details - dsouther

Metadata Format

[Metadata from components paper]

Due to the wide availability of libraries, the computable file will be released in JSON format. The metadata will be keyed at top level to ISO country code. Each ISO country code key will have an object with three keys - name, formats, and enums. Name is a string Display Name for the ISO code, in English. The UI may perform internationalization. Formats is a list of the supported formats for a country. Each format in the list is itself a list of address field descriptors. The enums key contains a map of enum names to list of possible enum values.

An address field descriptor has information about one token of the address data. Fields in an address must be in the same order as they appear in the format list. A descriptor has a key `field` that is a string. Descriptors have a key `optional` to indicate that they may be skipped. Descriptor `format` describes how the data is interpreted, with two approaches. If the `enum` field is present and set to true, `format` is a reference to the `enum` map, and values must come from that enum list. Otherwise, `enum` is interpreted as a regex pattern. This pattern is limited to a subset of regex that operates for both java.util.Pattern, as well as Javascript's RegExp.

Full example for Australia:

```
{
"AU": {
    "name": "Australia",
    "formats": [
        [
            { "field": "street_number", "format": "[0-9]+" },
            { "field": "street_name", "format": ".+" },
            { "field": "street_type", "format": "STREET_TYPE",
                "optional": true , "enum" : true},
            { "field": "city", "format": "AU_CITY", "enum" : true},
            { "field": "province", "format": "AU_PROVINCE", "enum" : true},
            { "field": "zip", "format": "[0-9]{4}" },
            { "field": "country", "format": "Australia", "optional": true }
        ]
    ],
    "enums": {
        "AU_CITY": ["Sidney", "Perth", "..."],
        "AU_PROVINCE": ["ACT", "NSW", "NT", "..."],
        "STREET_TYPE": ["ALLEY", "ARC", "AVE", "..."]
    }
}
}
```

API Format

Requests to the search endpoint will be in JSON. The body will have a single JSON object. The keys will be the values of the `field`s from one metadata format. The values should correspond to the format for the equivalent metadata field. However, this will be validated in the search RPC.

```
{
    "street_number": "123",
    "street_name": "Main",
    "street_type": "ST",
    "city": "Brisbane",
    "province": "QLD",
    "zip": "4500"
}
```

Search RPC

The search server will expose two RPCs. For country specific searches, clients will POST an API payload `/search/{ISO_CODE}` where `ISO_CODE` is the country of interest. If ISO_CODE is not a country the system has metadata for an address format, this POST will return 404. Otherwise, the server will validate the payload against the format and enum values in the appropriate metadata. If the payload does not meet the required format, the server will return `400 Bad Request`, and a JSON payload with each invalid field having an appropriate error message.

```
{
    "street_number": "'ABC' must match '\d+'",
    "province": "'Adelaide' is not a known province"
}
```

If the request payload validates, the fields are then mapped from country-specific fields in the metadata to the appropriate database columns. Enum fields are queried as-is, and must match specifically and entirely. Non-enum fields are queried using `LIKE %value%`, to find similar addresses. All field clauses are `AND`ed in the search. If no results are found, the server will respond with status `200` and body `{"search": "NOT FOUND"}`. If one or more results are found, the server will respond with status `200` and body `{"search": "FOUND", "addresses": [{}, ...]}`. Where addresses contains each address found, with fields translated back from the database columns into the appropriate metadata field names for that country.

Cross-country search will behave similarly. The API endpoint will be at `/search` (no ISO code). The request body can specify any fields, but those fields must match at least one country format field. The server will validate each field against all known format field types, with the same error responses as above. If the body is valid, the server will use the same query rules, and same response types, as the country specific case.

TODO Should match actually be a fulltext search for performance in both cases?

TODO Can we have a pre-match against known formats? That is, when the cross country search comes in, compare the payload and make a list of which contries have a possibly matching format?

UI Generation

The UI will begin with a single drop down, allowing country selection. The first option in the dropdown will be "all", followed by a separator, followed by countries with a known metadata format listed in alphabetical order. When selecting a country, the form will present inputs and dropdowns appropriate to the metadata format descriptors. For instance, a field `{ "field": "street_name", "format": ".+" }` would get an `<input>` with a validator against `/.+/`. The descriptor `{ "field": "street_type", "format": "STREET_TYPE", "optional": true , "enum" : true}` would get a dropdown with all the `STREET_TYPE` values, as well as a "None" option. The form will post its submission to the appropriate `/search` endpoint based on country selected. When the response comes back, the page will display either the error(s), or the list of found addresses.