JLIFF: Where we are, where we're going
The Story So Far

https://www.flickr.com/photos/anirvan/12002658/
XLIFF has always been XML-based

- No abstraction independent of the syntax
- Limits our ability to reason about the format
- Limits its adoption and (possibly) longevity
The OASIS XLIFF

Object

Model

&

Other

Serializations

Technical Committee

“Since December 2015”
XLIFF-OMOS Goals

- Develop an Object Model (OM)
- Develop non-XML representations of the OM
- Also in scope:
  - Mappings between different *LIFFs
  - Future versions of TMX
  - APIs related to XLIFF or related standard data exchange
Brief Digression

The XLIFF Object Model

OM Goals

● Define “LIFF” independent of representation
● Streamline introduction to XLIFF/JLIFF/etc
● Standardize terminology of language interchange concepts
OM Status

- Working draft of prose spec
- UML of XLIFF core
- Work ongoing at https://github.com/oasis-tcs/xliff-omos-om
OM: Partial UML Rendering

«DataType»
- LIFF [primary root object]
  - attributes
    - + trgLang: EString [0..1]
    - + version: EString [1]
    - + srcLang: EString [1]
    - + *module or extension attributes: <Undefined> [*]
    - + xmlns: EString [0..1]

LIFF contains files
  - + file

«DataType»
- file [secondary root object]
  - attributes
    - + id: EString [1]
  - 1+ file

file can contain or reference a skeleton
files have to contain groups or units

«DataType»
- skeleton
  - attributes
  - 1+ group

1+ file

groups are recursive

1+ group

«DataType»
- group [option...]
  - attributes
  - + group

«DataType»
- unit [smallest]
  - attributes
  - 1

«DataType»
- source

1+ target

sub-unit has to have units
contain one target

sub-units can contain one target

groups can contain units

+ source

«DataType»
- target0..1
Why JSON?

- Ubiquitous in web service implementations
- Syntactically simple
- Widely understood
- Good tooling
Use Cases

- Webservices
- Online translation environments
- Data storage in JSON-based stores
Goals for JLIFF

- Proof of concept for OM idea
- Improve availability of support for LIOM-compatible implementations in various development scenarios
- Data should be interchangeable without loss between JLIFF and XLIFF!
JSON: Strengths and Limitations

- Simple
- Reasonably concise
- Widely supported
- Good Unicode support
- Incompletely specified

(Nicolas Seriot, “Parsing JSON is a Minefield”)
Complementary Technologies

- **JSON-Schema**
  - Data typing, validation

- **JSON-LD**
  - Namespacing
XLIFF always works at the file level

```xml
<xlfiff>
  <file id="f1">
    ...
  </file>
  <file id="f2">
    ...
  </file>
</xlfiff>
```
JLIFF is attempting a more flexible approach

```json
{
   "files": [...]
}
```

* These representations may change

```json
{
   "groups": [...]
}
```

** What are the semantics of converting this to XLIFF?

```json
{
   "fragment": {...}
}
```
Schema work has been straightforward
What about Namespaces?

<unit id="1">
  <gls:glossary>
    <gls:glossEntry ref="#m1">
      <gls:term source="publicTermbase">TAB key</gls:term>
      <gls:translation id="1" source="myTermbase">Tabstopptaste</gls:translation>
      <gls:translation ref="#m2" source="myTermbase">TAB-TASTE</gls:translation>
      <gls:definition source="publicTermbase">A keyboard key that is traditionally used to insert tab characters into a document.</gls:definition>
    </gls:glossEntry>
  </gls:glossary>
  <segment>
    <source>Press the <mrk id="m1" type="term">TAB key</mrk>.</source>
    <target>Drücken Sie die <mrk id="m2" type="term">TAB-TASTE</mrk>.</target>
  </segment>
</unit>
Namespacing via JSON-LD

Defined at http://docs.oasis-open.org/xliff-omos/jliff/v2.1/jliff-v2.1.jsonld:

```
{
   "@context": {
      "gls": "urn:oasis:names:tc:xliff:glossary:2.0:glossary",
   }
}
```

```
{
   "@context": "http://docs.oasis-open.org/xliff-omos/jliff/v2.1/jliff-v2.1.jsonld",
   "gls:glossary": [
      {
         "gls:definition": { ... }
      }
   ]
}
```

Use of a colon is a minor inconvenience for some languages (Javascript)
Source, Target data are flat lists of objects
  - snippet of text
  - marker
  - inline code

Segments, Ignorables collected as "subunits" array

Default property values in JSON-schema make things more concise
A real example, part 1

```json
{
  "@context": "http://docs.oasis-open.org/xliff-omos/jliff/v2.1/jliff-v2.1.jsonld",
  "jliff": "2.1",
  "srcLang": "en",
  "trgLang": "de",
  "units": [
    {
      "id": "u1",
      "subunits": [
        {
          "source": [
            {
              "text": "Press the "
            },
            {
              "kind": "sm",
              "id": "m1",
              "type": "term"
            },
            {
              "text": "TAB key"
            },
            {
              "kind": "em",
              "startRef": "m1"
            },
            {
              "text": "."
            }
          ],
          "target": [...]
        },
        {
          "type": " ignorable",
          "source": "<br/>"
        }
      ]
    }
  ]
}
```

(These may change)

Always preserve space

Markers
A real example, part 2

```
"gls:glossary": [ {
  "ref": "m1",
  "term": { "text": "TAB key", "source": "publicTermbase" },
  "translations": [ {
    "id": "1",
    "source": "myTermbase",
    "text": "Tabstopptaste"
  } ]
},
  "definition": { "text": "A keyboard key that is traditionally used to insert tab characters into a document." },
  "source": "publicTermbase"
} ]
```
This looks... fine to a machine

"@context": "http://docs.oasis-open.org/xliff-omos/jliff/v2.1/jliff-v2.1.jsonld",
"jliff": "2.1",
"srcLang": "en",
"trgLang": "de",
"units": [{
"id": "u1",
"subunits": [{
"source": [{
"text": "Press the 
"},
"id": "m1",
"kind": "sm",
"type": "term"},
"text": "TAB key"},
"startRef": "m1",
"kind": "em"},
"text": "."]},
"target": [{
"text": "Drücken Sie die ",
"id": "m2",
"kind": "sm",
"type": "term"},
"text": "TAB-TASTE"},
"startRef": "m1",
"kind": "em"},
"text": ""]},
"type": "ignorable",
"source": "<br/>"}],
"gls:glossary": [{
"ref": "m1",
"term": [{
"text": "TAB key",
"source": "publicTermbase"}],
"translations": [{
"id": "1",
"source": "myTermbase",
"text": "Tabstopptaste"}],
"definition": [{
"text": "A keyboard key that is traditionally used to insert tab characters into a document."
"source": "publicTermbase"}]}]}

Obstacles

https://www.flickr.com/photos/fernando/2620041065
Obstacle: XML Haunts Us

It's just a bunch of key/value pairs!
"mda:metadata": [ 
  {
    "mda:category": "document_xml_attribute",
    "mda:meta": {
      "mda:version": 3,
      "mda:phase": "draft"
    }
  }
]
Second Attempt at JLIFF mda

"mda:metadata": {
  "id": "optional-id",
  "mda:metaGroups": [
    {
      "mda:category": "document_xml_attribute",
      "mda:meta": [
        { "type": "version", "value": 3 },
        { "type": "phase", "value": "draft" }
      ]
    }
  ]
}
5.4.4.2 metadata

Container for metadata associated with the enclosing element.
Contains:
- One or more `<metaGroup>` elements

Attributes:
- `id`, OPTIONAL

- metadata must be an object to hold this property

5.4.5.4 type

type - indicates the type of metadata contained by the enclosing element.

*Value description:* Text.

*Default value:* undefined.

*Used in:* `<meta>`.

- Duplicate type keys are possible
"Writers that do not support a given custom namespace based user extension SHOULD preserve that extension without Modification."

This works across XLIFF/JLIFF conversion, right?
Imagine a Custom XLIFF extension

```xml
<foo:data id="id123"
  xmlns:foo="urn:foo:bar">
  <foo:value>50</foo:value>
</foo:data>
```

This is easy to pass through in JLIFF! I can just use a local `@Context` declaration for the namespace, and then....

Oh.

*Oh dear.*
The Revenge of Untyped Values

What is the data type of this value?
Is the appropriate JLIFF representation

"value" : 50

or

"value" : "50"
Looking Ahead

https://www.flickr.com/photos/pmillera4/13570027834/
Goals for 2018

- Finish the schema! (Modules, modules)
- Write the doc!
- Implementations! (Okapi, ???)
- (Finish OM)
- (Look at other formats)
Where to learn more

JLIIFF work on GitHub:

https://github.com/oasis-tcs/xliff-omos-jliff

OM work on GitHub:

https://github.com/oasis-tcs/xliff-omos-om
Discussion!

Questions?

chase@spartansoftwareinc.com

@ctatwork