National Parking Platform

API Specification

Version 2.2

Publication 24th of July, 2025

Contents

Introduction

This document should be read with the Service Specification, that will provide details of stakeholders, relationships, and operations. The definitions of NPP specific terms used in this document are also available in the Service Specification.

In summary, the National Parking Platform (NPP) is a backend system that enables all connected stakeholders to exchange parking information, considering their respective roles (see Service Specification, Roles and Responsibilities):

- Parking Operator
- Parking Service Provider
- Connected Supplier

The NPP is a so-called headless system, mainly communicating via an application programming interface (API). This API is based on APDS, the parking data exchange model and format standardised by the Alliance for Parking Data Standards. This document specifies the APDS subset used by the NPP, supplemented by some extensions the publication of which is still pending. The currently released version of APDS is version 4.

All connected NPP Clients must adhere to the interface specification set out here.

Use Cases

This document specifies the actual NPP API. A second document (**NPP Developer Guide)** presents selected use cases and provides further guidance for developers.

API Scope, NPP Development

Pilot Scope

The NPP started out as a proof-of-concept project with 10 Local Authorities and 5 Service Providers. Due to time and budgetary constraints, the pilot phase was limited to a specific set of use cases:

- **Inventory Data** (semi-static information about on-street and off-street parking locations, including infrastructure details, charging hours, and tariffs)
- **Occupancy Information** (dynamic occupancy/availability information from parking locations with appropriate sensory equipment)
- Parking using **Payment on Arrival**
- Parking using **Check in / Check out**
- **Enforcement** (using either direct NPP queries or the NPP PUSH service)

Full Platform Scope

Following the formation of the not for profit company (NPP Consortium Limited) the original pilot system is scaled up to a full-size platform and further use cases are added such as e.g.

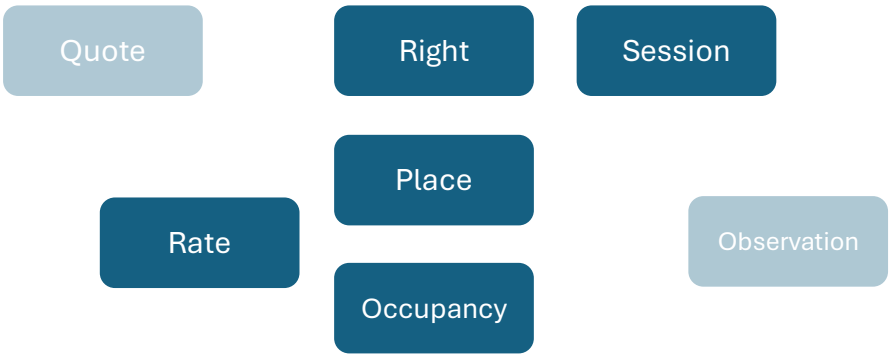
- **PODBA** (pay on departure by app)
- **Adjacent Sessions Handling** (technical policy enforcement)
- **Permit Parking**
- **Frictionless Parking** (parking based on a pre-registered account)

Furthermore, this version introduces API endpoints for the automated exchange of **Reconciliation** Information. Finally, the option to retrieve health and performance information has been added (**Observability)**.

APDS Scope

APDS Data Domains

This document specifies the endpoints and underlying data structures of the NPP's API. It is not intended as an introduction to APDS. We strongly recommend undertaking the available APDS training courses to familiarise themselves with the APDS model and/or procuring independent advice from APDS specialists if necessary. The APDS model defines a number of data domains:



The official APDS specification documents can be requested via the APDS website.

APDS Data Domains not used by the NPP

The current version of the NPP does not engage all APDS-defined data domains / model packages. In particular, the following parts are not used:

- Quote
- Observation
- Electrical Infrastructure (sub-package of Place) *(though this may be added later)*

Non-APDS Data Domains used by the NPP

Note: the NPP endpoints defined to cover **Reconciliation** and **Observability** are not defined in the APDS standard and hence have been specified as project-specific interface components.

API Endpoints

This section provides an overview of all endpoints offered by the NPP. Developers should consult the machine-processible Open API specification of the NPP interface which can be found in Annex 1: Open API Specification below. This contains all details concerning request and response payloads, available operations and error codes.

Inventory Service

The inventory and Dynamic Availability endpoints are important for the connection between Service Providers (and their parking apps) and the NPP.

Inventory and Availability API Endpoints

Inventory information (Locations and their properties including charging hours, tariffs, etc) shall be read by the Service Provider backend systems to

- populate the list of offered parking locations in their apps and
- feed their own rate engines with the effective tariff information.

Operators will provide up-to-date inventory information to the NPP (either directly via the inventory API endpoints or indirectly via the location spreadsheet which is then automatically processed and uploaded.

Dynamic Availability API Endpoints

These are provided within the Place Hierarchy to provide up to date occupancy for Locations where that information is available.

Endpoint List

Place Hierarchy

The *Place Hierarchy* endpoints accept/provide details about parking locations (*Places*).

Method	URI	Description
GET	/places	Read a (paginated) list of places, potentially filtered by a combination of optional parameters: • **latitude**, **longitude** and **radius** : retrieve all places within a geographic area, • **type** : retrieve all places of the given type, • **right_type** : retrieve all places associated with the right specification type, • **structure_type** : retrieve all places associated with the specified structure type, • **structure_grade** : retrieve all places associated with the specified structure grade, • **name** : retrieve all places which match the name of the facilities, • **layer** : specify the maximum hierarchy element layer desired, • **modified_since** : retrieve all places that have changed from that given instant onwards, • **page** : in case of multi-page results: the index of the page to be retrieved

		• **expand** : a comma-separated list of optional object attributes associated to a place that should be included in the result set: ○ **all**: provide all available details ○ **none**: only provide id and version information ○ **rightSpecifications**: include list of applicable right specifications ○ **occupancy**: include (dynamic) occupancy info where available ○ **characteristics**: include detailed location characteristics ○ **openingTimes**: include opening time information ○ **streetAddress**: include place's street address
POST	/places	Create a new *Place* record Returns a *ResponseStatus* record indicating success/failure of the operation
GET	/places/{id}	Read the latest version of a given *Place* specified by its unique identifier *{id}*. The level of verbosity in the response can be controlled via the **expand** query parameter. The *expand* parameter is a comma-separated list of one or more values from the following enumeration: • **all**: provide all available details • **none**: only provide id and version information • **rightSpecifications**: include list of applicable right specifications • **occupancy**: include (dynamic) occupancy info where available • **characteristics**: include detailed location characteristics • **openingTimes**: include opening time information • **streetAddress**: include place's street address Returns a single *Place* record (if it exists)
PUT	/places/{id}	Update existing *Place* with identifier {id} Returns a *ResponseStatus* record indicating success/failure of the operation
DELETE	/places/{id}	Delete a *Place* record. Please note that this operation is only possible if no *Assigned Right* has ever been issued for this *Place*. Returns a *ResponseStatus* record indicating success/failure of the operation

Rights (Right Specifications)

The *Right Specifications* endpoints accept/provide information about charging hours, eligibility criteria and point to applicable tariffs. Each *Right Specification* applies to one or more *Places*.

Method	URI	Description
GET	/rights/specs	Read a list of right specifications, potentially filtered by a combination of optional parameters: Returns a paginated list of all *Right Specifications* matching the provided criteria. The level of verbosity in the response can be controlled via the **expand** query parameter. The *expand* parameter is a comma-separated list of one or more values from the following enumeration: • **all**: provide all available details • **none**: only provide id and version information • **rateEligibility**: include eligibility and tariff reference information • **validity**: include validity details
POST	/rights/specs	Create a new *Right Specification*. Returns a *ResponseStatus* record indicating success/failure of the operation
GET	/rights/specs/{id}	Read the latest version of a given *Right Specification* specified by its unique identifier *{id}*. Returns a single *Right Specification* record (if it exists)
PUT	/rights/specs/{id}	Update existing *Right Specification* with identifier {id} Returns a *ResponseStatus* record indicating success/failure of the operation
DELETE	/rights/specs/{id}	Delete a *Right Specification* record. Please note that this operation is only possible if no *Assigned Right* has ever been issued referencing this *Right Specification*. Returns a *ResponseStatus* record indicating success/failure of the operation

Rates

The *Rates* endpoints accept/provide information about tariffs. While a tariff is a separately identifyable entity, its applicability is defined via a referencing *RateEligibility* definition (as part of a corresponding *Right Specification)*.

Method	URI	Description
POST	/rates	Create a new *Rate* record Returns a *ResponseStatus* record indicating success/failure of the operation
GET	/rates	Read a list of *Rates*, potentially filtered by a combination of optional parameters: • **latitude**, **longitude** and **radius** : retrieve all rates applicable within a geographic area (of places) • **place :** retrieve all rates applicable in the specified locations • **right_spec_ids** : retrieve all rates referenced by one of the listed right specifications • **modified_since** : retrieve all places that have changed from that given instant onwards, Returns a paginated list of *Rates* matching the specified criteria.
GET	/rates/{id}	Read the latest version of a given *Rate* specified by its unique identifier *{id}*. Returns a single *Rate* record (if it exists)
PUT	/rates/{id}	Update existing *Rate* with identifier {id} Returns a *ResponseStatus* record indicating success/failure of the operation
DELETE	/rates/{id}	Delete a *Rate* record. Please note that this operation is only possible if no existing *Right Specification* is referencing this *Rate*. Returns a *ResponseStatus* record indicating success/failure of the operation

Parking Rights Service

Assigned Rights API Endpoints ("Parking Rights")

In APDS, *Assigned Rights* are the representation of parking rights. *Assigned Rights* are issued by Parking Operators and/or Service Providers. As soon as a parking right has been issued to a driver, it is sent to the NPP by the issuing entity. *Assigned Rights* are the most important source of information for connected enforcement systems to check the eligibility of a parked vehicle.

Endpoints

Method	URI	Description
POST	/rights/assigned	Send a new *Assigned Right* record to the NPP. Returns a *ResponseStatus* record indicating success/failure of the operation
PUT	/rights/assigned/{id}	Update an existing *Assigned Right* record with identifier {id}. Returns a *ResponseStatus* record indicating success/failure of the operation
GET	/rights/assigned	Get a (paginated) list of *Assigned Right* records matching the provided search criteria. Will only return records in alignment with the caller's roles and permissions. Parameters are: • **latitude**, **longitude** and **radius** : retrieve all assigned rights associated with a place within a geographic area • **place** : filter by a ist of locations • **credential_id:** filter by vehicle identifier (e.g. VRM) • **start_before:** return only assigned rights with a validity that started before the given instant • **start_after:** return only assigned rights with a validity that started after the given instant • **end_before:** return only assigned rights with a validity that ended before the given instant • **end_after:** return only assigned rights with a validity that started after the given instant. If this parameter is not specified, it is automatically set to a default of "now". • **modified_since** : retrieve all *Assigned Right* records that have been created or changed from that given instant onwards • **page**: in case of multi-page responses: select the result page to retrieve
GET	/rights/assigned/{id}	Retrieve a specific *Assigned Right* record identified by {*id*}. Will only be returned if the caller has the appropriate role and permissions.
DELETE	/rights/assigned/{id}	Request to delete a specific *Assigned Right* record identified by {*id*}. Technically, the *Assigned Right* record will not be deleted, but moved to the archive database. As a best practice, it is recommended to – rather than using this endpoint – expire an *Assigned Right*.

Sessions API Endpoints

In APDS, *Sessions* and *Segments* describe the act of using of a previously-obtained parking right (*Assigned Right*). When drivers park their vehicles or leave the parking location, the Service Provider's backend or the operator's parking management system reports this to the NPP.

Session data can later be read from the NPP to support use cases like for example:

- Checking a challenged PCN (via the Session Viewer application)
- Checking the details of a parking session (via an enforcement system)

Endpoints

Method	URI	Description
POST	/sessions	Send a new *Session* record to the NPP. Returns a *ResponseStatus* record indicating success/failure of the operation
PUT	/sessions/{id}	Update an existing *Session* record with identifier {id}. Returns a *ResponseStatus* record indicating success/failure of the operation
GET	/sessions	Get a (paginated) list of *Session* records matching the provided search criteria. Will only return records in alignment with the caller's roles and permissions. Parameters are: • **latitude**, **longitude** and **radius** : retrieve all sessions within a geographic area (of places) • **place** : retrieve all sessions from the specified locations • **start_before:** return only sessions that started before the given instant • **start_after:** return only sessions that started after the given instant • **end_before:** return only sessions that ended before the given instant • **end_after:** return only sessions that started after the given instant • **modified_since** : retrieve all sessions that have been created or changed from that given instant onwards • **page**: in case of multi-page responses: select the result page to retrieve
GET	/sessions/{id}	Retrieve a specific *Session* record identified by {*id*}. Will only be returned if the caller has the appropriate role and permissions.
DELETE	/sessions/{id}	Request to delete a specific *Session* record identified by {*id*}. Technically, the Session record will not be deleted, but moved to the archive database.

Reconciliation Service

Reconciliation API Endpoints

Whilst the multi-vendor capability introduced by the NPP provides a high level of convenience to drivers, it makes the reconciliation process more complicated. The NPP offers a monthly consolidated reconciliation report. Operators can thus ensure that they have received all the parking fees to which they are entitled from the Service Providers.

For this purpose, all Service Providers must submit detailed reconciliation information to the NPP, which then creates the above-mentioned consolidated view based on this input.

Endpoints

Method	URI	Description
POST	/reconciliation/transactions	send an individual transaction to the NPP (used by Service Providers)
GET	/reconciliation/transactions/*{txnId}*	read-back of a previously sent transactions (optionally used by Service Providers)
POST	/reconciliation/submittals	send a reconciliation period close-out message to the NPP (used by Service Providers)
GET	/reconciliation/submittals/*{submittalId}*	Read back a previously submitted reconciliation data set
DELETE	/reconciliation/submittals/*{submittalId}*	Revoke a previous submittal, typically in preparation of sending a corrected version.
GET	/reconciliation/reports?month=&year=	Retrieve a machine-readable version of the consolidated monthly reconciliation report (used by operators) Expects two mandatory query parameters: • **month**: month of the report (1-12) • **year**: year including century

The API does not offer a PUT method to update a previous submittal. Updates must be provided by first invalidating the original submittal and then sending a new one. This situation should be a rare exception.

Further details regarding the expected contents can be found in a separate document: **Reconciliation Specification**. Additional information for developers is provided in the **NPP Developer Guide** (all documents available on request and in the NPP GitHub repo).

Submittal Methodology

The details of a monthly reconciliation data submittal are defined in the machine-readable API specification as well as the Reconciliation Specification document. The general approach is this:

- SP sends transactions (payment / refund) to the NPP in near real-time
- When the current reconciliation period has ended, the SP sends a close-out message listing all ids of the previously submitted transactions that shall be included in the reconciliation report
- When all close-out messages have been received, the NPP compiles the consolidated reconciliation report. This can then be retrieved by the operator

Operational Data

List of Contacts / Organisations

The NPP offers an endpoint to read a list of organisations known to the NPP (operators, service providers, enforcement system providers).

Endpoints

Method	URI	Description
GET	/contacts?type=*{contactType}*	Read the list of known organisations. By default, all organisations are returned. Optionally, the result can be filtered by *{contactType}* which can be one of: *operator, owner, serviceProvider, enforcementSystemProvider*
GET	/contacts/{contactId}	Retrieve the details of an organisation with the specified *{contactId}*

Observability: Health and Performance API Endpoints

Alerts

The NPP is constantly monitoring pre-defined performance indicators. Events that require connected clients to be notified, will trigger a corresponding alert. An alert can be characterised as follows:

Element	Description
Metric	The metric whose current value triggers the alert.
Condition	The condition that – if true – triggers the alert. A typical example is a count metric exceeding a certain number.
Window	Optionally, the condition may be constrained to a certain time window, e.g. number of specific failures in the past 5 minutes.
Action	When the alert is triggered, this action is performed. In this version of the API, supported actions are sending an email and pushing the event to a client-provided notification endpoint. Alerts can also be pulled via the /events API endpoint.
Frequency	Optionally, an alert can be configured to repeat the associated action at a defined interval as long as its condition remains true.

The NPP differentiates between *one-time alerts* and *repeating alerts*. Whilst one-time alerts do not repeat, repeating alerts can recur as long as the undesirable state persists.

The standard mechanism to share an alert is defined by its action (e.g. send email notification, push to client-provided notification endpoint). In addition, the NPP offers an endpoint for querying alerts/events that have occurred within a specified period of time.

Endpoint

Method	URI	Description
GET	/observability/events	Provides a (paginated) list of events that occurred within the period of time specified via the mandatory **modified-since** HTTP request header. Clients are encouraged to favour the alert push mechanism over this endpoint. For this reason, the pull interface only serves events from up to 2 weeks in the past.
POST	/observability/events	Lets a connected system communicate events that occurred and might be relevant for other connected clients

Subscription to Push Alerts

To subscribe to push alerts, please use the NPP's */webhooks* endpoint. The topics to subscribe to is ***SystemEvents***. Please note that the push mechanism considers client roles and permissions.

> *All connected NPP clients shall subscribe to the SystemEvents PUSH service*

Health and Performance

Besides individual event messages (see “Alerts”), the NPP offers the option of calling up data on the current health and performance status. This information can for instance be used for display on dashboards. It returns a selection of the most relevant metrics.

Currently, the following metrics are provided (subject to adjustment):

- Overall NPP status (OK, degraded, out of service)
- Sub-API status (OK, degraded, out of service) and min/max/average response time, broken down by
  - Inventory API
  - Assigned Rights and Sessions API
  - Push Notifications API (for *Assigned Rights* and *Sessions*)
- Number of *Assigned Rights* received within the last five minutes
  (plus an indication of whether this is within the expected range at the time of inquiry)
- Number of *Sessions* received within the last five minutes
  (plus an indication of whether this is within the expected range at the time of inquiry)
- Number of client errors within the last five minutes
- Number of server errors within the last five minutes

Note: where required, the information returned will take client roles and permissions into consideration. A Service Provider will for instance only see the metrics for *Assigned Rights* and *Sessions* created by themselves whilst an operator will see this information broken down by Service Provider.

Endpoint

Method	URI	Description
GET	/observability/status	Provides a current status snapshot in form of a selection of relevant KPIs/metrics. Please note that this endpoint applies a rate limit on incoming requests (current maximum is 1 in 5 seconds). It is not possible to retrieve historical snapshots.

Authentication

Context

Provision of data to and reading data from the NPP is only available to authenticated users. Client software will always have to provide an up-to-date access token ensuring that communication takes place between known and authorized entities/partners.

Endpoint

Method	URI	Description
PUSH	/auth	Generate a fresh access token for subsequent API calls. Expects the pre-shared ***client_id*** and ***client_secret*** credentials.

Using the API

Authentication

Please see the Client Authorisation section of the Service Specification before reading this section.

Authentication Mechanism

All endpoints of the NPP use OAUTH2-based authentication. Based on pre-shared credentials, Clients will have to obtain an access token. The tokens issued by the NPP's auth endpoint expire after a time specified in the response payload (see example below). After an access token has expired, a new one will have to be requested for subsequent calls.

> *This authentication method replaces the pilot's authentication method (which was based on a static access token provided in a custom HTTP header).*

Pre-shared Credentials

The following credentials will be pre-shared:

- **Application Id**: a unique id identifying the NPP-connected application
- **Client Id**: a client id identifying the caller organisation
- **Client Secret**: a secret to be used for requesting fresh access tokens

Access Token Request

The following example shows how to obtain a new access token.

```
APP_ID=3897bf01-b3fd-4c8f-ad11-0f33b8f717e2
CLIENT_ID=e861c136-7365-4713-8639-8fa4edf9b228
CLIENT_SECRET=b67a33e7-6a4b-4dac-9eeb-5baf720f98ce

curl -s -XPOST -H “Content-type: application/x-www-form-urlencoded” \
-u $CLIENT_ID:$CLIENT_SECRET \
-d “grant_type=client_credentials” \
https://api.npp.org.uk/v4/auth
```

If the credentials are valid, the NPP will respond with an access token in the response:

```
{
    “issued_at”: “1744780635”,
    “client_id”: “e861c136-7365-4713-8639-8fa4edf9b228”,
    “access_token”: “8bc41e2f-5d6a-4dd8-bacd-e37d68356b13”,
    “scope”: “npp:api”,
    “expires_in”: “86400”,
    “status”: “approved”
}
```

For calls to the NPP API endpoints, a client will have to use the value of the *access_token* property in the example above in the HTTP *Authorization* header (as a *bearer* token).

Note: only the *access_token*, *scope*, and *expires_in* properties can be safely expected in the response payload.

Hostnames

The NPP exposes its API under the fully qualified domain name

`api.npp.org.uk`

The NPP test system (used during the onboarding of new clients and for testing of new software versions) is available under

`staging-api.npp.org.uk`

Please make sure to always use the FQDN, refrain from using IP addresses as they might be subject to change. You will be issued separate sets of credentials for the different environments.

Content Type

This version of the NPP API expects and provides JSON-formatted payloads. Other content types are currently not supported. As this might change in the future, it is considered best practice to always provide the *Content-type:* and *Accept:* http content type negotiation headers.

Versioning

This document specifies the NPP API that is based on APDS v4. When using the API, this is reflected by the "/v4" path prefix for all URLs. For clients who are still using the first version of the NPP API (introduced in the Early Access phase), the "/v1" path prefix applies. For a limited period of time (approximately 1 year), both API versions will be supported in parallel. All new clients must however implement the APDS v4 version.

Example for reading place details using the (soon-to-be-deprecated) original version:

`GET https://api.npp.org.uk/`**`v1`**`/parking/places/909007`

APDS v4 API equivalent:

`GET https://api.npp.org.uk/`**`v4`**`/parking/places/909007`

Custom Request Headers

Custom HTTP Request Headers (to be sent by Clients)

X-Client Version *(mandatory)*

Over time, a client might be using different versions of their own, NPP-facing application. In order to have some level of visibility for this on the NPP side, clients are expected to provide the name and version of their application in a custom header named “X-Client-Version”.

Example:

```
X-Client-Version: SUPERPARK v1.3
```

X-Correlation-ID *(recommended)*

To support the process of trouble-shooting, it is considered good practice to provide a unique correlation id, especially when sending new/updated data to the NPP. This header is optional, but in case of problems, cross referencing between the two communicating system will be a lot easier.

Example:

```
X-Correlation-ID: cd2c003c-ada5-43b5-b8e7-cd7c8a8286ad
```

Custom HTTP Request Headers (sent by the NPP)

X-Correlation-ID

To support the process of trouble-shooting, all NPP responses will contain a unique correlation id. In case of responses, this will be the client-generated correlation id taken from the original request. In case of NPP-initiated requests, it will be an NPP-generated correlation id.

Example:

```
X-Correlation-ID: 0c8ef714-230f-4b0b-b5a0-d2d057da74a6
```

X-Event-Type *(NPP notification service only)*

For convenience purposes, the NPP push notification service (used to send event records to subscribers) adds a request header indicating the type of event. This allows clients to – instead of providing per-event-type endpoints – use a single endpoint to receive events and still be able to determine the type of event received.

Example:

```
X-Event-Type:  SessionCreated
```

Retry Policy

If a client fails to send a new or updated record to the NPP (i.e. receives back a bad status code), this operation shall be retried. Each client shall have – and communicate - a retry policy which ensures that

- a defined number of retry attempts is made (however not infinite)
- the interval between retries is incremented from very short (ms) to long

The NPP's PUSH service (information sent by the NPP to subscribers) also adheres to this convention.

Annexes

Annex 1: Open API Specification

The Open API specification of the NPP API uses Open API version 3.1. This machine-processible version of the API specification is meant for use by developer teams and can, for example be used to automatically generate boilerplate code. The most recent version can be downloaded from https://docs.npp.org.uk/apiv4.

Annex 2: Document Management

Date	Version	Author(s)	Description
22/05/2025	1.0	Markus Schneider, Keith Williams	Initial Version
03/07/2025	2.0	Emily Williams	Corrected typo for PODBA- now reads Pay on Departure instead of Pay on Arrival
17/07/2025	2.1	Markus Schneider	Revised Reconciliation Section
23/07/2025	2.2	Markus Schneider	• Add /contacts endpoint (Operational Data) • Add GET /transactions/{txnId} endpoint (Reconciliation)