22 July 2025

draft-ietf-regext-rdap-extensions

RegExt - IETF 123

Internet Engineering Task Force
© 2025 IETF Trust
Production by Meetecho

IETF

Namespaced Identifiers

Text describing extensions using bare identifiers introduces the term “namespaced identifiers” to more accurately describe the issue.

Previously the text focused primarily on the names of JSON properties, but there are other identifiers in RDAP that need namespaces (e.g. query parameters, object class names).

Names of things in RDAP

STD 95 says identifiers SHOULD be “extensionId_name”:

“It has been observed that these short identifiers aid software implementers with identifying the specification of the JSON member, and failure to use one could cause an implementer to assume the server is erroneously using a name from this specification.”

Feedback from implementers proves this out.

Extensions using Bare Identifiers

What is a bare identifier?

It is when an extension does not use the “_name” part, relying only on the extension ID to avoid collisions. Can only be used when an extension defines one identifier of a type (e.g. JSON name, object class name...)

In summary, it is when an extension violates the SHOULD in STD 95.

No Consensus On Bare Identifiers

Position 1: Disallow them (effectively turn the SHOULD to MUST).

Position 2: Always allow them (effectively remove the SHOULD).

After the interim, a compromise position:
Only if there is no way to solve the problem without one.

Why Disallow?

1. This *simplifies*, makes *consistent* the rules for writing extensions which will reduce mistakes.
2. Implementation feedback has indicated that they can cause problems under certain circumstances.
3. Bare identifiers look like core RDAP and can cause confusion.
4. Goes against many IETF RFCs on protocol design: RFC 1958, RFC 5218, & RFC 9413 [1].

[1] https://mailarchive.ietf.org/arch/msg/regext/7uEXtk1B-5_-KoFtJlhrRPlCcZ0/

Why Allow?

The only consistent reason given is that it looks better (“beauty”).

- but that is an aesthetics argument, not a technical argument.
- hidden from the user, so only software sees this beauty.
- unusable if more than one identifier of a type is defined.

Why compromise?

1. The compromise is allowance just in case there is something we did not consider (after 10 years, this is unlikely).
2. It does what all compromises are supposed to... move us past this issue.

What Next?

The current draft lists all 3 positions. Please read it and provide feedback.

And then we ask the chairs to make a very rough consensus call.