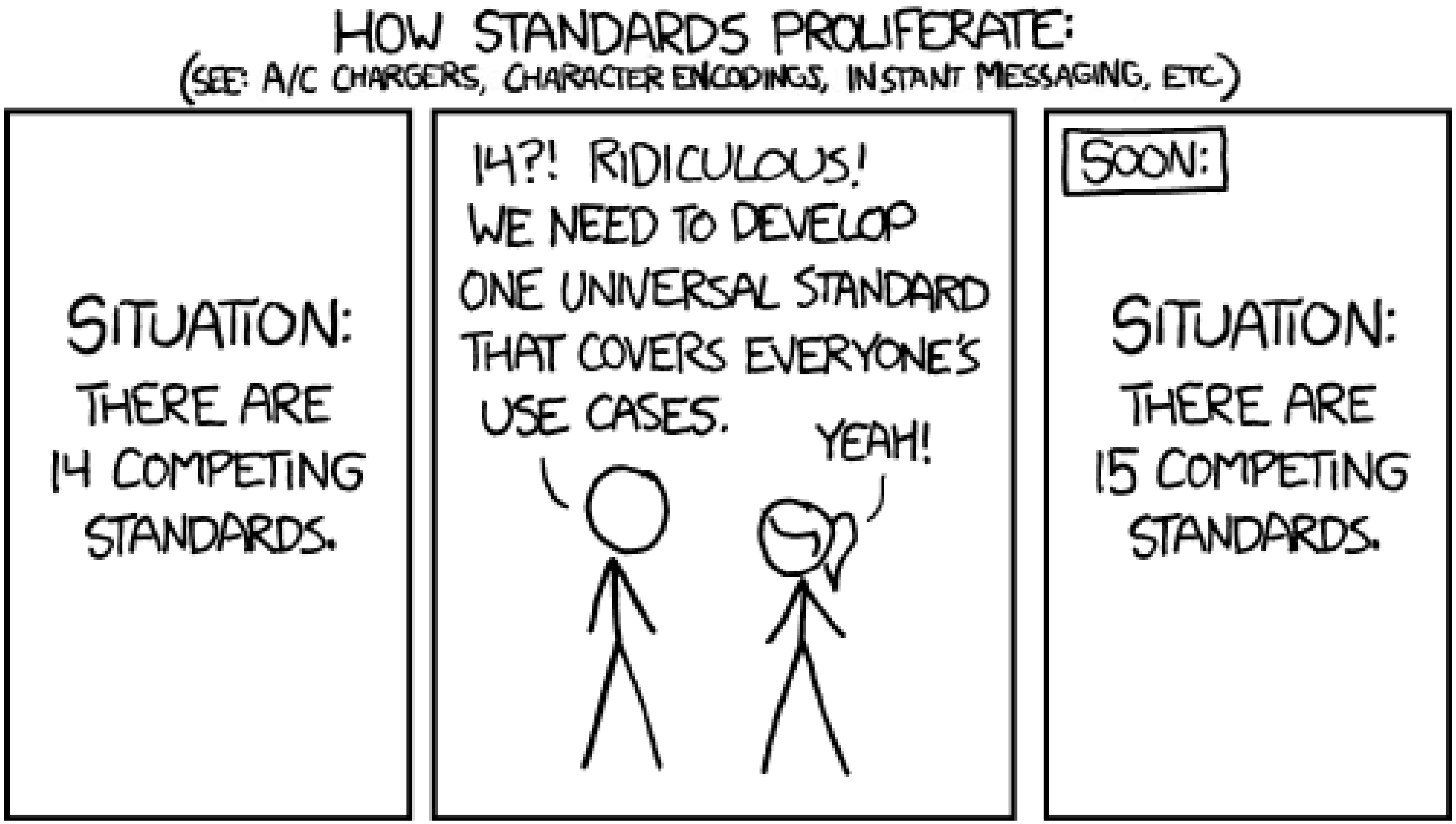
JSON Standards

Jamie Lennox

jamielennox@gmail.com

*@jamielennox_*


HOW STANDARDS PROLIFERATE:
(SEE: A/C CHARGERS, CHARACTER ENCODINGS, INSTANT MESSAGING, ETC.)
SITUATION: THERE ARE 14 COMPETING STANDARDS.
14?! RIDICULOUS! WE NEED TO DEVELOP ONE UNIVERSAL STANDARD THAT COVERS EVERYONE'S USE CASES.
YEAH!
SOON:
SITUATION: THERE ARE 15 COMPETING STANDARDS.

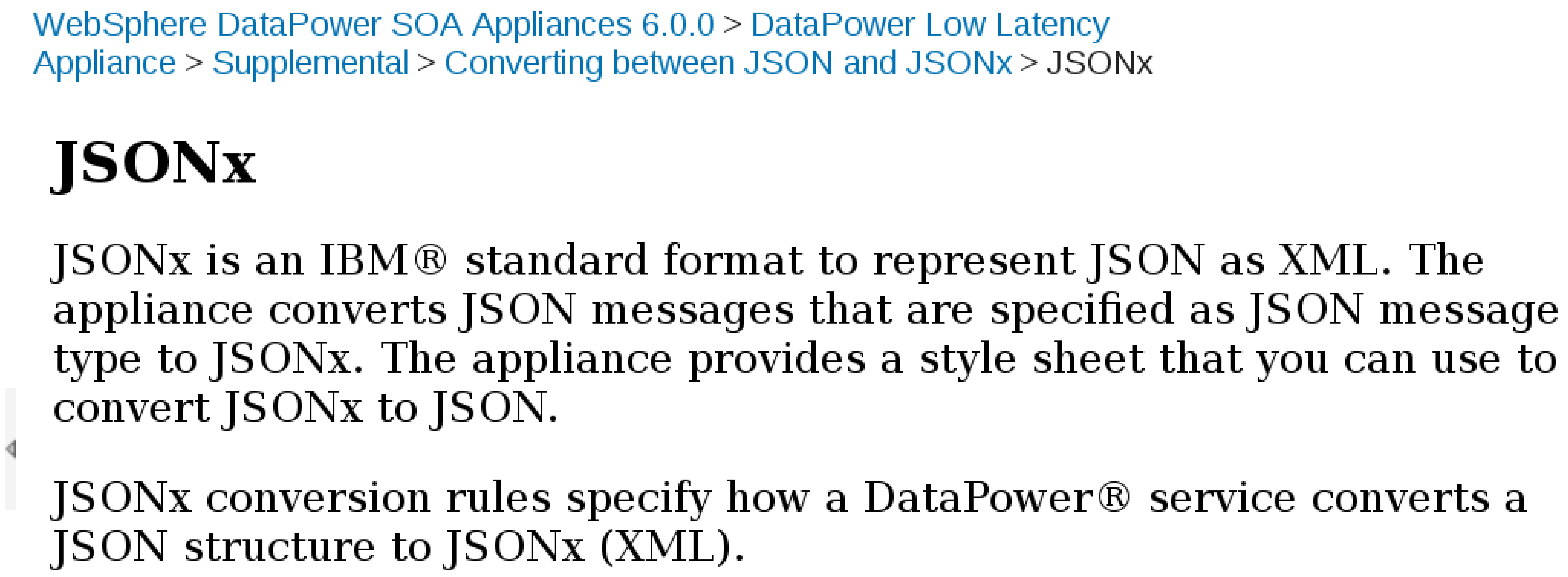
Source: http://xkcd.com/927/

SOAP

jsonrpc ~= xmlrpc

jsonpath ~= xpath

jsont ~= xslt



WebSphere DataPower SOA Appliances 6.0.0 > DataPower Low Latency Appliance > Supplemental > Converting between JSON and JSONx > JSONx

JSONx

JSONx is an IBM® standard format to represent JSON as XML. The appliance converts JSON messages that are specified as JSON message type to JSONx. The appliance provides a style sheet that you can use to convert JSONx to JSON.

JSONx conversion rules specify how a DataPower® service converts a JSON structure to JSONx (XML).

Source: http://www-01.ibm.com/support/knowledgecenter/SS9H2Y_6.0.0/com.ibm.dp.xm.doc/json_jsonx.html

JsonML.org

JsonML (JSON Markup Language) is an application of the JSON (JavaScript Object Notation) format. The purpose of JsonML is to provide a compact format for transporting XML-based markup as JSON which allows it to be losslessly converted back to its original form.

Native XML/XHTML doesn't sit well embedded in JavaScript. When XHTML is stored in script it must be properly encoded as an opaque string. JsonML allows easy manipulation of the markup in script before completely rehydrating back to the original form.

Source: http://httpstatusdogs.com/406-not-acceptable

jsonhome

```
{
    "resources": {
        "http://example.org/rel/widgets": {

            "href": "/widgets/",

            "hints": { ... }
        },

        "http://example.org/rel/widget": {

            "href-template": "/widgets/{widget_id}",
            "href-vars": {
                "widget_id": "http://example.org/param/widget"
            },

            "hints": { ... }
        }
    }
}
```

pip install jsonhome

uritemplate

```
http://example.com/~{username}/

http://www.example.com/foo{?query,number}

http://www.example.com/foo{?value*}


https://{mirror}/packages/source/{package:1}/{package}/
{package}-{version}.tar.gz{#md5}
```

rfc6570

pip install uritemplate

jsonhome

```
"resources": {
    "http://example.org/rel/widget": {
        "href-template": "/widgets/{widget_id}",
        "href-vars": {
            "widget_id": "http://example.org/param/widget"
        },

        "hints": {
            "allow": ["GET", "PUT", "DELETE", "PATCH"],
            "formats": {
                "application/json": {}
            },

            "accept-patch": ["application/json-patch"],
            "accept-post": ["application/xml"],
            "accept-ranges": ["bytes"]
        }
    }
}
```

pip install jsonhome

JSON Schema

JSON Schema ~= xsd (xml schema)

```
{
    "type": "string",
    "minLength": 10,
    "maxLength": 14,
    "pattern": "^\D*0(\D*\d){9}\D*$"
}
```

```
{
    "type": "boolean"
}
```

```
{
    "type": "number",
    "multipleOf": 10,
    "minimum": 0,
    "maximum": 100
}
```

```
{
    "type": "string",
    "enum": ["red", "green"]
}
```

pip install jsonschema

JSON Schema

```
{
    "type": "array",
    "items": {
        "type": "number",
        "maximum": 50
    },
    "minItems": 3,
}
```

```
{
    "type": "array",
    "items": [
        {
            "type": "number",
            "minimum": 1,
            "multipleOf": 1
        },
        {
            "type": "string"
        },
        {
            "type": "string",
            "enum": ["Street",
                     "Avenue",
                     "Boulevard"]
        }
    ],
    "additionalItems": false
}
```

JSON Schema

```
{
    "type": "object",
    "properties": {
        "name": {
            "type": "string"
        },
        "phone": {
            "type": "string"
        },
        "age": {
            "type": "number",
            "minimum": 5
        }
    },
    "additionalProperties": false,
    "required": ["name"]
}
```

```
{
    "name": "Trevor",
    "age": 44
}
```

```
{
    "age": 44
}
```

```
{
    "name": "Trevor",
    "legSeam": 81
}
```

JSON Schema

```
{
    "definitions": {
        "address": {
            "type": "object"
            "properties": {
                "street_address": {"type": "string"},
                "city": {"type": "string"},
                "state": {"type": "string"},
            },
            "required": ["street_address", "city", "state"]
        }
    }

    "type": "object",
    "properties": {
        "billing": { "$ref": "#/definitions/address" },
        "shipping": { "$ref": "#/definitions/address" }
    }
}
```

JSON Schema

```
{
    "anyOf": [
        {"type": "string"},
        {"type": "number"}
    ]
}
```

```
{
    "not": {
        "type": "string"
    }
}
```

```
{
    "shipping": {
        "allOf": [
            {"$ref": "#/definitions/address"},
            {
                "properties": {
                    "address_type": {
                        "enum": ["residential", "business"]
                    }
                }
            }
        ]
    }
}
```

Robustness Principal

“Be conservative in what you send, be liberal in what you accept”

Jon Postel – RFC 1122 – Oct 1989

The Harmful Consequences of Postel's Maxim

- https://tools.ietf.org/html/draft-thomson-postel-was-wrong-00
- Can lead to interoperable implementations
- Flaws become standards
- Backwards compatibility

  “Protocol designs and implementations should be maximally strict.”

- Fail Fast and Hard

```
>>> import jsl

>>> class Entry(jsl.Document):
...     name = jsl.StringField(required=True)
...

>>> class File(Entry):
...     content = jsl.StringField(required=True)
...

>>> class Directory(Entry):
...     content = jsl.ArrayField(jsl.OneOfField([
...         jsl.DocumentField(File, as_ref=True),
...         jsl.DocumentField(jsl.RECURSIVE_REFERENCE_CONSTANT)
...     ]), required=True)
...

>>> Directory.get_schema()
```

pip install jsl

```
>>> schema = {
    'name': 'Country',
    'properties': {
        'name': {'type': 'string'},
        'abbreviation': {'type': 'string'},
        'population': {'type': 'integer'},
    },
    'additionalProperties': False,
}

>>> import warlock
>>> Country = warlock.model_factory(schema)

>>> sweden = Country(name='Sweden', abbreviation='SE')

>>> sweden.name = 5
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "warlock/core.py", line 53, in __setattr__
    raise InvalidOperation(msg)
warlock.core.InvalidOperation: Unable to set 'name' to '5'
```

pip install warlock

jsonpatch

```
         {
Before:    "baz": "qux",
           "foo": "bar"
         }

         [
           { "op": "replace", "path": "/baz", "value": "boo" },
Patch:     { "op": "add", "path": "/hello", "value": ["world"] },
           { "op": "remove", "path": "/foo"}
         ]

         {
After:      "baz": "boo",
            "hello": ["world"]
         }
```

rfc6902

pip install jsonpatch

json hyper-schema

```
{
    "title": "Article",
    "type": "object",
    "properties": {
        "id": { ... },
        "title": { ... },
        "authorId": { ... },
    },
    "required" : ["id", "title", "authorId"],
    "links": [
        {
            "rel": "author",
            "href": "/user?id={authorId}"
            "targetSchema": { ... }
        },
        {
            "rel": "alternate",
            "href": "/{id}/rss",
            "mediaType": "application/rss+xml"
        }
    ]
}
```

jsonapi

:(

jsonapi.org

Don't use everything

Try and use something

Don't take my word on anything

jamielennox@gmail.com

*@jamielennox_*