WHITESPACE

GETTING STARTED WITH THE WHITESPACE API

Contents

Introduction

Whitespace launched the Whitespace Platform in August 2019. For the first time, insurance contracts agreed electronically in the London Market could be accessed, updated and processed via an Application Programming Interface (API). This opens up the possibility for integrating with other systems used by the platform's customers.

This document aims to explain how to use the API. It is intended for a technical audience who need to understand how to call the API, to support building integration(s). We will assume you have some understanding of the concepts behind calling a RESTful API, such as the use of JSON for input and output.

The Integration Process

Integrating with the Whitespace Platform typically follows a three-step process that involves becoming familiar with the platform and the API in our test environment, which is called Sandbox, before moving to the live platform, named Production. Sandbox and Production have identical functionality at all levels, but the organisations, users, and data held on Sandbox are all unreal. This allows for entirely safe experimentation with all user and integration systems. All integrative work should be extensively tested on Sandbox before being rolled out to Production.

Step	Purpose
Request Sandbox user accounts and an API service account for your organisation/s from support@whitespace.co.uk	To gain access to the platform UI, generate test data, and test API calls and solutions. Brokers can be given a dummy (Re)insurer organisation, and vice versa, to enable experimentation with the workflow. Dummy organisations are also available to third parties.
Request Sandbox service bus queue(s) from support@whitespace.co.uk	To gain access to the Azure service bus queues describing activity on your organisation's test account/s. Integrations are typically initiated from queues and so the triggering activities for your integration tasks need to be identified, and then integrations developed that work with a queue. These must be tested ahead of deployment into Production.
Request Production API service account(s) and queue(s) from support@whitespace.co.uk	To gain access to your organisation's live data and activity on the Production environment.

How We Demonstrate The API

The following section aims to reproduce the steps we would normally take in a live, person-to-person demonstration of the API system. As such, it should be considered a top-level overview of the functionality, rather than a set of steps that you specifically need to follow.

First, we check we have a connection by calling:

```
curl https://sandbox.whitespace.co.uk/api/version
```

This call does not require authentication, and so confirms that we are able to connect. It returns a block of information about the last update to the Whitespace Platform, similar to:

```
{
  "revapiCommit": "d8416ad12f200d4ae3ad9fa65256e7e429964817",
  "riskLibraryCommit": "7390b3c1263a9b1749a4ba69ff39cd0b7ea236bd",
  "revapiCommitMessage": "Merge pull request #402 from whitespace-software/almu/2.12.4\n\ninsert new stats doc
every minute",
  "buildDate": "2023-07-11@09:50:41"
}
```

Once we have confirmed that works, we use a token (see *Getting Your API Token* below) set in an environment variable to make the following call:

```
# on Unix/MacOS
curl -H "Authorization: Bearer $TOKEN" https://sandbox.whitespace.co.uk/api/user/myDetails
# on Windows
curl -H "Authorization: Bearer %TOKEN%" https://sandbox.whitespace.co.uk/api/user/myDetails
```

This returns information about the user that owns the token in the following format:

```
{"companyId":"PALERMO","uniqueID":"MUF8A41DCB-F66B-415B-B621-D9703124C130","companyName":"Palermo Insurance
Inc","role":"underwriter","username":"underwriter.palermo@wspt.co.uk","teams":[{"name":"All
Risks","teamId":"ALL","channel":"palermo_ALL"}]}
```

For display purposes, we format the JSON output to make it easier to read:

```
{
  "companyId": "PALERMO",
```

```
  "uniqueID": "MUF8A41DCB-F66B-415B-B621-D9703124C130",
  "companyName": "Palermo Insurance Inc",
  "role": "underwriter",
  "username": "underwriter.palermo@wspt.co.uk",
  "teams": [
    {
      "name": "All Risks",
      "teamId": "ALL",
      "channel": "palermo_ALL"
    }
  ]
}
```

In this set of examples, we are calling the API with the `curl` command which is available for both macOS/Unix and Windows. It is suitable for showing the low-level details of the interface. You may wish to use a tool such as Swagger (https://swagger.whitespace.co.uk/) to test the API, or to develop systems using Visual C# or JavaScript.

To show the output formatted with new lines and tabs, on macOS/Unix we use the jq utility. On Windows, we use "python -m json.tool" to re-format the output.

For the rest of these examples, we are omitting the -H token authorization header for reasons of brevity. Please note that without it, almost all calls to the API will be rejected. Similarly, we use the $ROOT environment variable (which would be %ROOT% on Windows) in place of the full server name. You will need to configure $ROOT for either the test environment or the live platform.

```
SANDBOX for testing: $ROOT = https://sandbox.whitespace.co.uk
LIVE PLATFORM: $ROOT = https://www.whitespaceplatform.com
```

To continue the demonstration we typically run the following command:

```
curl $ROOT/api/summary
```

With no parameters, this returns an array identifying the sixty most recently updated risks that the user can see. If there are no risks available, the result is simply:

```
[]
```

Otherwise, the result is of the form:

```
[
  {
        (risk1)
  },
  ...
  {
        (riskN)
  }
]
```

Unless the token's user account is quite new, the output will be too much to see on the screen. We redirect the formatted output to a text file, and then use a text editor to view it:

```
curl $ROOT/api/summary | jq . > risks.txt
```

Each risk array includes the unique identifier of that risk, a 38-digit reference which always starts with IC (for "Insurance Contract"). This identifier can be found as the `"rootID"` key. We set this as an environment variable named `IC` using "`export`" on macOS/Unix ("`export IC=ICB...77E`") and "`set`" on Windows ("`set IC=ICB...77E`"), so that we can use it in subsequent calls:

```
curl $ROOT/api/risks/root/$IC
```

This returns an array listing all the documents that are associated with that IC. Each one is identified in an `"id"` key-pair, and contains at least one `::` delimiter followed by further data. The document ID can then be used to view the document in full using the /api/risks/$ID call, or to perform other functions appropriate to the document type, such as writing a line or taking up a quote to firm order.

In our demonstrations, we use the formatted output of the /api/risks/$ID call to navigate to and display a simple contract heading and associated text, such as COUNTRY OF ORIGIN.


```
    {
      "nameVariation": "CountryOfOrigin",
      "fragments": [],
      "subItems": [],
      "originalHeading": "COUNTRY OF ORIGIN",
      "mrcHeading": "CountryOfOrigin",
      "originalSection": "",
      "mrcSection": "FiscalAndRegulatory",
      "elements": [
        {
          "text": "United States of America"
        }
      ],
      "placeholders": [],
      "index": 38
    },
```


See the section *Your First GET Calls* below for more details. API calls for working with tagged data are explained in the guide http://apidocs.whitespace.co.uk/Options_for_Extracting_Risk_Data.pdf.

Finally, we demonstrate using the /api/labels endpoint to set a new label on the risk, and then to check that it has been saved. This process is covered in *Your First POST Calls* below.

Having demonstrated reading and writing data, we typically discuss some of the various use cases for the API:

- Names clearance
- Sanctions checking
- Direct creation from a broking system
- Import to an underwriting system
- Clause checking
- Pricing

There is support for a very broad range of integrations with APIs on the Whitespace Platform.

Getting Your Sandbox API Token

The Whitespace Platform's test environment is called Sandbox. It allows new users and integrators to experiment with the platform and API without any consequence. All integrations should be tested robustly on Sandbox before deployment into the live Production server.

If you do not have any existing Sandbox users with the Administrator permission, the Whitespace Support Team (email `support@whitespace.co.uk`) can provide Sandbox user accounts. Please allow up to three working days.

For integrators, the best practice is to request a dedicated service account rather than using an API token attached to a specific user's account.

To obtain a service account, please contact the Support team with your request. You need to specify your organisation or, if you know it, your Whitespace COMPANYID, which can be retrieved from the admin portal by one of your internal platform administrators.

You also need to provide the email address of the person at your organisation that Whitespace should contact in case of issues or changes regarding the account. Finally, all service tokens have an expiry date. Please inform Support of the expiry date you prefer – the latest date available is 2099-12-31.

Whitespace Support will create a service account for your organisation using the format:

`wssvc_COMPANYID_contact.email@youremail.domain`

Note that 'wssvc' is an informative string confirming that this is a Whitespace Service account.

It is important to avoid changing the service account's details on the Platform via the admin portal without discussing the change with Support first. Doing so will prevent the service token from working. You will also need to make sure that the contact email address you select is monitored on your system, as we will need to send critical information to that account.

Service accounts do not require a platform log-in, but for individual users requiring a service token, please ensure that the user has been set up on Sandbox and set to 'Live' or 'ReadOnly'. Your internal platform administrators can do this for you via the admin portal.

When ready, email the Whitespace Integration Team or Support Team requesting an API service token, listing:

- That the token is for an individual user on the Sandbox environment.
- The email address of the user account.
- Your preferred token expiry date, up to 31st December 2099.
- If required, the IP address/es of the machines that you will call the API from.

We do not support SUMO (Single User, Multiple Organisation) service tokens. If your organisation has multiple corporate identities on the Whitespace platform, separate tokens will be required for each one. Ensuring that service token names contain the COMPANYID field helps keep token use purposes clear.

We will provide the token requested within 3 working days (GMT) in the form of a password-protected zip file containing a JSON file. The password will be sent in a following email flagged as confidential.

Service token files are JSON arrays containing three key-value pairs: `expiry`, `kid`, and `token`. 'Expiry' shows the date when the token becomes invalid. 'Kid' is a code required if the token is manually revoked. 'Token' is the data block to be passed to the API in the 'Authorization: Bearer' heading.

```
{
  "token": "eyJ0eX...7CG27g",
  "kid": "wssvc_WANTAGELTD_sally.adams@wantage.co.uk_1648767599",
  "expiry": "2025-01-01"
}
```

On receipt of your token file, good practice is to copy the token value out of the JSON file, not including the speech marks bracketing it, and store it on your system as the $TOKEN/%TOKEN% environment variable as discussed in *How We Demonstrate The API* above.

Alternatively, the following bash script extracts the token value out of a JSON file named wssvc_WANTAGELTD_sally.adams in the same directory and passes it to an API call requesting user details:

```
TOKEN=`jq -r .token wssvc_WANTAGELTD_sally.adams`
curl https://sandbox.whitespace.co.uk/api/user/myDetails -H "Authorization: Bearer $TOKEN"
```

If a token needs to be urgently revoked, please email support@whitespace.co.uk quoting either the "kid" value of the token to be revoked, or, to revoke all of that user's tokens, the associated email address.

Your service token may fail validation if the token is being passed from an IP address that is not on the token's whitelist, or the token has expired/been revoked.

In such cases, the call will return an HTTP code of "401 Unauthorized", and a JSON response of:

```
{
  "error": true,
  "reason": "Invalid WSAUTH"
}
```

Progressing to the Live Server

While the requirements for an API service token for the live production environment are very similar to those for Sandbox, there are some extra considerations to bear in mind.

We require you to specify the IP addresses that the service token will be used from. You may list several different IPs, or provide a reasonable range of addresses, but we will not issue service tokens for IPs ranging from 0.0.0.0 to 255.255.255.255. If it is not possible to confirm an IP range, please contact the Integration team to discuss alternatives to service tokens.

If the Whitespace user account associated with a service token is set to 'ReadOnly' via the admin portal's 'Users' panel, the token will not be able to change data on the platform. This is useful for ensuring that a system can safely read data off the platform without accidentally altering it. For full read/write API functionality, the user account must be set to 'Live'.

It's vitally important to keep live service tokens strictly private. Access to these files could allow a third party to operate the platform fraudulently – and legally bind your organisation into deals – using your credentials.

The naming convention for service accounts on the live platform is identical to that described in *Getting Your Sandbox API Token* above. Your organisation's Whitespace Administrators can set up correctly-named Live or ReadOnly users in the live Production environment for you using the `wssvc_COMPANYID_contact.email@youremail.domain` format. As for Sandbox, this email address needs to be monitored at your organisation.

To request a service token for the live platform, email the Whitespace Integration Team or Support Team listing:

- That the token is for the live Production environment.
- The associated email address and your organisation's name/COMPANYID.
- Your preferred token expiry date, up to 31st December 2099.
- The IP address/es of the machines that you will call the API from. Multiple single IPs or a reasonable range of IPs (ie. 103.89.10.34 – 103.89.21.34) are acceptable. Please note that no token will be issued for 0.0.0.0 – 255.255.255.255 or similar.

In circumstances where it is not possible to specify an IP range for a service token, we may be able to provide a manually-renewable short-term token. Please email `support@whitespace.co.uk` for further information if necessary.

We will provide the token requested within 3 working days (GMT) in the form of a password-protected zip file containing a JSON file as described above. The password will be sent in a following email, flagged as confidential.

Your First GET Calls

Try the following calls:

```
$ROOT/api/user/myDetails
$ROOT/api/user/colleagues
$ROOT/api/summary
```

Note that as discussed in *How We Demonstrate The API* above, calling $ROOT/api/summary without parameters lists the sixty most recent risks you can see. From that list, you can take the value of a "id" key pair as a document ID to be used in subsequent calls. A document id always starts IC, followed by a 36-character unique code, followed by various values separated by double colons.

Alternatively, you can get the risk IC number from the address bar of the browser, when you are looking at the full detail of the contract:

Using $IC to indicate the risk id to be used, try the following:

```
$ROOT/api/risks/$IC
$ROOT/api/risks/$IC/getExtendedMRC
$ROOT/api/risks/$IC/lineitem/Period
$ROOT/api/activities/$IC
$ROOT/api/export/pdf/$IC
$ROOT/api/attachments/$IC
```

Each of these provide different types of information relating to the document. A full discussion of each is beyond the scope of this guide, but further information can be found at https://swagger.whitespace.co.uk.

WHITESPACE

Your First POST Calls

Call /risks/$IC as discussed in *Your First GET Calls* above and filter the output for the “channels” array. Your team name is the value in that array that includes your organisation’s name.

```
$ROOT/api/risks/$IC | jq .channels
[
  "messina_ALL",
  "juwon_ALL"
]
```

Prepare the following JSON snippet in a file called label.json, replacing “messina_ALL” with your team name.

```
{
   "title": "January Renewal",
   "channel": "messina_ALL"
}
```

Send label.json to the /api/labels/$IC POST endpoint. You will need to set the Content-type header to be application/json.

```
export CTAJ="Content-type:application/json"
$ROOT/api/labels/$IC -X POST -H $CTAJ -T label.json
```

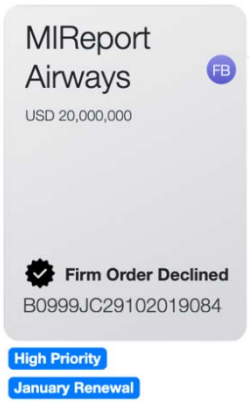
If all is well, the result will resemble:

```
{
  "rev": "1-8c20715ee8d47a180236249d5ef50389",
  "ok": true,
  "id": "ICB438...FAE5::FOD::MESSINA::ALL::LBL::MESSINA::ALL::January-Renewal::2022-11-26::13:53:34"
}
```

The /api/labels/$IC GET endpoint lists the labels on the risk, which should now include the new label:

```
[
  {
    "createdAt": "2022-11-26 13:53:34",
    "_rev": "1-8c20715ee8d47a180236249d5ef50389",
    "channels": [
      "messina_ALL"
    ],
    "teamID": "ALL",
    "rootID": "ICB438FF98-8D30-4E00-AE90-1A28CC4CFAE5::FOD::MESSINA::ALL",
    "_id": "ICB...E5::FOD::MESSINA::ALL::LBL::MESSINA::ALL::January-Renewal::2022-11-26::13:53:34",
    "companyID": "MESSINA",
    "title": "January Renewal"
  },
  {
    "createdAt": "2022-11-26 13:50:42",
    "_rev": "1-c566cd70ee6b70b8c00034987b744b4d",
    "channels": [
      "messina_ALL"
    ],
    "teamID": "ALL",
    "rootID": "ICB438FF98-8D30-4E00-AE90-1A28CC4CFAE5::FOD::MESSINA::ALL",
    "_id": "ICB4 ... E5::FOD::MESSINA::ALL::LBL::MESSINA::ALL::High-Priority::2022-11-26::13:50:42",
    "companyID": "MESSINA",
    "title": "High Priority"
  }
]
```

Within the platform, you will see the labels appear on the appropriate contract's Risk Overview screen, below the risk card:



We use labels as a safe introduction to POST calls. POST calls are designed to update data, and as such are to be used with extreme care. We recommend that you talk to Whitespace about your system requirements to ensure that you use POST calls correctly. Please Note that a number of steps in the system carry legal significance – such as writing a line with POST /risks/$ID/writeLine, or signing the contract with POST /risks/$ID/signSets – and require human interaction. They should therefore not be included in a fully automated process.

General Observations

We use GET endpoints to read data. In a few cases there are query string parameters as well, such as `/api/summary?from=2024-01-01`. Data is in JSON format, except when the endpoint exists only to return another format, such as XML or XLSX.

Endpoints that change data always use POST. In most cases there is further data to be sent, which is in JSON format required by the endpoint. We update https://swagger.whitespace.co.uk with examples of the input format. In some cases, the endpoint is just an instruction and needs no additional data. In this case we suggest passing {} as empty but valid JSON.

Errors are usually reported with a 400 status code. In general the API does not give feedback on the reason for the error to discourage malicious use. Some common mistakes are:

- using the wrong risk IC number – sometimes you need to specify the exact iteration, which comprises multiple parts separated by ::, while other occasions require the 'root' which has no :: separators.
- team IDs are uppercase (e.g. MARINE), while team channels are company name in lower case followed by underscore and team ID (priceforbes_MARINE). They cannot be used interchangeably.
- when a document revision is required it must be the most recent one; using a blank or a previously stored one will fail, as we need to keep data synchronised.

WHITESPACE

We're here to help

Email the Whitespace Integration Team (or Support team at support@whitespace.co.uk) including details of the endpoint you're trying to use and any JSON you're sending. We are committed to getting our customers and their suppliers up and running on our API.

Samples in macOS/Unix

In the following examples:

- $TOKEN has been set as covered in the *Getting Your API Token* section above
- $ROOT is https://sandbox.whitespace.co.uk
- $CTAJ is Content-type:application/json
- $IC, $LABELID are set to suitable values during the process

We regularly use the jq utility to format the JSON output:

```
curl -H "Authorization: Bearer $TOKEN" $ROOT/api/user/myDetails # displays:

{"companyId":"AJC","uniqueID":"MUD38EC011-780A-42D3-94DA-FD9063F5DAF9","companyName":"AJC Broking
Ltd","role":"broker","username":"broker.ajc@wspt.co.uk","teams":[{"name":"All
Risks","teamId":"ALL","channel":"ajc_ALL"}]}

curl $ROOT/api/user/myDetails | jq . # displays:

{
  "uniqueID": "MUD38EC011-780A-42D3-94DA-FD9063F5DAF9",
  "companyName": "AJC Broking Ltd",
  "teams": [
    {
      "name": "All Risks",
      "channel": "ajc_ALL",
      "teamId": "ALL"
    }
  ],
  "companyId": "AJC",
  "role": "broker",
  "username": "broker.ajc@wspt.co.uk"
}
```

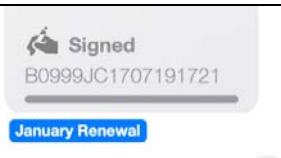
A good example of reading and writing data is in adding a label to a risk.

```
# set the IC variable to be the root of a risk (i.e. stopping before ::_
export IC=ICB0635326-CD0B-441F-A7B3-BD6FA881C77E

# List the labels on that risk. In this case, an empty array:
curl -H "Authorization: Bearer $TOKEN" $ROOT/api/labels/$IC | jq .
[]

# create suitable input JSON file
cat labeladd.json
{
    "title": "High Priority",
    "channel": "palermo_ALL"
}

# POST that JSON file; CTAJ is Content-type:application/json
curl -H "Authorization: Bearer $TOKEN" $ROOT/api/labels/$IC -X POST -H $CTAJ -T labeladd.json
{
    "ok": true,
    "rev": "1-f2030e315d1d831ae1b3c715ca1346a6",
    "id": "ICB0635326-CD0B-441F-A7B3-BD6FA881C77E::LBL::..."
}

# List the labels on that risk. Now one item.
```

```
curl -H "Authorization: Bearer $TOKEN" $ROOT/api/labels/$IC | jq .
[
  {
    "createdAt": "2022-11-16 10:04:25",
    "teamID": "ALL",
    "companyID": "PALERMO",
    "channels": [
      "palermo_ALL"
    ],
    "rootID": "ICB0635326-CD0B-441F-A7B3-BD6FA881C77E",
    "_rev": "1-f2030e315d1d831ae1b3c715ca1346a6",
    "_id": "ICB0635326-CD0B-441F-A7B3-BD6FA881C77E::LBL::PALERMO::ALL::High-Priority::2022-11-16::10:04:25",
    "title": "High Priority"
  }
]

# Use jq to modify the JSON file
cat labeladd.json | jq '.title="January Renewal"' > label2.json ; cat label2.json
{
  "title": "January Renewal",
  "channel": "palermo_ALL"
}

# use that as a second label
curl -H "Authorization: Bearer $TOKEN" $ROOT/api/labels/$IC -X POST -H $CTAJ -T label2.json | jq .
{
  "id": "ICB0635326-CD0B-441F-A7B3-BD6FA881C77E::LBL::PALERMO::ALL::...",
  "rev": "1-8d3926fb4ad889e535878c614891122e",
  "ok": true
}

# List the labels on that risk. Now two items.
curl -H "Authorization: Bearer $TOKEN" $ROOT/api/labels/$IC | jq .
[
  {
    "createdAt": "2022-11-16 10:04:25",
    "teamID": "ALL",
    "companyID": "PALERMO",
    "channels": [
      "palermo_ALL"
    ],
    "rootID": "ICB0635326-CD0B-441F-A7B3-BD6FA881C77E",
    "_rev": "1-f2030e315d1d831ae1b3c715ca1346a6",
    "_id": "ICB0635326-CD0B-441F-A7B3-BD6FA881C77E::LBL::PALERMO::ALL::...",
    "title": "High Priority"
  },
  {
    "createdAt": "2022-11-16 10:08:57",
    "teamID": "ALL",
    "companyID": "PALERMO",
    "channels": [
      "palermo_ALL"
    ],
    "rootID": "ICB0635326-CD0B-441F-A7B3-BD6FA881C77E",
    "_rev": "1-8d3926fb4ad889e535878c614891122e",
    "_id": "ICB0635326-CD0B-441F-A7B3-BD6FA881C77E::LBL::PALERMO::ALL::...",
    "title": "January Renewal"
  }
]

# use jq to reduce the output to just the text of the labels
curl -H "Authorization: Bearer $TOKEN" $ROOT/api/labels/$IC | jq -r .[].title
High Priority
January Renewal

# set LABELID from one of the items listed
export LABELID=ICB0635326-CD0B-441F-A7B3-BD6FA881C77E::LBL::PALERMO::ALL::...

# call the label delete API
curl -H "Authorization: Bearer $TOKEN" $ROOT/api/labels/$LABELID -X DELETE
{
    "id": "ICB0635326-CD0B-441F-A7B3-BD6FA881C77E::LBL::PALERMO::ALL::High-Priority::2022-11-16::10:04:25",
    "ok": true,
```

```
    "rev": "2-d64582eed315ae670f4745cc52fae351"
}

# list the labels again : there should be only one
curl -H "Authorization: Bearer $TOKEN" $ROOT/api/labels/$IC | jq .
[
  {
    "_id": "ICB0635326-CD0B-441F-A7B3-BD6FA881C77E::LBL::PALERMO::ALL::January-Renewal::2022-11-16::10:08:57",
    "rootID": "ICB0635326-CD0B-441F-A7B3-BD6FA881C77E",
    "title": "January Renewal",
    "teamID": "ALL",
    "_rev": "1-8d3926fb4ad889e535878c614891122e",
    "companyID": "PALERMO",
    "createdAt": "2022-11-16 10:08:57",
    "channels": [
      "palermo_ALL"
    ]
  }
]
```

The effect of these changes can be seen on the browser or iPad:

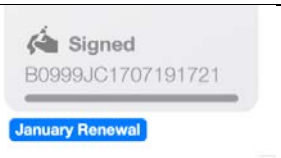


Samples in Windows Command Line

The following example shows typical Windows versions of the commands listed above in the *Samples in macOS/Unix* section. For sake of brevity, the detailed JSON output of the commands is not duplicated below.

```
set ROOT=https://sandbox.whitespace.co.uk
set TOKEN= eyJ...QuQ

curl -H "Authorization: Bearer %TOKEN%" %ROOT%/api/user/myDetails
curl -H "Authorization: Bearer %TOKEN%" %ROOT%/api/summary

set IC=IC0DAABA73-05F0-418A-82E8-B1F41C5F0269

curl -H "Authorization: Bearer %TOKEN%" %ROOT%/api/risks/%IC%

notepad labeladd.json
    {
        "title" : "January Renewal",
        "channel" : "ajc_ALL"
    }

curl -H "Authorization: Bearer %TOKEN%" %ROOT%/api/labels/%IC%
curl -H "Authorization: Bearer %TOKEN%" %ROOT%/api/labels/%IC% -X POST -H %CTAJ% -T labeladd.json
curl -H "Authorization: Bearer %TOKEN%" %ROOT%/api/labels/%IC%
```

The effect of these changes can be seen on the browser or iPad:



Jonathan Clarke
18 July 2023