PAWS
Internet-Draft
Intended status: Standards Track
Expires: June 21, 2013

V. Chen, Ed.
Google
S. Das
Applied Communication Sciences
Z. Lei
Huawei
J. Malyar
Telcordia Technologies Inc.
P. McCann
Huawei

December 18, 2012

Protocol to Access Spectrum Database
draft-ietf-paws-protocol-01

Abstract

Portions of the radio spectrum that are allocated to licensees are available for non-interfering use. This available spectrum is called "White Space." Allowing secondary users access to available spectrum "unlocks" existing spectrum to maximize its utilization and to provide opportunities for innovation, resulting in greater overall spectrum utilization.

One approach to manage spectrum sharing uses databases to report spectrum availability to devices. To achieve interoperability among multiple devices and databases, a standardized protocol must be defined and implemented. This document defines such a protocol, the "Protocol to Access White Space database" (PAWS).

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on June 21, 2013.

Table of Contents

1. Introduction 5
2. Conventions and Terminology 5
2.1. Conventions Used in This Document 5
2.2. Terminology 6
3. Protocol Overview 6
4. Protocol Functionalities 7
4.1. Database Discovery 7
4.2. Initialization 7
4.2.1. INIT_REQ 8
4.2.2. INIT_RESP 9
4.3. Device Registration 9
4.3.1. REGISTRATION_REQ 10
4.3.2. REGISTRATION_RESP 10
4.4. Available Spectrum Query 11
4.4.1. AVAIL_SPECTRUM_REQ 13
4.4.2. AVAIL_SPECTRUM_RESP 14
4.4.3. AVAIL_SPECTRUM_BATCH_REQ 15
4.4.4. AVAIL_SPECTRUM_BATCH_RESP 16
4.4.5. SPECTRUM_USE_NOTIFY 17
4.4.6. SPECTRUM_USE_RESP 18
4.5. Device Validation 19
4.5.1. DEV_VALID_REQ 20
4.5.2. DEV_VALID_RESP 20
5. Protocol Parameters 20
5.1. GeoLocation 21
5.2. DeviceDescriptor 22
5.3. AntennaCharacteristics 23
5.4. DeviceCapabilities 24
5.5. DeviceOwner 24
5.6. RulesetInfo 25

5.7. Spectrum 26
5.8. FrequencyRange 27
5.9. EventTime 28
5.10. SpectrumSchedule 28
5.11. GeoSpectrumSchedule 29
5.12. DeviceValidity 29
5.13. Error Element 30
5.13.1. REQUIRED Error 31
6. Message Encoding 31
6.1. JSON-RPC Binding 32
6.2. init Method 33
6.2.1. INIT_REQ Schema 33
6.2.2. INIT_RESP Parameters 34
6.3. register Method 35
6.3.1. REGISTRATION_REQ Parameters 35
6.3.2. REGISTRATION_RESP Parameters 37
6.4. getSpectrum Method 38
6.4.1. AVAILABLE_SPECTRUM_REQ Parameters 38
6.4.2. AVAIL_SPECTRUM_RESP Parameters 40
6.5. getSpectrumBatch Method 43
6.5.1. AVAIL_SPECTRUM_BATCH_REQ Parameters 43
6.5.2. AVAIL_SPECTRUM_BATCH_RESP Parameters 45
6.6. notifySpectrumUse Method 48
6.6.1. SPECTRUM_USE_NOTIFY Parameters 48
6.6.2. SPECTRUM_USE_RESP Parameters 50
6.7. verifyDevice Method 51
6.7.1. DEV_VALID_REQ Parameters 51
6.7.2. DEV_VALID_RESP Parameters 52
6.8. Sub-message Schemas 53
6.8.1. GeoLocation 53
6.8.2. DeviceDescriptor 55
6.8.3. AntennaCharacteristics 56
6.8.4. DeviceCapabilities 57
6.8.5. DeviceOwner 57
6.8.6. RulesetInfo 59
6.8.7. Spectrum 59
6.8.8. FrequencyRange 60
6.8.9. EventTime 61
6.8.10. SpectrumSchedule 62
6.8.11. GeoSpectrumSchedule 63
6.8.12. DeviceValidity 63
6.8.13. Additional Properties 64
7. HTTPS Binding 64
8. Example Messages 65
9. IANA Considerations 65
10. Security Considerations 65
10.1. Assurance of Proper Database 66
10.2. Protection Against Modification 67

10.3. Protection Against Eavesdropping 67
10.4. Client Authentication Considerations 67
11. Contributors 68
12. Acknowledgments 68
13. References 68
13.1. Normative References 68
13.2. Informative References 69
Appendix A. Changes / Author Notes. 70
Appendix B. Regulatory-specific Notes 70
B.1. US / FCC 70
B.2. UK / Ofcom 71
Authors' Addresses 71

1. Introduction

This section provides some high level introductory material. Readers are strongly encouraged to read Protocol to Access White Space database: PS, use cases and rqmts [I-D.ietf-paws-problem-stmt-usecases-rqmts] for use cases, requirements, and additional background.

A geospatial database can track available spectrum (in accordance with the rules of one or more regulatory domains) and make this information available to devices. This approach shifts the complexity of spectrum-policy conformance out of the device and into the Database. This approach also simplifies adoption of policy changes, limiting updates to a handful of databases, rather than numerous devices. It opens the door for innovations in spectrum management that can incorporate a variety of parameters, including user location and time. In the future, it also can include other parameters, such as user priority, time, signal type and power, spectrum supply and demand, payment or micro-auction bidding, and more.

In providing this service, a database records and updates information necessary to protect primary users -- for example, this information may include parameters such as a fixed transmitter's call sign, its geo-location, antenna height, power, and periods of operation. The rules that the Database must follow, including its schedule for obtaining and updating protection information, protection rules, and information reported to devices, vary according to regulatory domain. Such variations, however, should be handled by each database, and exposure to the variations by devices should be minimized.

This specification defines an extensible protocol to obtain available spectrum from a geospatial database by a device with geo-location capability. It enables a device to operate in any regulatory domain that implements the same protocol and in which the device is authorized to operate. The document describes the use of HTTP/TLS as transport for the protocol.

2. Conventions and Terminology

2.1. Conventions Used in This Document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in Key words for use in RFCs to Indicate Requirement Levels [RFC2119].

2.2. Terminology

Database or Spectrum Database: A database that provides spectrum availability information to devices.
Master Device: A device with geo-location capability that queries a database to find available spectrum.
Slave Device: A device without geo-location capability that uses the spectrum made available by a Master Device. It does not query the Database directly.
RAT: Radio Access Technology

3. Protocol Overview

A Master Device uses the PAWS protocol to obtain a schedule of available spectrum at its location. The security necessary to ensure the accuracy, privacy, and confidentiality of the Device's location is described in the Security Considerations (Section 10). This document assumes that the Master Device and the Database are connected to the Internet.

A typical sequence of PAWS operations is outlined as follows. See Protocol Functionalities (Section 4) and Protocol Parameters (Section 5) for details:

1. The Master Device locates or discovers the regulatory domain for its location and the URI for the Database to send subsequent PAWS messages. [Editor's Note: It is an open item whether database discovery should be a separate document.]
2. The Master Device establishes an HTTPS session with the Database.
3. The Master Device optionally sends an initialization message to the Database to exchange capabilities.
4. If the Database receives an initialization message, it responds with a message in the body of the HTTP response.
5. If required by regulatory domain, the Database registers the Master Device.
6. The Master Device sends an available-spectrum request message to the Database.
7. If the Master Device is obtaining the schedule on behalf of a Slave Device, and if required by the regulatory domain, the Database validates the Slave Device.
8. The Database responds with an available-spectrum response message in the body of the HTTP response.
9. Depending on regulatory domain requirements and database implementation, the Master Device sends a spectrum-usage notification message to the Database.

10. If the Database receives a spectrum-usage notification message, it responds by sending the Master Device a spectrum-usage acknowledgement message.

4. Protocol Functionalities

The PAWS protocol consists of several components:

- Database Discovery (Section 4.1) MUST be supported by the Master Device
- Initialization (Section 4.2) MAY be used by the Master Device and MUST be implemented by the Database.
- Device Registration (Section 4.3) MAY be used by the Master Device and MAY be implemented by the Database.
- Available Spectrum Query (Section 4.4) MUST be supported by Master Device and the Database.
- Device Validation (Section 4.5) MAY be used by the Master Device and MUST be implemented by the Database if the regulatory domain requires device validation.

This section describes the protocol components and their messages. Protocol Parameters (Section 5) contains a more thorough discussion of the parameters that comprise the PAWS request and response messages. Message Encoding (Section 6) provides details of the message encodings. HTTPS Binding (Section 7) describes the use of HTTPS HTTP Over TLS [RFC2818] for transporting PAWS messages and optional device authentication.

4.1. Database Discovery

The Device MUST determine the URI for the Database and applicable regulatory domain before it can send PAWS messages. The URI for the Database SHOULD be obtained from an authorized and authenticated entity, but it MAY be statically configured into the Device. [Editor's Note: It is an open item whether database discovery should be a separate document.]

4.2. Initialization

A Master Device SHOULD use the initialization procedure to exchange capability information with the Database whenever the Master Device powers up or initiates communication with the Database. The initialization response informs the Master Device of specific regulatory-dependent parameterized-rule values, such as threshold distances and time periods beyond which the Device must update its available-spectrum data (see RuleSetInfo (Section 5.6)). The Master Device MAY manually configure these parameterized-rule values. The

initialization message also represents extension points for database implementations or regulatory domains that require the extra handshake.

The Initialization request procedure is depicted in Figure 1.

- INIT_REQ (Section 4.2.1) is the initialization request message
- INIT_RESP (Section 4.2.2) is the initialization response message

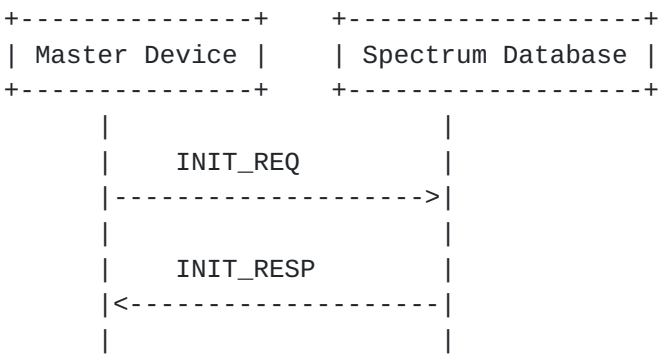


Figure 1

4.2.1. INIT_REQ

The initialization request message allows the Master Device to initiate exchange of capabilities with the Database.

INIT_REQ	
deviceDesc:DeviceDescriptor	required
location:GeoLocation	required
*other:any	depends

Parameters:

deviceDesc: The DeviceDescriptor (Section 5.2) for the Device is REQUIRED. If the Database does not support the regulatory domain specified by the "authority" parameter, it MUST return an error with the UNSUPPORTED (Table 1) code in the error response.

location: The GeoLocation (Section 5.1) for the Device is REQUIRED.

other: Depending on the regulatory domain or database implementation, the Master Device MAY specify additional handshake parameters in the INIT_REQ message. The Database MUST ignore all parameters it does not understand.

4.2.2. INIT_RESP

The initialization response message communicates database parameters to the requesting device.

```
+-----------------------------------+
|INIT_RESP                          |
+---------------------------+-------+
|rulesetInfo:RulesetInfo    | required|
|...................................|
|*other:any                 | depends |
+---------------------------+-------+
```

Parameters:

rulesetInfo: This RulesetInfo (Section 5.6) parameter MUST be included in the response. This parameter specifies the regulatory domain and parameters applicable for that domain. The Database MUST include the "authority" that defines the regulatory domain for the location specified in the INIT_REQ (Section 4.2.1) message.

other: Depending on the regulatory domain or database implementation, the Database MAY include additional handshake parameters in the INIT_RESP (Section 4.2.2) message. The Master Device MUST ignore all parameters it does not understand.

4.3. Device Registration

When a regulatory domain requires registration of a Master Device, the Device MUST send its registration information to the Database to establish certain operational parameters. FCC rules, for example, require that a 'Fixed Device' MUST register its owner and operator contact information, its device identifier, its location, and its antenna height.

The Database MAY support device registration as a separate Device Registration component, or as part of the Spectrum Availability component. If the Database does not support a separate Device Registration request, it MUST return an error with the UNIMPLEMENTED (Table 1) code in the error-response message.

The Device Registration request procedure is depicted in Figure 2.

- REGISTRATION_REQ (Section 4.3.1) is the device-registration request message
- REGISTRATION_RESP (Section 4.3.2) is the device-registration response message

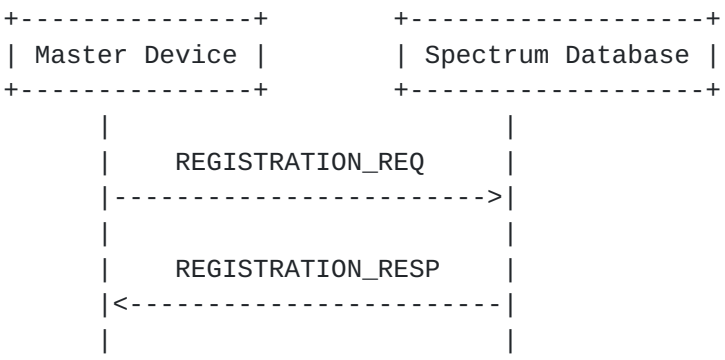


Figure 2

4.3.1. REGISTRATION_REQ

The registration request message contains the required registration parameters.

REGISTRATION_REQ	
deviceDesc:DeviceDescriptor	required
location:GeoLocation	required
deviceOwner:DeviceOwner	required
*other:any	depends

Parameters:

deviceDesc: The DeviceDescriptor (Section 5.2) for the Device is REQUIRED.
location: The GeoLocation (Section 5.1) for the Device is REQUIRED.
deviceOwner: The DeviceOwner (Section 5.5) information is REQUIRED.
other: Regulatory domains MAY require additional registration parameters. To simplify its registration logic, the Device MAY send a union of the registration information required by all supported regulatory domains. The Database MUST ignore all parameters it does not understand .

4.3.2. REGISTRATION_RESP

The registration response message simply acknowledges receipt of the request and is otherwise empty.

REGISTRATION_RESP	

```
+--------------------------+----------+
```

4.4. Available Spectrum Query

To obtain the available spectrum from the Database, a Master Device sends a request that contains its geo-location and any parameters required by the regulatory rules (such as device identifier, capabilities, and characteristics). The Database returns a response that describes what frequencies are available, at what permissible operating power levels, and a schedule of when they are available.

The Available Spectrum Query procedure is depicted in Figure 3.

- AVAIL_SPECTRUM_REQ (Section 4.4.1) is the available-spectrum request message
- AVAIL_SPECTRUM_RESP (Section 4.4.2) is the available-spectrum response message
- AVAIL_SPECTRUM_BATCH_REQ (Section 4.4.3) is an OPTIONAL batch version of the available-spectrum request message that allows multiple locations to be specified in the request
- AVAIL_SPECTRUM_BATCH_RESP (Section 4.4.4) is the response message for the batch version of the available-spectrum request that contains available spectrum for each location
- SPECTRUM_USE_NOTIFY (Section 4.4.5) is the spectrum-usage notification message
- SPECTRUM_USE_RESP (Section 4.4.6) is the spectrum-usage acknowledgment message

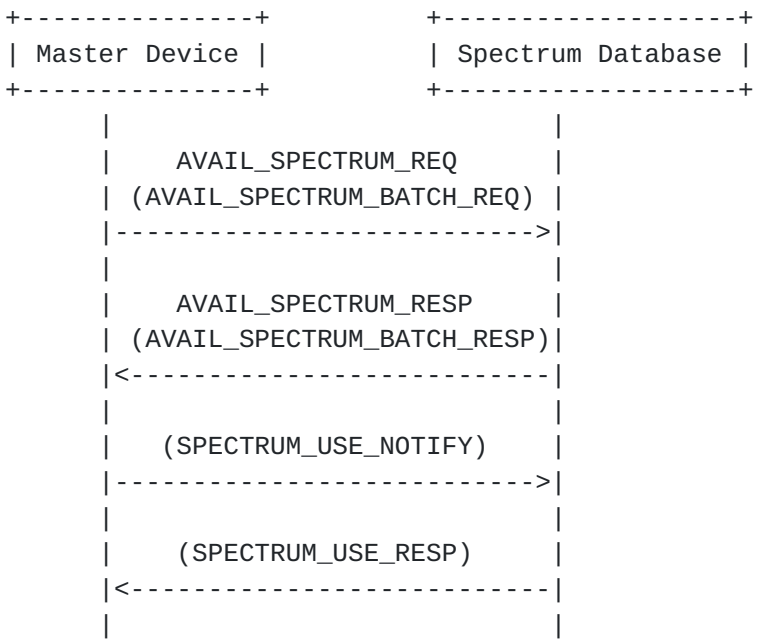


Figure 3

1. First, the Master Device sends an available-spectrum request message to the Database.
2. The Database MUST respond with an error using the NOT_REGISTERED (Table 1) code if:
   * registration information is required, and
   * the request does not include registration information, and
   * the Device has not previously registered
3. If the location specified in the request is outside the regulatory domain, the Database MUST respond with an OUTSIDE_COVERAGE (Table 1) error. If some locations within a batch request are outside the regulatory domain, the Database MAY return an OK response with available spectrum for only the valid locations. If all locations within a batch request are outside the regulatory domain, the Database MUST respond with an OUTSIDE_COVERAGE error.
4. The Database MAY perform other validation of the request, (e.g., checking for missing required parameters, authorizations). It MUST return an error with appropriate error code (Table 1), if validation fails. If the request is missing required parameters, the Database MUST respond with a REQUIRED (Table 1) error and SHOULD include a list of the missing parameters.
5. If the request is valid, the Database responds with an available-spectrum response message. If the regulatory domain requires that devices must report anticipated spectrum usage, the Database MUST indicate so in the response message.
6. If the available-spectrum response indicates that the Master Device must send a spectrum-usage notification message, the Master Device MUST send the notification message to the Database.
7. If the Database receives a spectrum-usage notification message, it MUST send a spectrum-usage acknowledgment message to the Master Device.

The procedure for asking for available spectrum on behalf of a Slave Device is similar, except that the process is initiated by the Slave Device. Also, the device identifier, capabilities, and characteristics communicated in the AVAIL_SPECTRUM_REQ message SHALL be those of the Slave Device. Although the communication and protocol between the Slave Device and Master Device is outside the scope of this document, the expected message sequence is shown in Figure 4.

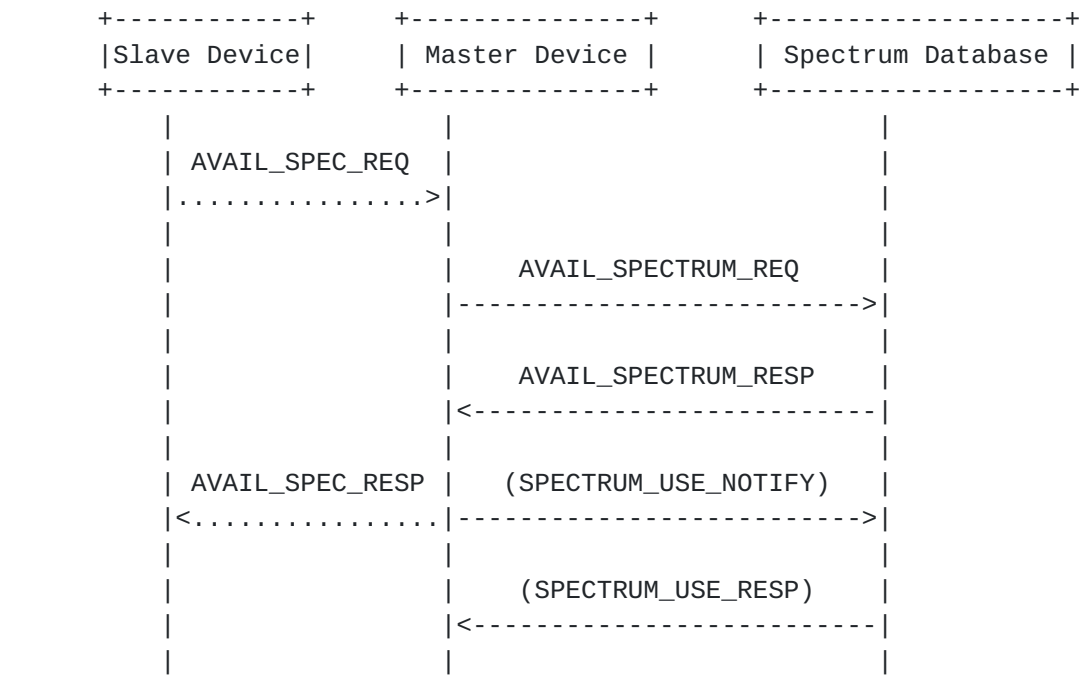


Figure 4

4.4.1. AVAIL_SPECTRUM_REQ

The request message for the Available Spectrum Query protocol MUST include the Device's geo-location. If allowed by the regulatory domain, the location MAY be an anticipated location.

AVAIL_SPECTRUM_REQ	
deviceDesc:DeviceDescriptor	required
location:GeoLocation	required
antenna:AntennaCharacteristics	depends on regulatory domain
owner:DeviceOwner	depends on regulatory domain
capabilities:DeviceCapabilities	optional

Parameters:

deviceDesc: The DeviceDescriptor (Section 5.2) for the Device is REQUIRED.

location: The GeoLocation (Section 5.1) for the Device is REQUIRED. The location SHOULD be the current location of the Device, but more precisely, the location of the radiation center of the Device's antenna. Depending on the regulatory domain, the location MAY be an anticipated position of the Device to support mobile devices. If the location specifies a region, rather than a

point, the Database MAY return an error with the UNIMPLEMENTED (Table 1) code, if it does not support query by region.

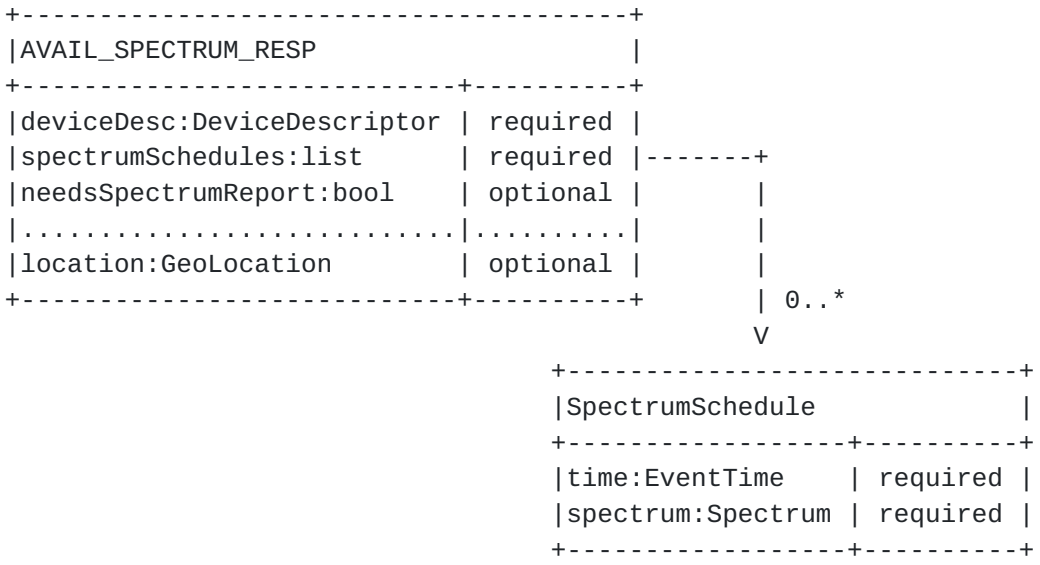
antenna: Depending on the device type and regulatory domain, the AntennaCharacteristics (Section 5.3) MAY be required.

owner: Depending on the device type and regulatory domain, the DeviceOwner (Section 5.5) information MAY be included to register the Device with the Database. This enables the Device to register and get spectrum-availability information in a single request.

capabilities: The Master Device MAY include its DeviceCapabilities (Section 5.4) to limit the available-spectrum response to the spectrum that is compatible with its capabilities. The Database SHOULD NOT return spectrum that is not compatible with the specified capabilities.

4.4.2. AVAIL_SPECTRUM_RESP

The response message for the Available Spectrum Query contains a schedule of available spectrum for the Device.



Parameters:

deviceDesc: The Database MUST include the DeviceDescriptor (Section 5.2) specified in the AVAIL_SPECTRUM_REQ message

spectrumSchedules: The SpectrumSchedule (Section 5.10) list is REQUIRED (though it MAY be empty if no spectrum is available). The Database MAY return more than one SpectrumSchedule (Section 6.8.10) to represent future changes to the available spectrum. How far in advance a schedule may be provided depends on the regulatory domain.

needsSpectrumReport: For regulatory domains that require a spectrum-usage report from devices, the Database MUST return true for this parameter. The default value is false.

location: The Database MAY copy other elements from the request, such as the GeoLocation (Section 5.1) of the Device. The Device MUST ignore any parameters it does not understand.

4.4.2.1. Update Requirements

When the stop time specified in the schedule has been reached, the Device:

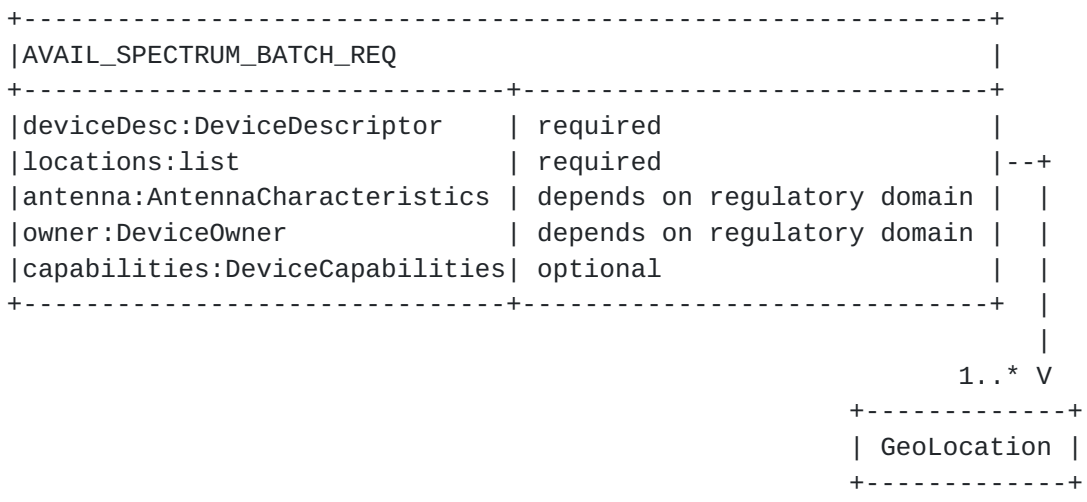
- MUST obtain a new spectrum-availability schedule, either by using the next one in the list (if provided) or making another Available Spectrum Query (Section 4.4)
- If the new schedule indicates the in-use spectrum is no longer available, the Device MUST stop operation immediately.
- If the Device is unable to contact the Database to obtain a new schedule, depending on the regulatory domain, the Device MAY continue to operate for a period of time, as indicated by parameters returned in the INIT_RESP (Section 4.2.2) message.

When the Device moves beyond a threshold distance (established by regulatory rules) away from the actual location and all anticipated location(s) it reported in previous AVAIL_SPECTRUM_REQ or AVAIL_SPECTRUM_BATCH_REQ requests (see "maxLocationChange" in RulesetInfo (Section 5.6)), it:

- MUST obtain a new spectrum-availability schedule by making another Available Spectrum Query (Section 4.4).
- If the new response indicates the in-use spectrum is no longer available, the Device MUST stop operation immediately.
- If the Device is unable to contact the Database to obtain a new schedule, depending on the regulatory domain, the Device MUST stop operation immediately.

4.4.3. AVAIL_SPECTRUM_BATCH_REQ

The Database MAY support the batch request that allows multiple locations to be specified. This allows a portable Master Device to get available spectrum for a sequence of anticipated locations using a single request. The Database MUST interpret each location in the batch request as if it were an independent request and MUST return results consistent with multiple individual AVAIL_SPECTRUM_REQ (Section 4.4.1) requests. The request message for the batch Available Spectrum Query protocol MUST include at least one GeoLocation (Section 5.1). If the Database does not support batch requests, it MUST return a UNIMPLEMENTED (Table 1) error.



Parameters:

deviceDesc: The DeviceDescriptor (Section 5.2) for the Device is REQUIRED.

locations: The GeoLocation (Section 5.1) list for the Device is REQUIRED. This allows the Device to specify its actual location plus additional anticipated locations, when allowed by the regulatory domain. At least one location MUST be included. This specification places no upper limit on the number of locations, but the Database MAY restrict the number of locations it supports by returning a response with fewer locations than specified in the request.

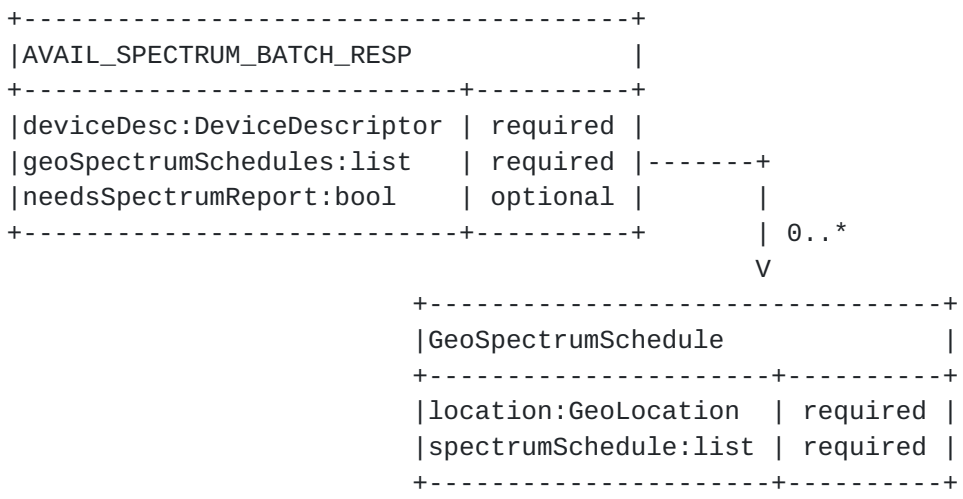
antenna: Depending on the device type and regulatory domain, the AntennaCharacteristics (Section 5.3) MAY be required.

owner: Depending on the device type and regulatory domain, the DeviceOwner (Section 5.5) information MAY be included to register the Device with the Database. This enables the Device to register and get spectrum-availability information in a single request.

capabilities: The Master Device MAY include its DeviceCapabilities (Section 5.4) to limit the available-spectrum response to the spectrum that is compatible with its capabilities. The Database SHOULD NOT return spectrum that is not compatible with the specified capabilities.

4.4.4. AVAIL_SPECTRUM_BATCH_RESP

The response message for the batch Available Spectrum Query contains a schedule of available spectrum for the Device at multiple locations.



Parameters:

deviceDesc: The Database MUST include the DeviceDescriptor (Section 5.2) specified in the AVAIL_SPECTRUM_REQ message

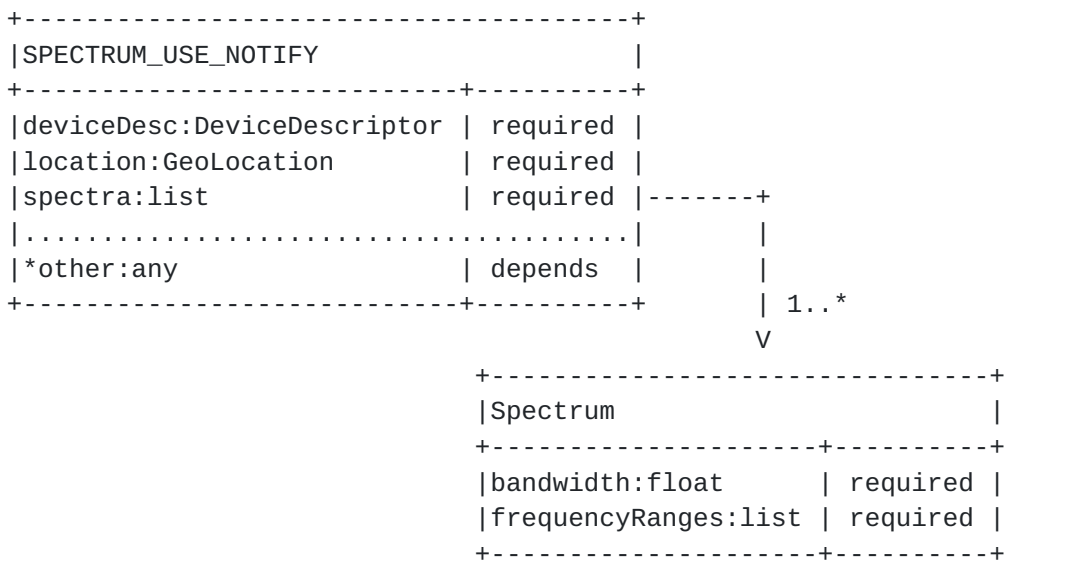
geoSpectrumSchedules: The geoSpectrumSchedule (Section 5.11) list is REQUIRED (though it MAY be empty if spectrum is unavailable). For each location, the Database MAY return more than one GeoSpectrumSchedule (Section 6.8.11) to represent future changes to the available spectrum. How far in advance a schedule may be provided depends on the regulatory domain. The Database MAY return available spectrum for fewer locations than requested. The Device MUST NOT make any assumptions on the order of the entries in the list and MUST use the location value in each GeoSpectrumSchedule entry to match available spectrum to a location.

needsSpectrumReport: For regulatory domains that require a spectrum-usage report from devices, the Database MUST return true for this parameter. The default value is false.

See Update Requirements (Section 4.4.2.1) for when the Device must update its available spectrum data.

4.4.5. SPECTRUM_USE_NOTIFY

The spectrum-use notification message MUST contain the geo-location of the Device and parameters required by the regulatory domain.



Parameters:

deviceDesc: The DeviceDescriptor (Section 5.2) for the Device is REQUIRED.

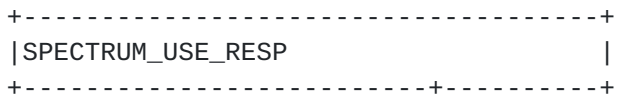
location: The GeoLocation (Section 5.1) for the Device is REQUIRED.

spectra: The Spectrum (Section 5.7) list is REQUIRED, and specifies the spectrum anticipated to be used by the Device, which includes frequency ranges and maximum power levels. For consistency, the "bandwidth" value SHOULD match that from one of the Spectrum (Section 5.7) elements in the corresponding AVAIL_SPECTRUM_RESP message, and the maximum power levels in the Spectrum element MUST be expressed as total power (EIRP) computed over the specified "bandwidth" value. The actual bandwidth to be used (as computed from the start and stop frequencies) MAY be different from the "bandwidth" value. As an example, when regulatory rules express maximum power spectral density in terms of maximum power over any 100 kHz band, then the "bandwidth" value should be set to 100 kHz, even though the actual bandwidth used can be 20 kHz.

other: Depending on the regulatory domain, other parameters MAY be required. To simplify its logic, the Device MAY include the union of all parameters required by all supported regulatory domains. The Database MUST ignore all parameters it does not understand.

4.4.6. SPECTRUM_USE_RESP

The spectrum-use response message simply acknowledges receipt of the notification.



```
+--------------------------+----------+
```

4.5. Device Validation

Typically, a Slave Device needs a Master Device to ask the Database on its behalf for available spectrum. Depending on the regulatory domain, the Master Device also must validate with the Database that the Slave Device is permitted to operate. When regulatory rules allow a Master Device to "cache" the available spectrum for a period of time, the Master Device MAY use the simpler Device Validation component, instead of the full Available Spectrum Query component, to validate a Slave Device.

When validating one or more Slave Devices, the Master Device sends the Database a request that includes the device identifier -- and any other parameters required by the regulatory rules -- for each Slave Device. The Database MUST return a response that indicates whether each device is permitted to use the spectrum.

A typical sequence for using the Device Validation request is illustrated in Figure 5, where the Master Device already has a valid set of available spectrum for Slave Devices. Note that the communication and protocol between the Slave Device and Master Device is outside the scope of this document.

- DEV_VALID_REQ (Section 4.5.1) is the device-validation request message
- DEV_VALID_RESP (Section 4.5.2) is the device-validation response message

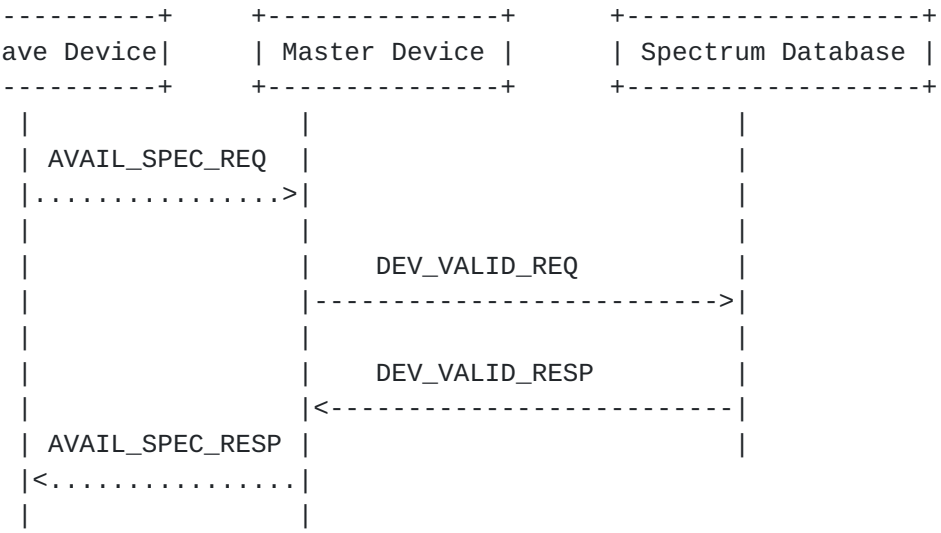
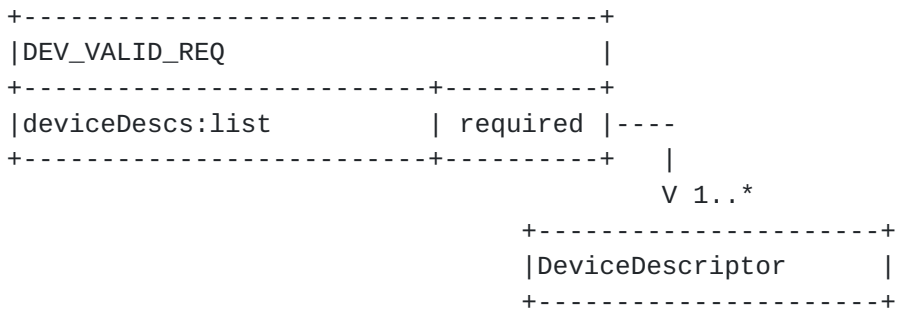


Figure 5

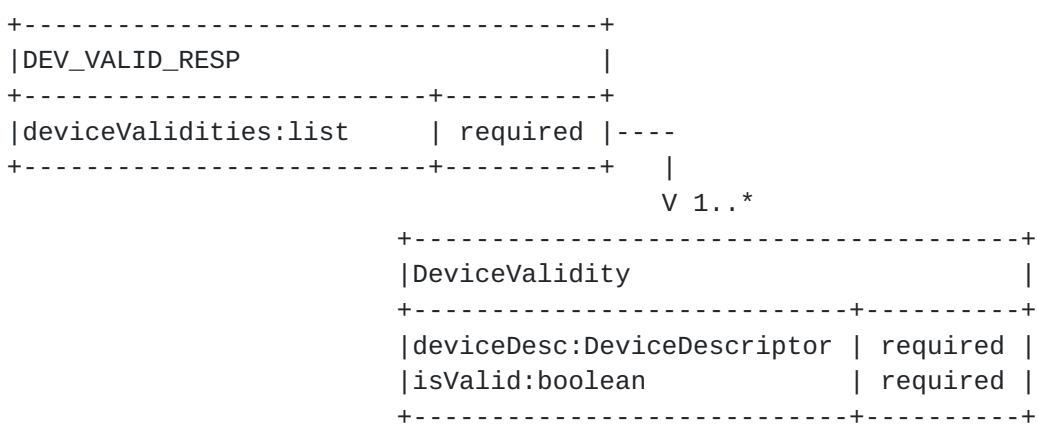
4.5.1. DEV_VALID_REQ



Parameters:

deviceDescs: A DeviceDescriptor (Section 5.2) list is REQUIRED, which specifies the list of Slave Devices that to be validated.

4.5.2. DEV_VALID_RESP



Parameters:

deviceValidities: A DeviceValidities (Section 5.12) list is REQUIRED to report the list of Slave Devices and whether each listed Device is valid. The number of entries MUST match the number of DeviceDescriptors (Section 5.2) listed in the DEV_VALID_REQ message.

5. Protocol Parameters

This section presents more details of the parameters that make up the PAWS request and response messages. It also includes a sub-section defining response codes.

5.1. GeoLocation

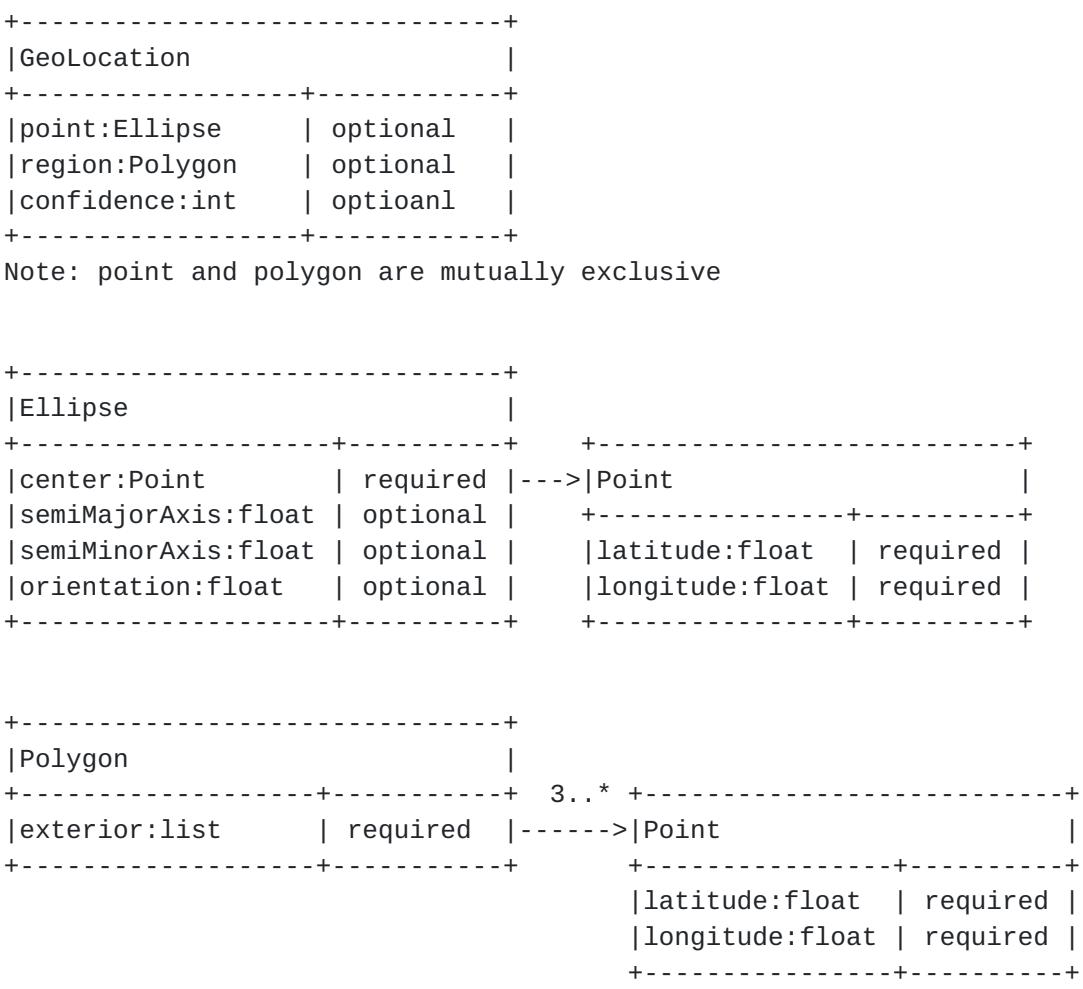
This parameter is used to specify the geo-location of the Device. It may be used to specify one of the following:

- A single point with optoinal uncertainty
- A region described by a polygon

These are represented using geometric shapes defined in GEORIV Presence Information Data Format Location Object [RFC5491], where:

- A "point" with uncertainty is represented using the Ellipse shape
- A region represented using the Polygon shape

The coordinates are expressed using the WGS84 datum, and units are degrees or meters. The data model for GeoLocation is illustrated below:



Parameters:

point: If present, it indicates that the GeoLocation represents a point. Paradoxically, a "point" is parameterized using an Ellipse, where the center represents the location of the point and the distances along the major and minor axes represent the uncertainty. The uncertainty values MAY be required, depending on the regulatory domain.

region: If present. in indicates that the GeoLocation represents a region. Database support for regions is OPTIONAL.

center: The center refers to the location of a GeoLocation pont and is the represented as the center of an ellipse. REQUIRED.

latitude, longitude: Floating-point numbers that express the latitude and longitude in degrees using the WGS84 datum. REQUIRED.

semiMajorAxis, semiMinorAxis: If required by the regulator domain, the location uncertainty, in meters, is parameterized using distances along the major and minor axes of the ellipse. When uncertainty is optional, the default value of each is 0.

orientation: This defines the orientation of the ellipse, expressed as the rotation, in degrees, of the semi-major axis from North towards the East. For example, when the uncertainty is greatest along the North-South direction, orientation is 0 degrees; conversely, if the uncertainty is greatest along the East-West direction, orientation is 90 degrees. When orientation is optoinal, its default value is 0.

exterior: When GeoLocation describes a region, the "exterior" field refers to a list of latitude/longitude points that represents the vertices of a polygon. A minimum of 3 points is required, and they must be in counter-clockwise direction.

confidence: The location confidence level, as an integer percentage, MAY be required, depending on the regulatory domain. When the parameter is optional, its default value is 100. This values is only meaningful when GeoLocation refers to a point.

5.2. DeviceDescriptor

The device descriptor contains parameters that identify the specific device, such as its manufacturer serial number, regulatory-specific ID (e.g., FCC ID), and any other device characteristics required by regulatory domains.

DeviceDescriptor	
serialNumber:string	required
fccId:string	depends on regulatory domain
*deviceType:string	depends on regulatory domain
*RAT:string	depends on regulatory domain
*other:any	

Parameters:

serialNumber: The manufacturer's device serial number is REQUIRED.
fccId: The Device's FCC ID may be required for some regulatory domains.

Additional parameters in the DeviceDescriptor depend on each regulatory domain, including certification IDs for other regulatory domains.

[Editor's Note: The descriptor probably should include an optional a list of rule-set identifiers supported by the Device. This would enable a database to determine the appropriate response. Also consider using IANA to define the names of the rule sets and parameters within the Device Descriptor.]

5.3. AntennaCharacteristics

Antenna characteristics provide additional information, such as the antenna height, antenna type, etc. Whether antenna characteristics must be provided in a request depends on the device type and regulatory domain.

AntennaCharacteristics	
height:float	depends on regulatory domain
heightType:enum	optional
heightUncertainty:float	depends on regulatory domain
*characteristics: various	depends on regulatory domain

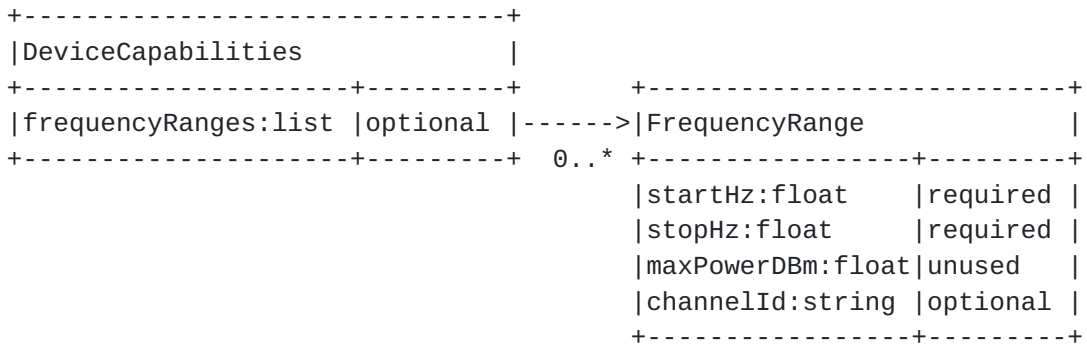
Parameters:

height: The antenna height in meters. Whether the antenna height is required depends on the device type and the regulatory domain. Note that the height may be negative.
heightType: If the height is required, then heightType is also REQUIRED. Valid values are:
AGL Above ground level (default)
AMSL Above mean sea level
heightUncertainty: The height uncertainty in meters. Whether this is required depends on the regulatory domain.

Depending on the regulatory authority, additional antenna characteristics may be required, such as:
- antenna direction
- antenna radiation pattern
- antenna gain
- antenna polarization

5.4. DeviceCapabilities

Device capabilities provide additional information that MAY be used by the Device to provide additional information to the Database that may help it to determine available spectrum. If the Database does not support device capabilities it MUST ignore the parameter altogether.



Parameters:

frequencyRanges: Optional FrequencyRange (Section 5.8) list. Each FrequencyRange element MUST contain start and stop frequencies, and optionally, channel IDs, in which the Device can operate. When specified, the Database SHOULD NOT return available spectrum that falls outside these ranges (or channel IDs).

5.5. DeviceOwner

This parameter contains device-owner information required as part of device registration. Regulatory domains MAY require additional

parameters.

DeviceOwner	
owner:vcard	required
operator:vcard	optional

Parameters:

owner: The vCard contact information for the individual or business that owns the Device is REQUIRED.
operator: The vCard contact information for the device operator is OPTIONAL, but may be required by specific regulatory domains

NOTE: Depending on the regulatory domain, the Database MAY be required to validate the device-owner information. In these cases, the Database MUST respond with an error if validation fails.

All contact information MUST be expressed using the vCard Format [RFC6350]. Only the contact fields of vCard are supported:

fn Full name of an individual
org Name of the organization
adr Address fields
tel Telephone numbers
email Email addresses

5.6. RulesetInfo

This contains parameters for the rule set of a regulatory domain that is communicated using the Initialization component (Section 4.2).

RulesetInfo	
authority:string	required
maxLocationChange:float	required
maxPollingSecs:int	required
maxValiditySecs:int	required
*other:any	depends

Parameters:

authority: A string that indicates the regulatory domain to which the ruleset applies is REQUIRED. It MUST use the 2-letter country codes defined by Country Codes - ISO 3166 [ISO3166-1].

maxLocationChange: The maximum location change in meters is REQUIRED. When the Device changes location by more than this specified distance, it MUST contact the Database to get the available spectrum for the new location. If the Device is using spectrum that is no longer available, it MUST stop operation in those frequencies immediately.

maxPollingSecs: The maximum duration, in seconds, between requests for available spectrum is REQUIRED. The Device MUST contact the Database to get available spectrum no less frequently than this duration. If the new spectrum information indicates that the Device is using spectrum that is no longer available, it MUST stop operation in those frequencies immediately.

maxValiditySecs: The maximum duration that the available-spectrum information may be considered valid is REQUIRED. When the Device is unable to contact the Database, it MAY continue to use the latest available-spectrum information at its location until its validity expires. The expiration is determined by adding maxValiditySecs to the timestamp of the AVAIL_SPECTRUM_RESP that provided the latest available-spectrum information.

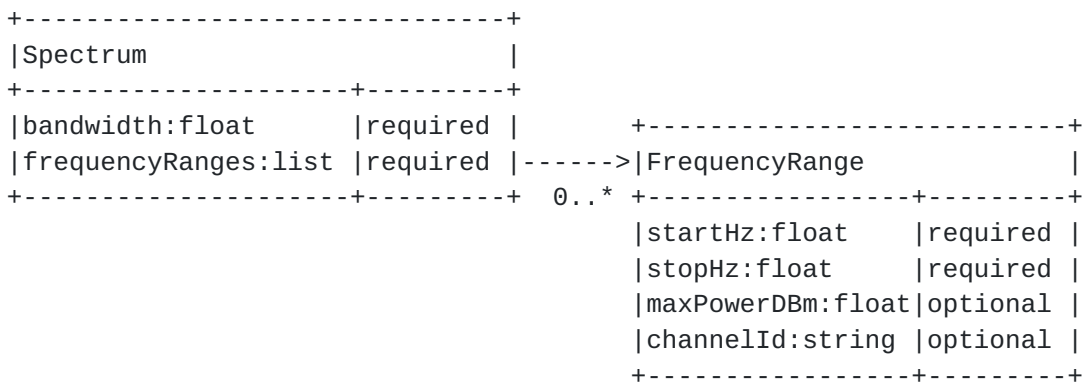
other: This message is intended to be extensible with other regulatory-specific parameters. Devices MUST ignore all parameters in the message it does not understand.

[Editor's Note: Rule-set information contain parameters from the Database that control behavior of the Device. Rules that govern device behavior may be hard to characterize, in a declarative fashion, with just a set of parameters that is easy to understand, especially when there may be coupling between parameters.

Would it be simpler to characterize rule sets by a concise list of identifiers, such as "FCC-WhiteSpace-2010", "Ofcom-WhiteSpace-2013"? An advantage of named rule sets is that it allows new regulatory domain to easily adopt existing rulesets and allow existing Devices to work "automatically" within a new regulatory domain with just a few configured rule sets.]

5.7. Spectrum

Available spectrum can logically be characterized by a list of frequency ranges and permissible power levels for each range.



Parameters:

bandwidth: This parameter is REQUIRED to define the operating bandwidth for which permissible power levels is to be specified. For example, FCC regulation would require only one spectrum specification at 6MHz bandwidth, but Ofcom regulation would require 2 specifications, at 0.1MHz and 8MHz. This parameter MAY be empty if there is no available spectrum.

frequencyRanges: A FrequencyRange (Section 5.8) list is REQUIRED to specify frequency ranges and permissible power levels. The list MAY be empty if there is no available spectrum.

5.8. FrequencyRange

The FrequencyRange parameter specifies the maximum permissible power levels within a frequency range.

startHz: The inclusive start of the frequency range is REQUIRED.

stopHz: The exclusive end of the frequency range is REQUIRED.

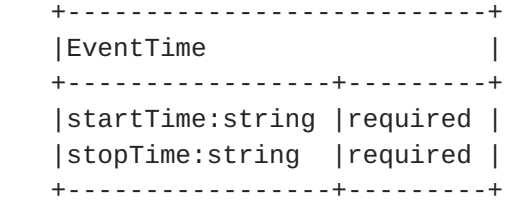
maxPowerDBm: The maximum total power level (EIRP) -- computed over the corresponding operating bandwidth -- that is permitted within the frequency range. Depending on the context in which the FrequencyRange element appears, maxPowerDBm may be REQUIRED. For example, it is REQUIRED in the AVAIL_SPECTRUM_RESP (Section 4.4.2), AVAIL_SPECTRUM_BATCH_RESP (Section 4.4.4), and SPECTRUM_USE_NOTIFY (Section 4.4.5) messages, but it would not be REQUIRED (nor applicable) when the FrequencyRange element appears in Device Capabilities (Section 5.4).

channelId: The server MAY include a channel identifier, when applicable. When it is included, the Master Device SHOULD treat it as informative.

NOTE: (maxPowerDBm / bandwidth) defines the maximum permitted EIRP spectral density.

5.9. EventTime

The EventTime element specifies the start and stop times of an "event". This is used to indicate the time period for which a Spectrum (Section 5.7) is valid.



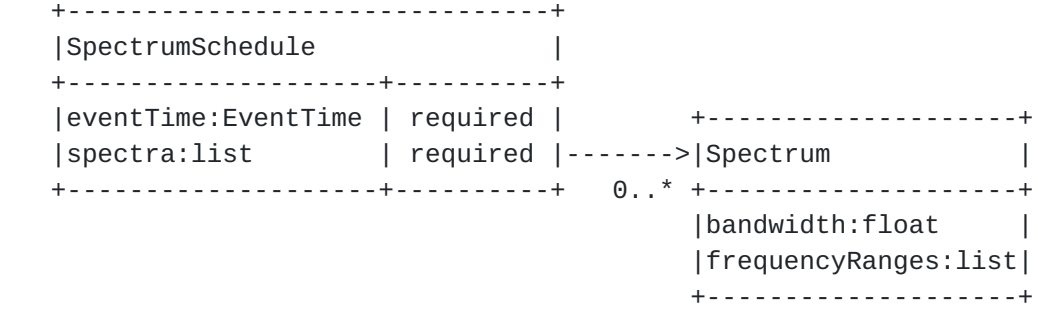
Parameters:

startTime: The inclusive start of the event is REQUIRED.
stopTime: The exclusive end of the event is REQUIRED.

Both times are expressed using the format, YYYY-MM-DDThh:mm:ssZ, as defined by Date and Time on the Internet: Timestamps [RFC3339]. The times MUST be expressed using UTC.

5.10. SpectrumSchedule

The SpectrumSchedule element combines EventTime with Spectrum to define a time period in which the spectrum is valid.

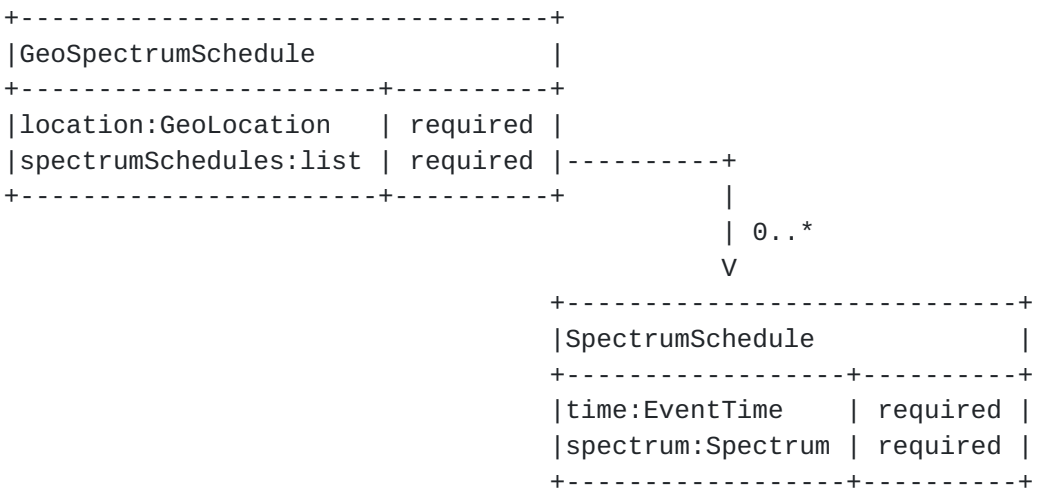


Parameters:

eventTime: The EventTime (Section 5.9) is REQUIRED to express "when" this specification is valid.
spectra: Spectrum (Section 5.7) list is REQUIRED to specify the available spectrum and permissible power levels, one per bandwidth. The list MAY be empty when there is no available spectrum.

5.11. GeoSpectrumSchedule

The GeoSpectrumSchedule element encapsulates the schedule of available spectrum at a location.



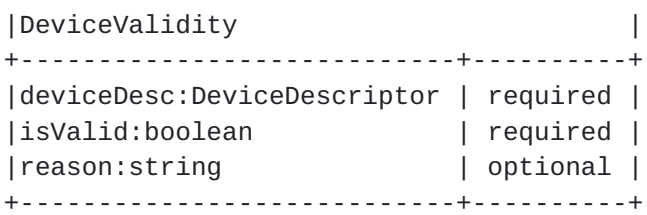
Parameters:

location: The GeoLocation (Section 5.1) is REQUIRED to identify the location at which the spectrum schedule applies

spectrumSchedules: The SpectrumSchedule (Section 5.10) list is REQUIRED. At least one schedule MUST be included (though it MAY be empty if there is no available spectrum). More than one schedule MAY be included to represent future changes to the available spectrum.

5.12. DeviceValidity

The DeviceValidity element is used to indicate whether a device is valid. See Section 4.5.2.



Parameters:

deviceDesc: The DeviceDescriptor (Section 5.2) that was used to check for validity is REQUIRED.

isValid: A REQUIRED boolean value that indicates whether the Device is valid.
reason: If the device identifier is not valid, the Database MAY include a reason. The reason MAY be in any language.

5.13. Error Element

If the Database responds to a PAWS request message with an error, it MUST include an Error element.

Error	
code:int	required
message:string	optional
data:any	optional

Parameters:

code An integer code that indicates the error type.
message A short description of the error. It MAY be in any language.
data The Database MAY include additional data. For some errors, additional data may be required. The Device MUST ignore any data parameters it does not understand.

The following table defines valid error codes. They are lossely grouped into the following categories:

-100s Indicates compatibility issues, e.g., version mismatch, unsupported or unimplemented features.
-200s Indicates that the Device request contains an error that needs to be modified before making another request.
-300s Indicates authorization-related issues.

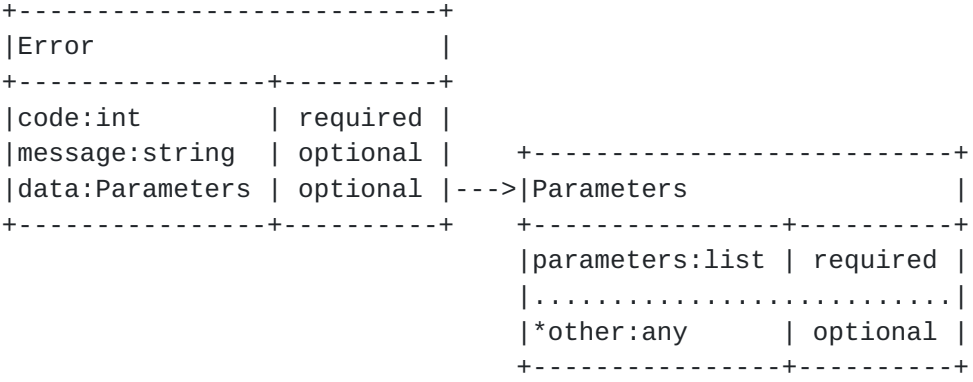
Code	Name	Description
-101	VERSION	The Database does not support the specified version of the message.
-102	UNSUPPORTED	The Database does not support the Device. For example, it does not support the regulatory domain specified in the request.
-103	UNIMPLEMENTED	The Database does not implement the optional request or optional feature.
-104	OUTSIDE_COVERAGE	The specified geo-location is outside the coverage area of the Database.

-201 REQUIRED	A required parameter is missing. The Database MUST include a list of the required parameter names. The Database MAY include only names of parameters that are missing, but MAY include a full list. When providing only a listing of missing parameters, the Database SHOULD include the full list to minimize number of re-queries from the Device.
-202 INVALID_VALUE	A parameter value is invalid in some way. The Database SHOULD include a message indicating which parameter(s) and why the value is invalid.
-301 UNAUTHORIZED	The Device is not authorized to used the Database. Authorization may be determined by regulatory rules or be dependent on prior arrangement between the Device and Database.
-302 NOT_REGISTERED	Device registration required, but the Device is not registered.

Table 1: Error Codes

5.13.1. REQUIRED Error

When the error code is REQUIRED, the Error element MUST include a Parameters element as its "data" field.



Parameters:
parameters List of parameter names.
other The Database MAY include other parameters. The Device MUST ignore all parameters it does not understand.

6. Message Encoding

The PAWS protocol is encoded using JSON-RPC [JSON-RPC] (see also JavaScript Object Notation (JSON) [RFC4627]). Each component described in Protocol Functionalities (Section 4) corresponds to one

or more JSON-RPC methods. This section provides the JSON schema for each of the protocol messages and parameters defined in sections Protocol Functionalities (Section 4) and Protocol Parameters (Section 5). JSON schemas are presented in accordance with A JSON Media Type for Describing the Structure and Meaning of JSON Documents [I-D.zyp-json-schema].

NOTE: In general, all messages defined in this section are extensible by adding additional properties to support regulatory-specific and database-specific requirements. In all cases, the Device or Database MUST ignore any parameter it does not understand.

6.1. JSON-RPC Binding

The JSON-RPC [JSON-RPC] protocol consists of two basic objects, Request and Response:

- The JSON-RPC Request object encapsulates a PAWS functionality (operation) and the request message
- The JSON-RPC Response object encapsulates a PAWS response message and Error element

The JSON-RPC Request for PAWS has the following form:

```
{
  "method": string,
  "params": <PAWS_REQ>,
  "id": string
}
```

where "method" is the name of a PAWS functionality (operation), and <PAWS_REQ> represents one of the PAWS request objects associated with the method.

The non-error JSON-RPC Response for PAWS has the following form:

```
{
  "result": <PAWS_RESP>,
  "id": string
}
```

where <PAWS_RESP> represents one of the PAWS response objects associated with the method.

The error JSON-RPC Response for PAWS has the following form:

```
{
  "error": {
    "code": integer,
    "message": string,
    "data": object,
  },
  "id": string
}
```

where the Error object and error codes are described by Error Element (Section 5.13).

Depending on prior arrangement between a Database and Device, the Request and Response MAY contain additional parameters. The Database or Device MUST ignore all parameters it does not understand.

6.2. init Method

This section describes the encoding for the JSON-RPC "init" method that represents the Initialization functionality (Section 4.2).

6.2.1. INIT_REQ Schema

The JSON encoding of the Initialization request message INIT_REQ (Section 4.2.1) is described by the following schema:

```
{
  "name": "INIT_REQ",
  "type": "object",
  "properties": {
    "type": "INIT_REQ",
    "version": {
      "type": "string",
      "required": true
    },
    "deviceDesc": {
      "type": "DeviceDescriptor",
      "required": true
    },
    "location": {
      "type": "GeoLocation",
      "description": "The location SHOULD be the current location " +
          "of the Device's antenna. Depending on the regulatory " +
          "domain, the location MAY be the anticipated position of " +
          "the Device.",
      "required": true
    }
  }
```

```
  }
```

Example "init" JSON-RPC request:

```
{
 "method": "init",
 "params": {
  "type": "INIT_REQ",
  "version": "1.0",
  "deviceDesc": {
   "serialNumber": "XXX",
   "fccId": "YYY",
   ...
  },
  "location": {
   "point": {
    "center": {"latitude": 37.0, "longitude": -101.3}
   }
  }
 }
 "id": "xxxxxx"
}
```

6.2.2. INIT_RESP Parameters

The JSON encoding of the Initialization response message INIT_RESP (Section 4.2.2) is described by the following schema:

```
{
  "name": "INIT_RESP",
  "type": "object",
  "properties": {
    "type": "INIT_RESP",
    "version": {
      "type": "string",
      "required": true
    },
    "rulesetInfo": {
      "type": "RulesetInfo",
      "description": "Indicates the active regulatory domain and " +
          "attributes that define the applicable rule set that " +
          "govern the device",
      "required": true
    }
  }
}
```

Example "init" JSON-RPC response:

```
{
 "result": {
  "type": "INIT_RESP",
  "version": "1.0",
  "rulesetInfo": {
    ...
  }
 },
 "id": "xxxxxx"
}
```

6.3. register Method

This section describes the encoding for the JSON-RPC "register" method that represents Device Registration functionality (Section 4.3).

6.3.1. REGISTRATION_REQ Parameters

The JSON encoding of the Registration request message REGISTRATION_REQ (Section 4.3.1) is described by the following schema:

```
{
  "name": "REGISTRATION_REQ",
  "type": "object",
  "properties": {
    "type": "REGISTRATION_REQ",
    "version": {
      "type": "string",
      "required": true
    },
    "deviceDesc": {
      "type": "DeviceDescriptor",
      "required": true
    },
    "deviceOwner": {
      "type": "DeviceOwner",
      "required": true
    },
    "location": {
      "type": "GeoLocation",
      "description": "The location SHOULD be the current location " +
          "of the Device's antenna. Depending on the regulatory " +
          "domain, the location MAY be the anticipated position of " +
          "the Device.",
      "required": true
    },
    "antenna": {
      "type": "AntennaCharacteristics",
      "description": "Antenna characteristics, including its " +
          "height and height type",
      "required": false
    }
  }
}
```

Example "register" JSON-RPC request:

```
{
 "method": "register",
 "params": {
  "type": "REGISTRATION_REQ",
  "version": "1.0",
  "deviceDesc": {
   "serialNumber": "XXX",
   "fccId": "YYY",
   ...
  },
  "deviceOwner": {
   "owner": {
    "fn": "John A. Smith",
    "org": {
     "text": "ACME",
    },
    "adr": {
      "street": "1234 A Street",
      "locality": "San Jose",
      "region": "CA",
      "code": "94423",
      "country": "US"
    },
    "tel": {
      "uri": "tel:+1-333-555-1212"
    },
    "email": {
      "text": "j.smith@email.com"
    }
   }
  },
  "location": {
   "point": {
    "center": {"latitude": 37.0, "longitude": -101.3}
   }
  },
  "antenna": {"height": 10.2, "heightType": "AGL"}
 },
 "id": "xxxxxx"
}
```

6.3.2. REGISTRATION_RESP Parameters

The JSON encoding of the Registraion response message REGISTRATION_RESP (Section 4.3.2) is described by the following schema:

```
{
  "name": "REGISTRATION_RESP"
  "type": "object",
  "properties": {
    "type": "REGISTRATION_RESP",
    "version": {
      "type": "string",
      "required": true
    }
  }
}
```

Example "register" JSON-RPC response:

```
{
 "result": {
  "type": "REGISTRATION_RESP",
  "version": "1.0"
 },
 "id": "xxxxxx"
}
```

6.4. getSpectrum Method

This section describes the encoding for the JSON-RPC "getSpectrum" method that represents the single-location query of the Available Spectrum Query functionality (Section 4.4) that enables a Device to obtain a set of available spectrum from the Database.

6.4.1. AVAILABLE_SPECTRUM_REQ Parameters

The JSON encoding of the Available Spectrum request message AVAIL_SPECTRUM_REQ (Section 4.4.1) is described by the following schema:

```
{
  "name": "AVAILABLE_SPECTRUM_REQ",
  "type": "object",
  "properties": {
    "type": "AVAILABLE_SPECTRUM_REQ",
    "version": {
      "type": "string",
      "required": true
    },
    "deviceDesc": {
      "type": "DeviceDescriptor",
      "required":true
    },
    "location": {
      "type": "GeoLocation",
      "description": "The location SHOULD be the current location " +
          "of the Device's antenna. Depending on the regulatory " +
          "domain, the location MAY be the anticipated position of " +
          "the Device.",
      "required": false
    },
    "antenna": {
      "type": "AntennaCharacteristics",
      "description": "Antenna characteristics, including its " +
          "height and height type. May required depending on " +
          "device type and regulatory domain",
      "required":false
    },
    "owner": {
      "type": "DeviceOwner",
      "description": "May be required if the Device is not yet " +
          "registered or if the DB does not implement a separate " +
          "device-registration request. Also depends on device type " +
          "and regulatory domain",
      "required": false
    },
    "capabilities": {
      "type": "DeviceCapabilities",
      "description": "The Database SHOULD NOT return spectrum that " +
          "is incompatible with the specified capabilities.",
      "required": false
    }
  }
}
```

Example "getSpectrum" JSON-RPC request:

```
{
 "method": "getSpectrum",
 "params": {
  "type": "AVAILABLE_SPECTRUM_REQ",
  "version": "1.0",
  "deviceDesc": {
   "serialNumber": "XXX",
   "fccId": "YYY",
   ...
  },
  "location": {
   "point": {
    "center": {"latitude": 37.0, "longitude": -101.3}
   }
  },
  "antenna": {"height": 10.2, "heightType": "AGL"}
 }
 "id": "xxxxxx",
}
```

6.4.2. AVAIL_SPECTRUM_RESP Parameters

The JSON encoding of the Available Spectrum response message AVAIL_SPECTRUM_RESP (Section 4.4.2) is described by the following schema:

```
  {
    "name": "AVAIL_SPECTRUM_RESP",
    "type": "object",
    "properties": {
      "type": "AVAILABLE_SPECTRUM_RESP",
      "version": {
        "type": "string",
        "required": true
      },
      "timestamp": {
        "type": "string",
        "description": "Timestamp of the response of the form " +
            "YYYY-MM-DDThh:mm:ssZ. This SHOULD be used by the Device " +
            "as a reference for the start and stop times in the " +
            "spectrum schedule",
        "required": true
      },
      "deviceDesc": {
        "type": "DeviceDescriptor",
        "required": true
      },
      "spectrumSchedules": {
        "type": "array",
        "description": "The Database MAY return more than one " +
            "schedule to represent future changes to the available " +
            "spectrum. This array MAY be empty if no spectrum is " +
            "is available.",
        "items": "SpectrumSchedule",
        "required": true
      },
      "needsSpectrumReport": {
        "type": "boolean",
        "description": "For regulatory domains that require a " +
            "spectrum-usage report from devices, the Database MUST " +
            "return true for this parameter.",
        "default": false,
        "required": false
      }
    }
  }
```

Example "getSpectrum" JSON-RPC response:

```
  {
    "result": {
      "type": "AVAILABLE_SPECTRUM_RESP",
      "version": "1.0",
      "timestamp": "2013-03-02T14:30:21Z",
```

```
    "deviceDesc": {
     "serialNumber": "XXX",
     "fccId": "YYY",
     ...
    },
    "spectrumSchedules": [
     {
      "eventTime": {
       "startTime": "2013-03-02T14:30:21Z",
       "stopTime": "2013-03-02T20:00:00Z",
      },
      "spectra": [
        {
         "bandwidth": 6e6,
         "frequencyRanges": [
           {"startHz":5.18e8, "stopHz":5.36e8, "maxPowerDBm":30.0},
           {"startHz":5.36e8, "stopHz":5.42e8, "maxPowerDBm":36.0},
           ...
         ]
        },
        {
         "bandwidth": 1e5,
         "frequencyRanges": [
           {"startHz":5.18e8, "stopHz":5.36e8, "maxPowerDBm":27.0},
           {"startHz":5.36e8, "stopHz":5.42e8, "maxPowerDBm":33.0},
           ...
         ]
        },
      ]
     },
     {
      "eventTime": {
       "startTime": "2013-03-02T22:00:00Z",
       "stopTime": "2013-03-03T14:30:21Z",
      },
      "spectra": [
       ...
      ]
     }
    ],
    "needsSpectrumReport": false
   },
   "id": "xxxxxx"
  }
```

6.5. getSpectrumBatch Method

This section describes the encoding for the JSON-RPC "getSpectrumBatch" method that represents the multiple-location query of the Available Spectrum Query functionality (Section 4.4) that enables a Device to obtain a set of available spectrum for multiple locations from the Database.

6.5.1. AVAIL_SPECTRUM_BATCH_REQ Parameters

The JSON encoding of the Batch Available Spectrum request AVAIL_SPECTRUM_BATCH_REQ (Section 4.4.3) is described by the following schema. This an OPTIONAL feature of the Database.

```
 {
   "name": "AVAIL_SPECTRUM_BATCH_REQ",
   "type": "object",
   "properties": {
     "type": "AVAILABLE_SPECTRUM_BATCH_REQ",
     "version": {
       "type": "string",
      "required": true
      },
     "deviceDesc": {
       "type": "DeviceDescriptor",
       "required": true
     },
     "locations": {
       "type": "array",
       "description": "At least one device location is required. " +
           "Additional (anticipated) locations can also be included, " +
           "as permitted by regulatory domain,",
       "items": "GeoLocation",
       "required": true
     },
     "antenna": {
       "type": "Antenna characteristics, including its " +
           "height and height type. May required depending on " +
           "device type and regulatory domain","AntennaCharacteristics",
       "required": false
     },
     "owner": {
       "type": "DeviceOwner",
       "description": "May be required if the Device is not yet " +
           "registered or if the DB does not implement a separate " +
           "device-registration request. Also depends on device type " +
           "and regulatory domain",
       "required": false
     },
     "capabilities": {
       "type": "DeviceCapabilities",
       "description": "The Database SHOULD NOT return spectrum that " +
           "is incompatible with the specified capabilities.",
       "required": false
     }
   }
 }
```

Example "getSpectrumBatch" JSON-RPC request:

```
{
 "method": "getSpectrumBatch",
 "params": {
  "type": "AVAILABLE_SPECTRUM_BATCH_REQ",
  "version": "1.0",
  "deviceDesc": {
   "serialNumber": "XXX",
   "fccId": "YYY",
   ...
  },
  "locations": [
   {
    "point": {
     "center": {"latitude": 37.0, "longitude": -101.3}
    }
   },
   {
    "point": {
     "center": {"latitude": 37.0005, "longitude": -101.3005}
    }
   },
   ...
  ],
  "antenna": {"height": 10.2, "heightType": "AGL"}
 }
 "id": "xxxxxx",
}
```

6.5.2. AVAIL_SPECTRUM_BATCH_RESP Parameters

The JSON encoding of the Batch Available Spectrum response AVAIL_SPECTRUM_BATCH_RESP (Section 4.4.4) is described by the following schema:

```
  {
    "name": "AVAIL_SPECTRUM_BATCH_RESP",
    "type": "object",
    "properties": {
      "type": "AVAILABLE_SPECTRUM_BATCH_RESP",
      "version": {
        "type": "string",
       "required": true
       },
      "timestamp": {
        "type": "string",
        "description": "Timestamp of the response of the form " +
            "YYYY-MM-DDThh:mm:ssZ. This SHOULD be used by the Device " +
            "as a reference for the start and stop times in the " +
            "spectrum schedule",
        "required": true
      },
      "deviceDesc": {
          "type": "DeviceDescriptor",
      "required": true
      },
      "geoSpectrumSchedules": {
        "type": "array",
        "description": "For each location, the Database MAY return " +
            "more than one schedule to represent future changes " +
            "to the available spectrum. This array MAY be empty if " +
            "there is no available spectrum.",
        "items": "GeoSpectrumSchedule",
        "required": true
      },
      "needsSpectrumReport": {
        "type": "boolean",
        "description": "For regulatory domains that require a " +
            "spectrum-usage report from devices, the Database MUST " +
            "return true for this parameter.",
        "default": false,
        "required": false
      }
    }
  }
```

Example "getSpectrumBatch" JSON-RPC response:

```
  {
   "result": {
    "type": "AVAILABLE_SPECTRUM_BATCH_RESP",
    "version": "1.0",
    "timestamp": "2013-03-02T14:30:21Z",
```

```
     "deviceDesc": {
      "serialNumber": "XXX",
      "fccId": "YYY",
      ...
     },
     "geoSpectrumSchedules": [
      {
       "location": {
        "point": {
         "center": {"latitude": 37.0, "longitude": -101.3}
        }
       },
       "spectrumSchedules": [
        {
         "eventTime": {
          "startTime": "2013-03-02T14:30:21Z",
          "stopTime": "2013-03-02T20:00:00Z",
         },
         "spectra": [
           {
            "bandwidth": 6e6,
            "frequencyRanges": [
              {"startHz":5.18e8, "stopHz":5.36e8, "maxPowerDBm":30.0},
              {"startHz":5.36e8, "stopHz":5.42e8, "maxPowerDBm":36.0},
              ...
            ]
           },
           {
            "bandwidth": 1e5,
            "frequencyRanges": [
              {"startHz":5.18e8, "stopHz":5.36e8, "maxPowerDBm":27.0},
              {"startHz":5.36e8, "stopHz":5.42e8, "maxPowerDBm":33.0},
              ...
            ]
           },
        ]
       },
       {
        "eventTime": {
         "startTime": "2013-03-02T22:00:00Z",
         "stopTime": "2013-03-03T14:30:21Z",
        },
        "spectra": [
         ...
        ]
       }
      ],
     },
```

```
     {
      "location": {
       "point": {
        "center": {"latitude": 37.0005, "longitude": -101.3005}
       }
      },
      "spectrumSchedules": [
       ...
      ]
     }
    ],
    "needsSpectrumReport": false
   },
   "id": "xxxxxx"
  }
```

6.6. notifySpectrumUse Method

This section describes the encoding for the JSON-RPC "notifySpectrumUse" method that represents the Spectrum-usage notification of Available Spectrum Query functionality (Section 4.4.5) for notifying the Database of anticipated spectrum usage.

6.6.1. SPECTRUM_USE_NOTIFY Parameters

The JSON encoding of the Spectrum Notification message SPECTRUM_USE_NOTIFY (Section 4.4.5) is described by the following schema:

```
{
  "name": "SPECTRUM_USE_NOTIFY",
  "type": "object",
  "properties": {
    "type": "SPECTRUM_USE_NOTIFY",
    "version": {
      "type": "string",
      "required": true
    },
    "deviceDesc": {
       "type": "DeviceDescriptor",
      "required": true
    },
    "location": {
      "type": "GeoLocation",
      "required": true
    },
    "spectra": {
      "type": "array",
      "description": "The spectrum anticipated to be used by " +
          "the Device.",
      "items": "Spectrum",
      "required": true
    }
  }
}
```

Example "notifySpectrumUse" JSON-RPC notification:

```
{
 "method": "notifySpectrumUse",
 "params": {
  "type": "SPECTRUM_USE_NOTIFY",
  "version": "1.0",
  "deviceDesc": {
   "serialNumber": "XXX",
   "fccId": "YYY",
   ...
  },
  "location": {
   "point": {
    "center": {"latitude": 37.0005, "longitude": -101.3005}
   }
  },
  "spectra": [
   {
    "bandwidth": 6e6,
    "frequencyRanges": [
     {"startHz":5.18e8, "stopHz":5.24e8, "maxPowerDBm":30.0}
    ]
   },
  ]
 },
  "needsSpectrumReport": false
 },
 "id": "xxxxxx"
}
```

6.6.2. SPECTRUM_USE_RESP Parameters

The JSON encoding of the Spectrum-usage response SPECTRUM_USE_RESP (Section 4.4.6) is decribed by the following schema:

```
{
  "name": "SPECTRUM_USE_RESP",
  "type": "object",
  "properties": {
    "type": "SPECTRUM_USE_RESP",
    "version": {
      "type": "string",
      "required": true
    }
  }
}
```

Example "notifySpectrumUse" JSON-RPC response:

```
{
 "result": {
  "type": "SPECTRUM_USE_RESP",
  "version": "1.0"
 },
 "id": "xxxxxx"
}
```

6.7. verifyDevice Method

This section describes the encoding for the JSON-RPC "verifyDevice" method that represents the Device Validation functionality (Section 4.5). This is used by a Master Device to validate Slave Devices.

6.7.1. DEV_VALID_REQ Parameters

The JSON encoding of the Device Validation request DEV_VALID_REQ (Section 4.5.1) is described by the following schema:

```
{
  "name": "DEV_VALID_REQ",
  "type": "object",
  "properties": {
    "type": "DEV_VALID_REQ",
    "version": {
      "type": "string",
      "required": true
    },
    "deviceDescs": {
      "type": "array",
      "description": "List of Slave Devices to be validated",
      "items": "DeviceDescriptor",
      "required": true
    }
  }
}
```

Example "verifyDevice" JSON-RPC request:

```
{
 "method": "verifyDevice",
 "params": {
  "type": "DEV_VALID_REQ",
  "version": "1.0",
  "deviceDescs": [
   {
    "serialNumber": "XXX",
    "fccId": "YYY",
    ...
   },
   {
    "serialNumber": "XXX3",
    "fccId": "YYY2",
    ...
   },
   ...
  ]
 },
 "id": "xxxxxx"
}
```

6.7.2. DEV_VALID_RESP Parameters

The JSON encoding of the Device Validation response DEV_VALID_RESP (Section 4.5.2) is described by the following schema:

```
{
  "name": "DEV_VALID_RESP",
  "type": "object",
  "properties": {
    "type": "DEV_VALID_RESP",
    "version": {
      "type": "string",
      "required": true
    },
    "deviceValidities": {
      "type": "array",
      "description": "List of DeviceValidity objects that shows the " +
          "validity of each device included in the original Device " +
          "Validity Request message.",
      "items": "DeviceValidity",
      "required": true
    }
  }
}
```

Example "verifyDevice" JSON-RPC response:

```
{
 "result": {
  "type": "DEV_VALID_RESP",
  "version": "1.0",
  "deviceValidities": [
   {
    "deviceDesc": {
      "serialNumber": "XXX",
      "fccId": "YYY",
      ...
    },
    "isValid": true
   },
   {
    "deviceDesc": {
      "serialNumber": "XXX3",
      "fccId": "YYY2",
      ...
    },
    "isValid": false,
    "reason": "Not authorized"
   }
  ]
 },
 "id": "xxxxxx"
}
```

6.8. Sub-message Schemas

This section defines the schema for Protocol Parameters (Section 5) embedded in PAWS request and response messages.

6.8.1. GeoLocation

This parameter is used to specify the GeoLocation (Section 5.1) of the Device. The geometric shapes represent the JSON encoding shapes defined in GEOPRIV Presence Information Data Format Location Object [RFC5491].

```
 {
    "name": "GeoLocation",
    "type": "object",
    "properties": {
      "point": {
        "description": "A single location, with optional " +
            "uncertainty measures",
        "type": "Ellipse",
        "required": false
```

```
        },
        "region": {
          "description": "A region described by a polygon",
          "type": "Polygon",
          "required": false
        },
        "confidence": {
          "description": "Confidence interval when location " +
              "is a point with uncertainty. 0 to 100."
          "type": "integer",
          "required": false,
          "default": 100
        }
      }
    }
    {
      "name": "Point",
      "type": "object",
      "properties": {
        "latitude": {
          "description: "Floating-point degrees. WGS84 datum.",
          "type": "number",
          "required": true
        },
        "longitude": {
          "type": "number",
          "description: "Floating-point degree. WGS84 datum.",
          "required":true
        }
      }
    }
    {
      "name": "Ellipse",
      "type": "object",
      "properties": {
        "center": {
          "type": "Point",
          "required": true
        },
        "semiMajorAxis": {
          "description": "Floating-point meters that describe " +
              "location uncertainty along the major axis of " +
              "the ellipse."
          "type": "number",
          "required": false,
          "default": 0
        },
        "semiMinorAxis": {
```

```
          "description": "Floating-point meters that describe " +
              "location uncertainty along the minor axis of " +
              "the ellipse."
          "type": "number",
          "required": false,
          "default": 0
        },
        "orientation": {
          "description": "Orientation of the ellipse, as rotation " +
              "of the major axis from North towards East. Degrees."
          "type": "number",
          "required": false,
          "default": 0
        }
      }
    }
    {
      "name": "Polygon",
      "type": "object",
      "properties": {
        "exterior": {
          "description": "List of Points in counter-clockwise " +
              "order. They must form a loop with no edges that " +
              "cross each other. Minimum of 3 points."
          "type": "array",
          "items": "Point",
          "required": true
        }
      }
    }
```

6.8.2. DeviceDescriptor

The DeviceDescriptor (Section 5.2) contains parameters that identify the specific device, such as its manufacturer serial number, regulatory-specific ID (e.g., FCC ID), and any other device characteristics required by regulatory domains, such as device-type classification.

```
{
  "name": "DeviceDescriptor",
  "type": "object",
  "properties": {
    "serialNumber": {
      "type": "string",
      "required": true
    },
    "fccId": {
      "type": "string",
      "description":"The device's FCC ID.",
      "required": false
    }
  }
}
```

6.8.3. AntennaCharacteristics

AntennaCharacteristics (Section 5.3) provide additional information, such as the antenna height, antenna type, etc.

```
{
   "name": "AntennaCharacteristics",
   "type": "object",
   "properties": {
     "height": {
       "description": "Height of the antenna, in meters",
       "type": "number",
       "required": false
     },
     "heightType": {
       "description": "Reference type for height: " +
           "Above Ground Level (AGL), or Above Mean Sea " +
           "Level (AMSL)."
       "type": "string",
       "enum": "["AGL","AMSL"]",
       "default": "AGL",
       "required": false
     },
     "heightUncertainty": {
       "description": "Uncertainty of the height measurement, " +
           "in meters."
       "type": "number",
       "required": false
     }
   }
}
```

6.8.4. DeviceCapabilities

Device capabilities (Section 5.4) provide additional information that MAY be used by the Device to provide additional information to the Database to help the Database determine available spectrum. If the Database does not support device capabilities, it MUST ignore the parameter.

```
{
  "name": "DeviceCapabilities",
  "type": "object",
  "description": "Device capabilities to help DB determine " +
      "available spectrum. The DB SHOULD NOT return available " +
      "spectrum that falls outside the given frequency ranges.",
  "properties": {
    "frequencyRanges": {
      "type": "array",
      "items": "FrequencyRange",
      "required": false
    }
  }
}
```

6.8.5. DeviceOwner

The DeviceOwner (Section 5.5) parameter contains device-owner information required as part of device registration. Regulatory domains MAY require additional parameters. JSON encoding of vCard is described in A JavaScript Object Notation (JSON) Representation for vCard [I-D.bhat-vcarddav-json].

```
    {
      "name": "DeviceOwner",
      "type": "object",
      "description": "Device-owner information required as part of " +
          "Device registration. Regulatory domains MAY require " +
          "additional parameters.",
      "properties": {
        "owner": {
          "type": "vCard",
          "description":"Contact information for the individual " +
               "or business that owns the device.",
          "required": true
        },
        "operator": {
          "type": "vCard",
          "description":"Contact information for the device operator.",
          "required": false
        }
      }
    }
```

Example:

```
    {
      "deviceOwner": {
        "owner": {
          "org": {
            "text": "Racafrax, Inc."
          }
        },
        "operator": {
          "fn": "John Frax",
          "adr": {
            "street": "100 Main Street",
            "locality": "Summersville",
            "region": "CA",
            "code": "90034",
            "country": "USA"
          },
          "tel": {
            "uri": "tel:+1-213-555-2344"
          },
          "email": {
            "text": "j.frax@rackafrax.com"
          }
        }
      }
    }
```

6.8.6. RulesetInfo

RulesetInfo (Section 5.6) contains parameters for the rule set of a regulatory domain that is communicated using the Initialization component (Section 4.2).

```
{
  "name": "RulesetInfo",
  "type": "object",
  "description": "The rule set of a regulatory domain that is " +
      "communicated to Devices in the Initialization Response " +
      "message.",
  "properties": {
    "authority": {
      "type": "string",
      "description": "The regulatory domain at the specified " +
          "location. It is a 2-letter country codes defined by " +
          "ISO3166-1.",
      "required": true
    },
    "maxLocationChange": {
      "type": "number",
      "description": "Maximum location change in meters.",
      "required": true
    },
    "maxPollingSecs": {
      "type": "integer",
      "description": "Maximum duration, in seconds, between " +
          "requests for available spectrum.",
      "required": true
    },
    "maxValiditySecs": {
      "type": "string",
      "description": "Maximum duration that the available-spectrum " +
          "information may be considered valid.",
      "required": true,
    }
  }
}
```

6.8.7. Spectrum

Available Spectrum (Section 5.7) can logically be characterized by a list of frequency ranges and permissible power levels for each range.

```
{
  "name": "Spectrum",
  "type": "object",
  "description": "A per-bandwidth list of frequency ranges with " +
      "permissible power levels. For example, In US, In US, FCC " +
      "requires only one spectrum specification at 6MHz " +
      "bandwidth; in UK, Ofcom requires two (at 0.1MHz and " +
      "8MHz).",
  "properties": {
    "bandwidth": {
      "type": "number",
      "description": "Operating bandwidth for which permissible " +
          "power levels are applicable.",
      "required": true
    },
    "frequencyRanges": {
      "type": "array",
      "description": "List of FrequencyRange objects to specify " +
          "frequency ranges and permissible power levels for " +
          "a given bandwidth. The list MAY be empty when there " +
          "is no available spectrum.",
      "items": "FrequencyRange",
      "required": true
    }
  }
}
```

6.8.8. FrequencyRange

The FrequencyRange (Section 5.8) element describes a frequency range and permissible power level within the specified range.

```
{
  "name": "FrequencyRange",
  "type": "object",
  "properties": {
    "startHz": {
      "type": "number",
      "description": "The inclusive start of the frequency range.",
      "required": true
    },
    "stopHz": {
      "type": "number",
      "description": "The exclusive end of the frequency range.",
      "required": true
    },
    "maxPowerDBm": {
      "type": "number",
      "description": "The maximum total power level (EIRP), " +
          "computed over the corresponding operating bandwidth, " +
          "that is permitted within the frequency range. This "
          "field is optional when specifying device " +
          "capabilities, but is otherwise required.",
      "required": false
    },
    "channelId": {
      "type": "string",
      "required": false
    }
  }
}
```

6.8.9. EventTime

The EventTime (Section 5.9) element specifies the start and stop times of an "event." It is used to indicate the time period for which a Spectrum (Section 5.7) is valid.

```
    {
      "name": "EventTime",
      "type": "object",
      "properties": {
        "startTime": {
          "type": "string",
          "description": "YYYY-MM-DDThh:mm:ssZ RFC3339 format.",
          "format": "date-time",
          "required": false
        },
        "stopTime": {
          "type": "string",
          "description": "YYYY-MM-DDThh:mm:ssZ RFC3339 format.",
          "format": "date-time",
          "required": false
        }
      }
    }
```

6.8.10. SpectrumSchedule

The SpectrumSchedule (Section 5.10) element combines EventTime with Spectrum to define a time period during which the spectrum is valid.

```
{
  "name": "SpectrumSchedule",
  "type": "object",
  "description": "The SpectrumSchedule element combines EventTime " +
      "with Spectrum to define a time period during which spectrum " +
      "is valid.",
  "properties": {
    "eventTime": {
      "type": "EventTime",
      "description": "Period when the spectra is valid.",
      "required": true
    },
    "spectra": {
      "type": "array",
      "description": "List of available spectra and permissible " +
          "power levels; one spectrum object per bandwidth. The " +
          "list MAY be empty when there is no available spectrum.",
      "items": "Spectrum",
      "required": true
    }
  }
}
```

6.8.11. GeoSpectrumSchedule

The GeoSpectrumSchedule (Section 5.11) element encapsulates the schedule of available spectrum at a location.

```
{
  "name": "GeoSpectrumSchedule",
  "type": "object",
  "description": "The GeoSpectrumSchedule element encapsulates " +
      "the schedule of available spectrum at a location.",
  "properties": {
    "eventTime": {
      "type": "GeoLocation",
      "description": "The location at which the spectrum " +
          "schedule applies.",
      "required": true
    },
    "spectrumSchedules": {
      "type": "array",
      "description": "At least one schedule MUST be included " +
          "(though it MAY be empty if there is no available " +
          "spectrum. More than one schedule MAY be included " +
          "to represent future changes to the available spectrum.",
      "items": "spectrumSchedule",
      "required": true
    }
  }
}
```

6.8.12. DeviceValidity

The DeviceValidity (Section 5.12) element is used to indicate whether a device is valid. See Section 4.5.2.

```
{
  "name": "DeviceValidity",
  "type": "object",
  "description": "The GeoSpectrumSchedule element encapsulates " +
      "the schedule of available spectrum at a location.",
  "properties": {
    "deviceDesc": {
      "type": "DeviceDescriptor",
      "required": true
    },
    "isValid": {
      "type": "boolean",
      "description": "Boolean that indicates if the Device is " +
          "valid",
      "required": true
    },
    "reason": {
      "type": "string",
      "description": "If the device identifier is not valid, " +
          "the Database MAY include a reason. The reason MAY be " +
          "in any language.",
      "required": false
    }
  }
}
```

6.8.13. Additional Properties

Note that A JSON Media Type for Describing the Structure and Meaning of JSON [I-D.zyp-json-schema] allows, as default behavior, the inclusion of additional properties by instances that are not explicitly defined in the JSON schema that the instance implements. The schema elaborated in this document adopts this default behavior. Hence, the instance MAY provide additional properties and associated values (which may be "any" JSON type) not explictly listed in this schema. Further note that the Database and Device MUST ignore any such additional properties and their associated values that it does not understand.

7. HTTPS Binding

This section describes the use of HTTP over TLS (HTTPS) HTTP Over TLS [RFC2818] as the transport mechanism for the PAWS protocol. TLS provides message integrity and confidentiality between the Master Device and the Database. The Master Device MUST implement server authentication, as described in Section 3.1 of HTTP Over TLS [RFC2818]. The Device uses the URI determined (either statically

configured or dynamically discovered) to authenticate the server. The Device SHOULD fail a request if server authentication fails.

Depending on prior relationship between a database and device, the server MAY require client authentication, as described in the Transport Layer Security (TLS) Protocol [RFC5246], to authenticate the device.

To enable databases to handle large numbers of requests from large numbers of devices, the Database MAY support and Devices SHOULD support Stateless TLS Session Resumption [RFC5077].

A PAWS request message is carried in the body of an HTTP POST request. A PAWS response message is carried in the body of an HTTP response. A PAWS response SHOULD include a Content-Length header.

The POST method is the only method REQUIRED for PAWS. If a database chooses to support GET, it MUST be an escaped URL, but the encoding of the URL is outside the scope of this document. The database MAY refuse to support the GET request by returning an HTTP error code, such as 404 (not found).

The Database MAY redirect a PAWS request. The Master Device MUST handle redirects by using the Location header provided by the server in a 3xx response. When redirecting, the Master Device MUST observe the delay indicated by the Retry-After header. The Master Device MUST authenticate the Database that returns the redirect response before following the redirect. The Master Device MUST authenticate the Database indicated in the redirect.

8. Example Messages

TBD

9. IANA Considerations

TBD

10. Security Considerations

PAWS is a protocol whereby a Master Device requests a schedule of available spectrum at its location (or location of its Slave Devices) before it (they) can operate using those frequencies. Whereas the information provided by the Database must be accurate and conform to applicable regulatory rules, the Database cannot enforce, through the

protocol, that a client device uses only the spectrum it provided. In other words, devices can put energy in the air and cause interference without asking the Database. Hence, PAWS security considerations do not include protection against malicious use of the White Space spectrum. For more detailed information on specific requirements and security considerations associated with PAWS, see Protocol to Access White Space database: PAWS Use Cases and Requirements [I-D.ietf-paws-problem-stmt-usecases-rqmts].

By using the PAWS protocol, the Master Device and the Database expose themselves to the following risks:

- Accuracy: The Master Device receives incorrect spectrum-availability information.
- Privacy: An unauthorized entity intercepts identifying data for the Master Device, such as serial number and location.

Protection from these risks depends on the success of the following steps:

1. The Master Device must determine a proper database.
2. The Master Device must connect to the proper database.
3. The Database must determine or compute accurate spectrum-availability information.
4. PAWS messages must be transmitted unmodified between the Database and the Master Device.
5. PAWS messages must be encrypted between the Database and the Master Device to prevent exposing private information.
6. For a Slave Device, the spectrum-availability information also must be transmitted unmodified and secure between the Master Device and the Slave Device.

Of these, only steps 2, 4, and 5 are within the scope of this document. [Editor's note: It is still open whether Step 1 is within the scope of this document]. Step 3 dependent on specific database implementations and regulatory rules and is outside the scope of this document. Step 6 requires a protocol between master and slave devices and is thus outside the scope of this document.

10.1. Assurance of Proper Database

This document assumes that the Database is contacted using a domain name or an IP address. Using HTTP over TLS HTTP Over TLS [RFC2818], the Database authenticates its identity, either as a domain name or IP address, to the Master Device by presenting a certificate containing that identifier as a "subjectAltName" (i.e., as a dNSName or IP address). If the Master Device has external information as to the expected identity or credentials of the proper database (e.g., a certificate fingerprint), these checks MAY be omitted. Note that in order for the presented certificate to be valid at the client, the

client must be able to validate the certificate. In particular, the validation path of the certificate must end in one of the client's trust anchors, even if that trust anchor is the Database certificate itself. [Editor's note: a database can change certificate authorities (CAs) over time. Should there be a recommendation about not relying on a single, statically configured CA certificate in the Master Device?]

10.2. Protection Against Modification

To prevent a PAWS response message from being modified en route, messages must be transmitted over an integrity-protected channel. Using HTTP over TLS, the channel will be protected by appropriate cyphersuites.

10.3. Protection Against Eavesdropping

Using HTTP over TLS, messages protected by appropriate cyphersuites are also protected from eavesdropping or otherwise access by unauthorized parties en route

10.4. Client Authentication Considerations

Although the Database can inform a device of available spectrum it can use, the Database cannot enforce that the Master Device uses any/only those frequencies. Indeed, a malicious device can operate without ever contacting a database. Consequently, client authentication is not required for the core PAWS protocol (although it may be required by specific regulators). Depending on a prior relationship between a Database and Master Device, the Database MAY require client authentication. TLS provides client authentication, but there are some considerations:

- As indicated in Section 3.2 of HTTP Over TLS [RFC2818], the TLS client authentication procedure only determines that the device has a certificate chain rooted in an appropriate CA (or a self-signed certicate). The database would not know what the client identity ought to be, unless it has some external source of information. Distribution and management of such information, including revocation lists, are outside the scope of this document.
- Authentication schemes are secure only to the extent that secrets or certificates are kept secure. When there are a vast number of deployed devices using PAWS, the possibility that device keys will not leak becomes small. Implementations should consider how to manage the system in the eventuality that there is a leak.

11. Contributors

This document draws heavily from the following Internet Draft documents, [I-D.das-paws-protocol] and [I-D.wei-paws-framework]. The editor would like to specifically call out and thank the contributing authors of these two documents.

Donald Joslyn
Spectrum Bridge Inc.
1064 Greenwood Blvd.
Lake Mary, FL 32746
U.S.A.
Email: d.joslyn at spectrumbridge dot com

Xinpeng Wei
Huawei
Phone: +86 13436822355
Email: weixinpeng@huawei.com

12. Acknowledgments

The authors gratefully acknowledge the contributions of: Gabor Bajko, Teco Boot, Nancy Bravin, Rex Buddenberg, Gerald Chouinard, Stephen Farrell, Michael Fitch, Joel M. Halpern, Jussi Kahtava, Warren Kumari, Paul Lambert, Andy Lee, Anthony Mancuso, Basavaraj Patil, Scott Probasco, Brian Rosen, Andy Sago, Peter Stanforth, John Stine, and Juan Carlos Zuniga.

13. References

13.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC3339] Klyne, G., Ed. and C. Newman, "Date and Time on the Internet: Timestamps", RFC 3339, July 2002.

[RFC5077] Salowey, J., Zhou, H., Eronen, P., and H. Tschofenig, "Transport Layer Security (TLS) Session Resumption without Server-Side State", RFC 5077, January 2008.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, August 2008.

[RFC5491] Winterbottom, J., Thomson, M., and H. Tschofenig, "GEOPRIV Presence Information Data Format Location Object (PIDF-LO) Usage Clarification, Considerations, and Recommendations", RFC 5491, March 2009.

[RFC6350] Perreault, S., "vCard Format Specification", RFC 6350, August 2011.

13.2. Informative References

[I-D.bhat-vcarddav-json]
Bhat, R. and P. Saint-Andre, "A JavaScript Object Notation (JSON) Representation for vCard", draft-bhat-vcarddav-json-00 (work in progress), June 2012.

[I-D.das-paws-protocol]
Das, S., Malyar, J., and D. Joslyn, "Device to Database Protocol for White Space", draft-das-paws-protocol-02 (work in progress), July 2012.

[I-D.ietf-paws-problem-stmt-usecases-rqmts]
Probasco, S. and B. Patil, "Protocol to Access White Space database: PS, use cases and rqmts", draft-ietf-paws-problem-stmt-usecases-rqmts-08 (work in progress), August 2012.

[I-D.wei-paws-framework]
Wei, X., Zhu, L., and P. McCann, "PAWS Framework", draft-wei-paws-framework-00 (work in progress), July 2012.

[I-D.zyp-json-schema]
Zyp, K. and G. Court, "A JSON Media Type for Describing the Structure and Meaning of JSON Documents", draft-zyp-json-schema-03 (work in progress), November 2010.

[ISO3166-1]
"Country Codes", <http://www.iso.org/iso/country_codes.htm>.

[JSON-RPC]
"JSON-RPC 2.0 Specification", <http://www.jsonrpc.org/specification>.

[RFC2818] Rescorla, E., "HTTP Over TLS", RFC 2818, May 2000.

[RFC4627] Crockford, D., "The application/json Media Type for JavaScript Object Notation (JSON)", RFC 4627, July 2006.

Appendix A. Changes / Author Notes.

Notes:

- Change needed: maxValidityTime in RulesetInfo doesn't work for US/FCC, since the rule indicates 11:59 of following day (no timezone specified), which cannot be expressed as a duration

Changes from 00:
- Add JSON encoding
- Adopt RFC5491 for GeoLocation
- Adopt vCard for contact information
- Add Response Code section and update text referencing the defined response codes
- Change DeviceIdentifier to be DeviceDescriptor, allowing identifiers and device-characteristic fields to be included.

Appendix B. Regulatory-specific Notes

This section contains regualtory-specific details that should be moved to other documents.

B.1. US / FCC

Device Registration
- The "antenna" parameter is REQUIRED for FIXED device types. It MUST specify the antenna height the and height type.
- The "owner" parameter is REQUIRED for FIXED device types and if the Device has not been registered yet with the Database or if the Database does not implement a separate device registration request.

Device Identifier
- The authority value MUST be "US"
- The identity value is the Device's FCC ID
- The "deviceType" parameter is REQUIRED to indicate the device type. Valid values are FIXED, MODE_1, MODE_2

Antenna Height
- The height and heightType parameters are REQUIRED for FIXED device types.

RuleSet Info

- "authority" MUST be the string, "US"
- "maxLocationChange" MUST be 50 meters
- "maxPollingSecs" MUST be 86400 seconds, corresponding to 24 hours

- "maxValiditySecs" MUST be 172800 seconds, corresponding to 48 hours

B.2. UK / Ofcom

Device Identifier

- The authority value MUST be "UK"
- The "RAT" parameter is required to specify the Radio Access Technology. Valid values are TBD.
- A "deviceClass" parameter is required to identify, among other things, the emission mask of the Device. Valid values are TBD.

GeoLocation

- Requires uncertainty in the X (longitude) and Y (latitude) directions

Authors' Addresses

Vincent Chen (editor)
Google
1600 Amphitheatre Parkway
Mountain View, CA 94043
US

Email: vchen@google.com

Subir Das
Applied Communication Sciences
444 Hoes Lane
Piscataway, NJ 08854
U.S.A.

Phone:
Fax:
Email: sdas at appcomsci dot com
URI:

Zhu Lei
Huawei

Phone: +86 13910157020
Fax:
Email: lei.zhu@huawei.com
URI:

John Malyar
Telcordia Technologies Inc.
1 Ericsson Drive
Piscataway, NJ 08854
U.S.A.

Phone:
Fax:
Email: jmalyar at telcordia dot com
URI:

Peter J. McCann
Huawei
400 Crossing Blvd, 2nd Floor
Bridgewater, NJ 08807
USA

Phone: +1 908 541 3563
Fax:
Email: peter.mccann@huawei.com
URI: