Contents

[Overview 3](#_Toc311041986)

[Page Integrations 3](#_Toc311041987)

[Overview 3](#_Toc311041988)

[Server-Requested HTML 3](#_Toc311041989)

[IFrames 3](#_Toc311041990)

[Server-Requested Integrations 3](#_Toc311041991)

[Search Listings (Get Request) 3](#_Toc311041992)

[Overview 3](#_Toc311041993)

[Endpoint 4](#_Toc311041994)

[QueryString Parameters 4](#_Toc311041995)

[Required Css 4](#_Toc311041996)

[Required JavaScript 5](#_Toc311041997)

[Sample Implementation (.Net Sample) 5](#_Toc311041998)

[View Listing 5](#_Toc311041999)

[Overview 5](#_Toc311042000)

[Endpoint 6](#_Toc311042001)

[QueryString Parameters 6](#_Toc311042002)

[Required Css 6](#_Toc311042003)

[Required JavaScript 6](#_Toc311042004)

[Sample Implementation (.Net Sample) 7](#_Toc311042005)

[User’s Properties 7](#_Toc311042006)

[Overview 7](#_Toc311042007)

[Endpoint 7](#_Toc311042008)

[QueryString Parameters 7](#_Toc311042009)

[Successful Response 7](#_Toc311042010)

[No User Response 7](#_Toc311042011)

[Invalid Parameters Response 7](#_Toc311042012)

[Remarks 7](#_Toc311042013)

[User’s Saved Properties 8](#_Toc311042014)

[Overview 8](#_Toc311042015)

[Endpoint 8](#_Toc311042016)

[QueryString Parameters 8](#_Toc311042017)

[Successful Response 8](#_Toc311042018)

[No User Response 8](#_Toc311042019)

[Invalid Parameters Response 8](#_Toc311042020)

[Remarks 9](#_Toc311042021)

[IFrame-Requested Integrations 9](#_Toc311042022)

[Overview 9](#_Toc311042023)

[Create Listing 9](#_Toc311042024)

[Overview 9](#_Toc311042025)

[Endpoint 9](#_Toc311042026)

[QueryString Parameters 9](#_Toc311042027)

[Sample Implementation (Html Sample) 9](#_Toc311042028)

[Required JavaScript 9](#_Toc311042029)

[Authentication Integration 10](#_Toc311042030)

[User Authentication 10](#_Toc311042031)

[Overview 10](#_Toc311042032)

[Api Key 10](#_Toc311042033)

[Auth Token 10](#_Toc311042034)

[General Flow 10](#_Toc311042035)

[Getting an Auth Token for a User 10](#_Toc311042036)

[Authenticating a User 12](#_Toc311042037)

[Registering a New User 13](#_Toc311042038)

[Changing a User’s Password 14](#_Toc311042039)

[Connecting a KSL User Who Already Has a Rentler Account 15](#_Toc311042040)

[Other Helpful Integration Pieces 15](#_Toc311042041)

[List of API Status Codes 15](#_Toc311042042)

[Links 15](#_Toc311042043)

[My Listings (Portfolio) 15](#_Toc311042044)

[Interested In link 15](#_Toc311042045)

[Listing Counts (Get Request) 15](#_Toc311042046)

[Description 15](#_Toc311042047)

[Endpoint 15](#_Toc311042048)

[Sample Response 16](#_Toc311042049)

# Overview

## Page Integrations

### Overview

Displaying information on the KSL site from Rentler can be done in two ways: IFrames and server-requested Html. Both have their strengths and weaknesses, but we are going to be utilizing both for this integration so it is important to know the differences. For those familiar with IFrames and POST requests feel free to skip to the next section.

### Server-Requested HTML

This method is KSL requesting pieces of the Rentler site to place on the KSL site. It offers a completely custom way to style and display the page before presenting it on the KSL site. It is good for common elements where authentication is not that big of an issue, and customization is huge. The downside is that every page must be built and styled custom for the application.

### IFrames

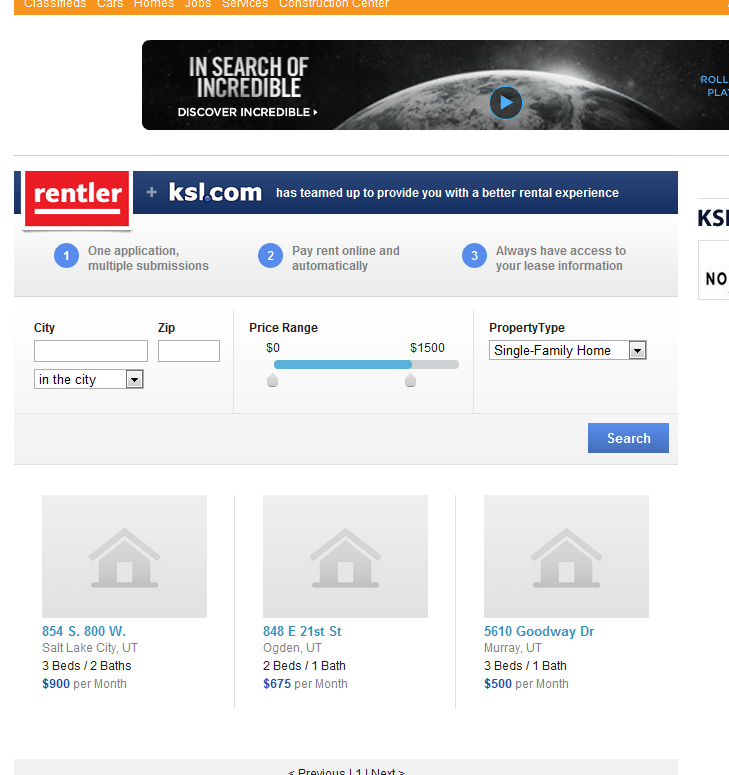
This method is where a slice of the Rentler website is placed directly into the KSL application via an IFrame. This is the easiest method to place information. The downside is that since the user is visiting both websites they must be authenticated on both websites at the same time.

# Server-Requested Integrations

## Search Listings (Get Request)

### Overview

This is the page where people come to search for the listings they are interested in. This search page will automatically post back to the page it is hosted on.



### Endpoint

<https://www.rentler.com/api/1/search.html>

### QueryString Parameters

* City – Optional to define a default city
* Zip – Optional to define a default zip code
* Miles – Optional to define the miles difference
* PriceLower – Optional to define the default lowest price
* PriceUpper – Optional to define the default top price
* PropertyType – Optional to define the default property type
* PriceUnlimited – Optional to define if the default price is unlimited
* PageSize – Optional Define the number of items per page
* Key – Required API Key

### Required Css

* <https://www.rentler.com/css/embed/ksl-rentler-integration.css>
* <https://www.rentler.com/css/embed/jquery-ui-1.8.6.custom.css>

### Required JavaScript

* <https://ajax.googleapis.com/ajax/libs/jquery/1.4.4/jquery.min.js> (or your favorite JQuery provider)
* <https://www.rentler.com/js/embed/jquery-ui-1.8.6.custom.min.js>
* <https://www.rentler.com/js/embed/ksl-rentler-search.js>

### Sample Implementation (.Net Sample)

// if there is no query passed in build one otherwise

// request the same querystring passed

string query = string.IsNullOrWhiteSpace(this.Request.Url.Query) ?

"?City=&Zip=&Miles=0&PriceLower=0" +

"&PriceUpper=1500&PropertyType=Any&PriceUnlimited=&PageSize=9" :

this.Request.Url.Query;

// add the query information

query += "&key={your api key}";

var url = host + "https://www.rentler.com/api/1/search.html" + query;

// get the listings and return our view

var client = new WebClient();

ViewBag.SearchHtml = client.DownloadString(url);

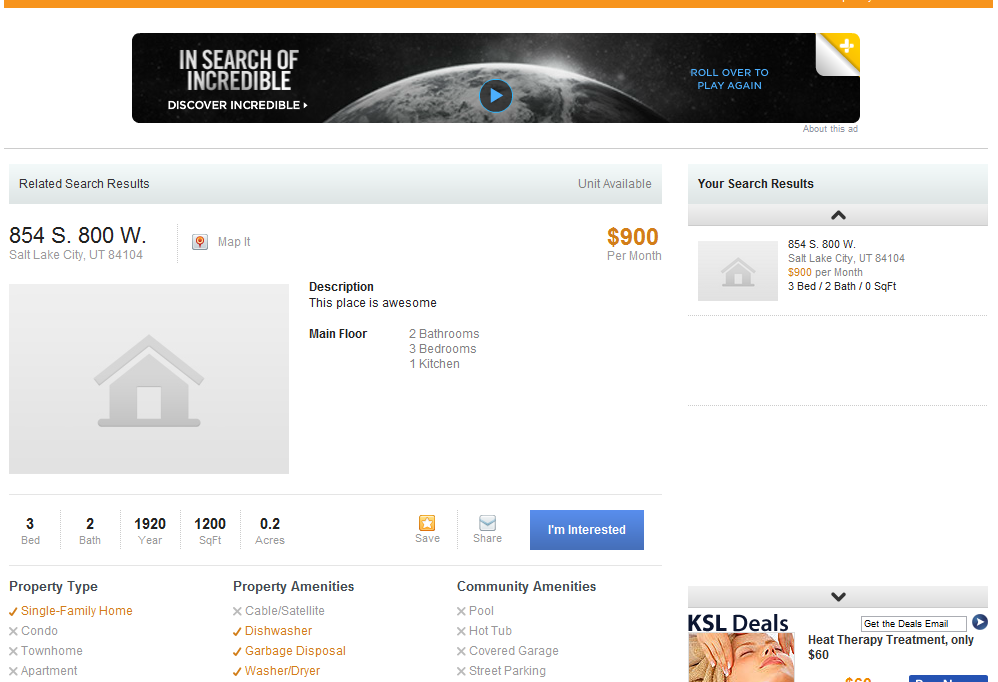
ViewBag.RentlerHost = host;

return View();

## View Listing

### Overview

This page allows users to view a listing. All outbound links go to the Rentler site.



### Endpoint

<https://www.rentler.com/api/1/listing.html>

### QueryString Parameters

* Id – Required id of the listing
* Key – Required API Key
* Email – Optional email address of the user

### Required Css

* <https://www.rentler.com/css/embed/ksl-rentler-integration.css>

### Required JavaScript

* <https://ajax.googleapis.com/ajax/libs/jquery/1.4.4/jquery.min.js> (or your favorite JQuery provider)
* <https://maps.google.com/maps/api/js?sensor=false>
* <https://s7.addthis.com/js/250/addthis_widget.js#pubid=ra-4e495ef7138737dd&domready=1>
* <https://ajax.microsoft.com/ajax/jquery.templates/beta1/jquery.tmpl.js>
* <https://www.rentler.com/js/cmn/lib/jcarousellite.js>
* <https://www.rentler.com/js/cmn/lib/modernizr-1.5.min.js>
* https://www.rentler.com/js/cmn/lib/jquery.expander.js
* <https://www.rentler.com/js/cmn/loader.js>
* <https://www.rentler.com/js/cmn/lib/plugins.js>
* <https://www.rentler.com/js/api/rentler.templates.js>
* <https://www.rentler.com/js/cmn/global.js>
* <https://www.rentler.com/js/embed/listing.js>

### Sample Implementation (.Net Sample)

var url = "https://www.rentler.com/api/1/listing.html?id={your api key}";

var client = new WebClient();

ViewBag.ListingHtml = client.DownloadString(url);

ViewBag.RentlerHost = host;

return View();

## User’s Properties

### Overview

When a user has created one or more properties to manage on Rentler.com, they can view them in their Rentler Portfolio. A Json service has been created for KSL to pull that user’s list of properties.

### Endpoint

[https://www.rentler.com/api/1/portfolio/properties.json?key={apiKey}&email={email of the user}](https://www.rentler.com/api/1/portfolio/properties.json?key=%7bapiKey%7d&email=%7bemail%20of%20the%20user%7d)

### QueryString Parameters

* Key – Required Api Key
* Email – Required email address of the user

### Successful Response

If the key and email are valid, a similar response to the following will be returned:

{"Success":true,"Message":"success","ErrorCode":0,"Result":[{"Latitude":"40.7694196","Longitude":"-111.9099254","HasTenant":false,"UserId":"A Guid of the User’s Rentler UserId","Address1":"650 W S Temple","Address2":"C208","City":"Salt Lake City","State":"UT","ZipCode":"84104","PrimaryPhoto":"A Guid of the Property’s Primary Photo, or null","PropertyInfoId":194769,"Id":194770}]}

### No User Response

The following will be returned if the user is not an authorized affiliate user:

|  |  |
| --- | --- |
|  | {"Success":false,"Message":"user not found","ErrorCode":100,"Result":false} |

### Invalid Parameters Response

Other errors can happen if the correct parameters above aren’t matched. See the “List of API Status Codes” for the full possible list.

### Remarks

This method does not support any format other than Json.

## User’s Saved Properties

### Overview

When a user has saved one or more properties they find while searching, they can view them in their Rentler Account. A Json service has been created for KSL to pull that list of saved properties.

### Endpoint

[https://www.rentler.com/api/1/account/savedproperties.json?key={apiKey}&email={email of the user}](https://www.rentler.com/api/1/account/savedproperties.json?key=%7bapiKey%7d&email=%7bemail%20of%20the%20user%7d)

### QueryString Parameters

* Key – Required Api Key
* Email – Required email address of the user

### Successful Response

If the key and email are valid, a similar response to the following will be returned:

{"Success":true,"Message":"success","ErrorCode":0,"Result":[{"House":{"Latitude":"40.753005","Longitude":"-111.8995624","HasTenant":false,"UserId":"8e102b24-659c-4913-b2f3-1874ad3aa379","Address1":"739 S. 300 W.","Address2":"","City":"Salt Lake City","State":"UT","ZipCode":"84101","PrimaryPhoto":null,"PropertyInfoId":194706,"Id":194707},"PropertyInfo":{"Bedrooms":2,"Bathrooms":2,"Type":"Apartment","SquareFootage":1050,"YearBuilt":0,"Acres":0,"Price":1435.0000,"IsAvailable":true,"DateAvailableUtc":"\/Date(1319278290163)\/","Deposit":null,"Id":194706},"Listing":{"IsActive":true,"Description":"2 MONTHS FREE ON A 12 MONTH LEASE! This special ends 09/09/11.BRAND NEW 2 & 3 BEDROOM APARTMENTS & PENTHOUSES. Located at 739 South and 300 West our Brand New Luxury apartments are full of upgrades including controlled access entry, garden tubs, upgraded appliances with designer kitchens and gas stoves. There are also hardwood floors, vaulted ceilings and fireplaces in select units. Fitness center, theater room, game room, lounge room, back patio wi...","Photos":null,"Building":{"Latitude":"40.753005","Longitude":"-111.8995624","HasTenant":false,"UserId":"8e102b24-659c-4913-b2f3-1874ad3aa379","Address1":"739 S. 300 W.","Address2":"","City":"Salt Lake City","State":"UT","ZipCode":"84101","PrimaryPhoto":null,"PropertyInfoId":194706,"Id":194707},"UserHasApplied":false,"SearchCount":59,"ViewCount":45,"Id":192649},"FullAddress":"739 S. 300 W. , Salt Lake City, UT 84101"}]}

### No User Response

The following will be returned if the user is not an authorized affiliate user:

|  |  |
| --- | --- |
|  | {"Success":false,"Message":"user not found","ErrorCode":100,"Result":false} |

### Invalid Parameters Response

Other errors can happen if the correct parameters above aren’t matched. See the “List of API Status Codes” for the full possible list.

### Remarks

This method does not support any format other than Json.

## Making a User’s Property Private or Public

### Overview

When a user has found a tenant for their property, or their property is vacant and they want to re-list the property, they need only activate or deactivate the listing. Two Json services allow for this functionality.

### Endpoints

Both of these endpoints are HTTP POST Requests.

<https://www.rentler.com/api/1/listing/activate>

<https://www.rentler.com/api/1/listing/deactivate>

### POST Request Values

* apiKey – Required Api Key
* email – Required email address of the user
* id – Required ListingId

### Successful Response

{"Success":true,"Message":"success","ErrorCode":0,"Result":true}

### Invalid Parameters Response

Other errors can happen if the correct parameters above aren’t matched. See the “List of API Status Codes” for the full possible list.

### Sample Implementation (.Net Sample)

var client = new WebClient();

var reqObj = new NameValueCollection();

reqObj["apiKey"] = "{your api key}";

reqObj["email"] = "dustin.dahl@gmail.com";

reqObj["id"] = "1939173";

string url = "http://localhost:15020/api/1/listing/activate";

Encoding encoding = Encoding.ASCII;

string response = encoding.GetString(client.UploadValues(url, reqObj));

### Remarks

This method uses an Api Key, and should only be called server side. Rentler recommends that this feature be promoted over deleting a property, as simply deactivating a property serves the purpose of removing properties from search results.

## Deleting a User’s Property

### Overview

When a user chooses to delete their property, KSL can use this Json REST service to remove it from Rentler.

### Endpoint

This endpoint is an HTTP Post Request.

[https://www.rentler.com/api/1/property/delete](https://www.rentler.com/api/1/property/delete?apiKey=%7bapiKey%7d&propertyInfoId%7bPropertyInfoId%20of%20the%20property%20to%20delete%7d&email=%7bemail%20of%20the%20user%7d)

### POST Request Values

* apiKey – Required Api Key
* email – Required email address of the user
* propertyInfoId – Required PropertyInfoId of the property to delete.

### Successful Response

If the Api Key, Email address, and PropertyInfoId are all valid, and the user found with the email address provided owns the property found, the following response will be returned:

{"Success":true,"Message":"success","ErrorCode":0,"Result":true}

### Invalid Parameters Response

Other errors can happen if the correct parameters above aren’t matched. See the “List of API Status Codes” for the full possible list.

### Sample Implementation (.Net Sample)

var client = new WebClient();

var reqObj = new NameValueCollection();

reqObj["apiKey"] = "{your apiKey}";

reqObj["email"] = "dustin.dahl@gmail.com";

reqObj["propertyInfoId"] = "1959820";

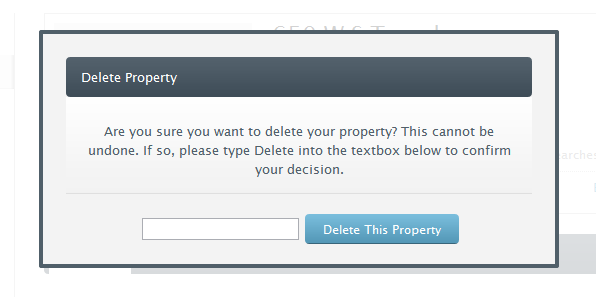
string url = "http://www.rentler.com/api/1/property/delete";

Encoding encoding = Encoding.ASCII;

string response = encoding.GetString(client.UploadValues(url, reqObj));

### Remarks

This method uses an Api Key, and should only be called server side. Rentler recommends the user be prompted to confirm their decision to delete a property, as it is usually an unnecessary action to take. For example, Rentler.com prompts the user with the following when they want to delete a property:



Where appropriate, the user should be encouraged to instead set their listing as *private*.

# IFrame-Requested Integrations

## Overview

All IFrame-Requested integrations require that the Authentication Integration procedures be followed. These procedures guarantee that the user is authenticated correctly on both KSL and Rentler before a page is presented to them.

## Create Listing

### Overview

Creating a listing requires Rentler authentication. See the authentication procedures to make sure that the user is authenticated before implementing creating of a listing.

### Endpoint

<https://www.rentler.com/api/1/property/index>

### QueryString Parameters

* BuildingId – Optional parameter for if the user is editing one of their listings. *Note: This is not the same as the listing id*
* Type – Optional parameter for creating a listing when you want to pass the property type so that the Basic Information form will automatically set the property type field

### Sample Implementation (Html Sample)

<iframe src="https://www.rentler.com/api/1/property/index?BuildingId=120" seamless

frameborder=0 border=0></iframe>

<iframe src="https://www.rentler.com/api/1/property/index?Type=Condo%2FTownhome" seamless

frameborder=0 border=0></iframe>

### Required JavaScript

* <https://www.rentler.com/js/embed/createlisting.js>

Once the user has finished the final step in creating a listing, the iframe needs to be able to redirect the browser to the listing. This is accomplished by createlisting.js. If the Post Message Api is supported by the browser, it will use that to redirect. Otherwise, it uses an alternative hash-value/polling method, to ensure support for older browsers.

# Authentication Integration

## User Authentication

### Overview

User authentication will be required for various sections of the site. Since Rentler does not have a list of all the KSL users this process must be created to make sure that users are authenticated correctly in the various sections.

### Api Key

The Api Key is the Key required by KSL to make a request to Rentler. This key is shared in private between the two companies.

### Auth Token

An Auth Token is generated by Rentler when a user needs to be authenticated. This Auth Token is verification that both companies have user information, and KSL is ready for them to be authenticated.

### General Flow

Figure 1: Basic Successful Authentication Flow



### Getting an Auth Token for a User

When a user is ready to be authenticated KSL will request the Auth Token for a user. If Rentler has the user information for the user the Auth Token will be passed back to KSL. This token can then be used for authentication.

Figure 2: Requesting Auth Token Workflow



#### Auth Token Endpoint

##### Auth Token Request Url (POST)

<https://www.rentler.com/api/1/auth/authtoken>

##### Request Format

POST Resquest with parameters:

* email: The required email address of the user to get the token for
* apiKey: The required api key given for access

##### Successful Response

The following is returned if the user is an authorized affiliate user. If we have that email address in our database the same response is returned, but when registration happens we will force the user to link their KSL and Rentler accounts:

{"Success":true,"Message":"","ErrorCode":0,"Result":{"AuthToken":"{The Auth Token for the User"}}

##### No User Response

The following is returned if the user is not an authorized affiliate user:

{"Success":false,"Message":"No User","ErrorCode":100,"Result":false}

##### Invalid Parameters Response

Other errors can happen if the correct parameters above aren’t matched. See the “List of API Status Codes” for the full possible list.

### Authenticating a User

Once an Auth Token is requested KSL can then redirect the user to our authentication section. If they are a KSL affiliate then we just authenticate them. If they aren’t a KSL affiliate, but they have a Rentler account we will have them connect their accounts.



#### Authentication Redirection Endpoint

##### Redirect Url (Get)

[https://www.rentler.com/account/auth?token={User’s](https://www.rentler.com/account/auth?token=%7bUser's) Auth Token}&returnUrl={where Rentler sends them back to}

##### Parameters

The parameters are passed via url:

* token: The required Auth Token that is delivered to KSL from Rentler for that user.
* returnUrl: This is required to point Rentler where the user is sent after Rentler authenticates them.

##### Response

There is no direct response. The user is redirected to the returnUrl.

### Registering a New User

If Rentler doesn’t have record of the user when KSL requests the user a new one can be created. Since there is no full user database to go off of there are a couple of scenarios that need to be accounted for.

Figure 3: Registering a User Implementation



#### Registering a User Endpoint

##### Registration Url (POST)

https://www.rentler.com/api/1/auth/createuser

##### Parameters

* apiKey: Required Api Key shared between Rentler and Ksl
* email: Required email address for the user
* passwordHash: Required password hash for the user
* firstName: Required first name for the user
* lastName: Required last name for the user
* username: Required username for the user

##### Success Response

If the user is created successfully the following response will be passed back:

{"Success":true,"Message":"","ErrorCode":0,"Result":{"AuthToken":"{The user’s Auth Token}"}}

##### Email Exists Response

If the email already exists the following response will be passed back:

{"Success":false,"Message":"Email Already Exists","ErrorCode":2,"Result":false}

##### Username Exists Response

If the username already exists the following response will be passed back:

{"Success":false,"Message":"Username Already Exists","ErrorCode":3,"Result":false}

##### Invalid Api Key Response

If the api key is invalid the following response will be passed back:

##### Invalid Parameters Response

Other errors can happen if the correct parameters above aren’t matched. See the “List of API Status Codes” for the full possible list.

### Changing a User’s Password

This method allows KSL to change the KSL password for a Rentler user. *Note: This will not affect the Rentler user’s Rentler password if they have one.* KSL will have to send Rentler password changes in order for the user’s to stay consistent. If there is no user attached, we will send back an error, but most likely this error can be ignored as Rentler will not have all the same users as KSL.

#### Changing a KSL User’s Password Endpoint

##### Change Password Url (POST)

<https://www.rentler.com/api/1/changepassword>

##### Parameters

* apiKey: The KSL API Key
* email: The email address of the user to change their password
* newPasswordHash: The new password hash for the user

##### Success Response

If everything works out fine Rentler will return:

##### {"Success":true,"Message":"success","ErrorCode":0,"Result":true}

##### Invalid Email Response

If we don’t have that email address on file or the user is not an affiliate of Rentler the server will return:

{"Success":false,"Message":"Invalid email","ErrorCode":2,"Result":false}

##### Invalid Parameters Response

Other errors can happen if the correct parameters above aren’t matched. See the “List of API Status Codes” for the full possible list.

### Connecting a KSL User Who Already Has a Rentler Account

As shown in the diagram Rentler handles the case where a KSL user has the same email address as the Rentler user. Ksl will request the Auth Token anyways. Rentler will bring the user to a page allowing them to connect the accounts together, and then Rentler passes the user back as if nothing happened.

# Other Helpful Integration Pieces

## List of API Status Codes

* 0 – Success
* 1 – Invalid Api Key
* 2 – Invalid email
* 3 – Invalid username
* 4 – Invalid password
* 5 – Invalid first name
* 6 – Invalid last name
* 7 – Invalid affiliate user key
* 8 – Invalid propertyInfoId
* 100 – No User
* 201 – Unknown Error

## Links

### My Listings (Portfolio)

<https://www.rentler.com/portfolio> - Links to the list of listings a user has created on Rentler

### Interested In link

[https://www.rentler.com/listing/{listingId}#interested](https://www.rentler.com/listing/%7blistingId%7d#interested) – Links to the rentler site listing page that will pop up the interested link.

## Listing Counts (Get Request)

### Description

This is a utility designed to help obtain the counts of Rentler’s listings. It returns a Json object with the types and the counts of all the Rentler listings.

### Endpoint

https://www.rentler.com/api/1/search/listingcount?key={your api key}

### Sample Response

{"SingleFamilyHome":0,"Apartment":101,"CondoTownhome":0,"Loft":0,"MultiFamilyHome":0,"ManufacturedHome":0,"HorseLivestock":0}