VWRAP Type System
Experience & Issues

March 2010

Mark Lentczner / “Zero Linden”
Experience Report
Implementations

C++: Linden Lab
C#: OpenMetaverse
Haskell: Linden Lab
Java: Linden Lab, University of St. Andrews
JavaScript: Linden Lab
Perl: Linden Lab
PHP: Linden Lab, SignpostMarv
Python: Linden Lab
Ruby: Linden Lab
Usage

Code:

9% of modules in LL main codebase use LLSD

(≈8k modules, ≈3M lines of code)
Usage

Counts:

- 253k log messages / minute
- 9.7M objects created / day

**daily peak (notation) (XML)**
Usage

Data Volume:

6.4 TiB / day for 2M teleports
average of 3.3MiB each (binary)
big LLSD!

55.7 TiB / day for 749k simulation checkpoints
gzip’d to 42% of original (XML)
Pain Points

Content Negotiation
Unicode Strings vs. XML
Mixing LLSD and non-LLSD
Documentation
Validation
Big data sets
Issues
Types: Date Range

The range of dates is currently bounded:

“Data of type Date may have the value of any time in the from January 1, 1970 though at least January 1, 2038, to at least second accuracy.”

Is this a worry?
Encoding: JSON Subtlety

1) RFC 4627 vs. ECMA-262

2) RFC 4627’s JSON-text vs. ECMA-262’s JSONValue

3) String literals are escaped UTF-16, not Unicode!
   U+1D11E (𝄞) is encoded in JSON as:
   "\ud834\udd1e"
Encoding: XML Base64

1) The `encoding` attribute is defaulted to `base64`, it is not `#REQUIRED`

2) Which alphabet?
   RFC 4648 § 4. Base 64 Encoding

3) Linebreaking, Padding and Non-Alphabet
   RFC 4648 § 3. Implementation Discrepancies
LLIDL: REST vs. HTTP

1) Clear community dislike of binding to HTTP

2) Agreement on focus on REST semantics

3) Which REST like operations do we support?

<table>
<thead>
<tr>
<th>Operational</th>
<th>POST</th>
<th>&lt;--  --&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Readable</td>
<td>GET</td>
<td>&lt;&lt;</td>
</tr>
<tr>
<td>Read/Write</td>
<td>GET/PUT</td>
<td>&lt;&gt;</td>
</tr>
<tr>
<td>Read/Write/Deletable</td>
<td>GET/PUT/DELETE</td>
<td>&lt;x&gt;</td>
</tr>
</tbody>
</table>
LLIDL: Events

An event, as cast into REST, would be like a request with no response.

This does seem to map onto the communication needs of VWRAP.

Do we add this? Perhaps:

<table>
<thead>
<tr>
<th>Event</th>
<th>POST?</th>
<th>&gt;&gt;</th>
</tr>
</thead>
</table>

Tuesday, March 23, 2010
LLIDL: Semantics of Matching

LLIDL describes shapes
LLSD describes defaulting and conversion

What constitutes conformance of an LLSD value to an LLIDL description?

One approach:

- `match(actual)` -- matches structurally and all conversions are valid (non-defaulted)
- `valid(actual)` -- matches structurally though defaulted or additional data is acceptable
- `has_additional(actual)` -- has additional data
- `has_defaulted(actual)` -- has defaulted data
- `incompatible(actual)` -- the value is incompatible
LLIDL: Stand Alone Values

There a need to be able to reference (and check conformance) for individual values.

The named type facility serves this purpose for now.
LLIDL: Path Arguments

Astute readers will have noticed the addition of query argument specifications to LLIDL (the ?? syntax).

While not strictly part of the current VWRAP usage, internally the need for specifying the query arguments to a resource was pressing.

The need for path arguments is similar, though at present, LLIDL doesn’t consider the URL to access a resource.
VWRAP Foundation
Issues

March 2010

Mark Lentczner / “Zero Linden”
Additions:

- Minor nit: “Capability Host”
- Minor nit: Resource Base URI
- Medium issue: Required Serialization Formats

Tuesday, March 23, 2010
Issues
Seed Capability Format

In draft:

```javascript
%% seed
-> { capabilities: [ string, ... ] }  
<- { capabilities: { $: uri } }  
```

But should it be?

```javascript
%% seed
-> { capabilities: [ string, ... ] }  
<- { capabilities: [ &cap, ... ]
&cap = { name: string, loc: uri }  
```
Event Queues

1) Are events to be handled differently than other resource types?

2) How should server invoked resources be implemented?

3) Is the current long-poll queue mechanism good enough for now?
Capability Host

The “capability host” is responsible for both granting and proxying

Should split out into two:

“granting host”
“proxying host”
Resource Name Base URI

Currently:
http://xmlns.secondlife.com/capability/name

Propose something like:
urn:vwrap:res:
Serialization

Currently “XML and JSON ... **MUST** be supported”

Is there a point to forcing everyone to do both?

We could put the burden on:
   just providers?
   just consumers?

How does that work in our more symmetric world?