

Strawman 39: Update following the Working Group meeting of April 24, 2013

The group basically chose Option 3 from Juli's previous document: the ILS will decide which fields will be included in its 64 responses to any given client. The assumption is that the ILS will present granular options to the librarian for which fields to include on a per-SIP-client basis, and that the ILS will identify clients via their 93 login information. (Which could also be done by remote IP address, by port, or some other method.)

The group also decided that somehow the server should tell the client which fields it's including and which it's excluding. Client software can then compare that list against the fields it needs, and can present a useful and precise error message to the librarian if the server isn't going to give it enough information, rather than the troubleshooting nightmare of random pieces of data not showing up. This strawman presents some options for how to achieve this.

We should encourage ILSes to also deny access to the appropriate (most?) SIP messages over connections where privacy is a concern. This should be designated via the BX field, and also should be actually enforced.

General question: We're proposing to define groups in the 64 response: ADDRESS, PHONE, and EMAIL. This strawman assumes that a server either supports sending a group or not; in other words, a server can choose whether or not to send any addresses at all, but if it does, then it must send all information about all addresses. This assumption simplifies the specification. But is it an oversimplification? Would there be a time when, for example, a ZIP code might need to be sent for some kind of authentication?

Option A – Add bytemap field to 98:

Use a method much like the existing BX field, which describes which messages are supported and which aren't. Let's call it the BD field (most of the good Bs are taken). Each field in the 64 message would be assigned a position, and that position would have a Y/N for whether that field is enabled by the server for that connection. It would be part of the 98 (or 94) response, just like the BX field is.

Pros:

- Concise
- Parallels the existing BX field nicely, which is good because they have very similar functions
- Fits perfectly into the existing 98/99 (or 93/94) messages where BX is now
- BX is a variable-length field, and this would be as well, so that new 64 fields could be added and handled
- One-time setup information about the connection really does belong in 98/99

Cons:

- Requires an arbitrary (but well-defined) mapping between byte position and the 64 field described
- Is more SIP2-ish than SIP3-ish (but then the BX field has this same problem)
- If a SIP server adds a field in 64 which is not described in the spec, then including that field in this scheme gets tricky. I believe it could be done (perhaps by assigning bytes 50 and up to be "user-defined" or similar) but it wouldn't be very tidy.

Option B – Overloading 63/64:

When the client wants to know what 64 fields a server supports, it sends a specially-crafted 63 message. For example (the AA field is the interesting one):

```
63|UL000|TD20130424 124530|AOwhatever|AA_WHATFIELDS_|AY1
```

The server then sends back a 64 response which contains all the fields which it supports for that connection. The fields are all empty, but their presence indicates they'll be sent, when appropriate, for an actual patron lookup. For any group that is supported, one empty instance of the group is included.

Example response (include patron status, language, canonical card number, patron last name, and phone numbers):

```
64|TD20130424 124530|SP|UL|AA|UQ|GRPHONE|GX|AY1
```

Pros:

- The client sees an “example” 64 response, and process it with slight modifications to its existing 64 processor
- Directly states which fields are supported; good for extensibility/readability
- Doesn't require a new message

Cons:

- Overloading is ugly, both conceptually and in code
- Is like a new message, just one that uses the same identifier

Option C – New Message

When the client wants to know what 64 fields a server supports, it sends a (new!) 61 message, expecting to get back a 62 response. The 61 request has no fields, viz:

```
61|AY2
```

The 62 response has only one repeatable field: the UY field. Each UY field contains the field identifier of a supported field in 64. Example response (include patron status, language, canonical card number, patron last name, and phone numbers):

```
62|UYSP|UYUL|UYAA|UYUQ|UYPHONE|AY2
```

Pros:

- Directly states which fields are supported; good for extensibility/readability

Cons:

- It's a new message, requiring some other changes in the spec (like the BX field), and generally

- is a larger change to SIP than other options
- Requires an additional setup message to be sent, in addition to 93 and/or 99, before useful information can be exchanged

Option D – Add repeatable field to 98

Make the new, repeatable UY field (from Option C) part of the 98 response. This is similar to one of the language proposals, where all the languages the server supports are sent in a repeatable field in 98. For example (the BX field's data is completely made up and wrong):

```
98|OAY|DT20130424 124530|VR3.00|AOwhatever|BXYYYYNNYNYN|UYSP|UYUL|UYAA|
UYUQ|UYPHONE|AY3
```

Pros:

- Directly states the supported fields; good for extensibility/readability
- One-time setup information about the connection really does belong in 98/99

Cons:

- Makes the 98 response quite lengthy

Conclusions:

In order of my preference at this point (of course I may have missed some option, or some critical pro or con):

- 1) Option D – Add repeatable field to 98
- 2) Option A – Add bytemap field to 98
- 3) Option C – New message
- 4) Option B – Overload 64

I think this is exactly the kind of thing that belongs in the 98 – ILS Status message.