**About Zero-length Components**

Updated August 27, 2014

*Empty component is very strange thing with not clear semantics.*

Two questions or points about this:

1. What are the potential negative consequences of leaving support for empty components intact?
2. Why reduce the expressiveness of the namespace unnecessarily?

Negative Consequences

As forwarding can ignore the empty component, I have trouble seeing why the protocol definition should not allow this case.

When you don’t know, grant freedom.[[1]](#footnote-1)

Update: An additional argument about ruling out an empty component based not being able to think of how an application would use it: Maybe we can't think of how an application would use a component that is all whitespace, or a string with whitespace before or after it. Should we rule those out too?

Expressiveness

In languages, the explicit *null* or *None* is quite useful, and often preferable to an application-defined non-value. Why would similar support not be valuable in the NDN namespace?

For example, consider the application that maps directly the contents of a variable to a particular component. In the building automation and management system application, this might be a name like /routable-prefix/<building>/<room>/<system>/etc… For building-wide systems, an E-BAMS application could just encode <room> as null, represented as a zero-length component, rather than implement a special name. /routable-prefix/melnitz//electrical/foo becomes a reasonable name.[[2]](#footnote-2)

As a point of comparison, RFC 3986, which defines URIs, allows empty components.[[3]](#footnote-3)

Response to Lixia’s questions August 23

*Lixia’s email shown in italics.*

*- The following is the reasoning for keep zero-length name component (from your attached doc file): you said*

*"For example, consider the application that maps directly the contents of a variable to a particular component.  In the building automation and management system application, this might be a name like /routable-prefix/<building>/<room>/<system>/etc…   For building-wide systems, an E-BAMS application could just encode <room> as null, represented as a zero-length component, rather than implement a special name.*

*/routable-prefix/melnitz//electrical/foo becomes a reasonable name.[2]*

*[2] There are some application-side disadvantages to this, especially in URI-encoded names, because errors could be introduced through mistyping an extra slash.   But that’s an application-side problem."*

*Besides application-side problems, what kind of forwarding behavior does one expect for a name with a no-value component in the middle?*

A zero-length component is unique, so couldn't it be matched in the same way as any other component?  I don't think there has to be any ambiguity or special treatment.

*- the longest match may get the Interest to the right building, how would one expect the Interest to be further forwarded?  broadcast to all rooms inside the melnitz building?*

I did not assume any special behavior in the forwarding.

In this case, there would be one or more BMS processes that have registered /<root>/melnitz// and would handle the interest.  (Those processes, for example, might be responsible for reading all of the rooms and generating aggregate values that don’t have a “room”.)

*If so, then multiple rooms may reply?*

No, to get multiple rooms’ data you would issue repeated interests for /<root>/melnitz and, in this application scenario, exclude or ignore the zero-length child, or use sync and not fetch the zero-length child. This could, for example, be the aggregating process’ behavior.

One could argue the use of a null component actually makes forwarding more consistent in this BMS deployment example. Consider the following namespace (dropping the /<root>) that does not use a null component:

 /melnitz/1471/electrical intended for a device responsible for room 1471

  /melnitz/1471/plumbing intended for another device resp. for room 1471

 /melnitz/electrical intended for an aggregator process

  /melnitz/plumbing intended for the same aggregator process

In this case:

1. Every consumer has to know that "electrical" and "plumbing" are different than "1471" even though they are on the same level
2. The aggregator has to register each aggregate prefix individual or get all interests for individual devices and ignore them.

So I wouldn't be surprised if the design evolved to keep the same **field to component mapping**:

 /melnitz/all/electrical

or something similar if zero-length components were note allowed.

Isn’t using a null component instead of the string "all" in this case a similar, reasonable naming choice?

I'm not sure that the null component is the best namespace design, but I don't see why it's not allowable.

*- what name would the returning Data packet carry?*

*I suppose the room part of the name would be filled in?*

*But that would not allow the data going back to the consumer (well, unless one wants to make routers do special handling with no-value components...)*

Again, I wasn't assuming any special forwarding semantics.  Wildcard components are an interesting idea but that seems like a different discussion!

The motivation for this comes for a naïve but realistic approach to mapping fields to names. For example, an application developer might use a data structure with fields like the following for the data in question:

 building: "melnitz"

 room: None

 system: "electrical"

This is a pretty reasonable way of representing the case when there is no room constraint being placed on the "query". Here “None” is the language’s null value.

One encoding in XML (or JSON, etc.) might be

 <building>melnitz</building

 <room></room>

 <system>electrical</system>

This seems very reasonable.

If we look at the name as a one-to-one **field to name mapping** from this data structure, we get:

 <component id="0">melnitz</component>

 <component id="1"></component>

 <component id="2">electrical</component>

And the null component still seems quite reasonable (to me) in this case.

If the network just forwards on this name considering the zero-length component like any other unique component, the application semantics are preserved and apps interested in the "None" room just register for /melnitz// or its children.

I wonder how much the URI representation is contributing to discomfort here. (We don't like the double slash because we don't see or use it.)  But writing it in this XML-ish format reinforces it as a reasonable option for me. Also, it seems allowable in the URI spec, even.

*Pop up a level, what would be the desired result one wants to get from sending an Interest with name "/routable-prefix/melnitz//electrical/foo" ?*

*getting a read of "electrical" value from any of the rooms in melnitz building?*

This would be an aggregator-generated value for all of the electrical/foo in the building, or perhaps one of different possible aggregate objects, foo/mean/<start-date>/<end-date>. (This is just an example, still a lot to do in namespace design for this application.)

*I tend to share the feeling with Alex that this looks a bit confusing.*

I don’t disagree that it’s confusing in the URI representation, but given that a zero-length component is unique when compared to any other component, it can be forwarded like any other component. Why should it be disallowed?

Response to Lixia’s questions August 26

*Lixia’s email shown in italics.*

*1/ Interest carrying a name with some zero-length component does not need any special handling. There ought to be someone announcing the prefix to direct the forwarding (otherwise forwarding daemon treats the Interest in the same way as any others that dont have a FIB match)*

*2/ So this implies some kind of understanding/agreement between consumers and producers (or aggregators), one can send Interest with skipped name component, and on the data producing side there should be some party taking care Interests with such names.*

Yes, just like any other component. (Note the idea that the zero-length component is “skipped” is actually application-specific and doesn’t have to be the case in all applications.)

*take your example: given both consumers and producing side share the understanding about the meaning of "/melnitz//electrical", an alternative way to achieve the same is just to all agree on the meaning of the name ""/melnitz/electrical".*

*So it seems zero-lenght component is not absolutely necessary.*

Yes, it is not necessary. But putting different categories of information into the same place in the tree may cause a non-trivial amount of confusion and failures failure of utilities that try to do application-agnostic things in the namespace.

*3/ Now, is it beneficial?*

*We have been struggling with finding a way to access data along multiple naming dimensions instead of just one, e.g. now we name sensing data by location hierarchy, but also want to access by time hierarchy (power consumption at a given time window across all rooms, or even buildings); your example showed by data type classification.*

*So I wondered whether allowing empty name components in a name could be one pragmatic way to get there, without having to specifically define multiple hierarchies a prior -- in this regard maybe allowing empty components in a name helps "the expressiveness of the namespace" that you argued to keep.*

Maybe, I am not sure. I think that has to do more with ordering, but maybe this could help somehow.

*Of course, the apps need to be engineered to handle names with skipped components including making proper prefix announcements (or configs). This also calls for more thought on the ordering of name components*

Yes, though again I am not sure this is different than any other component. (I prefer to think of it asthe null component rather than a skip.)

*Given there is no technical roadblock to support this, I got the impression of consensus to support it in next release.*

*does anyone else in the NFD chat yesterday want to add/comment more on this?*

*It's not in 0.2.0 as we can't delay the release any further given NDNComm is coming next week (and adding a feature does not cause compatibility problem)*

1. F. P. Brooks, *The Design of Design*, talking about the design principle of generality in computer architecture. “Generality is the ability to use a function for many ends. It expresses the professional humility of the designer, his conviction that users will be inventive beyond his imagination and that needs may change beyond his ability to forecast. The designer should avoid limiting a function by his own notions about its use. When you don’t know, grant freedom.” [↑](#footnote-ref-1)
2. There are some application-side disadvantages to this, especially in URI-encoded names, because errors could be introduced through mistyping an extra slash. But that’s an application-side problem. [↑](#footnote-ref-2)
3. <http://tools.ietf.org/html/rfc3986#section-3.3> [↑](#footnote-ref-3)