
Lemonade Partner API

Implementation Guide for Power Mortgage

Offload Insurance Process

Enhance your mortgage offer without building any infrastructure. Developers make simple requests to our service, and we handle the rest. No complicated setup required.





1. Features

One stop shop. Full support for everything related to Home Insurance.

→ **Quotes**

Create and store quotes for clients

→ **Payment**

Process payment immediately or let us handle it.

→ **Policies**

Create and store policies based on quotes

Why work with Lemonade instead our competitors?



Lemonade Ranks #1

US News ranks
Lemonade #1 in their list
of "Best Homeowners
Insurance Rating"

Ease of use and Documentation.



Technical

Code is on Github

Postman Collection

API Spec

Detailed Setup Guide

Endpoints available right now

Items Basic Table Operations, supports both quotes and policies

Readme: <https://github.com/Dev-Drew/LemonadeAPI/blob/master/README.md> ^

GET **/item{id}** Retrieve a quote or policy

PUT **/item{id}** Update an existing quote or policy

DELETE **/item{id}** Delete an existing quote or policy

Quote Everything to do with creating Quotes

GET **/quote** Return all quotes stored in table

POST **/quote** Creates a home insurance quote

Payment Payment processing and validation

Bundled with Mortgage or Pay with us through Stripe: <https://stripe.com/> ^

POST **/Payment** Creates a PaymentConfirmation given a valid PaymentInformation

Policy We've got you covered

GET **/Policy** Return all policies stored in table

POST **/Policy** Creates a Policy given a valid PaymentConfirmation

Create a Quote /quote

POST /quote Creates a home insurance quote

Each quote generates a unique ID and is saved in QuotesTable

Parameters

Cancel

Name	Description
body * required	Quote Input Object
object (body)	<div>Edit Value Model</div> <pre>{ "firstName": "Drew", "lastName": "Clark", "email": "dclark6783@aol.com", "birthDate": "2/1/1992", "address": "1234 5th Street, Charlotte NC", "propertySize": "599", "propertyYearBuilt": "1880" }</pre> <div>Edit</div>

Parameter content type

application/json

So, what's a quote?

Object that contains a unique id, premium / deductible, and client information.

```
{
  "id": "LP2729558958854",
  "createdTime": "2021-09-06T17:24:52.998Z",
  "lastUpdateTime": "2021-09-06T17:24:52.998Z",
  "premium": 500,
  "coverageType": "HOME",
  "QuoteDetails": {
    "status": "READY",
    "clientDetails": {
      "firstName": "Drew",
      "lastName": "Clark",
      "email": "dclark6783@aol.com",
      "birthDate": "2/1/1992",
      "address": "1234 5th Street, Charlotte NC",
      "propertySize": "599",
      "propertyYearBuilt": "1880"
    },
    "deductible": 2000,
    "effectiveDate": "2021-09-06T17:24:52.998Z"
  }
}
```


Process Payment /payment

Payment Payment processing and validation

Bundled with Mortgage or Pay with us through Stripe: <https://stripe.com/>

POST

/Payment Creates a PaymentConfirmation given a valid quoteId

If no mortgageId is provided, a stripe session will be created. Note: stripe session tokens are only valid for 24 hours..

Parameters

Try it out

Name

Description

body * required

Requires a PaymentInformation object. Includes the Id of a saved quote. Optional mortgageId

object

(body)

Example Value | Model

```
{
  "id": "LQ1630872416720",
  "mortgageId": "MID163084932900"
}
```

Parameter content type

application/json

How do you handle payments?

Stripe handles payment through credit card or Google Pay

```
{
  "id":
  "cs_test_a1G84XYRf1nMjj5qpMj7qBXtIwdj97FAZmwpjyY6NLAgadGz4EH4CnJpRS",
  "amount": 500,
  "quoteId": "LQ1630872416720",
  "success": true,
  "confirmationDate": "2021-09-06T01:37:35.243Z",
  "sessionExpirationDate": 1630978655
}
```

Request sample in Postman. Checkout documents folder on the repo

How to validate if payment was successful?

Stripe UI gives detailed information

Note to run on your local, you'll need a stripe token.

Payments

All

Succeeded

Refunded

Uncaptured

AMOUNT

DESCRIPTION

\$500.00 USD

Succeeded ✓

pi_3JWwBLvFtdPa1wm1aCyb3Vr

\$500.00 USD

Incomplete ⌚

pi_3JWVERLvFtdPa1wm0KVZl1j0

\$500.00 USD

Succeeded ✓

pi_3JWUxwLvFtdPa1wm1DoQYXZH

\$500.00 USD

Incomplete ⌚

pi_3JWUqZLvFtdPa1wm0AJWp7y

\$500.00 USD

Succeeded ✓

pi_3JWRcLLvFtdPa1wm1sK3uh9W

\$500.00 USD

Succeeded ✓

pi_3JWQ4ZLvFtdPa1wm0WoatCjv

\$500.00 USD

Canceled ✕

pi_3JWPtdLvFtdPa1wm0BjExiZR

Create a Policy

POST

/Policy Creates a Policy given a valid PaymentConfirmation



The ID contained in the payment information must be a valid mortgageID or stripe session token.

Parameters

Try it out

Name

Description

body * required

Requires a PaymentConfirmation object.

object

(body)

Example Value | Model

```
{
  "id": "cs_test_a1G84XYRf1nMjj5qpMj7qBXtIwdj97FAZmwpjyY6NLAgaDGz4EH4CnJpRS",
  "amount": 500,
  "quoteId": "LQ1630872416720",
  "success": true,
  "confirmationDate": "2021-09-06T01:37:35.243Z",
  "sessionExpirationDate": "2021-09-06T01:37:35.243Z"
}
```

Parameter content type

application/json



What's a policy look like?

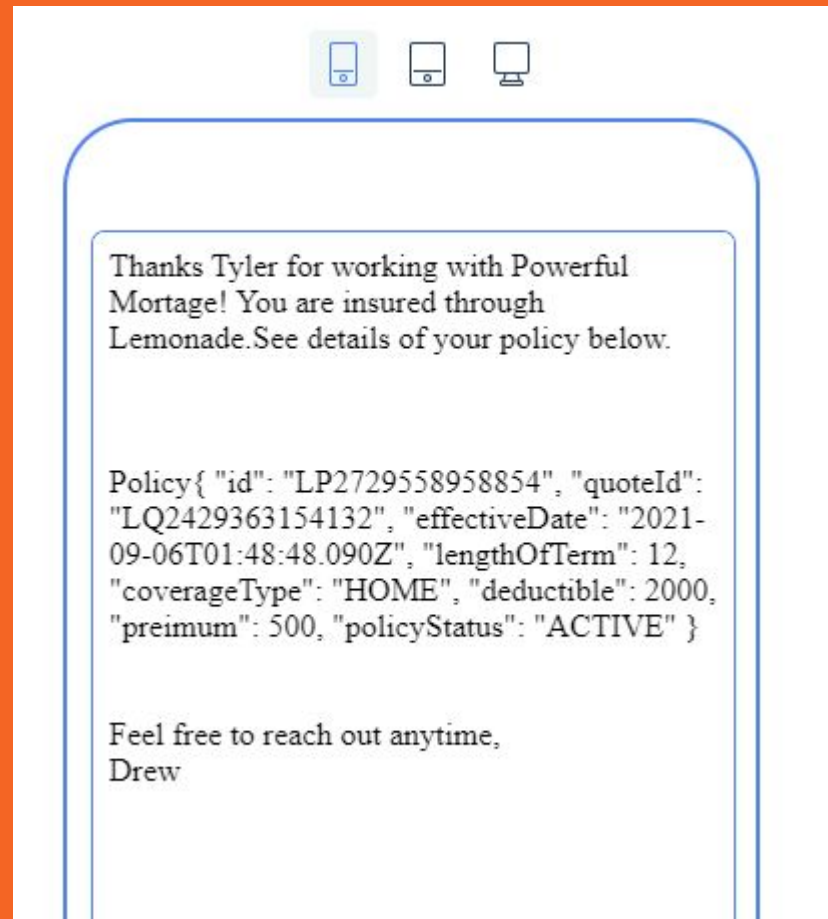
Stripe UI gives detailed information

```
{
  "id": "LP4769739024855",
  "quoteId": "LP4769739024855",
  "deductible": 500,
  "effectiveDate": "2021-09-06T01:38:40.884Z",
  "lengthOfTerm": 12,
  "coverageType": "HOME",
  "premium": 500,
  "policyStatus": "ACTIVE"
}
```

Note: You may only create a Policy for a Quote that is "READY"

How does a client know their policy was created?

Email automatically sent.



Partner API Flow



What if I just want to do basic operations?

We've got you covered

Note: All QuoteIDs start with "LQ", PolicyIDs start with "LP"

Items Basic Table Operations, supports both quotes and policies

GET

`/item{id}` Retrieve a quote or policy

PUT

`/item{id}` Update an existing quote or policy

DELETE

`/item{id}` Delete an existing quote or policy



Need all the data?

→ **GET /policy**

Returns all policy data stored in the Policy Dynamo Table.

→ **GET /quote**

Returns all quote data stored in the Quotes Dynamo Table.

Tech Stack

- **TypeScript:** Primary Language
 - **NodeJS:** Runtime
 - **NestJS:** NodeJS Framework
 - **NestJS-stripe:** Stripe Wrapper
 - **DyanmoDB:** NoSQL DB
 - **AWS SDK:** Access Tables
 - **Nodemailer:** Send Emails
 - **Open:** Open Links in Browser
 - **Postman:** Send HTTP
 - **DB Browser:** GUI for DB
-

Note: See full list in the README

Demo