# Protocol for Web Description Resources (POWDER): Formal Semantics

## W3C Working Draft — 15 August 2008

**This version**
http://www.w3.org/TR/2008/WD-powder-formal-20080815/
**Latest version**
http://www.w3.org/TR/powder-formal/
**Previous version**
http://www.w3.org/TR/2008/WD-powder-formal-20080709/

**Editors:**
Stasinos Konstantopoulos, Institute of Informatics & Telecommunications (IIT), NCSR
Phil Archer, Family Online Safety Institute



---

## Abstract

This document underpins the Protocol for Web Description Resources (POWDER). It describes how the relatively simple operational format of a POWDER document can be transformed through two stages: first into a more tightly constrained XML format (POWDER-BASE), and then into an RDF/OWL encoding (POWDER-S) that may be processed by Semantic Web tools. Such processing is only possible, however, if tools implement the semantic extension defined within this document. The formal semantics of POWDER are best understood after the reader is acquainted with the Description Resources [DR] and Grouping of Resources [GROUP] documents.

## Status of this document

*This section describes the status of this document at the time of its publication. Other documents may supersede this document. A list of current W3C publications and the latest revision of this technical report can be found in the W3C technical reports index at http://www.w3.org/TR/.*

This is the Last Call Working Draft of this document, the Last Call period being synchronized with two other documents in the set: Description Resources and Grouping of Resources. These three document are expected to be advanced to Recommendation Status. The POWDER Working Group welcomes comments on these through to 14 September 2008. Comments are equally welcome on the other documents that are also available as working drafts, in particular, the Primer and Test

Suite. Changes to this document since the previous version are recorded in the Change Log.

This document and the Description Resources [DR] document both show how POWDER can carry arbitrary RDF in the `attribution` and `descriptorset` elements. As the text and examples in Section 3.2.1 show, this is potentially problematic. POWDER works well when it transports RDF properties that have literals or RDF resources as objects (see Example 3-9), but the semantics are less clear when more complex graphs are included. We therefore flag this — that is, the support for arbitrary RDF in POWDER — as a **Feature at Risk**. Removing this feature would have the following effects:

First, descriptions would be limited to the two types below:

```
<descriptorset>
  <ex:color>red</ex:color>
  <ex:finish rdf:resource="http://example.org/vocab#shiny" />
</descriptorset>
```

Secondly, it would not be possible to include details of the entity that created the POWDER document within the document itself. Thus DR authors would be *required* to publish a separate RDF description of themselves (using FOAF or DC Terms).

Against these limitations, the benefit of removing this feature is substantial: it would be possible to build specialized software that would process POWDER just as XML, without having to process RDF. For instance: Example 2-2 can be transformed into POWDER-S and processed in a semantic environment, but it could equally be processed purely as XML if the unresolved URIrefs gave sufficient information for the application's needs.

It is worth noting that POWDER-S would be unaffected by the removal of this feature.

The Working Group is anxious to receive feedback on this trade-off between flexibility and ease of implementation.

Please send comments about this document to public-powderwg@w3.org (with public archive).

Publication as a Working Draft does not imply endorsement by the W3C Membership. This is a draft document and may be updated, replaced or obsoleted by other documents at any time. It is inappropriate to cite this document as other than work in progress.

This document was produced by a group operating under the 5 February 2004 W3C Patent Policy. W3C maintains a public list of any patent disclosures made in connection with the deliverables of the group; that page also includes instructions for disclosing a patent. An individual who has actual knowledge of a patent which the individual believes contains Essential Claim(s) must disclose the information in accordance with section 6 of the W3C Patent Policy.

## Table of Contents

# 1 Introduction

The Protocol for Web Description Resources, POWDER, offers a simple method of associating RDF data with groups of resources. Its primary 'unit of information' is the Description Resource (DR). This comprises three elements:

- attribution (who is providing the description)
- scope (defined as a set of IRIs over which the description applies to the resources de-referenced from those IRIs)
- the description itself (the 'descriptor set').

To some extent, this approach is in tension with the core semantics of RDF and OWL. To resolve that tension, it is necessary to extend RDF semantics as described below. In order to minimize the required extension, while at the same time preserving the relatively simple encoding of POWDER in XML which is generally readable by humans, we define a multi-layered approach. The operational semantics, i.e. the encoding of POWDER in XML, is first transformed into a more restricted XML encoding that is less easily understood by humans and depends on matching IRIs against regular expressions to determine whether or not they are within the scope of the DR. This latter encoding is, in its own turn, transformed into the extended-RDF encoding.

The data model makes the attribution element mandatory for all POWDER documents. These may contain any number of Description Resources (DRs) that effectively inherit the attribution of the document as a whole. Descriptor sets may also be included independently of a specific DR, and these too inherit the attribution. This model persists throughout the layers of the POWDER model, which are as follows:

**POWDER**

The operational encoding, a dialect of XML, that transports the RDF data. It is expected that POWDER will typically be published and processed in this form.

POWDER's resource grouping methods are mostly geared towards URLs and Information Resources as defined in the Architecture of the World Wide Web [WEBARCH].

**POWDER-BASE**

This is a largely theoretical XML encoding of POWDER that reduces all means of grouping resources according to their IRI into a single grouping method, that of matching IRIs against arbitrary regular expressions.

POWDER-BASE is provided as a means of formally specifying the semantics of the various IRI grouping methods defined in POWDER. POWDER-BASE is generated automatically from POWDER by means of the GRDDL transform [GRDDL] that is associated with the POWDER namespace.

Elements not concerned with IRI set definition are identical in POWDER and POWDER-BASE.

**POWDER-S (Semantic POWDER)**

The Semantic encoding uses a fragment of RDF/OWL that has been extended in a way that facilitates the matching of the string representation of a resource's identifier against a regular expression.

OWL classes are used to represent sets of resources, grouped according to their IRI and according to their properties (descriptors). Resources are described by asserting that a class that defines a set of IRIs is a sub class of a descriptor-defined class-set. Attribution is provided by way of an RDF description of the RDF graph as a whole.

A small RDF vocabulary is needed to support POWDER-S. Although it is valid RDF/OWL, generic tools will only be able to process the semantics of POWDER-S if they implement the necessary extension defined in this document.

POWDER-S is generated from POWDER-BASE by means of the GRDDL transform [GRDDL] that is associated with the POWDER namespace. POWDER-S MAY be created directly, but this is generally inadvisable since, whilst a POWDER Processor MUST understand and process POWDER-BASE and SHOULD understand POWDER, it MAY NOT understand and process POWDER-S. The aim of POWDER-S is to make the data available to the broader Semantic Web via GRDDL, not to create an alternative encoding.

The conformance criteria for a POWDER Processor are given in the Description

Resources document [DR].

The GRDDL transform from POWDER to POWDER-BASE to POWDER-S is achieved using multiple passes of a POWDER document through an XSLT [XSLT] instance. The Working Group has developed XSLT programs that perform the transforms described in this document and the other documents in the POWDER document suite, and are associated with the POWDER GRDDL namespaces. These transforms are consistent with the normative text in this document, but their syntactic specifics are not normative; in effect, a POWDER processor MAY use different transforms to produce syntactically different but semantically equivalent OWL/RDF for processing a POWDER document.

Description Resources are defined separately [DR] and a further document defines the creation of IRI sets [GROUP]. Readers should be familiar with those documents before proceeding with this one. The full set of POWDER documents also includes its Use Cases, Primer and Test Suite, together with the namespace documents [WDR, WDRS, WDRD] and GRDDL transform [PDR-GRDDL].

## 1.1 Namespaces, Terminology and Conventions Used in This Document.

The POWDER vocabulary namespace is `http://www.w3.org/2007/05/powder#` for which we use the prefix `wdr` The POWDER-S vocabulary namespace is `http://www.w3.org/2007/05/powder-s#` for which we use the prefix `wdrs` All prefixes used in this document, together with their associated namespaces, are shown in the table below.

| Prefix | Namespace |
|---|---|
| wdr | http://www.w3.org/2007/05/powder# |
| wdrs | http://www.w3.org/2007/05/powder-s# |
| rdf | http://www.w3.org/1999/02/22-rdf-syntax-ns# |
| rdfs | http://www.w3.org/2000/01/rdf-schema#" |
| owl | http://www.w3.org/2002/07/owl# |
| ox | http://www.w3.org/ns/owl2-xml# |
| dcterms | http://purl.org/dc/terms/ |
| foaf | http://xmlns.com/foaf/0.1/ |
| xsd | http://www.w3.org/2001/XMLSchema-datatypes# |
| xsl | http://www.w3.org/1999/XSL/Transform |
| ex | An arbitrary prefix used to denote an 'example vocabulary' |

Table 1: Prefixes and Namespaces used in this document

Unqualified elements in this document are from the `wdr` namespace.

In this document, the words MUST, MUST NOT, SHOULD, SHOULD NOT and MAY are to be interpreted as described in RFC2119 [RFC2119].

For convenience and transparency, we have used the RDF/XML serialization for POWDER-S as we have throughout the document set. Other serializations, such as N3 [N3], are equally valid for POWDER-S. The GRDDL Transformation associated with the POWDER namespace, which uses XSLT to effect the transform, produces RDF/XML as its output.

Examples in this document show fragments of data and each is linked to an external file that mirrors the data in the text. However, in order to be valid documents, the external files include generic data not shown in the text that has been taken largely from examples 2-1 and 2-3 in the Description Resources document [DR].

# 2 Attribution Element Semantics

The `attribution` element, present in all POWDER documents, provides data about the authorship, validity period, and other issues that a user or user agent can use when deciding whether or not to confer their trust on a POWDER document.

## 2.1 POWDER Attribution Semantics

Most `attribution` elements are not involved in IRI grouping, and as such are untouched during the transformation from POWDER to POWDER-BASE. The only exception is `abouthosts`, which sets an outer limit on the resources described by the DRs within the document. POWDER `abouthosts` elements are translated into POWDER-BASE `aboutregex` elements, as discussed in Section 4.5 below.

## 2.2 POWDER-S Attribution Semantics

Since the `attribution` element provides data about the document itself, it is transformed from POWDER (through POWDER-BASE) into POWDER-S as an `owl:Ontology` property for which the subject is null (i.e. the current document). This data does not receive OWL semantics, but is only meaningful to POWDER tools when deciding whether a POWDER document *as a whole* should be taken into account or discarded.

Child elements of the `attribution` element are RDF/XML statements about the document. With the explicit exception of `issued` and `abouthosts`, these are reproduced unchanged in the POWDER-S instance. This general rule applies to the POWDER elements `issuedby`, `validfrom`, `validuntil`, `certifiedby` and `supportedby` where the only transformation necessary is to make their (transformed) namespace explicit. In each case the same string is used as their element name and `wdrs` property name. The `wdrs:issuedby` property, however, is noteworthy as it is required for all POWDER documents and has particular semantics discussed below the following example.

As a shortcut designed to avoid the automatic need to declare the `rdf` namespace in all POWDER documents, where child elements of the `attribution` element of a POWDER document contain an external reference, denoted by the `src` attribute, this is transformed into `rdf:resource`.

RDF is copied verbatim. **TBD:** This is a Feature at Risk, see Status Section

Example 2-1 shows the generic semantics of the `attribution` element.

**Example 2-1: The Generic Semantics of the `attribution` Element**

POWDER [XML]

```
<attribution>
  <ex:property1>value</ex:property1>
  <ex:property2 rdf:resource="http://example.org/foo.rdf#frag" />
  <ex:property3 src="http://example.com/bar.rdf#frag" />
  <ex:property4>
    <ex:Class>
      <ex:property5>value_5</ex:property5>
    </ex:Class>
  </ex:property4>
</attribution>
```

POWDER-S [RDF/XML]

```
<rdf:Description rdf:about="">
  <ex:property1>value</ex:property1>
  <ex:property2 rdf:resource="http://example.org/foo.rdf#frag" />
  <ex:property3 rdf:resource="http://example.org/foo.rdf#frag" />
  <ex:property4>
    <ex:Class>
      <ex:property5>value_5</ex:property5>
    </ex:Class>
  </ex:property4>
</rdf:Description>
```

As noted above, some elements within the POWDER namespace that are treated differently or have noteworthy semantics:

**issuedby**

The `issuedby` element takes its semantics from both the Dublin Core [DC] and FOAF [FOAF] namespaces. `wdrs:issuedby` is defined as a sub property of **both** `dcterms:creator` **and** `foaf:maker`. These have a range of `dcterms:Agent` and `foaf:Agent` respectively, so that in this example triple:

```
<> wdrs:issuedby <http://example.org/company.rdf#me>
```

http://example.org/company.rdf#me *SHOULD* identify an instance of one of those classes (or a subclass thereof).

Alternatively, the `Agent` class (from either vocabulary) can be included in the POWDER document directly. Example 2-2 in the Description Resources document shows this.

**issued**

The semantics of the `issued` element are also defined in the `dcterms` namespace such that

```
<issued>2008-03-25T00:00:00</issued>
```

should be understood to mean

```
<dcterms:issued>2008-03-25T00:00:00</dcterms:issued>
```

**abouthosts**

The `abouthosts` element is discussed in Section 4.5 below.

The transformation of `validfrom`, `validuntil`, `certifiedby` and `supportedby` from POWDER to POWDER-S is straightforward in that, for example:

```
<validfrom>2008-01-01T00:00:00</validfrom>
<validuntil>2008-12-31T00:00:00</validuntil>
```

is transformed into

```
<wdrs:validfrom>2008-01-01T00:00:00</wdrs:validfrom>
<wdrs:validuntil>2008-12-31T00:00:00</wdrs:validuntil>
```

The data types for these elements and their `wdrs` properties are:

- `validfrom` and `validuntil`: W3C DTF
- `certifiedby` and `supportedby`: `xsd:anyURI`

Example 2-2 below shows all the POWDER-specific attribution elements. Note that there is no `abouthosts` element in the example, as it will be discussed in Section 4.3 below.

**Example 2-2: The Semantics of the POWDER-Specific Child Elements of the `att`**

POWDER [XML]

```
1  <attribution>
2    <issuedby src="http://example.org/company.rdf#me" />
3    <issued>2007-12-23T00:00:00</issued>
4    <validfrom>2008-01-01T00:00:00</validfrom>
5    <validuntil>2008-12-31T23:59:59</validuntil>
6    <certifiedby src="http://authority.example/powder.xml" />
7    <supportedby src="http://service.example.com?id=abc" />
8  </attribution>
```

POWDER-S [RDF/XML]

```
1  <rdf:Description rdf:about="">
2    <wdrs:issuedby rdf:resource="http://example.org/company.rdf#me" />
3    <dcterms:issued>2008-12-23T00:00:00</dcterms:issued>
4    <wdrs:validfrom>2008-01-01T00:00:00</wdrs:validfrom>
5    <wdrs:validuntil>2008-12-31T23:59:59</wdrs:validuntil>
6    <wdrs:certifiedby rdf:resource="http://authority.example/powder.xml
7    <wdrs:supportedby rdf:resource="http://service.example.com?id=abc"
8  </rdf:Description>
```

# 3 Description Resource Semantics

Description Resources use vocabularies defined in RDF and/or plain string literals (tags) to describe resources de-referenced from instances of the IRI set. Since descriptor set elements are not involved in the specification of the IRI set itself, they

are transferred verbatim from POWDER to POWDER-BASE. Example 3-1 below shows a generic example of a DR in which the IRI set has been elided for clarity (the semantics of the IRI set is discussed in Section 4 below).

**Example 3-1: A Generic Example of A Descriptor Set and Tag Set within a DR [X**

```
<dr>
  <iriset>…</iriset>

  <descriptorset>
    <ex:finish rdf:resource="http://example.org/vocab#shiny"/>
    <ex:shape>square</ex:shape>
  </descriptorset>

  <tagset>
    <tag>red</tag>
    <tag>light</tag>
  </tagset>
</dr>
```

The `ex:finish` element specifies that the `ex:finish` relation holds between all resources specified by `iriset` and the http://example.org/vocab#shiny resource.

The content of `ex:shape` is interpreted as a string literal. The `ex:shape` element specifies that all resources in `iriset` has the value "square" for the `ex:shape` dataproperty.

`tag` is a string property defined by POWDER. Its content is a single string literal, possibly including spaces.

The overall description of the resources in `iriset` is the union of the descriptions in the `descriptorset` and the `tagset`. In our example these are:

an ex:finish relation to http://example.org/vocab#shiny
**AND**
an ex:shape of "square"
**AND**
the tags "red" and "light"

We formally interpret the above as follows: there is an OWL class containing all resources that share all of these properties, and there is an OWL class of all resources denoted by `iriset`, and the latter is a subset of the former. In POWDER-S we say:

**Example 3-2 The POWDER-S Encoding of Example 3-1 [RDF/XML]**

```
<owl:Class rdf:nodeID="iriset_1">
  all resources specified by <iriset>...</iriset>
</owl:Class>

<owl:Class rdf:nodeID="descriptorset_1">
  <owl:intersectionOf rdf:parseType="Collection">
    <owl:Restriction>
      <owl:onProperty rdf:resource="http://example.org/vocab#finish"/
      <owl:hasValue rdf:resource="http://example.org/vocab#shiny"/>
    </owl:Restriction>
    <owl:Restriction>
      <owl:onProperty rdf:resource="http://example.org/vocab#shape"/>
      <owl:hasValue>square</owl:hasValue>
    </owl:Restriction>
  </owl:intersectionOf>
</owl:Class>

<owl:Class rdf:nodeID="tagset_1">
  <owl:intersectionOf rdf:parseType="Collection">
    <owl:Restriction>
      <owl:onProperty rdf:resource="http://www.w3.org/2007/05/powder-
      <owl:hasValue>red</owl:hasValue>
    </owl:Restriction>
    <owl:Restriction>
      <owl:onProperty rdf:resource="http://www.w3.org/2007/05/powder-
      <owl:hasValue>light</owl:hasValue>
    </owl:Restriction>
  </owl:intersectionOf>
</owl:Class>

<owl:Class rdf:nodeID="iriset_1">
  <rdfs:subClassOf rdf:nodeID="descriptorset_1"/>
  <rdfs:subClassOf rdf:nodeID="tagset_1"/>
</owl:Class>
```

It is possible to have more than one `iriset` element, in which case a resource receives all of the descriptions by belonging to any one of the corresponding IRI sets. For example:

**Example 3-3: A DR with Multiple IRI Sets in its Scope [XML]**

```
<dr>
 <iriset>.1.</iriset>
 <iriset>.2.</iriset>

 <descriptorset xml:id="silver">
   <ex:finish rdf:resource="http://example.org/vocab#shiny"/>
 </descriptorset>
</dr>
```

receives the following semantics:

**Example 3-4: The POWDER-S encoding of Example 3-3 [RDF/XML]**

```
1  <owl:Class rdf:nodeID="iriset_1">
2     all resources specified by <iriset>.1.</iriset>
3  </owl:Class>

4  <owl:Class rdf:nodeID="iriset_2">
5     all resources specified by <iriset>.2.</iriset>
6  </owl:Class>

7  <owl:Class rdf:ID="silver">
8    <owl:intersectionOf rdf:parseType="Collection">
9      <owl:Restriction>
10       <owl:onProperty rdf:resource="http://example.org/vocab#finish"/
11       <owl:hasValue rdf:resource="http://example.org/vocab#shiny"/>
12     </owl:Restriction>
13   </owl:intersectionOf>
14 </owl:Class>

15 <owl:Class rdf:nodeID="iriset_1">
16   <rdfs:subClassOf rdf:resource="#silver"/>
17 </owl:Class>

18 <owl:Class rdf:nodeID="iriset_2">
19   <rdfs:subClassOf rdf:resource="#silver"/>
20 </owl:Class>
```

Examples 3-3 and 3-4 also show that if a `descriptorset` element has an ID of its own, this is used in the POWDER-S document. This is reflected in the way that the sub class relationship (lines 15 - 20) is asserted. Where no `xml:id` attribute is set in the original POWDER document, the transform assigns `rdf:nodeID` identifiers for the blank nodes. As a result, `rdf:nodeID` attributes are used within the POWDER-S document in the sub class assertion (see lines 28 -31 in Example 3-2). However, where the original POWDER document includes an `xml:id` attribute, as in line 4 of Example 3-3, the sub class assertions in lines 16 and 19 of Example 3-4 is correctly asserted using the `rdf:resource` attribute.

A POWDER processor is free to choose any traversal policy for treating multiple `iriset` elements in a DR: first match wins, last match wins, shortest `iriset` first, and so on, as long as all `iriset` elements are tried before deciding that DR does not apply to a candidate resource (candidate resource is defined in the Grouping of Resources document [GROUP]). However, DR authors may use the order of the `iriset` elements to suggest an efficient scope evaluation strategy, by putting the `iriset` with the widest coverage first, so that a processor that chooses to follow the `iriset` elements in document order is more likely to terminate the evaluation after fewer checks.

## 3.1 Multiple Description Resources in a Single POWDER Document

A POWDER document may have any number of DRs, all of which are simultaneously asserted and ordering is not important. So, for example:

**Example 3-5: A POWDER Document Containing Multiple DRs [XML]**

```
1  <powder>

2    <dr>
3      <iriset>.1.<code>iriset</code>
4      <descriptorset>
5        <ex:shape>square</ex:shape>
6      </descriptorset>
7    </dr>

8    <dr>
9      <iriset>.2.<code>iriset</code>
10     <descriptorset>
11       <ex:finish rdf:resource="http://example.org/vocab#shiny"/>
12     </descriptorset>
13   </dr>

14 </powder>
```

receives the following semantics:

**Example 3-6: The POWDER-S Encoding of Example 3-5 [RDF/XML]**

```
1  <owl:Class rdf:nodeID="iriset_1">
2    all resources specified by <iriset>.1.</iriset>
3  </owl:Class>

4  <owl:Class rdf:nodeID="descriptorset_1">
5    <owl:intersectionOf rdf:parseType="Collection">
6     <owl:Restriction>
7        <owl:onProperty rdf:resource="http://example.org/vocab#shape"/>
8        <owl:hasValue>square</owl:hasValue>
9      </owl:Restriction>
10   </owl:intersectionOf>
11 </owl:Class>

12 <owl:Class rdf:nodeID="iriset_1">
13   <rdfs:subClassOf rdf:nodeID="descriptorset_1"/>
14 </owl:Class>

15 <owl:Class rdf:nodeID="iriset_2">
16   all resources specified by <iriset>.2.</iriset>
17 </owl:Class>

18 <owl:Class rdf:nodeID="descriptorset_2">
19   <owl:intersectionOf rdf:parseType="Collection">
20     <owl:Restriction>
21       <owl:onProperty rdf:resource="http://example.org/vocab#finish"/
22       <owl:hasValue rdf:resource="http://example.org/vocab#shiny"/>
23     </owl:Restriction>
24    </owl:intersectionOf>
25 </owl:Class>

26 <owl:Class rdf:nodeID="iriset_2">
27   <rdfs:subClassOf rdf:nodeID="descriptorset_2"/>
28 </owl:Class>
```

The `owl:intersectionOf` of a singleton collection in both descriptor sets, although redundant, is a result of the GRDDL transformation. As noted above, syntactically different but semantically equivalent representations are equally valid.

Note that `iriset_1` and `iriset_2` are not necessarily disjoint — some resources may be both shiny AND square.

A POWDER document may have an `ol` element which is an ordered list of DRs. Such a list receives first-match semantics, that is, when seeking the description of a candidate IRI, processors extract the descriptor set from the first DR in the ordered list in which it is in scope. `ol` elements allow the easy expression of exceptions to more general rules. So, for example:

**Example 3-7: An Ordered List of DRs [XML]**

```
<ol>

  <dr>
   <iriset>.1.</iriset>
   <descriptorset>
     <ex:shape>square</ex:shape>
   </descriptorset>
  </dr>

  <dr>
   <iriset>.2.</iriset>
   <descriptorset>
     <ex:shape>round</ex:shape>
   </descriptorset>
  </dr>

  <dr>
   <iriset>.3.</iriset>
   <descriptorset>
     <ex:shape>triangle</ex:shape>
   </descriptorset>
  </dr>

</ol>
```

receives the following semantics, where belonging to description_1 automatically precludes belonging to description_2 and description_3; and belonging to description_2 automatically precludes belonging to description_3:

### Example 3-8: The POWDER-S Encoding of Example 3-7 [RDF/XML]

```
<owl:Class rdf:nodeID="iriset_1">
  all resources specified by <iriset>.1.</iriset>
</owl:Class>

<owl:Class rdf:nodeID="descriptorset_1">
  <owl:intersectionOf rdf:parseType="Collection">
    <owl:Restriction>
      <owl:onProperty rdf:resource="http://example.org/vocab#shape"/>
       <owl:hasValue>square</owl:hasValue>
    </owl:Restriction>
  </owl:intersectionOf>
</owl:Class>

<owl:Class rdf:nodeID="iriset_1">
  <rdfs:subClassOf rdf:nodeID="descriptorset_1"/>
</owl:Class>

<owl:Class rdf:nodeID="iriset_2">
  all resources specified by <iriset>.2.</iriset>
</owl:Class>

<owl:Class rdf:nodeID="descriptorset_2">
  <owl:intersectionOf rdf:parseType="Collection">
    <owl:Restriction>
      <owl:onProperty rdf:resource="http://example.org/vocab#shape"/>
      <owl:hasValue>round</owl:hasValue>
    </owl:Restriction>
  </owl:intersectionOf>
</owl:Class>

<owl:Class>
  <owl:intersectionOf rdf:parseType="Collection">
    <owl:Class rdf:nodeID="iriset_2"/>
    <owl:Class>
      <owl:complementOf rdf:parseType="Collection">
        <owl:Class rdf:nodeID="iriset_1"/>
      </owl:complementOf>
    </owl:Class>
  </owl:intersectionOf>
  <rdfs:subClassOf rdf:nodeID="descriptorset_2"/>
</owl:Class>

<owl:Class rdf:nodeID="iriset_3">
  all resources specified by <iriset>.3.</iriset>
</owl:Class>

<owl:Class rdf:nodeID="descriptorset_3">
  <owl:intersectionOf rdf:parseType="Collection">
    <owl:Restriction>
      <owl:onProperty rdf:resource="http://example.org/vocab#shape"/>
      <owl:hasValue>triangular</owl:hasValue>
    </owl:Restriction>
  </owl:intersectionOf>
</owl:Class>

<owl:Class>
  <owl:intersectionOf rdf:parseType="Collection">
    <owl:Class rdf:nodeID="iriset_3"/>
    <owl:Class>
      <owl:complementOf rdf:parseType="Collection">
        <owl:Class>
          <owl:unionOf rdf:parseType="Collection">
            <owl:Class rdf:nodeID="iriset_2"/>
            <owl:Class rdf:nodeID="iriset_1"/>
          </owl:unionOf>
        </owl:Class>
```

## 3.2 Descriptor Set Semantics

### 3.2.1 Descriptor Sets expressed as RDF Properties and Values

In the simplest case, a descriptor set contains RDF properties that have literals or RDF Resources as their values as shown in Example 3-9 below. Note that that http://example.org/vocab#shiny *MUST NOT* identify an RDFS or OWL class.

**Example 3-9: A Descriptor Set Containing an RDF Property with a Literal Value**

POWDER [XML]

```
1  <descriptorset>
2    <ex:shape>square</ex:shape>
3    <ex:finish rdf:resource="http://example.org/vocab#shiny"/>
4  </descriptorset>
```

POWDER-S [RDF/XML]

```
1  <owl:Class rdf:nodeID="descriptorset_1">
2    <owl:intersectionOf rdf:parseType="Collection">
3      <owl:Restriction>
4        <owl:onProperty rdf:resource="http://example.org/vocab#shape"/>
5        <owl:hasValue>square</owl:hasValue>
6      </owl:Restriction>
7      <owl:Restriction>
8        <owl:onProperty rdf:resource="http://example.org/vocab#finish"/
9        <owl:hasValue rdf:resource="http://example.org/vocab#shiny"/>
10     </owl:Restriction>
11   </owl:intersectionOf>
12 </owl:Class>
```

These simple cases will normally be what is required for Description Resources.

**TBD:** The remainder of section 3.2.1 is a Feature at Risk. See the Status section.

More complex RDF descriptions are possible but are likely to cause problems and should not normally be used. The following example highlights the semantic problem of using blank nodes within a POWDER descriptor set.

**Example 3-10: A Descriptor Set Containing a Blank Node**

POWDER [XML]

```
<descriptorset>
  <ex:material>
    <ex:Wood>
      <ex:finish rdf:resource="http://example.org/vocab#shiny"/>
      <ex:madeof>cedar</ex:madeof>
    </ex:Wood>
  </ex:material>
</descriptorset>
```

POWDER-S [RDF/XML]

```
<owl:Class rdf:nodeID="descriptorset_1">
 <owl:intersectionOf rdf:parseType="Collection">
   <owl:Restriction>
      <owl:onProperty rdf:resource="http://example.org/vocab#materia
      <owl:hasValue>
        <ex:Wood>
          <ex:finish rdf:resource="http://example.org/vocab#shiny"/>
          <ex:madeof>cedar</ex:madeof>
        </ex:Wood>
      </owl:hasValue>
    </owl:Restriction>
  </owl:intersectionOf>
</owl:Class>
```

The semantics of the blank node (`ex:Wood`) are that there **is at least one identifiable resource** that is of the type ex:Wood that has the color brown, and that this is the value filler for the properties `ex:finish` and `ex:madeof`. Although this may be semantically valid, it is very unlikely that such a construct will be appropriate for use in POWDER. This is because it is possible for a descriptor set to be defined independently of any Description Resource and therefore not associated with any resources directly. Equally, a descriptor set may be part of a DR for which there are no resources that are within its scope at the time of its publication. It is the nature of POWDER that descriptions may well be published before or after the resources that any given DR describes. This lack of direct connection means that the use of blank nodes is strongly discouraged.

Identifying nodes within a descriptor set does not cause any of the semantic problems discussed for blank nodes; however, it does effectively create or duplicate vocabulary terms every time a DR is processed. Consider the following example, which is a variation on the previous one.

**Example 3-11: A Descriptor Set Containing an Identified Node**

POWDER [XML]

```
1  <descriptorset>
2    <ex:material>
3      <ex:Wood rdf:about="http://my.example.org/myVocab#PolishedCedar">
4        <ex:finish rdf:resource="http://example.org/vocab#shiny"/>
5        <ex:madeof>cedar</ex:madeof>
6      </ex:Wood>
7    </ex:material>
8  </descriptorset>
```

POWDER-S [RDF/XML]

```
1   <owl:Class rdf:about="http://my.example.org/myVocab#PolishedCedar">
2     <owl:intersectionOf rdf:parseType="Collection">
3       <owl:Restriction>
4         <owl:onProperty rdf:resource="http://example.org/vocab#finish"
5         <owl:hasValue rdf:resource="http://example.org/vocab#shiny"/>
6       </owl:Restriction>
7       <owl:Restriction>
8         <owl:onProperty rdf:resource="http://example.org/vocab#madeof/
9         <owl:hasValue>cedar</owl:hasValue>
10      </owl:Restriction>
11    </owl:intersectionOf>
12  </owl:Class>

13  <owl:Class rdf:nodeID="descriptorset_1">
14    <owl:intersectionOf rdf:parseType="Collection">
15      <owl:Restriction>
16        <owl:onProperty rdf:resource="http://example.org/vocab#materia
17        <owl:hasValue rdf:resource="http://my.example.org/myVocab#Poli
18      </owl:Restriction>
19    </owl:intersectionOf>
20  </owl:Class>
```

By adding rdf:about="http://my.example.org/myVocab#PolishedCedar" to line 3 of the descriptor set, a new class of `PolishedCedar` has been created in the http://my.example.org/myVocab# vocabulary. It is this that is described as having a shiny finish and made of cedar. Creating new vocabulary terms in this way should only be done where the DR author has no alternative since the triples concerning the new class will be created repeatedly as the document is processed. It is always better to create a separate new vocabulary and use that or, better still, to re-use an existing one. Nevertheless, POWDER does support the usage shown in Example 3-11.

As a further point, notice that the IRI used to identify the Polished Cedar class is an absolute one. A relative URI, or a value supplied to the RDF/XML attribute `rdf:ID` in line 3, would be relative to the candidate IRI (as defined in the Grouping of Resources document [GROUP]) — i.e. each and every IRI that is described by the DR of which the descriptor set is a part. This is almost certainly not what is intended.

To summarize the discussion of examples 3-9, 3-10 and 3-11: the semantics of POWDER descriptor sets work well with properties that take literals or RDF Resources as values. More complex RDF constructs are likely to lead to unintended or unintelligible results.

### 3.2.2 Asserting the `rdf:type` Relationship

Asserting the `rdf:type` property, i.e. that all elements within an IRI set are instances of a particular OWL or RDFS Class, is achieved most easily using the `typeof` element which takes the URI of the class as the value of its `src` attribute as shown in Example 3-12 below. The POWDER-S translation of the `descriptorset` element intersects `typeof` classes with the property restrictions (if any) in the `descriptorset`.

**Example 3-12: A Descriptor Set Asserting that IRIs within its Scope are Instance**

POWDER [XML]

```
<descriptorset>
  <typeof src="http://example.org/vocab#Conformance_Class" />
  <ex:shape>square</ex:shape>
</descriptorset>
```

POWDER-S [RDF/XML]

```
<owl:Class rdf:nodeID="descriptorset_1">
  <owl:intersectionOf rdf:parseType="Collection">
    <owl:Class rdf:about="http://example.org/vocab#Conformance_Class"
    <owl:Restriction>
      <owl:onProperty rdf:resource="http://example.org/vocab#shape"/>
      <owl:hasValue>square</owl:hasValue>
    </owl:Restriction>
  </owl:intersectionOf>
</owl:Class>
```

This is particularly useful in the context of the POWDER use cases [USECASES] when claiming that resources on a Web site conform to a published set of criteria. In such situations, multiple criteria can be grouped together by defining the class of resources that satisfy all the criteria as the intersection of a number of property restrictions; series of increasingly stricter conformance levels can be defined as a subsumption hierarchy of such classes.

If used directly, the `rdf:type` property will be treated in the same way as the `typeof` element in the POWDER to POWDER-S transform.

### 3.2.3 Referring to External Descriptor Sets

A descriptor set may defer to a second descriptor set in another POWDER document using the `src` attribute. However, this cannot express POWDER semantics since, at the time of processing, the remote document may be unknown, unavailable or not a valid POWDER document. Therefore the formal semantics are limited as shown below.

**Example 3-13: A Descriptor Set Referring to one in an External Document**

POWDER [XML]

```
<descriptorset src="http://remote.example.org/powder2.xml#d1" />
```

POWDER-S [RDF/XML]

```
<owl:Class rdf:nodeID="descriptorset_1">
  <rdfs:seeAlso rdf:resource="http://remote.example.org/powder2.xml#d1"
</owl:Class>
```

Informally, a processor MAY apply full semantics to a descriptor set referred to in this way if it is known to be part of a valid POWDER document, but only once it too has been transformed into POWDER-S.

### 3.2.4 Further Descriptors

There are two POWDER elements that can be included as child elements of `descriptorset` that are mapped to property restrictions in POWDER-S. In both cases the same string is used as the element name in POWDER and vocabulary term in POWDER-S:

**`sha1sum`**
: A SHA-1 sum of the described resource

**`certified`**
: An element of type `xsd:boolean` used when a DR certifies another resource.

The usage of both `sha1sum` and `certified` is shown in section 5.2 of the Description Resources document [DR].

We define further elements that can be included as child elements of `descriptorset` that, when transformed into POWDER-S, become annotation properties of the descriptive OWL class (not property restrictions).

**`displaytext`**
: is transformed to `dcterms:description`. The text supplied as the value of this element may be displayed in user agents.

**`displayicon`**
: has a `src` attribute, the value of which is a URI (or IRI) *u* which, in POWDER-S, becomes `foaf:depiction rdf:resource="`*u*`"`. The referred-to image may be displayed in user agents.

**`seealso`, `label`, `comment`**
: Each of these elements is transformed into an annotation of the OWL class using the similar term from the `rdfs` vocabulary with which it shares its name. For the avoidance of doubt:

```
<seealso src="http://www.example.com/page.html" />
<label>An example to us all</label>
<comment>Comments make code easier to read</comment>
```

are transformed into:

```
<rdfs:seeAlso rdf:resource="http://www.example.com/page.html" />
```

```
<rdfs:label>An example to us all</rdfs:label>
<rdfs:comment>Comments make code easier to read</rdfs:comment>
```

Usage of these elements is exemplified in the following section. As with `rdf:type`, they are provided as shortcuts within POWDER — the direct use of `rdfs:seeAlso`, `rdfs:comment` and `rdfs:label` will be rendered in exactly the same way (as annotations and not property restrictions) by the transform.

It is unlikely that other terms from the `rdfs` vocabulary can be used in a meaningful way in a POWDER context.

## 3.3 Tag Set Semantics

The semantics of the (free text) tags are similar to those for properties with literal values. Each tag given in a `tagset` element in a POWDER document is a value for the RDF datatype property `wdrs:tag` as shown below. Note also the use of the `seealso` (which puts the tags in context), `label` and `comment` elements described in the previous section.

**Example 3-14: A Tag Set.**

POWDER [XML]

```
<tagset>
  <label>Tags for the London landmark</label>
  <tag>London</tag>
  <tag>Swiss Re</tag>
  <tag>gherkin</tag>
  <seealso src="http://encyclopaedia.example.com/gherkin.html" />
  <seealso src="http://photo.example.com/gherkin.jpg" />
  <comment>Tags are linked to specific resources that contextualize t
</tagset>
```

POWDER-S [RDF/XML]

```
<owl:Class rdf:nodeID="tagset_1">
  <rdfs:label>Tags for the London landmark</rdfs:label>
  <owl:intersectionOf rdf:parseType="Collection">
    <owl:Restriction>
      <owl:onProperty rdf:resource="http://www.w3.org/2007/05/powder#
      <owl:hasValue>London</owl:hasValue>
    </owl:Restriction>
    <owl:Restriction>
      <owl:onProperty rdf:resource="http://www.w3.org/2007/05/powder#
      <owl:hasValue>Swiss Re</owl:hasValue>
    </owl:Restriction>
    <owl:Restriction>
      <owl:onProperty rdf:resource="http://www.w3.org/2007/05/powder#
      <owl:hasValue>gherkin</owl:hasValue>
    </owl:Restriction>
  </owl:intersectionOf>
  <rdfs:seeAlso rdf:resource="http://encyclopaedia.example.com/gherki
  <rdfs:seeAlso rdf:resource="http://photo.example.com/gherkin.jpg" /
  <rdfs:comment>Tags are linked to specific resources that contextual
</owl:Class>
```

# 4 IRI Set Semantics

The previous sections have shown that the semantics of several elements of a

POWDER document can be obtained by applying the GRDDL transform associated with the namespace to generate native RDF/OWL as POWDER-S. This is not so for the IRI set element which, although transformed into valid RDF/OWL syntax, does not express the full semantics.

The IRI constraints defined in the POWDER Grouping of Resources document [GROUP] are given regular-expression semantics by the first part of the GRDDL transform from POWDER to POWDER-BASE. Regular-expression IRI groups are, in their turn, given semantics using datarange restrictions by the POWDER-BASE to POWDER-S transformation. It is noteworthy that the value space of POWDER's IRI constraints is, for the most part, a white space separated list of alternative values. This makes POWDER in its XML form relatively simple, but the implications for the semantics are substantial.

## 4.1 White Space and List Pre-Processing

Many elements of a POWDER IRI set definition have white space separated lists of strings as their value. White space is any of U+0009, U+000A, U+000D and U+0020. A space-separated list is a string in which the items are separated by one or more space characters (in any order). The string may also be prefixed or suffixed with zero or more of those characters. The GRDDL transform associated with the POWDER namespace converts these into components of a regular expression for use in POWDER-BASE and POWDER-S by following the steps set out below:

- Replace any sequence of space characters with a single space (U+0020) character, dropping any leading or trailing U+0020 characters
- Replace each remaining space (U+0020) character with the vertical bar character | (U+0124)
- Escape all instances of the following characters within the string using a \ character

  ```
  . \ ? * + { } ( ) [ ] ! " # % & ' , - / : ; = > @ [ ] _ ` ~
  ```

- Enclose the resulting string in parentheses

The resulting string is used in a template regular expression to give the element and list's desired semantics. For example

```
<includehosts>example.com example.org </includehosts>
```

becomes

```
<owl:Restriction>
  <owl:onProperty rdf:resource="http://www.w3.org/2007/05/powder-s#matche
  <owl:hasValue  rdf:datatype="http://www.w3.org/2001/XMLSchema-datatypes
</owl:Restriction>
```

## 4.2 POWDER and POWDER-BASE IRI Set Semantics

POWDER's use cases involve information resources available on the Web, identified by IRIs containing host names, directory paths, IP addresses, port numbers, and so on. To make it as easy as possible to create IRI sets we define a series of IRI constraints in the Grouping of Resources document [GROUP]. These all receive semantics through being mapped to `includeregex` and `excluderegex` elements in

POWDER-BASE.

Re-visiting the example given in the previous section, the POWDER element

```
<iriset>
  <includehosts>example.com example.org</includehosts>
</iriset>
```

is expressed in POWDER-BASE as:

```
<iriset>
  <includeregex>\:\/\/(([^\/\?\#]*)\@)?([^\:\/\?\#\@]+\.)?(example\.com|e
</iriset>
```

IRIs are always interpreted as strings, even if they include numerical parts such as ports and IP numbers as shown in the following example:

**Example 4-1: The POWDER and POWDER-BASE Encoding of an Example IRI Se**

POWDER: [XML]

```
<iriset>
  <includehosts>example.com example.org</includehosts>
  <includeports>80 8080 8081 8082</includeports>
</iriset>
```

POWDER-BASE: [XML]

```
<iriset>
  <includeregex>\:\/\/(([^\/\?\#]*)\@)?([^\:\/\?\#\@]+\.)?(example\.com|
  <includeregex>\:\/\/(([^\/\?\#]*)\@)?([^\:\/\?\#\@]+\.)*[^\:\/\?\#\@]+
</iriset>
```

This approach is applied to several of the POWDER IRI set elements. The following table shows these and their associated template regular expressions. In each case, *var* means the value of the POWDER element after processing as defined in Section 4.1.

| **POWDER IRI Constraint (`include/exclude...`** | **POWDER-BASE Regular Expression (used in `includeregex/excluderegex`)** |
|---|---|
| schemes | ^***var***\:\/\/ |
| hosts | \:\/\/(([^\/\?\#]*)\@)?([^\:\/\?\#\@]+\.)?***var***(\:([0-9]+))?\/ |
| ports | \:\/\/(([^\/\?\#]*)\@)?([^\:\/\?\#\@]+\.)*[^\:\/\?\#\@]+\:***var***\/ |
| exactpaths | \:\/\/(([^\/\?\#]*)\@)?([^\:\/\?\#\@]*)(\:([0-9]+))?***var***($\|\?\|\#) |
| pathcontains | \:\/\/(([^\/\?\#]*)\@)?([^\:\/\?\#\@]*)(\:([0-9]+))?\/[^\?\#]****var***[^\?\#]*[\?\#]? |
| pathstartswith | \:\/\/(([^\/\?\#]*)\@)?([^\:\/\?\#\@]*)(\:([0-9]+))?***var*** |
| pathendswith | \:\/\/(([^\/\?\#]*)\@)?([^\:\/\?\#\@]*)(\:([0-9]+))?\/[^\?\#]****var***($\|\?\|\#) |
| resources | ^***var***$ |

Table 3. Template regular expressions for IRI constraints that take a white space separated list of values.

| POWDER IRI Constraint (`include/exclude...` | POWDER-BASE Regular Expression (used in `includeregex/excluderegex`) |
|---|---|

Note that the Grouping of Resources document [GROUP] sets out a canonicalization process that must be followed. This has particular implications for the matching of ports: where the port number is constrained, default port numbers for the relevant scheme must be taken into account.

Two further pairs of IRI set constraints defined in the Grouping of Resources document undergo additional processing when transformed from POWDER to POWDER-BASE: `includequerycontains` and `includeiripattern` (and their 'exclude' counterparts). Each of these maps to multiple elements in the POWDER-BASE document.

`includequerycontains` and `excludequerycontains` take a single value, not a white space separated list of values. Furthermore, an attribute `delimiter` takes a single character that delimits the name/value pairs in the query string. If no such attribute is set, the ampersand (`&`) character is used as the default. To transform these elements from POWDER to POWDER-BASE regular expressions the following steps are carried out:

- Split the supplied value at the delimiter, ***d***, dropping that character in the process
- For each resulting sub string, ***q***, create an `includeregex` or `excluderegex` as appropriate using the following regular expression template:

  `\:\/\/(([^\/\?\#]*)\@)?([^\:\/\?\#\@]*)(\:([0-9]+))?\/[^\?\#]*\?([^\#]*`***d***`)?`***q***`(`***d***`|$)`

This transformation is exemplified below.

**Example 4-2: The POWDER and POWDER-BASE Encoding of an Example IRI Se**

POWDER: [XML]

```
<iriset>
  <includehosts>example.org</includehosts>
  <includequerycontains>id=123456&group=abcdefg</includequerycontains>
</iriset>
```

POWDER-BASE: [XML]

```
<iriset>
  <includeregex>\:\/\/(([^\/\?\#]*)\@)?([^\:\/\?\#\@]+\.)?(example\.org
  <includeregex>\:\/\/(([^\/\?\#]*)\@)?([^\:\/\?\#\@]*)(\:([0-9]+))?\/[
  <includeregex>\:\/\/(([^\/\?\#]*)\@)?([^\:\/\?\#\@]*)(\:([0-9]+))?\/[
</iriset>
```

`includeiripattern` and `excludeiripattern` also take a single value, not a white space separated list of values, and generate `includeregex` and `excluderegex` elements in POWDER-S as follows:

- Match the value given against this regular expression:

  `(([^\:\/\?\#\.]+)\:)?(\/\/)?([^\:\/\?\#\@]*)(\:([0-9]+))?`

  from which $2 is a constraint on the scheme, $4 is a constraint on the host and $6 is a constraint on the port (the host is always constrained, $2 and $6 may be empty).

- Let the value of $2 be *s*, $4 be *h* and $6 be *p*
- If *s* is empty then let *s* be `[A-Za-z]+`.
- If *p* is empty then let *p* be `(\:[0-9]+)?`, else let *p* be `\:`***p***
- If *h* is exactly ***** then let *h* be ([^\:\/\?\#\@]+\.)*[^\:\/\?\#\@]+
- Else if *h* matches the regular expression `^\*\.(.*)` then let *h* be `([^\:\/\?\#\@]+\.)*`**$1** where **$1** refers to `^\*\.(.*)`
- Else let *h* be `([^\:\/\?\#\@]+\.)*`***h***
- Create the following element in the POWDER-BASE document: `<includeregex>^`***s***`\:\/\/`***hp***`</includeregex>`

The following example shows these steps.

**Example 4-3: The POWDER and POWDER-BASE Encoding of an Example IRI Se**

POWDER: [XML]

```
<iriset>
  <includeiripattern>http://*.example.org:8080</includeiripattern>
</iriset>
```

POWDER-BASE: [XML]

```
<iriset>
  <includeregex>^http\:\/\/([^\:\/\?\#\@]+\.)?example.org:8080</include
</iriset>
```

Incidentally, the IRI set defined here is 'all resources on all subdomains of example.org (but not on example.org) accessed via HTTP through port 8080.'

## 4.3 POWDER-S IRI Set Semantics

Providing OWL/RDF semantics for `iriset` elements is not directly possible, since RDF does not provide any means for accessing or manipulating the string representation of an IRI. We extend RDF with a datatype property `wdrs:matchesregex` as shown below.

```
wdrs:matchesregex rdf:type owl:DatatypeProperty .
wdrs:matchesregex rdf:type owl:Property .
wdrs:matchesregex rdfs:domain rdfs:Resource .
```

```
wdrs:matchesregex rdfs:range xsd:string .
```

We further stipulate that for triples:

```
x wdrs:matchesregex reg .
```

<*x*, *reg*> is in IEXT(I(wdrs:matchesregex)) if and only if:

- *reg* conforms with regular expression syntax, AND
- there is some *uuu* in the value space of xsd:anyURI such that:
    - *uuu* matches the regular expression *reg*, AND
    - *uuu* is in the domain of I, with I(*uuu*)=*x*

It is now possible to express `includeregex` and `excluderegex` as a `owl:hasValue` restriction [OWL] on this dataproperty and build up an OWL Class to represent the IRI set in the POWDER-S encoding. Furthermore, the sub class relationship between the IRI set and the descriptor set is asserted.

The following example takes a complete example POWDER document through POWDER-BASE to POWDER-S. Note that the only change from POWDER to POWDER-BASE is in the elements within the IRI set.

### Example 4-4: The Full Transformation of an Example from POWDER Through P

POWDER [XML]

```
1  <?xml version="1.0"?>
2  <powder xmlns="http://www.w3.org/2007/05/powder#"
3          xmlns:ex="http://example.org/vocab#">

4    <attribution>
5      <issuedby src="http://authority.example.org/company.rdf#me" />
6      <issued>2007-12-14T00:00:00</issued>
7    </attribution>

8    <dr>
9      <iriset>
10       <includehosts>example.com example.org</includehosts>
11       <includeports>80 8080 8081 8082</includeports>
12     </iriset>
13     <descriptorset>
14      <ex:color>red</ex:color>
15      <ex:shape>square</ex:shape>
16      <displaytext>Everything on example.org and example.com is red an
17      <displayicon src="http://example.org/icon.png" />
18    </descriptorset>
19   </dr>

20 </powder>
```

POWDER-BASE [XML]

```
1  <?xml version="1.0"?>
2  <powder xmlns="http://www.w3.org/2007/05/powder#"
3          xmlns:ex="http://example.org/vocab#">

4    <attribution>
5      <issuedby src="http://authority.example.org/company.rdf#me" />
6      <issued>2007-12-14T00:00:00</issued>
7    </attribution>

8    <dr>
9      <iriset>
10       <includeregex>\:\/\/([^\/\?\#]*)\@)?([^\:\/\?\#\@]+\.)?(exampl
11       <includeregex>\:\/\/([^\/\?\#]*)\@)?([^\:\/\?\#\@]+\.)*[^\:\/\
12     </iriset>
13     <descriptorset>
14      <ex:color>red</ex:color>
15      <ex:shape>square</ex:shape>
16      <displaytext>Everything on example.org and example.com is red an
17      <displayicon src="http://example.org/icon.png" />
18    </descriptorset>
19   </dr>

20 </powder>
```

POWDER-S [RDF/XML]

```
1  <?xml version="1.0"?>
2  <rdf:RDF
3     xmlns:wdrs="http://www.w3.org/2007/05/powder-s#"
4     xmlns:foaf="http://xmlns.com/foaf/0.1/"
5     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
6     xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
7     xmlns:owl="http://www.w3.org/2002/07/owl#"
8     xmlns:dcterms="http://purl.org/dc/terms/0.1/"
9     xmlns:ex="http://example.org/vocab#">

10    <rdf:Description rdf:about="">
```

The formal encoding of an `excluderegex` element uses the `owl:complementOf` property to link to the IRI set specified by the regular expression as exemplified below (note line 8).

**Example 4-5: The POWDER, POWDER-BASE and POWDER-S Encoding of `exclu`**

POWDER [XML]

```
1  <iriset>
2    <incudehosts>example.org</includehosts>
3    <excludeports>8080</excludeports>
4  </iriset>
```

POWDER-BASE [XML]

```
1  <iriset>
2    <includeregex>\:\/\/(([^\/\?\#]*)\@)?([^\:\/\?\#\@]+\.)?(example\.o
3    <excluderegex>\:\/\/(([^\/\?\#]*)\@)?([^\:\/\?\#\@]+\.)*[^\:\/\?\#\
4  </iriset>
```

POWDER-S [RDF/XML]

```
1  <owl:Class rdf:nodeID="iriset_1">
2    <owl:intersectionOf rdf:parseType="Collection">

3      <owl:Restriction>
4        <owl:onProperty rdf:resource="http://www.w3.org/2007/05/powder-
5        <owl:hasValue  rdf:datatype="http://www.w3.org/2001/XMLSchema-d
6      </owl:Restriction>

7      <owl:Class>
8        <owl:complementOf>
9          <owl:Restriction>
10           <owl:onProperty rdf:resource="http://www.w3.org/2007/05/pow
11           <owl:hasValue rdf:datatype="http://www.w3.org/2001/XMLSchem
12         </owl:Restriction>
13       </owl:complementOf>
14     </owl:Class>

15   </owl:intersectionOf>
16 </owl:Class>
```

## 4.4 Direct Descriptions

POWDER and, consequently, POWDER-BASE documents might include `descriptorset` elements that are not inside a `dr` element but directly subsumed by the document's root. Such descriptions are not meant to be implicitly applied to any IRI groups, but are only made available by explicit reference by resources, as explained in Section 2.5 of the Description Resources document [DR].

In POWDER-S, such descriptions are translated into classes, but no subsumption of an IRI group is asserted, as shown in Example 4-6. Note, however, that as shown in Section 3, each derived OWL class is given an `rdf:ID` equivalent to the `xml:id` in the original POWDER document, not an `rdf:nodeID`, so that it can be referred to from outside.

**Example 4-6: The Semantics of Direct Description Elements**

POWDER [XML]

```
<descriptorset xml:id="square">
  <ex:finish rdf:resource="http://example.org/vocab#shiny"/>
  <ex:shape>square</ex:shape>
</descriptorset>

<descriptorset xml:id="round">
  <ex:finish rdf:resource="http://example.org/vocab#matt"/>
  <ex:shape>round</ex:shape>
</descriptorset>

<descriptorset xml:id="hexagonal">
  <ex:finish rdf:resource="http://example.org/vocab#eggshell"/>
  <ex:shape>hexagonal</ex:shape>
</descriptorset>
```

POWDER-S [RDF/XML]

```
<owl:Class rdf:ID="square">
  <owl:intersectionOf rdf:parseType="Collection">
    <owl:Restriction>
      <owl:onProperty rdf:resource="http://example.org/vocab#finish"/
      <owl:hasValue rdf:resource="http://example.org/vocab#shiny"/>
    </owl:Restriction>
    <owl:Restriction>
      <owl:onProperty rdf:resource="http://example.org/vocab#shape"/>
      <owl:hasValue>square</owl:hasValue>
    </owl:Restriction>
  </owl:intersectionOf>
</owl:Class>

<owl:Class rdf:ID="round">
  <owl:intersectionOf rdf:parseType="Collection">
    <owl:Restriction>
      <owl:onProperty rdf:resource="http://example.org/vocab#finish"/
      <owl:hasValue rdf:resource="http://example.org/vocab#matt"/>
    </owl:Restriction>
    <owl:Restriction>
      <owl:onProperty rdf:resource="http://example.org/vocab#shape"/>
      <owl:hasValue>round</owl:hasValue>
    </owl:Restriction>
  </owl:intersectionOf>
</owl:Class>

<owl:Class rdf:ID="hexagonal">
  <owl:intersectionOf rdf:parseType="Collection">
    <owl:Restriction>
      <owl:onProperty rdf:resource="http://example.org/vocab#finish"/
      <owl:hasValue rdf:resource="http://example.org/vocab#eggshell"/
    </owl:Restriction>
    <owl:Restriction>
      <owl:onProperty rdf:resource="http://example.org/vocab#shape"/>
      <owl:hasValue>hexagonal</owl:hasValue>
    </owl:Restriction>
  </owl:intersectionOf>
</owl:Class>
```

## 4.5 Semantics of `abouthosts` and `aboutregex`

The value of `abouthosts` is a whitespace-separated list of hosts. This list receives

identical semantics to the value of the `includehosts` grouping element. In consequence, the transformation from POWDER to POWDER-BASE transforms the `abouthosts` elements into an `aboutregex` element using the same processing steps as for transforming `includehosts` into `includeregex`, as described in Section 4.2 above.

The `aboutregex` element of POWDER-BASE documents sets an outer limit on the resources described by the DRs within the document. That is to say, it restricts the resources that may receive a description not only implicitly (via subsumption by an IRI set) but also explicitly as described in Section 2.5 of the Description Resources document [DR].

In order to capture these semantics, the POWDER-BASE to POWDER-S transformation must add an implicit restriction to all descriptor sets in the document, effectively subsuming all resource classes created by `descriptorset` elements under the resource class that is created by the `aboutregex` element. In this manner, the implicit or explicit assignment of a description to a resource (cf. Section 2.5 of the Description Resources document [DR]) will create an inconsistency if the resource lies outside the `aboutregex` class.

It should be noted that `iriset` classes are *not* implicitly intersected with the `aboutregex` class, and it is the responsibility of the POWDER document author to ensure that all IRI sets in the document are within the scope defined by `abouthosts/aboutregex`.

This process is demonstrated by Example 4.7. Notice that in the POWDER-S document, each descriptor class is intersected with the 'aboutset' (lines 31, 51 and 63). A logical inconsistency will arise (i.e. an error) if an IRI set is not a subset of the aboutset class.

### Example 4-7: The Semantics of `abouthosts`

POWDER Document [XML]

```
<attribution>
  <issuedby src="http://authority.example.org/company.rdf#me" />
  <abouthosts>example.org example.com</abouthosts>
</attribution>

<dr>
  <iriset>
    <includehosts>square.example.org</includehosts>
  </iriset>
  <descriptorset>
    <ex:shape>square</ex:shape>
  </descriptorset>
</dr>

<dr>
  <iriset>
    <includehosts>round.example.com</includehosts>
  </iriset>
  <descriptorset>
    <ex:shape>round</ex:shape>
  </descriptorset>
</dr>

<descriptorset xml:id="silver">
  <ex:finish rdf:resource="http://example.org/vocab#shiny"/>
  <ex:shape>square</ex:shape>
</descriptorset>
```

POWDER-BASE Document [XML]

```
<attribution>
  <maker ref="http://authority.example.org/company.rdf#me" />
  <aboutregex>\:\/\/(([^\/\?\#]*)\@)?([^\:\/\?\#\@]+\.)?(example\.org
</attribution>

<dr>
  <iriset>
    <includeregex>\:\/\/(([^\/\?\#]*)\@)?([^\:\/\?\#\@]+\.)?(square\.
  </iriset>
  <descriptorset>
    <ex:shape>square</ex:shape>
  </descriptorset>
</dr>

<dr>
  <iriset>
    <includeregex>\:\/\/(([^\/\?\#]*)\@)?([^\:\/\?\#\@]+\.)?(round\.e
  </iriset>

  <descriptorset>
    <ex:shape>round</ex:shape>
  </descriptorset>
</dr>

<descriptorset xml:id="silver">
  <ex:finish rdf:resource="http://example.org/vocab#shiny"/>
  <ex:shape>square</ex:shape>
</descriptorset>
```

POWDER-S Document [RDF/XML]

```
<?xml version="1.0"?>
<rdf:RDF
```

As discussed in Section 3.2.3, a DR MAY refer to `descriptorset` elements in other POWDER documents. In such a situation, although it is not possible using XSLT (the technology used to effect the POWDER transforms) to generate `aboutregex` elements for POWDER-BASE as shown in the previous example, a conformant POWDER Processor MUST take account of `abouthosts` elements in both documents. The `abouthosts` element is designed to place an outer limit on the scope of any description in a POWDER document so that publishers descriptions retain effective control over the assertions made using their descriptions, even when such assertions are not attributed to them. For example, publishers of descriptions can use `abouthosts` to state that their descriptions are only meaningful for a particular set of domains, and may not be used to describe anything else.

## 4.6 POWDER-BASE IRI Set Semantics in OWL 2 (Informative)

At the time of this writing, the OWL-2 [OWL2] working draft provides for user-defined datatypes, using the restriction facet mechanism in XSD 2 [XSD2]. As this includes regular expression patterns, it is possible to translate POWDER into OWL 2 requiring a simpler extension than the one in Section 4.3.

More specifically, we extend RDF semantics [RDF-SEMANTICS] with a datatype property `hasIRI`, defined as:

```
hasIRI rdf:type owl:DatatypeProperty .
hasIRI rdf:type owl:Property .
hasIRI rdfs:domain owl:Thing
hasIRI rdfs:range xsd:anyURI .
```

and the further stipulation that:

<*x*, *uuu*> is in IEXT(I(wdrs:hasIRI)) if and only if *uuu* is in the domain of I, with I(*uuu*)=*x*

Such an extension makes it possible to provide semantics to `iriset` by constructing an RDF datatype for each `iriset` and restricting the values of `hasIRI` to this datatype's range. In this manner, the POWDER-S translation of Example 4-4 becomes as shown in Example 4-8, where `iriset_1` is a class of abstract resources, the concrete IRI string of which is within a user-defined datarange (lines 20-25 and 31-36).

**Example 4-8: The POWDER-S Encoding of Example 4-4 With User-defined Datat**

```
<?xml version="1.0"?>
<rdf:RDF
  xmlns:wdrs="http://www.w3.org/2007/05/powder-s#"
  xmlns:dcterms="http://purl.org/dc/terms/"
  xmlns:foaf="http://xmlns.com/foaf/0.1/"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns:owl="http://www.w3.org/2002/07/owl#"
  xmlns:owl2="http://www.w3.org/2006/12/owl2#"
  xmlns:ex="http://example.org/vocab#">

  <rdf:Description rdf:about="">
    <wdrs:issuedby rdf:resource="http://authority.example.org/company
    <dcterms:issued>2007-12-14</dcterms:issued>
  </rdf:Description>

  <owl:Class rdf:nodeID="iriset_1">
    <owl:intersectionOf rdf:parseType="Collection">
      <owl:Restriction>
        <owl:onProperty rdf:resource="http://www.w3.org/2007/05/powde
        <owl:hasValue>
          <owl:DataRange>
            <owl2:pattern rdf:datatype="http://www.w3.org/2001/XMLSc
              \:\/\/(([^\/\?\#]*)\@)?([^\:\/\?\#\@]+\.)?(example\.co
            </owl2:pattern>
            <owl2:onDataRange rdf:resource="http://www.w3.org/2001/X
          </owl:DataRange>
        </owl:hasValue>
      </owl:Restriction>
      <owl:Restriction>
        <owl:onProperty rdf:resource="http://www.w3.org/2007/05/powd
        <owl:hasValue>
          <owl:DataRange>
            <owl2:pattern rdf:datatype="http://www.w3.org/2001/XMLSc
              \:\/\/(([^\/\?\#]*)\@)?([^\:\/\?\#\@]+\.)*[^\:\/\?\#\@
            </owl2:pattern>
            <owl2:onDataRange rdf:resource="http://www.w3.org/2001/X
          </owl:DataRange>
        </owl:hasValue>
      </owl:Restriction>
    </owl:intersectionOf>
  </owl:Class>

  <owl:Class rdf:nodeID="descriptorset_1">
    <owl:intersectionOf rdf:parseType="Collection">
      <owl:Restriction>
        <owl:onProperty rdf:resource="http://example.org/vocab#color
        <owl:hasValue>red</owl:hasValue>
      </owl:Restriction>
      <owl:Restriction>
        <owl:onProperty rdf:resource="http://example.org/vocab#shape
        <owl:hasValue>square</owl:hasValue>
      </owl:Restriction>
    </owl:intersectionOf>
    <dcterms:description>Everything on example.org and example.com i
    <foaf:depiction rdf:resource="http://example.org/icon.png" />
  </owl:Class>

  <owl:Class rdf:nodeID="iriset_1">
    <rdfs:subClassOf rdf:nodeID="descriptorset_1"/>
  </owl:Class>

</rdf:RDF>
```

# 5 Acknowledgements

The Working Group would like to thank Jeremy Carroll for his substantial contribution to the development of the semantics of POWDER.

# 6 References

**[WEBARCH]**
*Architecture of the World Wide Web, Volume One*, Section 2.2. W3C Recommendation 15 December 2004. I. Jacobs, N. Walsh. This document is at http://www.w3.org/TR/webarch/#id-resources

**[GRDDL]**
*Gleaning Resource Descriptions from Dialects of Languages (GRDDL)*, W3C Recommendation 11 September 2007. D. Connolly. This document is at http://www.w3.org/TR/grddl/

**[XSLT]**
*XSL Transformations (XSLT)*, W3C Recommendation 16 November 1999. J. Clark. This document is at http://www.w3.org/TR/xslt

**[DR]**
*Protocol for Web Description Resources (POWDER): Description Resources*, P. Archer, A. Perego, K. Smith. This document is at http://www.w3.org /TR/powder-dr/

**[GROUP]**
*Protocol for Web Description Resources (POWDER): Grouping of Resources*, A. Perego, P. Archer This document is at http://www.w3.org/TR/powder-grouping/

**[USECASES]**
*POWDER: Use Cases and Requirements* W3C Working Group Note 31 October 2007, P. Archer . This document is at http://www.w3.org/TR/powder-use-cases/

**[PRIMER]**
*Protocol for Web Description Resources (POWDER): Primer* 2008, K. Scheppe, D. Pentecost. (URI TBC)

**[TESTS]**
*Protocol for Web Description Resources (POWDER): Test Suite* 2008, A. Kukurikos. (URI TBC)

**[WDR]**
*Protocol for Web Description Resources (POWDER): Web Description Resources XML Schema (WDR)*, A. Perego, K. Smith. This document is at http://www.w3.org/2007/05/powder

**[WDRS]**
*Protocol for Web Description Resources (POWDER): POWDER-S Vocabulary (WDRS)*, P. Archer. This document is at http://www.w3.org/2007/05/powder-s

**[WDRD]**
*Protocol for Web Description Resources (POWDER): Web Description Resources Datatypes (WDRD)*, A. Perego, K. Smith. This document is at http://www.w3.org/TR/powder-xsd/

**[PDR-GRDDL]**
*Protocol for Web Description Resources (POWDER): GRDDL Transform* 2008, K. Smith, A. Perego (URI TBC)

**[XSD2]**
*XML Schema Part 2: Datatypes Second Edition* W3C Recommendation 28 October 2004. P V Biron, A. Malhotra. This document is at http://www.w3.org

/TR/2004/REC-xmlschema-2-20041028/datatypes.html#rf-pattern

**[N3]**
*Notation3 (N3): A readable RDF syntax*, T. Berners-Lee, D. Connolly. This document is at http://www.w3.org/TeamSubmission/n3/

**[DC]**
*Dublin Core Metadata Initiative Terms*. See http://dublincore.org/documents/dcmi-terms/

**[FOAF]**
*FOAF Vocabulary Specification 0.9* Namespace Document 24 May 2007, D. Brickley, L. Miller. This document is at http://xmlns.com/foaf/spec/

**[W3C DTF]**
*W3C Date and Time Formats* W3C Note, 15 September 1997. M. Wolf, C. Wicksteed. This document is at http://www.w3.org/TR/NOTE-datetime

**[OWL]**
*OWL Web Ontology Language, Semantics and Abstract Syntax*, W3C Recommendation 10 February 2004. P. F. Patel-Schneider, I. Horrocks. This document is at http://www.w3.org/TR/owl-semantics/syntax.html#2.

**[OWL2]**
*OWL 2 Web Ontology Language: Primer*, W3C Working Draft 11 April 2008. B. Parsia, P. F. Patel-Schneider. This document is at http://www.w3.org/TR/2008/WD-owl2-primer-20080411/

**[RDF-SEMANTICS]**
*RDF Semantics*, W3C Recommendation 10 February 2004, P. Hayes. The key text is in Section 1.3. This document is at http://www.w3.org/TR/2004/REC-rdf-mt-20040210/

**[XQXP]**
*XQuery 1.0 and XPath 2.0 Functions and Operators*, A. Malhotra, J. Melton, N. Walsh. W3C Recommendation 23 January 2007. This document is at http://www.w3.org/TR/xpath-functions/

**[XSLT2]**
*XSL Transformations (XSLT) Version 2.0*, W3C Recommendation 23 January 2007. M. Kay. This document is at http://www.w3.org/TR/xslt20/

**[Rabin]**
*URI Pattern Matching for Groups of Resources.*, Draft 0.1 17 June 2006, J. Rabin. This document is at http://www.w3.org/2005/Incubator/wcl/matching.html

**[RFC2119]**
*Key words for use in RFCs to Indicate Requirement Levels*, RFC 2119, S. Bradner. This document is at http://tools.ietf.org/html/rfc2119

# 7 Change Log

## 7.1 Changes since First Public Working Draft

- Status section updated
- Mention of separate POWDER-BASE namespace removed (a legacy from pre-publication draft)
- `<maker>` and `foaf:maker` replaced by `<issuedby>` and `wdrs:issuedby` in all examples. This is defined in the WDRS vocabulary as a sub property of both `foaf:maker` and `dcterms:creator` so that Agent classes from both vocabularies may be used. Support for both now included. See e-mail thread.

- Sentence added to clarify that the `dcterms/foaf:Agent` class can be included directly
- Where previous examples have had <ex:color rdf:resource="...#red" /> this has been changed to <ex:finish rdf:resource="...#shiny" /> to avoid confusion following comments by Ivan Herman
- Correction of error in text following example 3-1
- Use of `rdf:nodeID` corrected throughout following comments by Masahide Kanzaki and Ivan Herman.
- Unnecessary detail elided from Example 3-8
- Line numbers added to all examples in the text following comment from Ivan Herman (see the P.S.)
- Section 3.2 substantially re-written and tidied up, introducing `typeof, seealso, label and comment` elements. Blank nodes no longer forbidden but their usage is strongly discouraged. Text flows from e-mail discussion.
- Wording of section on tags tidied up, `ref` attribute removed in favor of using `seealso`.
- Slight change in the presentation of the semantic extension in Section 4.3. This now has the fragment identifier of #SE. The semantic extension in the informative example in Section 4.6 now has the fragment identifier of #SE2
- Error in preamble to example 4-4 fixed. This referred to an extra namespace for POWDER-BASE which is no longer used.
- Section 4.4 and Example 4-6 amended slightly to state that descriptor sets not inside a DR are transformed into OWl classes with `rdf:ID` identifiers cf. `rdf:nodeIDS`.
- In section 4.5, the logical inconsistency of an IRI set being outwith an abouthosts limit is made clear. Example 4-7 reworked a little to make consistent with other examples.
- Slight revision to text concerning semantics of descriptor sets referred to in external documents as it relates to `abouthosts`. Reference is now to section 3.2.3 of this document cf. the DR document.
- Use of `rdf:Description` to describe the POWDER-S document changed to `owl:Ontology` after e-mail discussion on the member list.
- The use of `owl:complementOf` in Example 3-8 and related examples corrected following comment from Ivan Herman
- Collection removed from line 8 of POWDER-S example in Example 4-5 following comment from Ivan Herman
- Template regular expressions (Table 3 etc.) updated to support candidate resource IRIs that include user info following e-mail comment.
- Support for arbitrary RDF in the `attribution` and `descriptorset` elements flagged as a Feature at Risk - see Status section.