Abstract

This document defines PHTAL, a generic representation format for hypertext applications guided by REST constraints. PHTAL could be compared to HTML without graphical requirements.

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 https://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 August 30, 2019.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (https://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.
1. Introduction

This document defines PHTAL, a generic representation format for hypertext applications guided by REST constraints. PHTAL could be compared to HTML without graphical requirements.

This document registers two media-type identifiers with the IANA: "application/phtal+json" and "application/phtal+xml". This registration is for community review and will be submitted to the IESG for review, approval, and registration with IANA.

1.1. Definitions and Terminology
1.1.1. Terminology and Conformance Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", 
"SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT REQUIRED", "MAY", and 
"OPTIONAL" in this document are to be interpreted as described in 
BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all 
capitals, as shown here.

1.1.2. General

Representational State Transfer, or *REST*, is an architectural style 
for distributed hypermedia systems. Introduced and first defined in 
2000 in Chapter 5, REST, of the doctoral dissertation "Architectural 
Styles and the Design of Network-based Software Architecture" by Roy 
Fielding.

*Hypermedia*, or hypertext, is defined by the presence of application 
control information embedded within, or as a layer above, the 
presentation of information. Hypermedia allows for a virtually 
unbound network of resources while also guiding users through an 
application as they navigate said relationships.

A *hypermedia relationship*, also known as a link relation, describes 
the semantics behind a virtual uni-directional association between 
two resources.

A *hypermedia relationship name* is an identifier for a hypermedia 
relationship.

A *resource* is the intended conceptual target of a hypertext 
reference.

*Representational state* indicates the current state of the requested 
resource, the desired state for the requested resource, or the value 
of some other resource, such as a representation of the input data 
within a client’s query form, or a representation of some error 
condition for a response.

*Application state* is the state of the user’s application of 
computing to a given task, controlled and stored by the user agent 
and can be composed of representations from multiple servers.

1.1.3. Documents description

A trailing question mark, for example *description?*, indicates an 
optional property.
1.2. Motivation

The essential trade-off that REST makes when compared to an architectural style like RPC is dynamic modifiability over efficiency. Dynamic modifiability is the degree to which an application can be changed without stopping and restarting the entire system. This is what REST promises through the Uniform Interface, and optionally Code-On-Demand, constraints.

Guided by these constraints PHTAL introduces the necessary elements to enable application authors to create evolvable and extensible applications.

2. PHTAL Representations

2.1. Hypermedia as the engine of application state

The Uniform Interface constraint dictates that hypermedia be the engine of application state. This means that the state of the application and its potential transitions are dictated by the presence of hypermedia relationships in-band and by the navigation of those relationships by an user (human or automated). In order for users to traverse a selected relationship they depend on the server to provide instructions for communicating with the target resource.

When servers provide control information at run-time instead of at deploy-time, they retain control of their implementation space and enable dynamic evolvability; they can change their implementation without having to restart or deploy clients. Applications servers are free to change their URI structure, they are free rearrange resources into different servers, they are free introduce new links that provide new features in existing representations, nothing will break already deployed components as long as links are not broken.

PHTAL introduces generic but comprehensive hypertext markup so that instead of creating and registering a new, application specific, hypertext enabled media type, authors can choose to make use of PHTAL’s markup. This frees authors to spend most of their descriptive efforts in defining application-specific representation and possibly on extended link relations to drive application state.

2.1.1. Document Root

The in-band elements defined by PHTAL aim to provide just what’s necessary for agents to evaluate the hypertext relationship alternatives provided and invoke the appropriate operation to get the agent to the next application state.
2.1.1.1. Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>links?</td>
<td>Map[&quot;string&quot;, [Link (Section 2.1.2)])]</td>
<td>The links element is a map where the keys are hypermedia relationship identifiers and the values are single or multiple Link elements. The relationship identifier MUST be a IANA registered relation type or an URI that when dereferenced resolves to an XREL [I-D.draft-montoya-xrel-00] document.</td>
</tr>
<tr>
<td>operations?</td>
<td>[Operation (Section 2.1.3)]</td>
<td>Informs an agent of what operations are allowed to be invoked on the context resource.</td>
</tr>
</tbody>
</table>

The operations element MAY be included as part of an HTTP GET response body, or as the response body to an HTTP OPTIONS, for example.

Detailed mappings to XML and JSON are provided through the appropriate schemas in Section #5 of this document.

2.1.1.2. PHTAL examples

*JSON Representation Example*
{  
  "name": [  
    {  
      "use": "official",  
      "family": "Chalmers",  
      "given": [  
        "Peter",  
        "James"  
      ]  
    }  
  ],  
  "_links": {  
    "https://api-docs.myclinic.com/fhir/rel/encounter": [{  
      "href": "http://fhir.myclinic.com/Encounter/1234",  
      "operation": {  
        "HTTP": {  
          "method": "GET",  
          "produces": "application/phtal+json;profile="http://hl7.org/fhir/json-schema/Encounter",  
          application/phtal+xml;profile="http://hl7.org/fhir/encounter.xsd""  
        }  
      }  
    }  
  },  
  "_operations": {  
    "HTTP": {  
      "method": "PUT",  
      "consumes": "application/phtal+json;profile="http://hl7.org/fhir/json-schema/Encounter",  
      application/phtal+xml;profile="http://hl7.org/fhir/encounter.xsd""
    }  
  }  
}  

*XML Representation Example*

```xml
<Encounter id="1234">
  <status value="active"/>
  <encounterPeriod>
    <start value="2019-01-01T00:00:00Z"/>
    <end value="2019-01-31T23:59:59Z"/>
  </encounterPeriod>
  <participant>
    <individual reference="Practitioner/123456"/>
  </participant>
</Encounter>
```
  <id value="example"/>
  <name>
    <use value="official"/>
    <family value="Chalmers"/>
    <given value="Peter"/>
    <given value="James"/>
  </name>
  <phtal:link rel="https://api-docs.myclinic.com/fhir/rel/encounter">
    <phtal:operation protocol="HTTP">
      <phtal:method>GET</phtal:method>
      <phtal:produces>application/phtal+json;profile="http://hl7.org/fhir/json-schema/Encounter",
        application/phtal+xml;profile="http://hl7.org/fhir/encounter.xsd"
      </phtal:produces>
    </phtal:operation>
  </phtal:link>
  <phtal:operation protocol="HTTP">
    <phtal:method>PUT</phtal:method>
    <phtal:consumes>application/phtal+json;profile="http://hl7.org/fhir/json-schema/Encounter",
      application/phtal+xml;profile="http://hl7.org/fhir/encounter.xsd"
    </phtal:consumes>
    <phtal:produces>application/phtal+json;profile="http://hl7.org/fhir/json-schema/OperationOutcome",
      application/phtal+xml;profile="http://hl7.org/fhir/operationoutcome.xsd"
    </phtal:produces>
  </phtal:operation>
</Patient>

2.1.2. Link

2.1.2.1. Properties
## Link Examples

### *JSON Representation Example*
The most important element that PHTAL representations provide is the operation element. This element instructs the agent on how to interact with a resource through a particular communication protocol. This allows the server to control its own URI space and the clients to be undisrupted when URI changes are made or new representation or protocol support is added.

Identifying protocol specific instructions allows servers to separate communication protocols from resource identification. This is consistent with the URI specification [RFC3986] and allows PHTAL to support arbitrary protocols.
### 2.1.3.1. Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>method?</td>
<td>&quot;string&quot;</td>
<td>Instructs the agent what protocol method to use when interacting with the identified resource. The default value is whatever protocol specific method results in information retrieval, eg. HTTP GET.</td>
</tr>
<tr>
<td>produces?</td>
<td>&quot;string&quot;</td>
<td>Indicates to the client what media types the server supports as response content to following the current link. It MUST be a media range and parameters according to Section 5.3.2 ‘Accept’ of the HTTP Specification [RFC7231].</td>
</tr>
<tr>
<td>consumes?</td>
<td>&quot;string&quot;</td>
<td>Indicates to the client what media types the server supports as request content to following the current link. It MUST be a media range and parameters according to Section 5.3.2 ‘Accept’ of the HTTP Specification [RFC7231].</td>
</tr>
<tr>
<td>requestContent?</td>
<td>&quot;boolean&quot;</td>
<td>Indicates to the client whether a request content is required or not for the following the current link. The default value is &quot;false&quot;.</td>
</tr>
<tr>
<td>onInvoke?</td>
<td>EventAttribute</td>
<td>Script function which is to be executed when this operation is invoked.</td>
</tr>
</tbody>
</table>

The quality weight parameters MAY be used in the "consumes" and "produces" properties to indicate to the client which media types are preferred, possibly allowing the client to know when a known media type has been superseded and a new one is preferred.
2.1.3.2. Operation Examples

*JSON Representation Example*

```json
{
  "method": "POST",
  "consumes": "application/phtal+xml;profile="http://hl7.org/fhir/json-schema/Appointment",
  "produces": "application/phtal+xml;profile="http://hl7.org/fhir/json-schema/OperationOutcome",
  "requestContent": true,
  "onInvoke": "...
}
```

*XML Representation Example*

```xml
<phtal:operation protocol="HTTP" onInvoke="...">
  <phtal:consumes>application/phtal+xml;profile="http://hl7.org/fhir/appointment.xsd"
  </phtal:consumes>
  <phtal:produces>application/phtal+xml;profile="http://hl7.org/fhir/operationoutcome.xsd"
  </phtal:produces>
  <phtal:requestContent>true</phtal:requestContent>
</phtal:operation>
```

2.1.4. HTTP Operation

The HTTP Operation element is an extension of the Operation element specifically for interactions using the HTTP protocol.

2.1.4.1. Added Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>securedBy?</td>
<td>[SecurityRequirement (Section 2.1.5)]</td>
<td>An array of SecurityRequirement elements, the operation can be authenticated by any of the specified security schemes.</td>
</tr>
<tr>
<td>headers?</td>
<td>Map[&quot;string&quot;, &quot;string&quot;]</td>
<td>A map where the keys are the names of the HTTP headers to be sent and the values are URIs that resolve to RAML 1.0 Spec [RAML] Data Type declarations explaining the format and semantics of the variables.</td>
</tr>
</tbody>
</table>
2.1.4.2. HTTP Operation Examples

*JSON Representation Example*

```json
{
  "method": "POST",
  "consumes": "application/phtal+json;profile="
      "http://hl7.org/fhir/json-schema/Appointment"",
  "produces": "application/phtal+json;profile="
      "http://hl7.org/fhir/json-schema/OperationOutcome"",
  "requestContent": true,
  "securedBy": [
    { "scheme": "https://api-docs.myclinic.com/fhir/security/basicAuth" },
    { "scheme": "https://api-docs.myclinic.com/fhir/security/oauth2.0",
      "scopes": [ "appointment:write" ]
    }
  ],
  "headers": {
    "trace-id": "https://api-docs.myclinic.com/fhir/traceId.raml"
  }
}
```

*XML Representation Example*

```xml
<phtal:operation protocol="HTTP">
  <phtal:consumes>application/phtal+xml;profile="http://hl7.org/fhir/appointment.xsd"
</phtal:consumes>
  <phtal:produces>application/phtal+xml;profile="http://hl7.org/fhir/operationoutcome.xsd"
</phtal:produces>
  <phtal:requestContent>true</phtal:requestContent>
  <phtal:securedBy scheme="https://api-docs.myclinic.com/fhir/security/basicAuth">
    <phtal:scope>appointment:write</phtal:scope>
  </phtal:securedBy>
  <phtal:securedBy scheme="https://api-docs.myclinic.com/fhir/security/oauth2.0"/>
</phtal:operation>
```

2.1.5. SecurityRequirement

2.1.5.1. Properties
<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>scheme</td>
<td>&quot;string&quot;</td>
<td>An URI that MUST resolve to RAML 1.0 Spec [RAML] Security Scheme declarations.</td>
</tr>
<tr>
<td>scopes?</td>
<td>[&quot;string&quot;]</td>
<td>A list of the scopes required for authorization. Scopes are required when the security scheme is of type OAuth 2.0.</td>
</tr>
</tbody>
</table>

2.1.6. Partial

Partial representation of the linked resource.

2.1.6.1. Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>&quot;string&quot;</td>
<td>Identifies the media type that describes the partial representation.</td>
</tr>
<tr>
<td>data</td>
<td>Any</td>
<td>The actual partial representation content.</td>
</tr>
</tbody>
</table>

Partial content SHOULD NOT be considered full representations even if their contents happen to be complete. It is RECOMMENDED partial representations provide just enough information for agents to be able to discern which link they want to follow and SHOULD NOT be used as mechanism to batch interactions.

In the case of XML it is the content of the "partial" element, in JSON it is the value of a "data" property.

2.1.6.2. Partial Examples

*JSON Representation Example*
2.2. Self-descriptive messages

The Uniform Interface constraint also dictates that messages be self-descriptive. This is achieved by message metadata, of which content type metadata is an important part. However, the purpose of content type metadata in web interactions is not only to indicate representation format or schema, but the sender’s preferred interpretation of that format, an application-specific format.

By making use of media type parameters, PHTAL representations allow participants to retain the ability to signal their preferred interpretation of a message through metadata. Message authors only have to identify the document that defines the application-specific format of their representation and attach it to an otherwise generic PHTAL representation.

When web participants identify an application-specific format in metadata they promote visibility and evolvability. Intermediaries (i.e., proxies and gateways) are able to accurately and more efficiently perform significant functions such as encapsulating legacy services, and enhancing client functionality.
2.2.1. Linking to a profile

For example consider the following interactions:

POST http://www.example.com/someIdentifier
Content-Type: application/json
Accept: application/json

200 OK
Content-Type: application/json

This interaction can only be accurately interpreted to mean that the client requested resource "http://www.example.com/someIdentifier" to process an "application/json" request and it successfully responded with an "application/json" response. "application/json" offers intermediaries no semantic information about the content of the message besides how it's (de)serialized.

To indicate the format and semantics of a PHTAL representation, the sender SHOULD identify a document that explains additional semantics using a "profile" media type parameter.

POST http://www.example.com/someIdentifier
Content-Type: application/phtal+json;profile="http://hl7.org/fhir/json-schema/Appointment"
Accept: application/phtal+json;profile="http://hl7.org/fhir/json-schema/OperationOutcome"

200 OK
Content-Type: application/phtal+json;profile="http://hl7.org/fhir/json-schema/OperationOutcome"

In contrast, this second interaction is perfectly clear. The client asked "http://www.example.com/someIdentifier" to process a clinical Appointment request and it successfully responded with an OperationOutcome response that details the results of the processing. Intermediaries are able to parse and manipulate the message, perhaps defaulting values of the appointment request, or adding and/or removing links from the response, or maybe redirecting the message to different resources based on the profile information.

The profile parameter SHOULD be a dereferenceable URI that resolves to a RAML 1.0 Spec [RAML] Data Type declaration. RAML data types are used because of their ability to describe representations independent of their runtime media type, as well as supporting XSD and JSON Schema documents.

TODO: Consider restricting profile parameter to a single value, forego the link rel definition.
2.3. Code-On-Demand

In order to further promote modifiability of a system REST defines code-on-demand as an optional constraint. An optional constraint would observe desired behavior where supported, but with the understanding that it may not be the general case. Code-on-demand is a style of code mobility in which the processing logic is moved from the server into the client, thus providing dynamic extensibility; functionality can be added to a deployed component without impacting the rest of the system.

Very similar to HTML’s script element PHTAL provides ways to embed code into its representations.

Ultimately, application servers may prefer if legacy clients could adapt to new representations or communication protocols instead of having to support overloaded versions of a feature. At the cost of visibility, code-on-demand allows application servers to re-program a deployed component to support new features, thus freeing the server from the responsibility of maintaining backwards compatibility.

It’s worthy to mention other advantages of code-on-demand outside of modifiability. Scalability of the server is improved, since it can off-load work to the client. User-perceived performance and efficiency are enhanced when the code can adapt its actions to the client’s environment and interact with the user locally rather than through remote interactions.

2.3.1. Script

A script element is equivalent to the script element in HTML and thus is the place for scripts (e.g., ECMAScript). Any functions defined within any script element have a "global" scope across the entire current document.

TODO: Consider what to include about DOM and Events.

2.3.1.1. Properties
<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>&quot;string&quot;</td>
<td>Identifies the media type that describes the script content.</td>
</tr>
<tr>
<td>source?</td>
<td>&quot;string&quot;</td>
<td>An URI that references the script’s content.</td>
</tr>
<tr>
<td>data?</td>
<td>Any</td>
<td>The actual contents of the script. It is mutually exclusive with the source property.</td>
</tr>
</tbody>
</table>

*JSON Representation Example*

```json
{
  "_scripts": [{
    "type": "text/javascript",
    "source": "http://fhir.myclinic.com/scripts/patientScript"
  }, {
    "type": "text/javascript",
    "data": "...
  }]
}
```

*XML Representation Example*

```xml
  <!--<![CDATA[
  function foo(evt) {
    ...
  }\]-->  
</phtal:script>
</Patient>
```

3. IANA Considerations

This specification establishes two media types: 'application/phtal+xml' and 'application/phtal+json'

TODO: Update schemas linked.
3.1. application/phtal+xml

*Type name:* application

*Subtype name:* phtal+xml

*Required parameters:* none

*Optional parameters:*

*charset:* This parameter has identical semantics to the charset parameter of the ‘application/xml’ media type as specified in [RFC7303].

*profile:* A whitespace-separated list of URIs identifying specific constraints or conventions that apply to a PHTAL document. A profile must not change the semantics of the resource representation when processed without profile knowledge, so that clients both with and without knowledge of a profiled resource can safely use the same representation. The profile parameter may also be used by clients to express their preferences in the content negotiation process. Profile URIs MUST be dereferenceable and resolve to a profile document as defined in Section #5.1 of this document.

*Encoding considerations:*

*binary:* Same as encoding considerations of application/xml as specified in [RFC7303].

*Security considerations:*

This format shares security issues common to all XML content types. The security issues of [RFC7303], section 10, should be considered.

Several PHTAL elements may cause arbitrary URIs to be referenced. In this case, the security issues of [RFC3986], section 7, should be considered.

In common with HTML, PHTAL documents may reference external scripting language media. Scripting languages are executable content. In this case, the security considerations in the Media Type registrations for those formats shall apply.

*Interoperability considerations:* An xsd document detailing the format of application/phtal+xml is located at http://www.phtal.org/schema/phtal+xml.xsd [1]
3.2.  application/phtal+json

*Type name:* application

*Subtype name:* phtal+json

*Required parameters:* none

*Optional parameters:*

*profile:* A whitespace-separated list of URIs identifying specific constraints or conventions that apply to a PHTAL document. A profile must not change the semantics of the resource representation when processed without profile knowledge, so that clients both with and without knowledge of a profiled resource can safely use the same representation. The profile parameter may also be used by clients to express their preferences in the content negotiation process. Profile URIs MUST be dereferenceable and resolve to a profile document as defined in Section #5.1 of this document.

*Encoding considerations:* binary
Security considerations:

This media type shares security issues common to all JSON content types. The security issues of [RFC8259], section 6, should be considered.

Several PHTAL elements may cause arbitrary URIs to be referenced. In this case, the security issues of [RFC3986], section 7, should be considered.

In common with HTML, PHTAL documents may reference external scripting language media. Scripting languages are executable content. In this case, the security considerations in the Media Type registrations for those formats shall apply.

Interoperability considerations:

A json-schema document detailing the format of application/phtal+json is located at http://www.phtal.org/schema/phtal+json.yaml [2]

Fragment identifier considerations:

none

Published specification:

This Document

Applications that use this media type:

Various

Additional information:

*magic number(s):* none

*file extensions:* .json

*macintosh type file code:* TEXT

*object identifiers:* none

Person to contact for further information:

*Name:* Jose Montoya

*Email:* jmontoya@ms3-inc.com

*Intended usage:* Common

*Author/change controller:* Jose Montoya
4. References

4.1. Normative References

[I-D. draft-montoya-xrel-00]
Montoya, J., "Extended Link Relationships", draft-montoya-xrel-00 (work in progress), February 2019.


4.2. Informative References

[I-D.draft-handrews-json-schema-hyperschema-01]

[I-D.draft-kelly-json-hal-08]
Kelly, M., "JSON Hypertext Application Language", draft-kelly-json-hal-08 (work in progress), May 2016.


4.3. URIs


Appendix A.  Acknowledgments

Thanks to Mike Kelly, Henry Andrews, Marcus Lanthaler, Mike Amundsen, Stu Charlton, and Jeff Michaud for their contributions in this space even if not directly related to PHTAL.

Appendix B.  Frequently Asked Questions

B.1.  How can I submit comments or feedback to the editors?

The issues list for this draft can be found at https://github.com/phtal-org/internet-draft-phtal/issues [3]. For additional information, see https://phtal-org.github.io/internet-draft-phtal/ [4].

To provide feedback, use this issue tracker, the communication methods listed on the homepage, or email the document editors.

Author’s Address

Jose Montoya
Mountain State Software Solutions

Email: jmontoya@ms3-inc.com