
    "products" : []                //Required

}
```

clientId: set clientId to zeros when typeOrder is online sponsoring
consultantId: set consultantId to zeros when typeOrder is online sponsoring
geocode: a geocode which matches the shipping address of the user (from StrikeIron)
typeOrder: 1414 = Online sponsoring, 1510 = Order entry
source: “P” = Online sponsoring, “Y” = Order entry
products - an array of sku/qty items and any relevant kitSelections

Success

Status Code: 200
Response:

```
{
  "SubTotal": 10.50,
  "SH": 5.00,
  "TaxRate": 10.50,
  "TotalBeforeTax": 15.50,
  "TaxAmount": 2.25,
  "Total": 17.75,
  "Carrier":"UPS"
}
```

Errors

Status Code: 404
Info: The specified product was not found.
Responses:

```
{
  "statusCode": 404,
  "errorCode": "taxNotFound",
  "message": "Tax and S&H not found"
}
```

Password Reset Request (LVI)

Method: GET
URI : /api/clients/passwordReset
Info: Generates a reset link sent to the user's email address in the form: {baseUrl}?email={email}&token={token}. See password reset sequence diagram for details. No more than one password reset can be requested within a 15 minute period.

Example: https://usa.jafra.com/reset?email=joe@jafra.com&token=a88hf87913g79f63g971g176381g6187

Request Body:

```
{
  "email": "joe@jafra.com",                    // the email
address of the user
  "language" : "en_US"                         // language for
reset email
}
```

email: the email address of the user
token: used to link from email -> db

Success

Status Code: 204
Response:

Errors

Status Code: 400
Info: The specified data was invalid
Responses:

The password didn't meet the server requirements (8 characters long).

```
{
  "statusCode": 400,
  "errorCode": "passwordResetInvalidData",
  "message": "Invalid data for password reset"
}
```

The email address was not found in the user database.

```
{
  "statusCode": 404,
  "errorCode": "clientEmailNotFound",
  "message": "Client email not found"
}
```

Password Reset Request (JCS)

JCS Method: POST
JCS URI: /cgidev2/JCD05009P.pgm
Info: Generates a reset link sent to the user's email address in the form: {baseUrl}?email={email}&token={token}. See password reset sequence diagram for details. No more than one password reset can be requested within a 15 minute period.

Example: https://usa.jafra.com/reset?email=joe@jafra.com&token=a88hf87913g79f63g971g176381g6187

Request Body:

```
{
  "email": "joe@jafra.com",
  "token": "a88hf87913g79f63g971g176381g6187",
  "url": "https://usa.jafra.com/reset",
  "language" : "en_US"
}
```

email: the email address of the user
token: used to link from email -> db
url: the URL to submit the token to
language: language for reset email

Success

Status Code: 204
Response:

```

```

Errors

Status Code: 400
Info: The specified data was invalid
Responses:

The password didn't meet the server requirements (8 characters long).

```
{
  "statusCode": 400,
  "errorCode": "passwordResetInvalidData",
  "message": "Invalid data for password reset"
}
```

The email address was not found in the user database.

```
{
  "statusCode": 404,
  "errorCode": "clientEmailNotFound",
  "message": "Client email not found"
}
```

Change Password (LVI)

Method: POST
URI : /api/clients/passwordReset
Info: Set the user password

Request Body:

```
{
  "email": "joe@jafra.com",
  "password": "somepassword",
  "token": "a88hf87913g79f63g971g176381g6187"
}
```

email: the email address of the user
password: the new password for the user
token: the token from the password reset request

Success

Status Code: 204
Response:

```

```

Errors

Status Code: 400
Info: The specified data was invalid
Responses:

The password didn’t meet the server requirements (8 characters long).

```
{
  "statusCode": 400,
  "errorCode": "invalidPassword",
  "message": "Invalid password"
}
```

The email address was not found in the user database.

```
{
  "statusCode": 404,
  "errorCode": ""clientEmailNotFound",
  "message": ""Client email not found"
}
```

Status Code: 500
Info: Internal error while processing password change.
Responses:

The specified token was expired.
{
“statusCode”: 500,
“errorCode”: “passwordResetTokenExpired”,
“message”: “Password Reset token expired”
}

Change Password (JCS)

Method: POST
URI : /api/clients/passwordReset
Info: Set the user password

Request Body:

```
{
  "email": "joe@jafra.com",
  "password": "somepassword"
}
```

email: the email address of the user
password: the new password for the user

Success

Status Code: 204
Response:

```
```

Errors

Status Code: 400
Info: The specified data was invalid
Responses:

The password didn't meet the server requirements (8 characters long).

```
{
  "statusCode": 400,
  "errorCode": "invalidPassword",
  "message": "Invalid password"
}
```

The email address was not found in the user database.

```
{
  "statusCode": 404,
  "errorCode": ""clientEmailNotFound",
  "message": ""Client email not found"
}
```

Get Inventory (JCS)

JCS Method: GET
JCS URI: /cgidev2/JCD05012P.pgm
Info: Get the current full inventory from JCS. This provides LVI with a way to determine current inventory levels and display out of stock messages in Client Direct.

Notes: the numbers returned have a built in reserved inventory deducted, which prevents frequent phone/order entry purchases from causing inventory availability issues at checkout. JCS is updated nightly by JDE and occasionally will be updated manually. LVI polls this data every 4 hours by default, but can force a sync as needed.

Success

Status Code: 200
Response:

```
{
    "inventory" : {
        "11335": 59,
        "55263": 9,
        "26463": 9,
        "98364": 9,
        "17365": 9
    },
    "lastUpdated": "1417759530"   // Unix time in seconds
}
```

Errors

Status Code: 500
Info: Specified data was invalid and should be rejected by the server
Responses:

An unknown error occurred while retrieving inventory.

```
{
  "statusCode": 500,
  "errorCode": "inventoryLookupFailed",
  "message": "Inventory lookup failed"
}
```

Get Inventory for Item (JCS)

JCS Method: GET
JCS URI: /cgidev2/JCD05021P.pgm?productId={productId}
Info: Get the current inventory for a specific product from JCS. This provides LVI with a way to determine current inventory levels and display out of stock messages in Client Direct.

Notes: the numbers returned have a built in reserved inventory deducted, which prevents frequent phone/order entry purchases from causing inventory availability issues at checkout. JCS is updated nightly by JDE and occasionally will be updated manually.

productId - the product ID of the product to get inventory levels for

Success

Status Code: 200
Response:

```
{
  "availableInventory": 10
}
```

Errors

Status Code: 404
Info: The specified product was not found.
Responses:

```
{
  "statusCode": 404,
  "errorCode": "productNotFound",
  "message": "Product not found"
}
```

Check Client/Consultant Email

Method: GET
URI : /api/consultants/{consultantId} or /api/clients/{clientId}
JCS URI: /cgidev2/JCD05011P.pgm?email={email}&type={type}
Info: Lookup consultant or client by email address

consultantId or ***clientId***: email address of the consultant or client, respectively
type: 1 = Consultant email, 2 = Client email

Request Body:

```
{
  "clientId": <clientId>
}

Or

{
  "consultantId": <consultantId>
}
```

email: the email address of the user
password: the new password for the user

Success

Status Code: 200
Response:

```
```

Errors

Status Code: 404
Info: The client or consultant was not found
Info:

Responses:

```
{
    "statusCode": 404,
    "errorCode": "emailNotFound",
    "message": "Email address not found"
}
```