Standaard voor Publicatie Dynamische Parkeerdata

Standard for Publishing Dynamic Parking Data

2014-02-09
Document version 1.0

DOCUMENT INFORMATION

Title: Standard for Publishing Dynamic Parking Data
Customer:
Author(s): Werkgroep SPDP
Version: 1.0
Date: 2014-02-09
File: Standard for the Publication of Dynamic Parking Data v1.0.docx
Project: Standaard voor Publicatie Dynamische Parkeerdata
Project number:

Document versions:

Version	Date	Author	Comment
0.1	2013-11-19	Erwin Gribnau	Initial version
0.2	2013-11-20	Erwin Gribnau	Detailed HTTP usage
0.3	2013-11-25	Leontien Balkenende	Added data model and message contents
0.4	2013-11-25	Erwin Gribnau	Changes to data model
0.5	2013-11-28	Erwin Gribnau	New document structure
0.6	2013-12-04	Leontien Balkenende	
0.7	2013-12-12	Kees Koster	Added generic project information & document structure
0.8	2013-12-20	Henk den Breejen	Changes to data model, overall-view, and description.
0.9	2014-01-03	Kees Koster	Addition of descriptions and review
1.0	2014-02-09	Kees Koster	Preparation of the initial release version after processing of review comments

TABLE OF CONTENTS

1. INTRODUCTION

1.1 Context

This document has been prepared by the “Werkgroep Standaard Publicatie Dynamische Parkeerdata”. This specification defines a standard for the publication of dynamic parking data in enclosed parking sites.
The standard has been created for the Dutch market by parking technology providers and parking management companies. The working group was initiated by the Netherlands’ Ministry of Infrastructure and the Environment. The working group consisted of representatives of the following organizations:

- P1 (www.p1.nl)
- Q-Park (www.q-park.nl)
- SKIDATA (www.skidata.nl)
- Technolution (www.technolution.eu)
- Schmit (www.schmit.nl)
- Vialis (www.vialis.nl)
- Imtech (www.imtech.nl)
- RDW (www.rdw.nl)
- Netherlands’ Ministry of Infrastructure and the Environment (www.rijksoverheid.nl/ministeries/ienm)

1.2 Purpose of this report

The purpose of the publication of dynamic parking data is to provide information about available parking spaces in enclosed parking facilities (controlled by barriers). For an accurate overview, some (semi-)static data about the facilities must also be provided. The main purpose of the publication is to provide dynamic data on the actual availability of parking spaces for visitors. Visitors are people accessing the parking facility who are not subscription holders.
A public parking facility is to be defined as parking spaces, to be accessed by visitors, passing the entry of the facility, at one common set of restrictions/rates and capacity control. Two separated sections with different restrictions and/or rates and capacity control are to be specified as two parking facilities. For example a facility with a lower deck with height limitations and an upper deck without height limitation should be specified as two facilities.
This technical specification specifies components required for the exchange and use of data in the field of parking. The components include the following:

- Domain model: Data structure and relationships;
- Data content (or data dictionary)
- Communications specification (or communication protocol)
- Licensing for the use of the dynamic parking data

1.3 Structure of this report

This document describes two protocols in one:

1. A protocol to get parking data from parking management systems (PMS’s) to central parking data servers (CPDS’s), called the ‘push protocol’;
2. A protocol to get parking data from CPDS’s to website and app developers, called the ‘pull protocol’.

The standard uses a domain model, which has been based on two sources:

- PRIS domain model
- Draft DATEX II standard for parking information

1.4 Reading guide

Examples of messages are formatted in a fixed width font, e.g.

```
PUT /parkingdata/v1/dynamic/<UUID>/ HTTP/1.1
Content-Type: application/json
Content-Length: ...
{
   ...
}
```

Within these blocks text marked with < and > and triple dots (…) are entries meant to be replaced by the producing software.

1.5 Acknowledgements

The standard was prepared with the valuable input of all members of the working group.
The contribution of Leontien Balkenende of Technolution in preparing the UML diagrams of the chosen domain model is gratefully acknowledged.

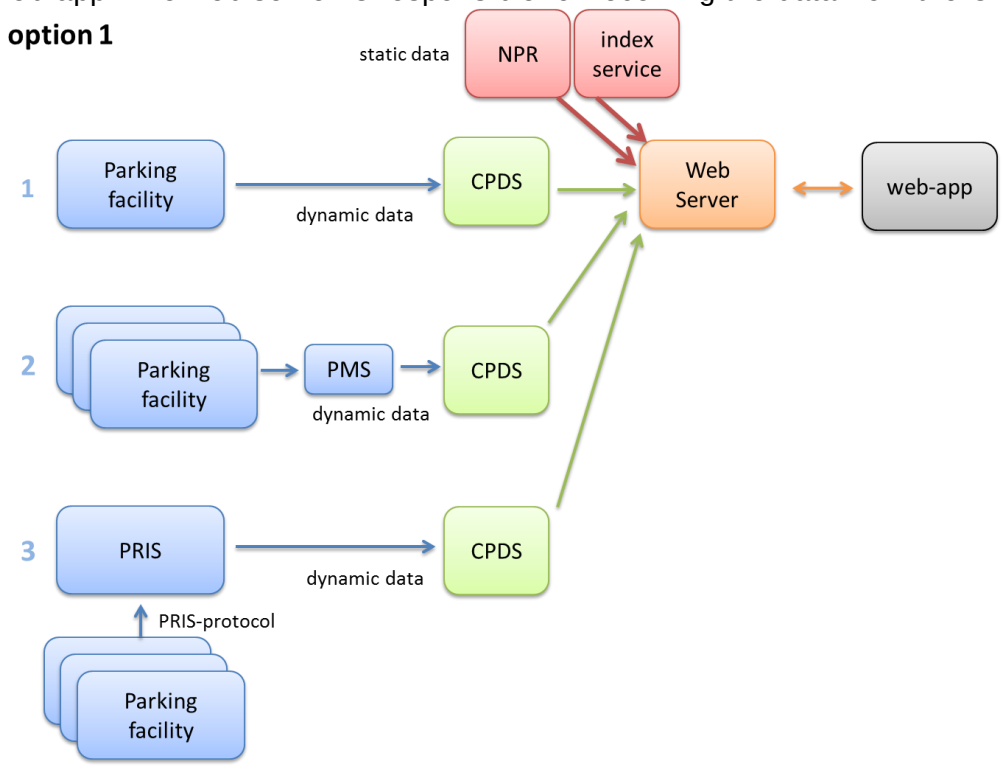
2. OVERALL VIEW

There are several options to use the static and dynamic parking data. In the sections below we visualize possible options of usage.
It is important to note that parking data providers will likely offer fewer outputs than these three usage options. Specifically option 2 is a less likely option to be offered.

Parking facilities provide dynamic data to Central Parking Data Servers (CPDS). Then the data is available for publishing in several ways. To give a clear view we visualize the complete information chain.

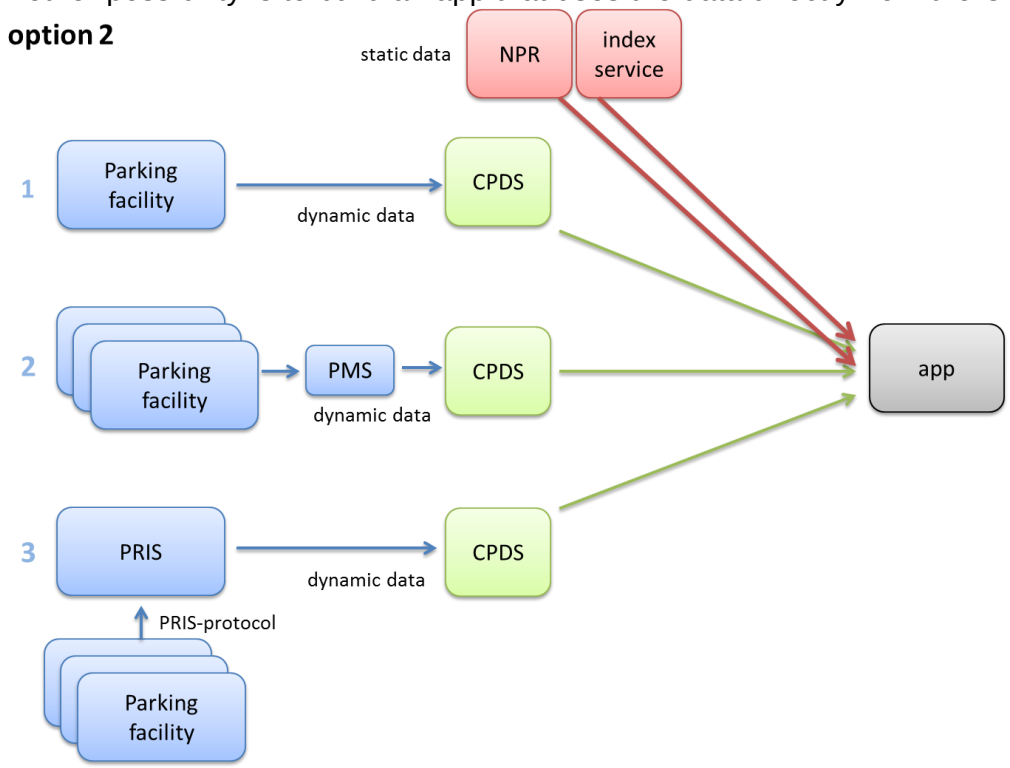
2.1 Option 1: Using a web server and a web-app

One possibility is to build a web-app and use a web server to bring the content to a web-app. The web server is responsible for receiving the data from the CPDS.



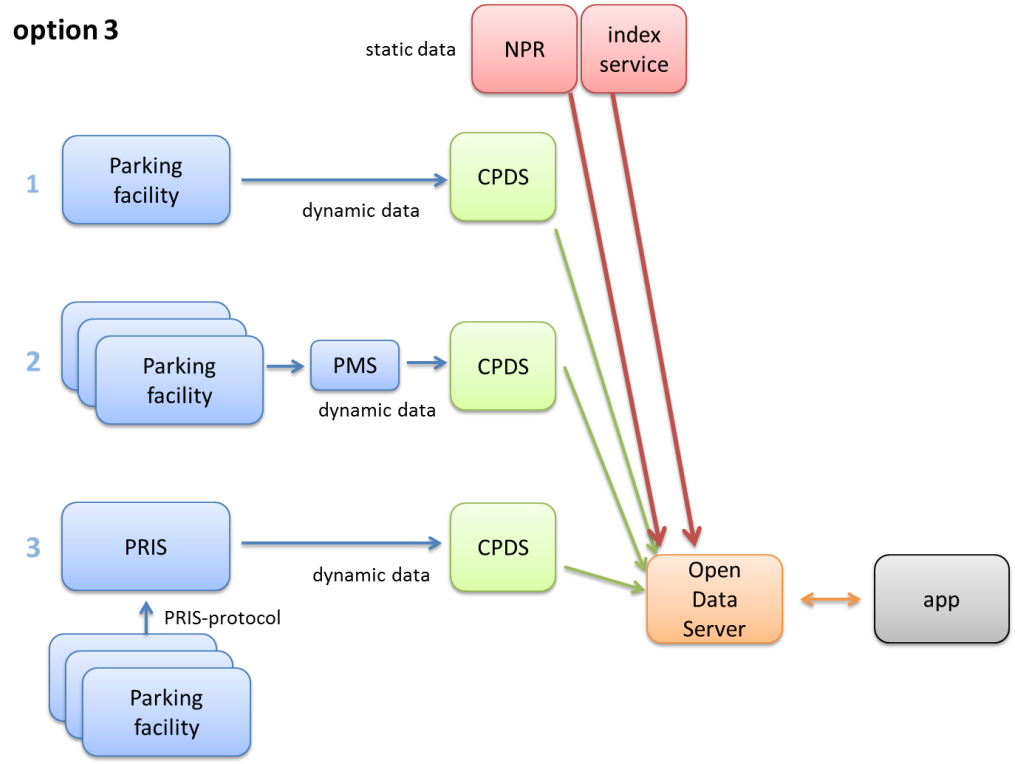
2.2 Option 2: Using an app directly

Another possibility is to build an app that uses the data directly from the CPDS.



2.3 Option 3: Using an Open Data Server and an app

In the third example there is an extra Open Data Server which is used by apps to retrieve data. The Open Data Server is responsible for receiving the data from the CPDS.



3. SYMBOLS AND ABBREVIATIONS

Abbreviation	Meaning
CPDS	Central Parking Data Server
HTTP	Hypertext Transfer Protocol
JSON	JavaScript Object Notation
NPR	“Nationaal Parkeer Register”
PMS	Parking Management System
PRIS	Parking Route Information System
SPDP	“Standaard voor Publicatie Dynamische Parkeerdata”
SSL	Secure Sockets Layer
UML	Unified Modeling Language
URL	Uniform Resource Locator
UUID	Universally Unique Identifier

4. BASIC TECHNOLOGY CHOICES

4.1 Transfer protocol

All data will be transferred using an HTTP connection. To query (‘pull’) data, clients execute an HTTP GET request. To push data, clients execute an HTTP PUT request.

4.2 Data format

The content type for all transferred data is ‘application/json’.

4.3 Roles

In the push protocol the PMS is an HTTP client, CPDS is an HTTP server. In the pull protocol the website or app is an HTTP client, CPDS is an HTTP server.

4.4 Authentication

To authenticate clients to servers (can be used for both push and pull), the connection can be secured using SSL. The client is identified and authenticated using HTTP basic authentication.
HTTP basic authentication uses a combination of username and password, which, in this case, is sent through an encrypted connection. It is advisable that the CPDS system enforces the use of strong passwords.

4.5 Identification

Names of parking facilities are not globally unique. Still, pieces of data need to be matched to other data using a form of unique identification. A central parking data server needs to match incoming dynamic data with the already known static data. This protocol chooses to use a UUID[1] to identify parking facilities.
In absence of a central registration party, the source, a PMS, chooses a random UUID for a parking facility once and uses this UUID in all communications thereafter. Using a UUID the chances of two parking facilities having the same UUID are negligible.
If a central registration party is present, this party can provide a UUID for a parking facility or simply validate that there are no UUID clashes.[2]
All popular programming languages support choosing a random UUID.

4.6 Versioning

The URL’s for all HTTP requests contain a version number to identify the version of the protocol used. This document describes version 1. URL’s contain ‘v1’ in their paths.

[1] http://en.wikipedia.org/wiki/Universally_unique_identifier
[2] UUID must be delivered by the parking operator. In NPR it is delivered by the Cities.

5. DOMAIN MODEL

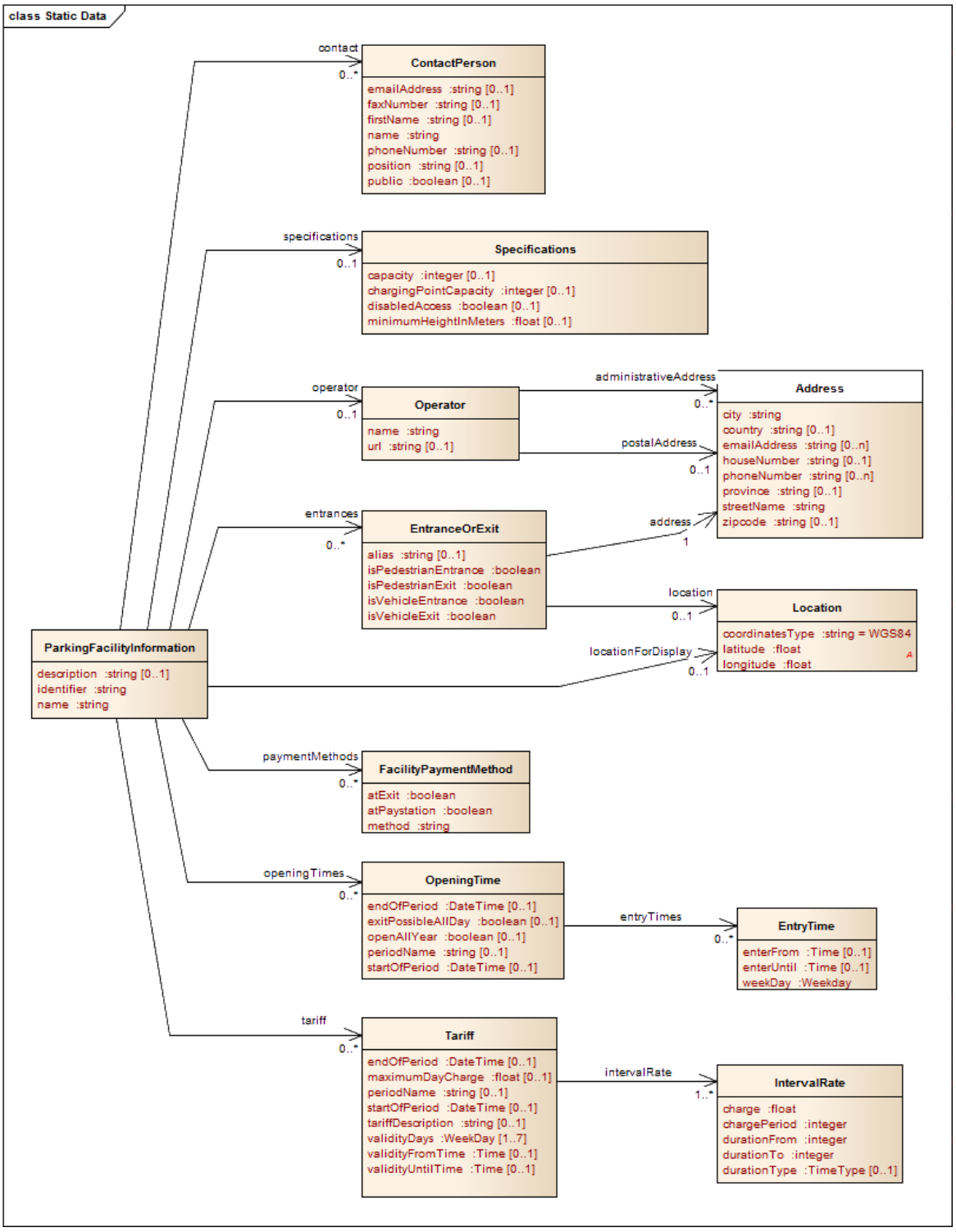
This chapter describes the domain model for transferring static and dynamic parking facility data. UML diagrams are used to show the structure and details of the model. Optional attributes and relations are marked with [0..1]. Attributes and relations that can have more than one value are marked with [0..*] or [0..n].

5.1 Complex data types

This chapter using some specific complex data types. They are explained in the following table:

Data type	Description
DateTime	A type containing a date and a time on that date.
Weekday	A day of the week. Names in the enumeration have been chosen in accordance with the standard the HTTP standard (see IETF RFC 0822, Standard for ARPA Internet Text Messages, paragraph 5.1).
Time	A type containing the hour, minutes and seconds (on any day).
TimeType	Type for indication of time granularity containing a string. Possible values are {“Seconds”, “Minutes”, “Hours”, “Days”, “Weeks”}.

5.2 Static data



5.2.1 ParkingFacilityInformation

Field	Description
description	Description of the parking facility
identifier	UUID of the parking facility
name	Name of the parking facility
entrances	List of entrances and/or exits of the parking facility
operator	Responsible operator of the parking facility
paymentMethods	List of accepted payment methods
openingTimes	List of availability hours to enter by car
tariff	List of tarification for visitors
specifications	Specifications and limitations of the parking facility
contact	List of contact persons for the parking facility
locationForDisplay	Location of the parking facility for displaying on maps (preferred in WGS84)

Field	Field type	Minimum occurrences	Maximum occurrences
description	xs:string	0	1
identifier	xs:string	1	1
name	xs:string	1	1
entrances	EntranceOrExit	0	unbounded
operator	Operator	0	1
paymentMethods	FacilityPaymentMethod	0	unbounded
openingTimes	OpeningTime	0	unbounded
tariff	Tariff	0	unbounded
specifications	Specifications	0	1
contact	ContactPerson	0	unbounded
locationForDisplay	Location	0	1

5.2.2 EntranceOrExit

Field	Description
alias	Name of the entrance and/or exit
isVehicleEntrance	Indicates whether it is possible for vehicles to entry the parking facility
isVehicleExit	Indicates whether it is possible for vehicles to exit the parking facility
isPedestrianEntrance	Indicates whether it is possible for pedestrians to entry the parking facility
isPedestrianExit	Indicates whether it is possible for pedestrians to exit the parking facility
address	Address of the entrance and/or exit
location	Location of the entrance or exit

Field	Field type	Minimum occurrences	Maximum occurrences
alias	xs:string	0	1
isVehicleEntrance	xs:boolean	1	1
isVehicleExit	xs:boolean	1	1
isPedestrianEntrance	xs:boolean	1	1
isPedestrianExit	xs:boolean	1	1
address	Address	1	1
location	Location	0	1

5.2.3 Operator

Field	Description
name	Name the operator of the parking facility
administrativeAddress	List of administrative addresses of the operator of the parking facility
postalAddress	Postal address of the operator of the parking facility

emailAddress	E-mail address of the operator of the parking facility
url	Website address of the operator of the parking facility

Field	Field type	Minimum occurrences	Maximum occurrences
name	xs:string	1	1
administrativeAddress	Address	0	unbounded
postalAddress	Address	0	1
url	xs:string	0	1

5.2.4 Contact person

Field	Description
emailAddress	E-mail address of the contact person
faxNumber	Fax number of the contact person
firstName	First name of the contact person
name	Name of the contact person
phoneNumber	Telephone number of the contact person
position	Function of the contact person
public	Indicates whether the contact details of the contact person can be made public or not

Field	Field type	Minimum occurrences	Maximum occurrences
emailAddress	xs:string	0	1
faxNumber	xs:string	0	1
firstName	xs:string	0	1
name	xs:string	1	1
phoneNumber	xs:string	0	1
position	xs:string	0	1
public	xs:boolean	0	1

5.2.5 Specifications

Field	Description
capacity	Total number of available parking spaces in the parking facility
chargingPointCapacity	Total number of available parking spaces with a charging point in the parking facility
disabledAccess	Possible access for disabled people (wheelchair)
minimumHeightInMeters	Lowest height in the facility in meters

Field	Field type	Minimum occurrences	Maximum occurrences
capacity	xs:integer	0	1
chargingPointCapacity	xs:integer	0	1
disabledAccess	xs:boolean	0	1
minimumHeightInMeters	xs:float	0	1

5.2.6 OpeningTimes

Field	Description
endOfPeriod	End date and time of the opening times period
exitPossibleAllDay	Indicates whether it is possible to exit parking facility all day. Implicitly, when exit is not possible all day, exit is possible in the same times as entry (as specified in the list of entry times.
openAllYear	Indicates whether the parking facility is open all year
periodName	Name of opening times period
startOfPeriod	Start date and time of the opening times period
entryTimes	List of entry times

Field	Field type	Minimum occurrences	Maximum occurrences
endOfPeriod	xs:dateTime	0	1
exitPossibleAllDay	xs:boolean	0	1
openAllYear	xs:boolean	0	1
periodName	xs:string	0	1
startOfPeriod	xs:dateTime	0	1
entryTimes	EntryTime	0	unbounded

5.2.7 EntryTime

Field	Description
enterFrom	Start time (of day) for possible entry
enterUntil	End time (of day) for possible entry
weekday	List of weekdays for which this entry period is valid

Field	Field type	Minimum occurrences	Maximum occurrences
enterFrom	xs:time	0	1
enterUntil	xs:time	0	1
weekDay	xs:string	1	1

5.2.8 Tariff

Field	Description
endOfPeriod	End date and time of the tariff period
maximumDayCharge	Maximum rate for a day
periodName	Name of tariff period
startOfperiod	Start date and time of the tariff period
tariffDescription	Description of the tariff period
validityDays	List of weekdays for which this tariff period is valid
validityFromTime	Start time (of day) of tariff
validityUntilTime	End time (of day) of tariff
intervalRate	Rate of the tariff period

Field	Field type	Minimum occurrences	Maximum occurrences
endOfPeriod	xs:dateTime	0	1
maximumDayCharge	xs:float	0	1
periodName	xs:string	0	1
startOfPeriod	xs:dateTime	0	1
tariffDescription	xs:string	0	1
validityDays	xs:string	1	7
validityFromTime	xs:time	0	1
validityUntilTime	xs:time	0	1
IntervalRate	IntervalRate	1	unbounded

5.2.9 IntervalRate

Field	Description
charge	Rate in euros
chargePeriod	Length of period for the given charge
durationFrom	Interval start indicator
durationTo	Interval end indicator (99999 means unlimited)
durationType	Time specification for this interval, of type TimeType (i.e. seconds, minutes, hours, days or weeks)

Field	Field type	Minimum occurrences	Maximum occurrences
charge	xs:float	1	1
chargePeriod	xs:integer	1	1
durationFrom	xs:integer	1	1
durationTo	xs:integer	1	1
durationType	xs:string	0	1

5.2.10 FacilityPaymentMethod

Field	Description
atExit	Indicates whether this payment method is available at the exits
atPaystation	Indicates whether this payment method is available at the pay stations
method	Description of the tariff method The following standard payment methods should at the minimum be recognized: • "Coins", • "Banknotes", • "Maestro", • "VPay", • "MasterCard", • "AMEX", • "Visa" It is allowed to give other payment method names, however these might not be recognized by users of the data.

Field	Field type	Minimum occurrences	Maximum occurrences
atExit	xs:boolean	1	1
atPaystation	xs:boolean	1	1
method	xs:string	1	1

5.2.11 Address

Field	Description
city	City of address
country	Country of address
emailAddress	Email address of address
houseNumber	House number of address / for postal address also allowed: P.O.Box number
phoneNumber	Telephone number of address
province	Province of address
streetName	Street name of address / for postal address also allowed: P.O.Box
zipcode	Zip code of address

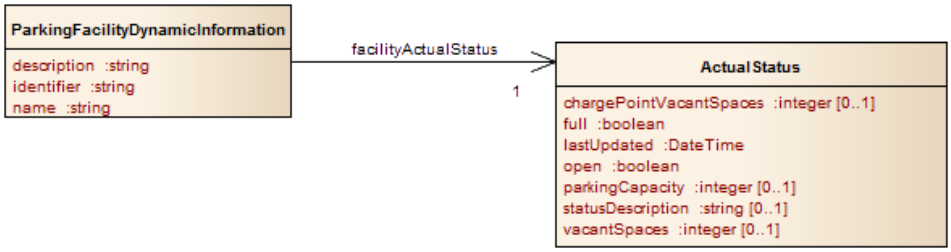
Field	Field type	Minimum occurrences	Maximum occurrences
city	xs:string	1	1
country	xs:string	0	1
emailAddress	xs:string	0	unbounded
houseNumber	xs:string	0	1
phoneNumber	xs:string	0	unbounded
province	xs:string	0	1
streetName	xs:string	1	1
zipcode	xs:string	0	1

5.2.12 Location

Field	Description
coordinatesType	Type of coordinate system, preferably WGS84
latitude	Latitude of coordinate in decimal degrees
longitude	Longitude of coordinate in decimal degrees

Field	Field type	Minimum occurrences	Maximum occurrences
coordinatesType	xs:string	1	1
latitude	xs:float	1	1
longitude	xs:float	1	1

5.3 Dynamic data

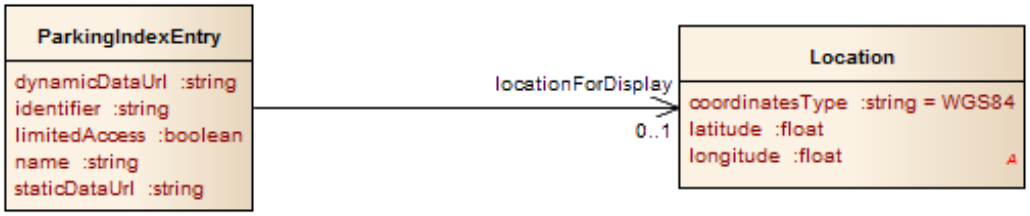


5.3.1 ActualStatus

Field	Description
chargePointVacantSpaces	Number of vacant parking spaces with a charge point
full	Indicates whether the facility currently has vacant spaces for visitors
lastUpdated	Timestamp of last update
open	Indicates whether the facility is currently open for visitors
parkingCapacity	Number of parking spaces for area, facility of assigned parking. This capacity may be dynamic, e.g. when a dynamic allocation is made for subscription holders. This number therefore signifies the total number of parking spaces (occupied and non-occupied) for visitors.
statusDescription	Explanation for the current actual status
vacantSpaces	Number of vacant spaces for area, facility or assigned parking

Field	Field type	Minimum occurrences	Maximum occurrences
chargePointVacantSpaces	xs:integer	0	1
full	xs:boolean	1	1
lastUpdated	xs:dateTime	1	1
open	xs:boolean	1	1
parkingCapacity	xs:integer	0	1
statusDescription	xs:string	0	1
vacantSpaces	xs:integer	0	1

5.4 Index data



5.4.1 ParkingIndexEntry

Field	Description
dynamicDataUrl	URL that must be used to retrieve the dynamic data of the parking facility
identifier	UUID of the parking facility
limitedAccess	Indicates whether authentication is necessary to request data for this parking facility
name	Name of the parking facility
staticDataUrl	URL that must be used to retrieve the static data of the parking facility
locationForDisplay	Location of the parking facility for displaying on maps (preferred in WGS84)

Field	Field type	Minimum occurrences	Maximum occurrences
dynamicDataUrl	xs:string	1	1
identifier	xs:string	1	1
limitedAccess	xs:boolean	1	1
name	xs:string	1	1
staticDataUrl	xs:string	1	1
locationForDisplay	Location	0	1

6. MAPPING DATA TO JSON

JSON is a simple data serialization format, having only a few basic types. This chapter contains the mapping of the types in the data model to JSON.

Type	Mapping to JSON	Restrictions
float	number	
integer	number	Only integers used
boolean	boolean literal 'true' or 'false'	
WeekDay	string	Restricted to the values in the enumeration according to IETF RFC 0822, Standard for ARPA Internet Text Messages, paragraph 5.1
DateTime	number	As seconds since Unix epoch[3]
string	string	
Time	object containing numbers for hours, minutes and seconds time: { h: 13, m: 12, s: 23 }	All values are integer only. h [0-23] m [0-59] s [0-59]

[3] See http://en.wikipedia.org/wiki/Unix_time. This format is supported in all major programming languages, see http://www.epochconverter.com/ for examples.

7. PUSH PROTOCOL

A PMS pushes parking facility data to a CPDS by issuing a HTTP PUT request. Two request types are needed:

1. one to push static data from the PMS to the CPDS;
2. one to push dynamic data from the PMS to the CPDS.

Typically, CPDS systems will want to authenticate incoming requests.
Authentication is done using the mechanism mentioned in paragraph 4.4.

7.1 Pushing static data

7.1.1 Client request

7.1.1.1 Message content

Field	Type
parkingFacility	ParkingFacilityInformation

7.1.1.2 JSON example

```
PUT /parkingdata/v1/static/<UUID>/ HTTP/1.1
Content-Type: application/json
Content-Length: ...
{
   parkingFacility: {
    "description": "Delft, Phoenixgarage",
    "identifier": "DELFT_01",
    "name": "Phoenixgarage",
    "entrances": {
      "alias": "Phoenix",
      "isPedestrianEntrance": "1",
      "isPedestrianExit": "1",
      "isVehicleEntrance": "1",
      "isVehicleExit": "1",
      "address": {
        "city": "Delft",
        "country": "Nederland",
        "streetName": "Phoenixstraat"
      },
      "location": {
        "coordinatesType": "WGS84",
        "latitude": "52.010781",
        "longitude": "43.54725"
      }
    },
    "operator": {
      "operatorName": "GemeenteDelft",
      "url": "www.delft.nl",
      "postalAddress": {
        "city": "Delft",
        "country": "Nederland",
        "emailAddress": "info@delft.nl",
        "houseNumber": "16",
        "phoneNumber": "015 260 22 22 ",
        "province": "ZH",
        "streetName": "Phoenixstraat",
        "zipcode": "2611 AL"
      },
      "administrativeAddress": {
        "city": "Delft",
        "streetName": "Phoenixstraat"
      }
    },
    "paymentMethods": [
      {
        "atExit": "true",
        "atPaystation": "true",
```

```
      "method": "Visa"
    },
    {
      "atExit": "false",
      "atPaystation": "true",
      "method": "Coins"
    }
  ],
  "openingTimes": {
    "endOfPeriod": "2014-12-31T00:00:00.000",
    "exitPossibleAllDay": "1",
    "openAllYear": "1",
    "periodName": "Hele jaar",
    "startOfPeriod": "2014-01-01T00:00:00.000",
    "entryTimes": [
      {
        "enterFrom": "00:00:00.000",
        "enterUntil": "23:59:59.000",
        "weekDay": "1"
      },
      {
        "enterFrom": "00:00:00.000",
        "enterUntil": "23:59:59.000",
        "weekDay": "2"
      },
      {
        "enterFrom": "00:00:00.000",
        "enterUntil": "23:59:59.000",
        "weekDay": "3"
      },
      {
        "enterFrom": "00:00:00.000",
        "enterUntil": "23:59:59.000",
        "weekDay": "4"
      },
      {
        "enterFrom": "00:00:00.000",
        "enterUntil": "23:59:59.000",
        "weekDay": "5"
      },
      {
        "enterFrom": "00:00:00.000",
        "enterUntil": "23:59:59.000",
        "weekDay": "6"
      },
      {
        "enterFrom": "00:00:00.000",
        "enterUntil": "23:59:59.000",
        "weekDay": "7"
      }
    ]
  },
  "tariff": {
    "endOfPeriod": "2014-12-31T00:00:00.000",
    "maximumDayCharge": "24",
    "periodName": "…",
    "startOfPeriod": "2014-01-01T00:00:00.000",
    "tariffDescription": "…",
    "validityDays": [
      "1",
      "2",
      "3",
      "4",
      "5"
    ],
    "validityFromTime": "00:00:00.000",
    "validityUntilTime": "23:59:59.000",
    "IntervalRate": {
      "charge": "0.2",
      "chargePeriod": "10",
      "durationFrom": "0",
      "durationTo": "180",
```

```
        "durationType": "Minutes"
      }
    },
    "specifications": {
      "capacity": "202",
      "chargingPointCapacity": "4",
      "disabledAccess": "true",
      "minimumHeightInMeters": "1.8"
    },
    "contact": {
      "emailAddress": "…",
      "faxNumber": "…",
      "firstName": "…",
      "name": "…",
      "phoneNumber": "…",
      "position": "…",
      "public": "true"
    },
    "locationForDisplay": {
      "coordinatesType": "WGS84",
      "latitude": "52.010781",
      "longitude": "43.54725"
    }
  }
}
```

7.1.2 Server response

The server can respond with one of the following HTTP Responses:

- Status-Code 200 (‘Ok’) if the data was accepted by the server;
- Status-Code 400 (‘Bad request’) if the server finds the request incorrect;
- Status-Code 401 (‘Unauthorized’) if the client is not authorized to push this data.

7.2 Pushing dynamic data

7.2.1.1 Message content

Field	Type
status	ActualStatus

7.2.1.2 JSON example

```
PUT /parkingdata/v1/dynamic/<UUID>/ HTTP/1.1
Content-Type: application/json
Content-Length: ...
{
   status:
      {
         lastUpdated: 1386166308,
         open: true,
         full: false,
         statusDescription: "...",
         parkingCapacity: 250,
         vacantSpaces: 123,
         chargePointVacantSpaces: 0
      }
}
```

7.2.2 Server response

The server can respond with one of the following HTTP Responses:

- Status-Code 200 (‘Ok’) if the data was accepted by the server;
- Status-Code 400 (‘Bad request’) if the server finds the request incorrect.
- Status-Code 401 (‘Unauthorized’) if the client is not authorized to push this data;

8. PULL PROTOCOL

A website or app pulls static parking facility data from the CPDS by issuing a HTTP GET request. Three request types are needed:

1. one to get a list of known parking facilities and the location (URL's) of the associated static and dynamic data;
2. one to get the static data of a parking facility;
3. one to get the dynamic data of a parking facility.

8.1 Pulling a list of known parking facilities

The purpose of this message is to allow website and app developers to get a list of all known facilities, filter it using a geographical filter, and then only pull the data of the facilities of interest.
The response also contains URL's where the static and dynamic data can be obtained. This allows for separation of indexing servers and data servers and for separation of static data servers and dynamic data servers.
The response also contains an field ('limitedAccess') to indicate whether accessing the static and dynamic data of that facility requires authentication.

8.1.1 Client request

```
GET /parkingdata/v1/ HTTP/1.1
Content-Type: application/json
```

8.1.2 Server response

The server can respond with one of the following HTTP Responses:

- Status-Code 200 ('Ok') if the request was accepted by the server and ;
- Status-Code 400 ('Bad request') if the server finds the request incorrect;
- Status-Code 401 ('Unauthorized') if the client is not authorized to get this data.

8.1.2.1 Message content

Field	Type
parkingFacilities	ParkingIndexEntry

8.1.2.2 JSON example of response

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: ...
{
   parkingFacilities: [
      {
         name: "...",
         identifier: "...",
         limitedAccess: true
         locationForDisplay: {
            {
               coordinatesType: "WGS84"
               latitude: 52.326701
               longitude: 4.98765
            },
         staticDataUrl: "http://...",
         dynamicDataUrl: "http://..."
      },
      {
         name: "...",
         identifier: "...",
         limitedAccess: true
         locationForDisplay: {
```

```
            {
               coordinatesType: "WGS84"
               latitude: 52.026701
               longitude: 5.18765
            },
         staticDataUrl: "https://...",
         dynamicDataUrl: "https://..."
      }
   ]
}
```

8.2 Pulling the static data of a parking facility

This request can be subject to authentication. Authentication is done using the mechanism mentioned in paragraph 4.4.

8.2.1 Client request

```
GET <URL OBTAINED FROM LIST> HTTP/1.1
Content-Type: application/json
```

8.2.2 Server response

The server can respond with one of the following HTTP Responses:

- Status-Code 200 (‘Ok’) if the request was accepted by the server and ;
- Status-Code 400 (‘Bad request’) if the server finds the request incorrect.
- Status-Code 401 (‘Unauthorized’) if the client is not authorized to get this data;
- Status-Code 404 (‘Not found') if the server has no static data for this parking facility;

8.2.2.1 Message content of response

Field	Type
parkingFacility	ParkingFacilityInformation

8.2.2.2 JSON example

See example for pushing static data.

8.3 Pulling the dynamic data of a parking facility

This request can be subject to authentication. Authentication is done using the mechanism mentioned in paragraph 4.4.

8.3.1 Client request

```
GET <URL OBTAINED FROM LIST> HTTP/1.1
Content-Type: application/json
```

8.3.2 Server response

The server can respond with one of the following HTTP Responses:

- Status-Code 200 (‘Ok’) if the request was accepted by the server and ;
- Status-Code 400 (‘Bad request’) if the server finds the request incorrect.
- Status-Code 401 (‘Unauthorized’) if the client is not authorized to get this data;
- Status-Code 404 (‘Not found') if the server has no static data for this parking facility;

8.3.2.1 Message content of response

Field	Type
status	ActualStatus

8.3.2.2 JSON example

See example for pushing dynamic data.

9. LICENSING

Data providers may elect to provide the dynamic parking data under license. Therefore, authentication is an integral part of the transport protocol. This standard does not prescribe which license model is the most appropriate for a data provider, nor does it set requirements for the registration procedures employed.