****

**March 2019**

**Version 0.2**

**[draft]**

**sfgs**

***Released for the purposes of the API Pilot***

**API Specification**

© Crown copyright

**Street Manager**

**Street Manger API Documentation**

[Introduction 3](#_Toc3972584)

[Onboarding 4](#_Toc3972585)

[Technical Overview 5](#_Toc3972586)

[Versioning 6](#_Toc3972587)

[Testing 7](#_Toc3972588)

[Security 9](#_Toc3972589)

[Sequencing 11](#_Toc3972590)

[Resource Guide 14](#_Toc3972591)

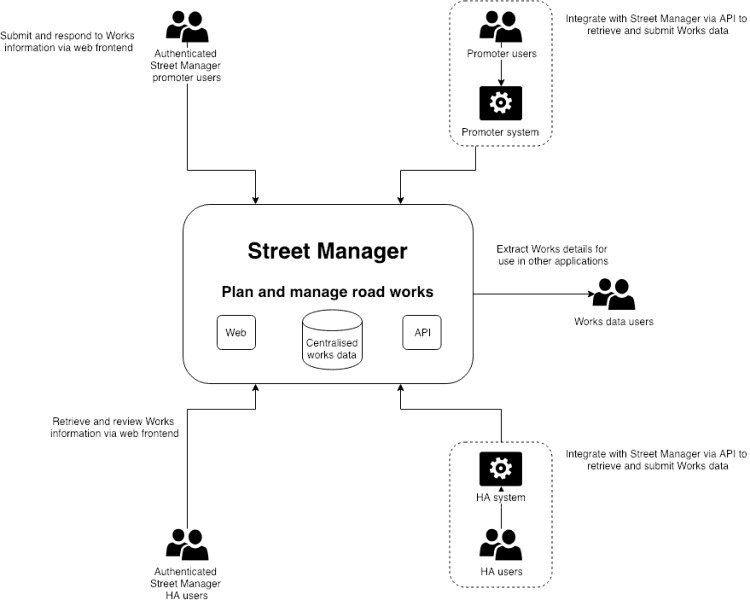
[Permit Alterations 28](#_Toc3972592)

[User and organisation registration 29](#_Toc3972593)

[Document Versions 30](#_Toc3972594)

# Introduction

This is the API guidance documentation for Street Manager, intended as a guide for developers integrating their systems to submit and retrieve information about street works. It is intended as a technical guide to be used with the API definition documentation and not a business change guide.



The Street Manager APIs are RESTful APIs to read/write data to the centralised Street Manager system following [GDS guidelines](https://www.gov.uk/guidance/gds-api-technical-and-data-standards), not a peer to peer solution.

### Updates

The following is a list of significant changes since version 0.1 of this document:

* Exposed **Street Lookup API**, **GeoJSON API** and **Reporting API** for external use, see *Technical Overview*
* Details added for Polling endpoint planned for **Reporting API,** see*Technical Overview*
* Plans added for Notifications, see *Technical Overview*
* Details added for User account roles and permissions, see *Security*
* Additional features added to **Works API**
  + Works history
  + Works comments
  + Submit immediate permit
  + Permit alterations
  + Scheduled inspections
  + Permit cancellation
  + Submit PAA
  + Progress PAA to PA

# Onboarding

### Access to API pilot and sandbox environment

Contact streetmanager@dft.gov.uk for details on how to join the API pilot and access to the sandbox environment.

### Organisation registration and access to production system

TBC

# Technical Overview

### Available APIs

Street Manager exposes a number of APIs to allow external applications to retrieve and submit data.

#### Works API

*This API is exposed externally.*

The street manager works API allows promoters and highway authority users to carry out a number of key workflows relevant to their organization and role. We will cover each in detail but at a high level they are:

**Promoter workflows:**

* Submit permit application
* Carry out a work
* Create reinstatement
* Action an FPN
* Add comments to a works record

**Highway authority workflows:**

* Assess permit application
* Issue an inspection
* Issue an FPN
* Add comments to a works record

In order to ensure a user has the appropriate permissions to carry out the above workflows they must authenticate to the API and acquire a JWT to be used as part of their request.

#### Street Lookup API

*This API is exposed externally.*

The street manager street lookup API allows querying of NSG and ASD data based on location and USRN. This function is only available as part of submitting a permit for a work.

Details TBC

#### GeoJson API

*This API is exposed externally.*

The street manager GeoJson API exposes works spatial data to authenticated users for use with mapping queries. Details TBC

#### Open data API

*This API is not yet available.*

Street Manager will support an API for Open Data users, details TBC.

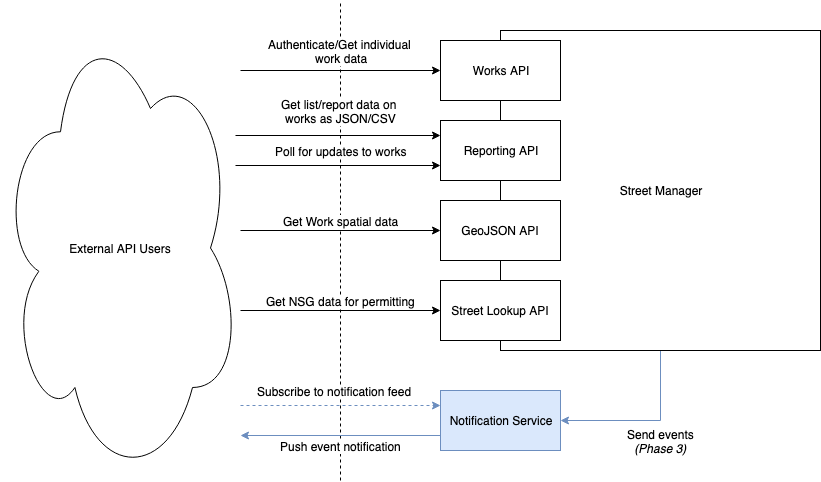
#### Reporting API

*This API is exposed externally.*

The reporting API allows promoters and highway authority users to carry out a number of data analysis and reporting workflows, allowing them to retrieve data with configurable filtering, sorting and paging. This is the backbone of our dashboard and list functionality. This API should be used as a primary API to retrieve large volumes of data about your works.

### Getting data from Street Manager

External systems integrating with Street Manager need to retrieve data from the service to give their users the most up-to-date view on what is going on with their works. Street Manager has a number of ways which you can extract data from the service.



**Individual work data**

The Works API provides endpoints which give the full detail available for individual Works and Permits. Use these endpoints to retrieve details such as timings, comments, history and changes.

**Reporting**

You can use the Reporting API to extract data from the service both as JSON and CSV format. These endpoints allow you to extract most Work information efficiently for your organisation.

**Continuous Polling**

*Note: The polling endpoint is planned for private beta, details will be added shortly.*

To find out what has changed to your Works recently, there is an API endpoint available in the Reporting API which will return a list of Works which have had changes with a defined time period. This allows external integrators to poll Street Manager for changes and use the results to retrieve further information from the Works or Reporting API.

**Notifications**

*Note: Notifications are currently in roadmap for Phase 3, not in private beta.*

Street Manager will offer a Notification service which will send Push notifications to organisations for updates to their Works to subscribed systems. Organisations who wish to receive notifications need to expose an HTTPS endpoint capable of receiving POST requests from the Notification service containing the update event data as JSON.

Notifications cannot offer guaranteed delivery (network issues, service downtime etc.) so to reconcile for missed Notifications you can use the Polling API endpoint to validate you have received notifications for all updated works.

# Versioning

*Note: API versioning will only be used for updates which change existing API functions, potentially breaking existing integration and will only be introduced after private beta.*

### Iteration and change

* API updates to be released at fixed intervals – potentially every 3 or 6 months (TBD)
* Only the newly released version, and one version previous to it, will be supported
* API is versioned via request URL path
* You don’t need to upgrade immediately, but you can only be one version out
* Release notes will be published for each release

### API Versioning

* Versioning will be controlled via the URL path.
* Versioning structure is TBC and will be researched during the pilot.

### Release Management

* API users will be notified of changes/updates prior to them being released (e.g. via DfT email, newsletter and GitHub).
* Releases / versions will consider the prevention of breaking change, with backwards compatibility being supported where appropriate.
* Sandbox / pre-production environments will be made available to support new releases.

# Testing

### Sandbox environment

A sandbox environment is available for authorised users to test their API integration.

Access details will be shared at time of onboarding.

### Other testing

#### Generating test client and server stubs

It is possible to generate test clients and servers using the available API documentation (Swagger JSON) which can be retrieved directly from the exposed APIs in Sandbox environments. This allows isolated testing of your integration without dependency on test environments and can speed up development.

[Swagger-codegen](https://swagger.io/tools/swagger-codegen/) supports generating client and server stubs for a variety of languages, see here for details. Below is an example of generating Java client and server stubs using the [Swagger-codegen](https://swagger.io/tools/swagger-codegen/) utility.

**Requires:**

* Java (tested with openjdk version "11.0.1" 2018-10-16)
* Maven for build
* Swagger Codegen

#### Client

Generated with command:

swagger-codegen generate -i swagger.json -l java -o ./streetmanager-apiclient-java

The generated code for the template had a number of errors which required manual corrections.

Corrections:

* pom.xml - updated <java.version>1.7</java.version> to <java.version>1.8</java.version> to support annotations
* pom.xml - added dependency javax.annotation to replace deprecated class
* Find/Replace body@ApiParam to body,@ApiParam due to template error on generated clients

• Find/Replace new BigDecimal() to new BigDecimal(0) due to template error on generated tests

To build:

mvn package

#### Server

Generated with command:

swagger-codegen generate -i swagger.json -l spring -o ./streetmanager-apistub-java-spring

The generated code for the template had a number of errors which required manual corrections.

Corrections:

* pom.xml - updated <java.version>1.7</java.version> to <java.version>1.8</java.version> to support annotations
* pom.xml - added dependencies for javax.xml.bind to replace deprecated class
* Find/Replace body@ApiParam to body,@ApiParam due to template error on generated controllers
* Find/Replace new BigDecimal() to new BigDecimal(0) due to template error on generated tests

To run:

mvn package

java -jar target/swagger-spring-1.0.0.jar --server.port=8080

# Security

### HTTPS

All Street Manager APIs are secured using Transport Layer Security (TLS) v1.2 certificates signed by a standard certificate authority.

### Authentication and Authorization

All resource endpoints in the API with the exception of authentication and health-checks, require a JWT to be passed in the 'token' header of the request. The JWT contains information about the user and allows them to access routes, services, and resources that are permitted with that token. Without it the request will be met with a 401 error response.

### User accounts and permissions

External systems integrating with Street Manager should use specific credentials setup for API users. This is to allow Street Manager to differentiate between Web UI user accounts and API users. User accounts are assigned specific roles, such as *planner* and *admin*.

*Note: Currently systems who need to act as users associated with multiple organisations, i.e. submitting permits for multiple utility companies, need to use separate user accounts for each organisation.*

### JWT

Json Web Token (JWT) is an open standard for exchanging information securely. The entities of street manager exchange information using JWTs and resources of the street manager API require that a JWT be provided as part of the request. This will be discussed in detail as part of the security section.

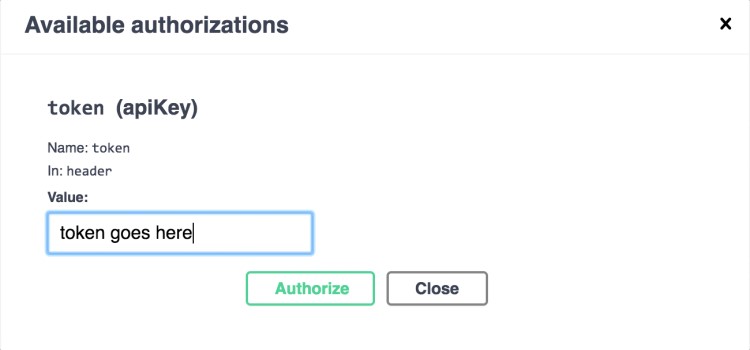
### Resource

POST /authenticate

The authenticate endpoint takes a case sensitive username (email address) and password, returning a JWT token if successful. **The JWT token is valid for one hour.** Once the token has been acquired it can be added to all protected resource requests made via swagger using the Authorize button.



When clicked this will present an input to enter the token. Once authorized then all protected resource requests made via swagger will have the token header set.



### Error responses

It's important to distinguish between authentication and authorization error responses as it will help narrow down where an issue is occurring.

### Authentication Failed

{ "message": "Authentication failed", "error": { "status": 401 }

Authentication fails when the token provided in the request is invalid. The token may have expired or the value set as the token was incorrect. You may also see this error when calling the POST /authenticate endpoint with invalid credentials I.e. wrong username or password.

### Access Restricted

{ "message": "Access restricted", "error": { "status": 401 }

The access restricted error indicates that although the token was valid, the user does not have permissions to perform the desired action. This error could arise for several reasons which will be listed in detail as part of the resource section, but common triggers would be attempting to manipulate entities (permit, reinstatement, inspection etc.) not related to your organization.

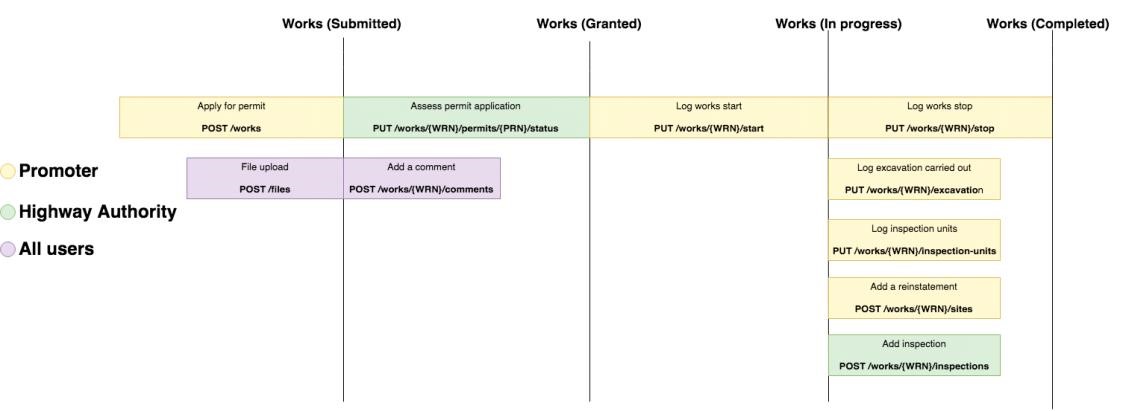
### Rate limiting

To protect the system from denial of service attacks, repeated calls made in a short period of time from a single IP source will receive 405 status responses. If you are receiving 405 responses ensure you are not sending calls

# Sequencing

As detailed in the Technical Overview section, the reporting API drives a large amount of data retrieval functionality whilst the street manager API drives a lot of key user workflows e.g. submit permit, assess permit, etc. These two APIs together form much of the common sequences a user is likely to perform.

Below is an example of sequence calls used to submit and respond to a permit application as well as how various actions affect the works lifecycle. These endpoints are all available as part of the street manager API and are discussed in more detail within the resource guide.



The following actions can be performed at any subsequent stage in the sequence from the stage they are available:

* File upload
* Add a comment
* Add reinstatement
* Add inspections

Whilst the above focuses much on data manipulation via the street manager API, here is an example of some data retrieval calls that may be performed alongside these actions via the reporting API.

* **Permits awaiting assessment:** GET /permits?status=submitted
* **Expiring interim reinstatements:** GET /sites?status=interim
* **Disputed FPNs**: GET /fixed-penalty-notices?status=disputed

### Understanding the status of a work

As a permit progresses through the sequence above the permit status changes. Knowing the various statuses of a permit allows you to filter lists of permits related to your organization through the reporting API. The statuses of a permit are:

* **submitted:** The permit is awaiting assessment
* **granted\_proposed:** The permit has been assessed as granted by an HA
* **granted\_with\_changes:** The permit has been assessed as granted with changes by an HA
* **refused:** The permit has been assessed as refused by an HA
* **granted\_in\_progress:** The permit has been started by the promoter after being granted
* **closed:** The permit has been stopped by the promoter
* **cancelled:** The permit has been cancelled by the promoter
* **deemed\_proposed:** The permit has been automatically deemed as it was not assessed before the deadline date
* **deemed\_in\_progress**: The permit has been started by the promoter after being deemed

Note: PAA/Major submission will be included as part of this sequence.

### Viewing works and permits

#### Street Manager API

When a permit has been submitted and a works record exists both promoters and HAs can view the works record via the GET work endpoint on the street manager API. This endpoint requires you to provide the **work reference number** which was supplied during the submission of the permit application.

GET /works/{work reference number}

This contains information about all of the entities associated with the work record, the properties of this response are:

* **Active permit:** The currently active permit associated with the works. In the sequence above this would contain the permit awaiting assessment
* **Sites:** Any reinstatement sites that have been added to the works (none initially)
* **Inspections:** Any inspections that have been issued on the works (none initially)
* **FPNs:** Any fixed penalty notices that have been issued on the works (none initially)
* **Permits:** Summary of all permits that have been associated with that works (I.e. multiple permits)

It's also possible to retrieve only information about the permit itself using the GET permit endpoint. This endpoint requires you to provide the **permit reference number** which is returned in the response of the permit application submission. As detailed in the submit permit application section of the resource guide, the permit reference number is simply the works reference number suffixed by an incrementing number e.g. {WRN}-01 for the first permit added to that work.

GET /works/{work reference number}/permits/{permit reference number}

#### Reporting API

The reporting API permits endpoint will be the most useful way to see all permits that are relevant to your organisation.

*Note: Reporting API data can be retrieved as CSV data to allow exporting data to tools such as Excel for more complex reporting needs, such as all permit details needed for Fee Reporting.*

For example, as an HA you can use the **status** property of a permit indicates the current state it is in, for submitted permits that are awaiting assessment the permit status will be "submitted".

GET /permits?status=submitted

The **status** query param can be changed to any of the values discussed above to retrieve permits in any stage of the sequence. This is discussed in more detail in the reporting API resource guide.

Promoters can use **status** values to find permits which the HA has responded to, see the Street Manager API resource guide for more details on Permit status values.

# Resource Guide

### Reporting API

As discussed in the sequencing section, the reporting API allows promoters and HAs to query resource lists for their organization, filtering them by various properties. The reporting API currently allows users to retrieve the following:

* Permits
* Comments
* Fixed penalty notices
* Sites

Most endpoints on the reporting API are driven with pagination. This means the following common query params are available for most endpoints:

* **Before**: cursor to filter pagination results backwards (I.e. retrieve everything before this item)
* **After**: cursor to filter pagination results forwards (I.e. retrieve everything after this item) Each paginated response in the reporting API contains the following meta-data:

{

"pagination": {

"page\_start": 1,

"total\_rows": "50"

},

"rows": […]

}

The page\_start and total\_rows properties indicate the current page of results returned in the response, in the context of the total number of available rows (records). The rows property contains the records for the current page.

By default there are a maximum 25 rows returned per page and each record in the rows array has a cursor property, which is that record's placement in the context of the total result set. Therefore in the example above, if you have 50 items total, with 25 items per page, to get the next page of results would simply involve taking the last item in the rows array, and using it's cursor value as the 'after' query param value. This would result in the next metadata response

containing rows 26-50. The reason we use cursor based paging is to handle real time data. That is to say, the same items will not show up on different pages even as items are inserted/removed from the database in-between queries.

### Permits

#### View permits

As discussed in the sequencing section, the reporting API allows promoters and HAs to query permits for their organization, filtering them by various properties such as status or works reference number. As well as this the reporting API allows users the retrieve count summary information about permits.

### Get permits endpoint

/permits

The permits endpoint retrieves a list of permits associated with the user. The endpoint allows configuration of results through the use of the following query params:

* **status**: The permit status I.e. submitted, granted
* **query**: The work reference number associated with the permit (partial match)
* **sort\_column**: The property of the permit to order results by
* **sort\_direction**: Ascending/descending

### Count permits endpoint

/permits/counts

The counts endpoint provides a total summary of permits grouped by their status. The response contains:

* Total refused applications
* Total submitted applications
* Total granted
* Total in-progress permits
* Total closed permits

Comments

#### View Comments

The reporting API allows users to view comments associated with their organization.

#### Get comments endpoint

/comments

Retrieves a list of comments that have been added to any works record associate with the currently authenticated user's organization. Comments are added via the street manager API (see comments section) and visible from this endpoint in the reporting API.

### **FPNs**

#### View Fixed Penalty Notices

The reporting API allows users to view FPNs associated with their organization.

#### Get FPNs endpoint

/fixed-penalty-notices

Retrieves a list of FPNs that have been added to any works record associate with the currently authenticated user's organization. FPNs are issued via the street manager API (see FPN section). Similarly to the permits endpoint, FPNs can be filtered by status. The status of an FPN are:

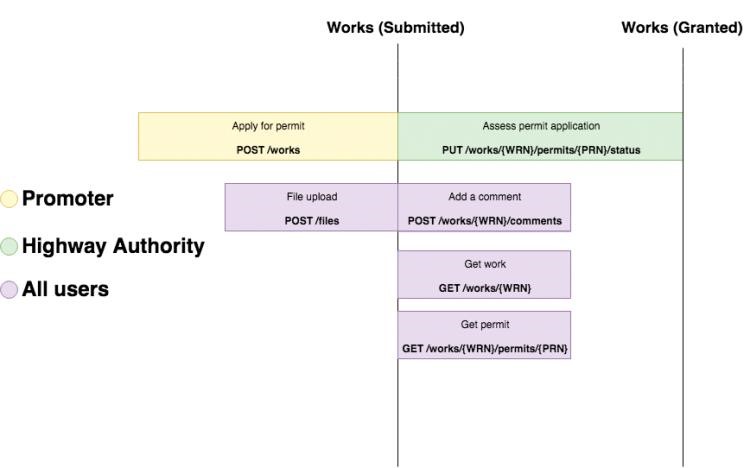
* **issued**: Issued by the HA
* **accepted**: Accepted by the promoter
* **paid**: Confirmed as paid by the HA
* **paid\_discounted**: Confirmed as paid by the HA with a discount
* **disputed**: Disputed by the promoter
* **withdrawn**: Withdrawn by the HA

Sites

Street Manager API supports retrieving details on sites, details TBC.

### **Street Manager API**

#### Submit permit application



As shown in the sequence, a promoter submits a permit application at which point an HA can assess it. Both a promoter and HA can add comments to the works record once it exists. Both users can upload files for various workflows, but in the case of applying for a permit application, a promoter may choose to upload a file as part of their application (I.e. providing a traffic management plan). Files need to be uploaded via the file upload endpoint before than can be used as part of other requests, so often this endpoint is called first. See the **File Upload** section of the resource guide for details.

It's important to clarify the technical relationship between a work and a permit. When creating a permit for the first time, this will in effect create a works record. A work can have multiple permits. You can create a work and a permit at the same time, or you can create a permit against an existing work. A work cannot exist without a permit.

There is also a concept of a work record's active permit, that is simply the most recently added permit on that works record. In essence, a work is a representation of all the permits, reinstatements, FPNs, inspections etc. that have been added to a particular location by a particular promoter.

#### Create work endpoint

POST /works

This endpoint takes all the information required to create a permit, as well as some key identification information about the works record as a whole.

#### Constraints

##### Work reference number

The works reference number provided must be unique in the context of our system in order to create a new work.

##### Works coordinates

The works coordinates field expects a [GeoJSON geometry](http://geojson.org/) and must be a point, line string or polygon.

{ "type": "Point", "coordinates": [80000.00, 80000.00] }

##### Promoter SWA code and highway authority SWA code

These are particularly important fields in the request. Currently only a promoter can create a permit and works record so the promoter SWA code provided in the request much match that of the user authenticated to the system. This is determined by the token header of the request, which contains the JWT. In effect this means the promoter can only add a work or permit for their own organization.

All SWA codes are left-padded to 4 digits, so for example if the SWA code of your promoter organization is 10, this should be entered as "0010".

As a promoter, the HA SWA code you choose is the organization which will be associated with the permit, and thus responsible for assessing the permit. This HA will also be the only organization able to add inspections or issue FPNs against the work record. Although generally most information in the system is viewable by any organization, only the HA and promoter responsible for the work can make actions against it.

### Get work endpoint

GET /works/{work reference number}

Once the work has been created it can be retrieved using the GET endpoint, passing the work reference number that was used as part of the create request.

### Create permit endpoint

POST /works/{work reference number}/permits

A work can have multiple permits associated with it so it possible to add a new permit to an existing works. This endpoint requires some of the same fields as the create work request but much of the information from the first permit will be used as the value for additional permits, and so only a subset of information is required.

#### Constraints

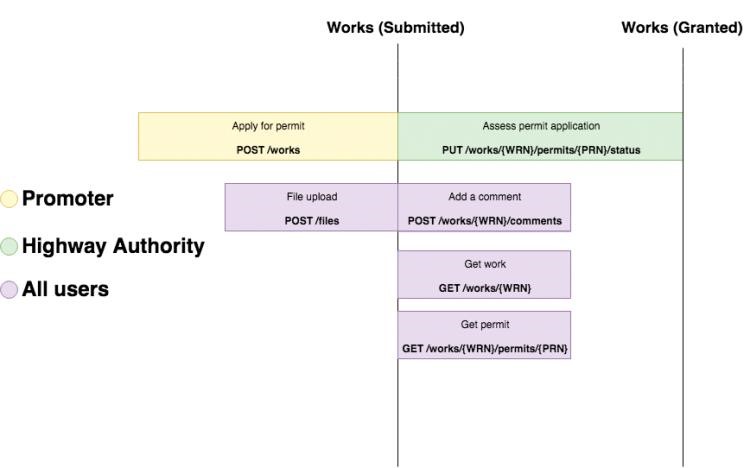
It's not possible to add an additional permit to an existing works, unless the work is in an inactive state. The state of the work is derived by the status of the most recently added permit. So in short, an additional permit can only be added to a work if the most recently added permit on the existing works record has a status of closed, cancelled or refused.

### Get permit endpoint

GET /works/{work reference number}/permits/{permit reference number}

Once an additional permit has been added to an existing works record, it can be retrieved using the work and permit reference number. The permit reference number is simply the works reference number suffixed by an incrementing number e.g. {WRN}-01 for the first permit added to that work.

#### Assess permit application



As shown in the sequence diagram, once a permit application has been submitted an HA user can provide an assessment decision for the permit. There are technically 4 assessment decision options:

* Granted
* Granted with changes
* Refused
* Deemed

*Note: Available options may change as functionality is updated/added*

Permits are deemed automatically when they exceed their response date, and granted with changes is a future function so these two options aren't explicitly supported as manually entry.

#### Update status endpoint

POST /works/{work reference number}/permits/{permit reference number}/status

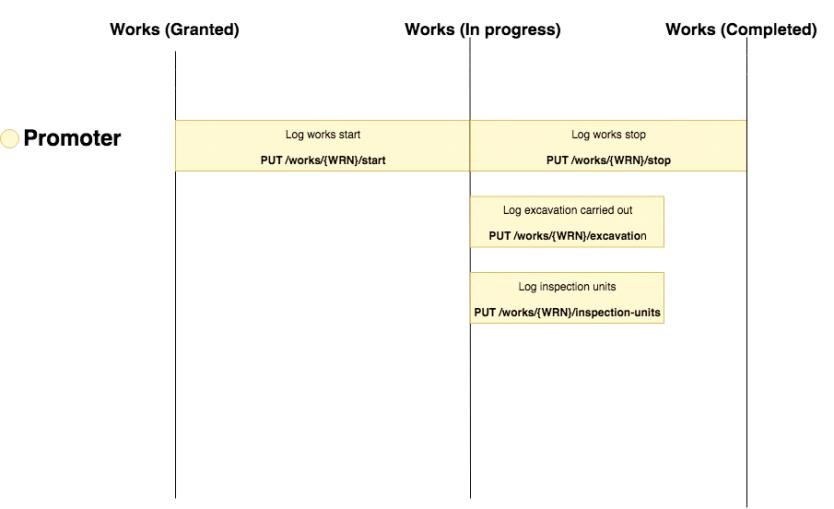
Assessing a permit application is achieved through this endpoint by providing an assessment status. This also changes the overall permit status. Once a permit has been assessed it cannot be assessed again.

#### Constraints

As mentioned in the submit permit application section, a permit can only be assessed by the highway authority that the permit was submitted to. This means HA users can only assess permits on work records they are associated with. If the permit was submitted to a different organization, they will not be able to assess it. Furthermore promoters cannot assess their own permits.

#### On site (start/stop works)

#### On site



Once a permit has been submitted and granted by an HA, the promoter which raised the permit is able to:

* Start a work
* Stop a work
* Provide inspection units
* Indicate whether or not an excavation was carried out

These actions control various stages of the works record life cycle as per the sequence diagram above.

#### Start work endpoint

PUT /works/{work reference number}/start

When a permit is submitted initially, a proposed start and end date for the work must be provided. The start endpoint is then used to provide the date the work has actually began, and thus officially starting the work.

By setting an actual start date, the active permit's status will change to in-progress. This allows promoter organization users to add reinstatements against the works record if an excavation was carried out and it also allows highway authority users to add inspections, which will be covered in separate sections.

#### Constraints

The date provided must be in the past, you can't set an actual start date ahead of time. You can only start a work if the active permit has been granted. You can only provide an actual start date once, afterwards it cannot be changed.

### Stop work endpoint

PUT /works/{work reference number}/stop

When a permit is submitted initially, a proposed start and end date for the work must be provided. The stop endpoint is then used to provide the date the work has actually ended, and thus officially stopping the work.

By setting an actual stop date via this endpoint, the active permit's status will change to closed. It's still possible to add reinstatements and inspections to closed works as they may be added retrospectively.

#### Constraints

The date provided must be in the past, you can't set an actual stop date ahead of time. You can only stop a work if the active permit is in progress I.e. a start date has been provided. You can only provide an actual stop date once, afterwards it cannot be changed. The actual stop date must be after the actual start date.

### Excavation carried out endpoint

PUT /works/{work reference number}/excavation

When a permit is submitted initially the excavation property indicates whether or not an excavation will need to be carried out as part of the work. Similar to proposed start and stop, this endpoint indicates whether or not an excavation was *actually* carried out.

Marking an excavation was carried out on an in-progress work record allows promoters to add reinstatements to that works record, which will be covered in a separate section.

#### Constraints

You can only make this request if the permit is in-progress or closed. You can only set whether or not an excavation was carried out once, afterwards it cannot be changed.

### Inspection units endpoint

PUT /works/{work reference number}/inspection-units

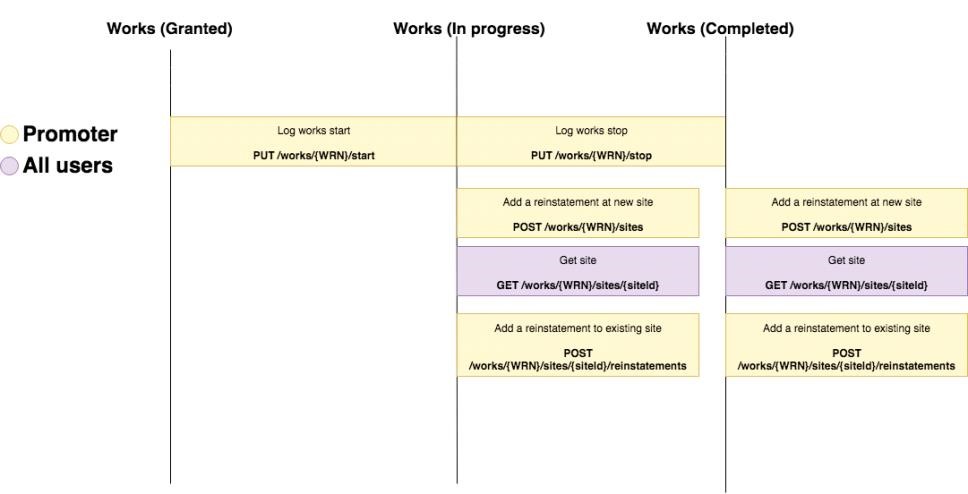
Once a permit is in progress and an excavation has been carried out, a promoter can log the inspection units associated with the currently active permit.

#### Constraints

You can only make this request if the permit is in-progress or closed. You can only provide inspection-units if a reinstatement currently exists on the works record, which will cover in a separate section.

#### Reinstatements

#### Add reinstatements



As shown in the sequence diagram, once the a work has been started by the promoter and an excavation was carried out, then a promoter can add reinstatements and sites. A promoter can continue to create and view these even after the work has been stopped and completed, as they made need to do this retrospectively.

In a similar vein to the relationship between a works and a permit, it's important to clarify the technical relationship between a site and a reinstatement. When creating a reinstatement for the first time, this will in effect create a site record. A site can have multiple reinstatement. You can create a site and a reinstatement at the same time, or you can create a reinstatement against an existing site. A site cannot exist without a reinstatement.

A site is a representation of all the reinstatements carried out at a particular location but the most recently added reinstatement reflects the sites properties.

### Create site endpoint

POST /works/{work reference number}/sites

This endpoint takes all the information required to create a reinstatement, a successful request will create a site with an associated reinstatement.

#### Constraints

##### Reinstatement coordinates

The reinstatement coordinates field expects a [GeoJSON geometry](http://geojson.org/) and must be a point, line string or polygon.

{ "type": "Point", "coordinates": [80000.00, 80000.00] }

### Get site endpoint

GET /works/{work reference number}/sites/{site id}

Once a site has been created it can be retrieved using the GET endpoint, passing the site id which is returned as part of the create request.

### Create reinstatement endpoint

POST /works/{work reference number}/sites/{site id}/reinstatements

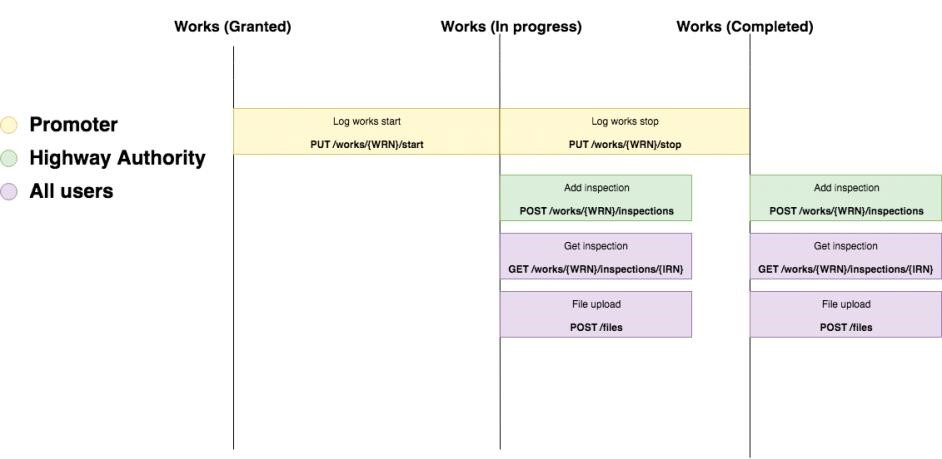
A site can have multiple reinstatements associated with it so it is possible to add a new reinstatement to an existing site. This endpoint requires all of the same fields as the create site request.

#### Constraints

A reinstatement cannot be added before the actual start date of a work. A reinstatement can be interim or permanent. If the most recently added reinstatement on a site is interim, then a new permit is required to be granted and started before that site can be made permanent.

#### Inspections

#### Add inspections



As shown in the sequence diagram above, once a work has been started then an HA can issue an inspection. Similar to reinstatements, this can be done even after the works has been stopped and completed, in cases where the HA needs to do this retrospectively.

There are several inspection outcome types supported:

* Passed
* Unable to complete
* Failed standard
* Failed 2 hour • Failed 4 hour

When issuing a failed inspection it's also possible to upload supporting evidence. This requires using the file upload flow. See the File Upload section of the resource guide for details.

### Create inspection endpoint

POST /works/{work reference number}/inspections

Creating an inspection will return a inspection reference number which can be used to retrieve an individual inspection via the GET endpoint provided. The inspection reference number is simply the works reference number suffixed by an incrementing number e.g. {WRN}-INSP-01 for the first inspection added to that work.

#### Get inspection endpoint

GET /works/{work reference number}/inspections/{inspection reference number}

#### FPNs

#### Issue a FPN (HA only)

An HA user can issue a fixed penalty notice against the work record at any point in the works life cycle. Promoters can view and dispute the FPN but they cannot issue their own. HA users may upload evidence to support their FPN but the workflow for this is explained in the file upload section. There are several statuses an FPN can have:

* Issued
* Accepted
* Paid
* Paid (discounted)
* Disputed
* Withdrawn

### Create FPN endpoint

POST /works/{work reference number}/fixed-penalty-notices

Creating a FPN will return a FPN reference number which can be used to retrieve the individual FPN via the GET endpoint provided. The FPN reference number is simply the works reference number suffixed by an incrementing number e.g. {WRN}-FPN-01 for the first FPN added to that work.

### Get FPN endpoint

GET /works/{work reference number}/fixed-penalty-notices/{FPN reference number}

### Action FPN endpoint

PUT /works/{work reference number}/fixed-penalty-notices/{FPN reference number}/status

Both a promoter and HA can action FPNs in different way. When an FPN is created by the HA it is considered issued. A promoter can accept or dispute FPNs, whilst an HA officer can mark an FPN as withdrawn or paid.

#### File Upload

#### File upload

Several flows discussed in the previous sections allow users to add files as part of their request:

* **Submit permit application:** planners can upload traffic management plans
* **Add reinstatement:** planners can upload evidence of the reinstatement
* **Issue an inspection:** HA users can upload evidence as part of issuing failed inspection
* **Issue a fixed penalty notice:** HA users can upload evidence as part of issuing an FPN

Uploading a file is achieved through the file upload endpoint. This endpoint is required as part of all the above flows as any files that the user wishes to associate with the above requests must first be uploaded using this endpoint. As mentioned in the Sequencing section, you can upload files at any point in the works lifecycle.

*Note: The sequence for uploading files may change to allow drafting permits before uploading files.*

### Upload file endpoint

POST /files

Once a file has been uploaded the response will contain a file ID. This is the unique identifier of the file in our system. Behind the scenes the file will be uploaded to S3 and virus scanned. This file ID can then be provided in the requests of the flows discussed above. Once a valid file ID is provided in the requests of the above flows, the file is then associated with the relevant entity.

### Get file endpoint

GET /files/{file ID}

Retrieves the file using the file ID.

#### Constraints

As the file is virus scanned at the point of upload, it is not immediately available for retrieval. It's possible that this endpoint could return a file not found error response if the file has not yet been virus scanned and marked as safe or if the file was deemed unsafe and removed from the system.

Typically the virus scanning process will only take a few seconds.

### Delete file endpoint

DELETE /files/{file ID}

Deletes a file from the system.

#### Constraints

You cannot remove a file that's been linked to an entity. As soon as the file id is used as part of a create permit/reinstatement/FPN/inspection request then it is considered linked to that entity.

### Comments

#### Add comment

Both promoter and highway authority users can add comments to a works record. When a comment is added to a works record it will be returned as part of the get works response.

### Create comment endpoint

POST /works/{work reference number}/comments

Add a comment to a work record.

#### View comments

The comments on an individual work record can be viewed as part of that work's history. This endpoint also returns audit events such as when a permit is assessed, started etc. You can distinguish history items as audits or comments by the **isComment** property.

### History endpoint

GET /works/{work reference number}/history

As well as viewing comments on a work record level, you can also call the reporting API comments endpoint to retrieve all comments associated with your users organization. See the Reporting API Comments section of the resource guide.

# Permit Alterations

API will support submitting permit alterations, details and sequencing TBC.

# User and organisation registration

API will support user and organisation registration, details and sequencing TBC.

# Document Versions

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Version | Status | Comment |
| 20/03/2019 | 0.2 | Draft | Draft for early API pilot |
| 31/01/2019 | 0.1 | Draft | Initial release |