# Go Planning

**A windows application that scrapes planning application data from UK/Scotlish councils.**

[Go Planning](#header-n0)  
 [Introduction](#header-n8)  
 [Quick Start](#header-n11)  
 [The Process](#header-n16)  
 [Step 1](#header-n19)  
 [Step 2](#header-n31)  
 [Step 3](#header-n36)  
 [step 4](#header-n42)  
 [Modes](#header-n115)  
 [Start](#header-n117)  
 [Restart](#header-n119)  
 [Options](#header-n122)  
 [goplanning start...](#header-n124)  
 [goplanning restart...](#header-n164)  
 [Supporting files](#header-n192)  
 [Config.json](#header-n193)  
 [Reset:](#header-n196)  
 [File locations:](#header-n199)  
 [Puppeteer](#header-n212)  
 [Property Settings](#header-n226)  
 [Logs](#header-n230)  
 [Note:](#header-n251)  
 [Console](#header-n253)  
 [Council-planning.json](#header-n257)  
 [Caveats](#header-n263)  
 [Only Address and URL returned](#header-n265)

## Introduction

The **goplanning** application allows a user to collected planning application data from a council's planning portal automatically. The user will select a type of process (*start*, initially) followed by a council identifier (Name or ID), optionally the user can also determine a start and finish letter for the applicaiton to scrape between.

## Quick Start

If you need to get an application process up and running, it's easy to start with this command. At your command promt (ensuring your in the right directory), type the following:

**goplanning start -c=0** or, to add the full list of defaulted options **goplanning start -c=0 -s=A -f=Z -r=false -t=true**

The above (left) command will start goplanning with the first council object.url it finds in the *./settings/council-planning.json* support file, also using the default options. The default options are highlighted in the above (right) command example. **-s=**{startLetter} **-f=**{finishLetter} **-r=**{reset boolean} **-t=**{typhon logging}. All of the default values can be omitted, meaning that the only option required is the council identifier.

## The Process

The entire application can be broken into 4 stages or steps; these steps define how far goplanning as gone into the planning portal in order to start/continue retrieving planning application data.

As covered above and after inputting a series of commands to initiate a process, there are four main steps that take place:

### Step 1

Concentrating on command line options, the delta between the startLetter and finishLetter is determined. Usually 26 letters and depending on the council (designated street naming) the user may find a few letters missing from the alphabet, usually X and/or Y, maybe Z too.   
There is a little user feedback to help troubleshoot something that might not be correct in the users opinion.

**Time**: 1-2mins

Each alphabrt URL is checked to ensure the page(s) are valid for the next step. At this point data is saved to the current working folder under the chosen ***lettersFile*** (accepted-letters.json) name. This is an array with a max length of 26.

**Example**:  
The output for the lettersFile is an array that currently looks like:

[..."https://publicaccess.chesterfield.gov.uk/online-applications/search.do?action=property&type=atoz&letter=X","https://publicaccess.chesterfield.gov.uk/online-applications/search.do?action=property&type=atoz&letter=Y","https://publicaccess.chesterfield.gov.uk/online-applications/search.do?action=property&type=atoz&letter=Z"]

**Time**: 15mins±

The returned data from *lettersFile* is further processed, from this all avaiable streets in the council district can be read and saved to file. This saved file ***allStreetsFile*** (accessible-streets.json) lists all the possible street URL's that can be read from the planning portal.

**Example**:  
The output for allStreetsFile has objects that currently look like:

[  
 ...  
 [  
 {  
 "street" : "Lace Lane , Buckingham , Buckinghamshire",  
 "link" : "https://publicaccess.aylesburyvaledc.gov.uk/online-applications/alphabeticalSearchResults.do?streetKeyValue=M5G6XBCL04A01&action=firstPage&streetLetter=L"  
 },  
 {  
 "street" : "Lacemakers , Long Crendon , Buckinghamshire",  
 "link" : "https://publicaccess.aylesburyvaledc.gov.uk/online-applications/alphabeticalSearchResults.do?streetKeyValue=0000H7CLSD000&action=firstPage&streetLetter=L"  
 },  
 ...  
 ]  
 ...  
]

### Step 2

**Time**: 10sec±

This step happens in the blink of an eye and probably doesn't warrent its own step. Though what it does is strip out all the street URL's from the previously created *allStreetsFile's* objects. This data is then re-stored in a new file named ***workableStreets*** (usable-street-urls.json) and will be referenced if a *properties restart* is required.

**Example**:  
The output for the workableStreets array currently looks like:

["https://planning.basildon.gov.uk/online-applications/alphabeticalSearchResults.do?streetKeyValue=0000JOCQSD000&action=firstPage&streetLetter=L","https://planning.basildon.gov.uk/online-applications/alphabeticalSearchResults.do?streetKeyValue=O7N01VCQ00301&action=firstPage&streetLetter=L","https://planning.basildon.gov.uk/online-applications/alphabeticalSearchResults.do?streetKeyValue=00017SCQSD000&action=firstPage&streetLetter=L"...]

### Step 3

**Time**: 17hours±,   
**Note**: restart can occur from this step.

The *workableStreets* gets chunked into array's of (max) 10 and processed async with each property on a street being saved to the ***propertiesFile*** (properties.json). One of the reasons this step starts to take exponentially longer is due to the heavy pagination incorporated in the Idox system.

**Example**:  
The output for a given street in the properties object currently looks like:

[  
 [  
 {  
 "property" : "1 Amber Crescent Walton Derbyshire S40 3DH",  
 "link" : "https://publicaccess.chesterfield.gov.uk/online-applications/propertyDetails.do?keyVal=000QVKEPLI000&activeTab=summary"  
 },  
 {  
 "property" : "2 Amber Crescent Walton Derbyshire S40 3DH",  
 "link" : "https://publicaccess.chesterfield.gov.uk/online-applications/propertyDetails.do?keyVal=000CHAEPLI000&activeTab=summary"  
 },  
 ...  
 ]  
 ...  
 ]

This information will be part of (future) cross-referencing that pertains to *Related Cases*.

### step 4

**Time**: 5days±,   
**Note**: restart can occur from this step.

The final step of the process ***propertyFile*** (property-data.json) is created when address URL's are fed into the sub-process from the *propertiesFile*. For each address URL data is gathered relating to a specific planning application at that address, and includes information such as:

* Summary:
  + Application URL.
  + A unique planning application reference (used to retrieve related documents on *some* Idox systems).
  + Relevant dates.
  + Property address.
  + Proposal summary.
  + Status.
  + Decision & date.
  + Appeal status & date.
* Important dates:
  + Application received & validated date.
  + Expiry data.
  + Committee, neighbour & standard consultation dates.
  + Press & site dates.
  + Agreed, statutory & decision dates.
  + Environmental impact assessment.
  + Determination deadline & temporary permission expity date.
* Contact details:
  + Agents
  + Councillors
  + Contractors
  + Others
* Additional information:
  + Application type.
  + Decision level.
  + Case officer.
  + Parish & ward.
  + Applicant.
  + Agent & company.
  + Environmental assessment (boolean).

*In addition to this,* ***future updates*** *will include document urls, related cases and constraint information*

**Note**:  
If any of the above (scraped) fields are a null state they can be omitted from future scrapes by adding the nullified question responce (ie, NA, unknown, not stated...etc) to the config.json propertySetting.uselessValuePairs propertoes key/value array.\*

**Example**:  
The output for a given planning application currently looks like:

[  
 ...  
 [  
 ...  
 {  
 "address" : "3 Church View Campsall Doncaster DN6 9RA",  
 "application" : {  
 "link" : "https://planning.doncaster.gov.uk/online-applications/applicationDetails.do?previousCaseType=Property&keyVal=ZZZYXMFXTS126&activeTab=summary",  
 "Reference" : "94/0869/P",  
 "Application Received" : "Wed 23 Mar 1994",  
 "Application Validated" : "Mon 28 Mar 1994",  
 "Address" : "3 Church View Campsall Doncaster South Yorkshire DN6 9RA",  
 "Proposal" : "ERECTION OF GROUND FLOOR PITCHED ROOF DINING ROOM EXTENSION (4.8M X 4.2M) TO REAR OF DETACHED BUNGALOW",  
 "Status" : "Decided",  
 "Decision" : "Planning Permission GRANTED",  
 "Decision Issued Date" : "Mon 09 May 1994",  
 "Application Type" : "Full Application",  
 "Case Officer" : "Diane Adams",  
 "Parish" : "Norton Parish Council",  
 "Ward" : "(Historic) Askern Spa",  
 "Applicant Name" : "A WOODHOUSE",  
 "Agent Name" : "A KIDGER",  
 "Agent Address" : "4, HAWKE CLOSE, NORTON, DONCASTER, DN6 9PG",  
 "Environmental Assessment Requested" : "No",  
 "Application Received Date" : "Wed 23 Mar 1994",  
 "Application Validated Date" : "Mon 28 Mar 1994",  
 "Statutory Expiry Date" : "Mon 23 May 1994",  
 "Decision Made Date" : "Mon 09 May 1994",  
 "Decision Printed Date" : "Mon 09 May 1994",  
 "Determination Deadline" : "Mon 23 May 1994"  
 }  
 ...  
 }  
 ...  
 ]  
 ...  
]

## Modes

There are two modes the user can choose from; either **start** or **restart**.

### Start

This mode will start an applicaiton process from the beginning of the **options** that have been given to the CLI.

### Restart

This mode will restart an application process from where it exited, providing the process reached step 3 or 4. The logs are scanned for the last complete entry, it is then determined whether step 3 or step 4 should be restarted, based on the scanned data. A temporary file is created and exerything before the last complete entry is removed. The step is then restarted with an array of data to work-through.

## Options

In either mode the only required option is **council** and a little caution will go a long way here. It is vital that you are aware of the council you want to interact with otherwise you stand the **risk of erasing any progress** you have made already!!!!

### goplanning start...

The user is required to supply at least 1 option when using the start command **-c=** all other options are set to their default.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Option | Shorthand | Longhand | Default | Notes |
| Council | -c=string/integer | -council="string/integer" | NONE<br />**Must include!** | You may not type ***the council name*** exactly, so goplanning will filter out similar councils relating to the name you have typed. <br />If you prefer to use numbers (to keep track of your progress), it would be easier to start your processes sequentially, 00, 01, 02... etc. |
| Start Letter | -s=g | -startLetter="g" | A | The ***start letter*** you choose to start your process with, both upper and lowercase is fine. This can be any letter, though some council's are very sparse with Y and X. |
| Finish Letter | -f=S | -finishLetter="S" | Z | The ***finish letter*** you choose to stop the process after. The process will check the entirity of the letter, finishing thereafter.<br />You cannot select a letter that ~~preceeds~~ the start letter. |
| Reset | -r=true | -reset="true" | FALSE | If you include a true value for ***reset***, the referenced council working folder will be **wiped clean** and "reset"!! Not something you want to do if you are near the end of a council process. |
| Typhon | -t=false | -typhon="false" | TRUE | ***Typhon*** logging is active by default, which enabled you (as a Typhon user) the ability to keep tabs on the progress of each of your running processes. By visiting [Typhon UI](https://typhon.sst-l.com) you can create a namespace (goplanning) and a name based on the council folder that has been created, otherwise known as the shortName as found in the *./settings/council-planning.json* support file. Once you are observing the live streram of logs coming in from your process, the last log you will see is the "Job Done" info notice. Alternatively, your log stream will just run dry, which proves that the process has errors-out or completed successfully. |

### goplanning restart...

The user is required to supply at least 1 option when using the restart command **-c=** all other options are set to their default.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Option | Shorthand | Longhand | Default | Notes |
| Council | -c=string/integer | -council="string/integer" | NONE<br />**Must include!** | You may not type ***the council name*** exactly, so goplanning will filter out similar councils relating to the name you have typed. <br />If you prefer to use numbers (to keep track of your progress), it would be easier to start your processes sequentially, 00, 01, 02... etc. |
| Reset | -r=true | -reset="true" | FALSE | If you include a true value for ***reset***, the referenced council working folder will be **wiped clean** and "reset"!! Not something you want to do if you are near the end of a council process. |
| Typhon | -t=false | -typhon="false" | TRUE | ***Typhon*** logging is active by default, which enabled you (as a Typhon user) the ability to keep tabs on the progress of each of your running processes. By visiting [Typhon UI](https://typhon.sst-l.com) you can create a namespace (goplanning) and a name based on the council folder that has been created, otherwise known as the shortName as found in the *./settings/council-planning.json* support file. Once you are observing the live streram of logs coming in from your process, the last log you will see is the "Job Done" info notice. Alternatively, your log stream will just run dry, which proves that the process has errors-out or completed successfully. |

## Supporting files

### Config.json

Location: *./settings/config.json*

The config file allows for some of the more common vaiables to be updated/changed as the user requires and to help with a more streamlined process. The file is (losely speaking) broken into five areas; the information (here) of which will be looked at in that manner, as opposed to linear through the json file.

#### Reset:

"resetOnStart": false,

This option is automatically updated through the command line option **-r=**, by default this option is set to false and will revert back to false unless the user specically states a true value at the command line.

#### File locations:

...  
"fileLocations": {  
 "outputFolder": "basildon",  
 "logsFile": "logs.txt",  
 "lettersFile": "accepted-letters.json",  
 "allStreetsFile": "accessible-streets.json",  
 "workableStreets": "usable-street-urls.json",  
 "propertiesFile": "properties.json",  
 "propertyFile": "property-data.json"  
 }  
 ...

The **outputFolder** automatically updates as a result of the user inputting an option at the command line. Therefore, manually changing this will have no effect on a new or restarted process.

All other file locations are interchangeable by the user. They are also listed in order of creation, the following list pertains to the steps of the process: ie, 1 = step 1, 2 = step 2...

1. lettersFile, allStreetsFile,
2. workableStreets,
3. propertiesFile,
4. propertyFile

#### Puppeteer

"puppeteer": {  
 "path": "chromiumPath",  
 "product": "chrome",  
 "headless": true,  
 "ignoreHTTPSErrors": false,  
 "slowMo": 50,  
 "selectorTimeout": "50000",  
 "waitUntil": "domcontentloaded",  
 "timeout": 0  
 },

The *path* option is predetermined when goplanning is ran, taking a *process.platform* reading and using the correct puppeteer instance. The *product* can work under Firefox, though as this is experimental and unnecessary the default chrome should be left alone. They are included as options for a time where chrome is unviable.

The next two options (*headless* and *ignoreHTTPSErrors*) are based on optimisation and can be changed if required.

The last four options in this section are all timing values with the *waitUntil* option having the following predetermined choices:

* load - consider navigation to be finished when the load event is fired.
* domcontentloaded - consider navigation to be finished when the DOMContentLoaded event is fired.
* networkidle0 - consider navigation to be finished when there are no more than 0 network connections for at least 500 ms.
* networkidle2 - consider navigation to be finished when there are no more than 2 network connections for at least 500 ms.

#### Property Settings

"propertySetting": {  
 "uselessValuePairs": ["Not Available", "Unknown", "None"]  
 }

Including additional values in this array will prevent unnecessary data being scraped, file bloating and time being lost.

As the **propertyFile** is created an object of a property planning application is generated, this object "can" include essentially a blank field. This is due to each council having a non-standardised method of dealing with an empty field. Therefore the ***uselessValuePairs*** option allows the user to omit any scraped field (table row) that includes one of these phrases.

#### Logs

"logs": {  
 "symbols": {  
 "info": "!!!",  
 "debug": "^^^",  
 "error": "\*\*\*",  
 "warning": "%%%",  
 "default": "---"  
 },  
 "typhon": true,  
 "typhonSettings": {  
 "url": "https://typhon-api.sst-l.com/events",  
 "headers": {  
 "Content-Type": "application/json"  
 },  
 "host": "",  
 "namespace": "goplanning",  
 "name": "logs"  
 }  
 },

All of these options pertain to local and/or remote logging. The symbols object determines how the local log file looks, making it easier to find/view a type of issue that has been logged.

This option is automatically updated through the command line option **-t=**, by default this option is set to true and will revert back to true unless the user specically states a false value at the command line. **This option is what activates remote logging**.

If the Typhon url changes over time, it can be reflected here, as can any additional headers needed to be sent when communicating with the API.

**Host** - This option defaults to the users local machine name. It can be overridden to something more meaningful if the user requires so or has similar processes running on multiple machines.

**Namespace** - The default is goplanning. If the user changes this, a new namespace will need to be "created" within the Typhon interface.

**Name** - The name is a concatenation of whatever the user used for the *fileLocations.outputFolder* followed with the contents of *logs.typhonSettings.name*.

|  |  |  |
| --- | --- | --- |
| fileLocations.outputFolder | . | logs.typhonSettings.name |
| derby.city | . | logs |
| myremote | . | Issues.space |

##### Note:

Due to Typhon naming conventions, the only seporator allowed is a .period. No special charecters or spaces are allowed in any part of a namespace name.

#### Console

"console": {  
 "logTimeout": "5000"  
 },

This option was more for testing than the end user and was set out to delay certain places where the console output might clear before a debug issue could be noted.

### Council-planning.json

Location: *./settings/council-planning.json*

The council planning file contains data used for local folder and typhon log creation, as well as the current url used for each of the available council's planning portals. An example of a council object:

[  
 {  
 "id" : 5,  
 "council" : "Chesterfield Borough",  
 "shortName" : "chesterfield",  
 "url" : "https://publicaccess.chesterfield.gov.uk/online-applications/search.do?action=property&type=atoz&letter=",  
 "internalDocuments" : true,  
 "docServer" : "https://publicaccess.chesterfield.gov.uk/online-applications/applicationDetails.do?activeTab=documents&keyVal="  
 }...  
]

Line 3 is the ID number you can use to initiate a process via the **-c=** option.  
Line 4 the typed (exact) name you would use via the **-c=** option.  
Line 5 is the local data folder name as well as the typhon name entity.  
Line 6 is the current url for that particual council's planning portal entry point.  
Line 7 determines how (future handling) of actual planning documents will be referenced.  
Line 8 is the planning document server base address; different (in some cases) from the planning portal server.

## Caveats

*Known instances where data is not returned.*

###### Only Address and URL returned

In some cases a property data object will only populate with an address and URL. This appears to be as a result of an application reference existing, but the underlying data is that old that it was never included in this "modern" system.

**Example JSON Output**:

[  
 {  
 "address" : "32 Abbey Road Billericay Essex CM12 9NF",  
 "application" : {  
 "link" : "https://planning.basildon.gov.uk/online-applications/applicationDetails.do?previousCaseType=Property&keyVal=HSX32LCQN2000&activeTab=summary"  
 }  
 }  
]

Known to affect:

* Basildon

Developed by **Joz**