ndn-cxx - Feature #4658

Encode and decode Interest ApplicationParameters

07/09/2018 02:41 PM - Davide Pesavento

Status:	Closed	**Start date:**	
Priority:	High	**Due date:**	
Assignee:	Davide Pesavento	**% Done:**	100%
Category:	Base	**Estimated time:**	6.00 hours
Target version:	v0.7		

Description

Extend ndn::Interest class to encode and decode the **ApplicationParameters** element.
This is part of Packet03Transition.

Interest::wireEncode function should support both v0.2 and v0.3 format.
If ApplicationParameters element is present, it encodes to v0.3 format; when v0.3 format is selected, if any v0.2-only element (such as MinSuffixComponents) was specified, they would not appear in the encoded packet.
Otherwise, it encodes to v0.2 format.

Interest::decode03 function should recognize and store ApplicationParameters element.
Interest::decode02 function remains unchanged.

Related issues:

Related to NFD - Feature #4535: Nack format for Interest v0.3	**Rejected**
Blocks ndn-cxx - Feature #4567: Encode Interest into v0.3 format and drop sup...	**Closed**
Blocked by NDN Specifications - Feature #4831: Redefine ParametersSha256Diges...	**Closed**

History

#1 - 07/09/2018 02:41 PM - Davide Pesavento

- Related to Feature #4535: Nack format for Interest v0.3 added

#2 - 07/10/2018 12:48 PM - Junxiao Shi

- Subject changed from Support Interest Parameters to Encode and decode Interest Parameters

- Description updated

- Assignee set to Arthi Padmanabhan

- Estimated time set to 4.50 h

#3 - 07/10/2018 12:49 PM - Junxiao Shi

- Blocks Feature #4567: Encode Interest into v0.3 format and drop support for v0.2 format added

#4 - 07/10/2018 12:51 PM - Davide Pesavento

- Target version set to v0.7

#5 - 07/10/2018 02:15 PM - Arthi Padmanabhan

To make sure I'm understanding this correctly, does this issue include building the Parameters class and its encoding? Based on the current wireEncode, it looks like I would need the Parameters implementation before I can check for the Parameters element and encode it. If so, is there any more information about what type of information can/should be contained in Parameters?

#6 - 07/17/2018 01:33 PM - Junxiao Shi

> does this issue include building the Parameters class and its encoding?

No. There would be no Parameters class. It's a Block same as Data packet **Content** element.

#7 - 07/18/2018 01:26 PM - Arthi Padmanabhan

- Status changed from New to Code review

#8 - 07/26/2018 04:30 PM - Alex Afanasyev

Apologies for a late response on this, but the API has to include updating of the name with a hash of the parameters (to ensure that the name is unique). The API can stay the same, just action for setParameters should include updating of the name; not quite sure what to do with unsetParameters.

#9 - 07/26/2018 04:38 PM - Davide Pesavento

In the NFD call that greenlit the current design (as implemented by Arthi), the consensus was that it's the application's responsibility to ensure that the name is unique. I don't have a strong opinion either way, but:

1. with Data packets and content, the uniqueness of the name is left to the application
2. hashing the parameters means agreeing on a hash function
3. what if the application knows better and wants to deal with name uniqueness on its own?
4. as you said, what component do we remove on unsetParameters?

If we go this way, my suggestion is to use a typed name component for the Parameters hash.

#10 - 07/26/2018 04:40 PM - Alex Afanasyev

At the very least, we need to have a helper and a quick "standard" way to add such a component. Ideally, as part of this commit. I still have a tiny preference to force this inside setParameters, but can also be flexible.

#11 - 07/26/2018 04:52 PM - Arthi Padmanabhan

Sure, maybe we can discuss at Monday's NFD call? And can we do this in a separate commit? Parameters aren't currently being used, though I can add a Redmine issue/comment if we're concerned about people trying to use it immediately

#12 - 07/26/2018 06:21 PM - Junxiao Shi

I disagree with setParameters automatically modifying Name. It would be surprising for a setter to change another field.

I'm fine with creating a helper function appendParameterDigestToName, but this requires defining a naming convention first, and should belong to a separate issue.

#13 - 07/26/2018 07:43 PM - Alex Afanasyev

It could be even more surpising to have two interests with the same name be aggregated or retrieve wrong data from caches (just because there is no suggestion/enforcement for name uniqueness). In any case, we shall proceed with the helper function (including the definition of the convention) ASAP, before we start using the parameters.

#14 - 07/26/2018 07:50 PM - Davide Pesavento

Alex Afanasyev wrote:

> In any case, we shall proceed with the helper function (including the definition of the convention) ASAP, before we start using the parameters.

Fair enough.

#15 - 07/31/2018 07:50 AM - Junxiao Shi

- Status changed from Code review to In Progress

Change 4889 adds an InterestParametersSha256DigestComponent name component type to ensure uniqueness of parameterized Interests.

I have a question: should InterestParametersSha256DigestComponent have its own prefix in URI encoding, or should it use the generic 2=xxx URI encoding?

#16 - 07/31/2018 07:55 AM - Alex Afanasyev

Logically, it should have its own name (the number is a possible URI representation). I was trying to add, but couldn't come up with a good name.

sha256params= ?

#17 - 07/31/2018 07:58 AM - Junxiao Shi

- Blocks Feature #4570: Redesign Name::getSuccessor added

#18 - 07/31/2018 08:00 AM - Junxiao Shi

> sha256params=

Yes this is fine.

To all: I'll take care of adding the name component type to ndn-cxx under #4570.
After that, Arthi can write the helper function for creating InterestParametersSha256DigestComponent from Parameters.

#19 - 09/07/2018 05:33 PM - Davide Pesavento

- % Done changed from 0 to 50

Helper function(s) that automatically take care of adding ParametersSha256DigestComponent are still missing. @Arthi, are you still working on this?

#20 - 09/07/2018 05:41 PM - Davide Pesavento

- Blocks deleted (Feature #4570: Redesign Name::getSuccessor)

#21 - 09/17/2018 01:43 PM - Arthi Padmanabhan

Yes, I'm still planning to but can only start next week

#22 - 09/22/2018 08:39 AM - Davide Pesavento

- Tags set to Packet03Transition

#23 - 12/04/2018 11:33 AM - Junxiao Shi

- Assignee changed from Arthi Padmanabhan to Xinyu Ma

> Helper function(s) that automatically take care of adding ParametersSha256DigestComponent are still missing.

https://gerrit.named-data.net/5027

#24 - 01/22/2019 09:27 PM - Davide Pesavento

- Priority changed from Normal to High

This seems to have stalled. Is anyone still working on this? If not, we need to reassign again.

#25 - 01/23/2019 01:49 PM - Xinyu Ma

Davide Pesavento wrote:

> This seems to have stalled. Is anyone still working on this? If not, we need to reassign again.

I'm still working on this. I will commit a new version soon.

#26 - 02/09/2019 02:01 PM - Davide Pesavento

I was wondering why Interest::getParameters() returns a Block (which includes the Parameters T and L)... why should an application care about how the element is encoded? This effectively means that the vast majority of apps will have to manually extract the TLV value of the returned Block, which leads to more boilerplate in the app's code. It's also inconsistent with the other getters... we don't include the T and L in the return value of getInterestLifetime or getNonce, for example.

My suggestion is to return a Buffer or ConstBufferPtr from getParameters().

#27 - 02/09/2019 02:44 PM - Junxiao Shi

Isn’t Data::getContent returning Block too?
A major benefit of Block is that its API allows easy decoding into a sequence of sub elements.

#28 - 02/09/2019 02:55 PM - Davide Pesavento

Junxiao Shi wrote:

> Isn’t Data::getContent returning Block too?

Yes, I don't like that either, but it's too late to change it.

> A major benefit of Block is that its API allows easy decoding into a sequence of sub elements.

It's only a "benefit" if you assume that apps will use a Block-based encoding (i.e. NDN-TLV) for their parameters. Still, returning a ConstBufferPtr from getParameters doesn't prevent this use case, since you can efficiently construct a Block from a ConstBufferPtr without copying the underlying buffer.

#29 - 02/21/2019 06:10 PM - Junxiao Shi

To ensure applications do not forget to check ParametersSha256DigestComponent, Interest::wireDecode should perform the check automatically.
To avoid slowing down the forwarder, this check can be turned off via Interest::setAutoCheckParametersDigest(false).
When an application leaves it at default, Interest::wireDecode throws on bad ParametersSha256DigestComponent.
Forwarder can turn it off.

#30 - 04/06/2019 12:34 PM - Davide Pesavento

- Subject changed from Encode and decode Interest Parameters to Encode and decode Interest ApplicationParameters

- Description updated

#31 - 04/06/2019 12:35 PM - Davide Pesavento

- Blocked by Feature #4831: Redefine ParametersSha256DigestComponent covered area added

#32 - 04/22/2019 03:36 PM - Junxiao Shi

- Assignee changed from Xinyu Ma to Yu Guan

#33 - 06/01/2019 07:50 PM - Davide Pesavento

- Assignee changed from Yu Guan to Davide Pesavento

- Estimated time changed from 4.50 h to 6.00 h

I'm taking over this task.

#34 - 06/02/2019 11:10 AM - Davide Pesavento

On gerrit, Alex says:

> I'm still unease that we modify name during encoding (which is const operation). No objection, but this is the root cause of this question.

Yeah I don't like that either, so I'm all ears if you have better suggestions for the API. I can think of two alternatives:

1. In wireEncode, only check that the digest component is present (and unique) if the Interest carries parameters, and that it's absent if the Interest does not carry parameters, throw an exception otherwise. Require the user to call a new updateParametersSha256Digest function at the right time (after adding/changing/removing parameters but prior to encoding/sending). One shortcoming of this alternative is that the checks in wireEncode won't detect mistakes such as modifying the parameters *after* calling updateParametersSha256Digest.

2. Automatically update (or add/remove) the digest component from setApplicationParameters and unsetApplicationParameters. Possibly add a boolean flag to these functions to disable the automatic update of the digest (will be useful when we implement the new signed Interest format). No extra work to do in wireEncode, or we could do the same checks as in alternative (1) just to be on the safe side. One drawback of this alternative is that it's not very future-proof if we add more Interest elements covered by the ParametersSha256Digest (i.e. elements after ApplicationParameters).

#35 - 06/06/2019 02:33 PM - Junxiao Shi

> I'm still unease that we modify name during encoding

No, Interest type should not modify name during encoding. Instead:

1. In constructor or setName:
   - If the Interest is expected to contain parameter elements, the name should contain exactly one ParametersSha256DigestComponent. Even if the Interest does not contain parameter elements at the moment, this operation does not throw.
   - If the Interest is expected to not contain parameter elements, the name should contain zero ParametersSha256DigestComponent. Even if the Interest contains parameter elements at the moment, this operation does not throw.
   - This operation throws if the name contains more than one ParametersSha256DigestComponent.

2. In getName:
   - If the Interest contains parameter elements, and the name contains a ParametersSha256DigestComponent, the return value should reflect

an updated digest.
    - If the Interest does not contain parameter elements, and the name does not contain ParametersSha256DigestComponent, the return value does not contain a digest.
    - In all other cases, the return value is the current name unchanged.

3. In wireEncode:

    - If the Interest contains parameter elements, and the name contains a ParametersSha256DigestComponent, the encoding result should reflect an updated digest.
    - If the Interest does not contain parameter elements, and the name does not contain ParametersSha256DigestComponent, the encoding result does not contain a digest.
    - In all other cases, this operation throws because ParametersSha256DigestComponent mismatches parameter elements.

Effectively,

- setName and any operation that adds or removes parameter elements can recompute the digest, iff the presence of ParametersSha256DigestComponent matches the presence of parameter elements.
- wireEncode fails if the presence of ParametersSha256DigestComponent mismatches the presence of parameter elements.

Notably, there's no automatic insertion or deletion of ParametersSha256DigestComponent in Name, because application should maintain control over the name structure.
The Interest type can update the digest in an existing ParametersSha256DigestComponent, but cannot insert or delete a ParametersSha256DigestComponent.

At callsite, Interest type is used as follows:

```
Interest iA("/A");
iA.setCanBePrefix(false);
iA.setMustBeFresh(false);
iA.wireEncode();

Interest iB("/B/params-sha256=0000000000000000000000000000000000000000000000000000000000000000");
iB.setCanBePrefix(false);
iB.setMustBeFresh(false);
iB.wireEncode(); // throws
iB.setApplicationParameters("240100"_block);
iB.getName(); // reflects updated digest
iB.wireEncode(); // reflects updated digest
```

Internally, Interest type may employ lazy computation to reduce overhead when multiple operations that adds or removes parameter elements are invoked in sequence. This could mean modifying name in getName and wireEncode functions, but the semantic remains the same: setName and any operation that adds or removes parameter elements can recompute the digest.

#36 - 06/06/2019 10:35 PM - Davide Pesavento

Junxiao Shi wrote:

> Notably, there's no automatic insertion or deletion of ParametersSha256DigestComponent in Name, because application should maintain control over the name structure.
> The Interest type can update the digest in an existing ParametersSha256DigestComponent, but cannot insert or delete a ParametersSha256DigestComponent.

Not sure if I agree with the above.

> Internally, Interest type may employ lazy computation to reduce overhead when multiple operations that adds or removes parameter elements are invoked in sequence. This could mean modifying name in getName and wireEncode functions, but the semantic remains the same: setName and any operation that adds or removes parameter elements can recompute the digest.

This is exactly what Alex doesn't like (unless I misunderstood his comment). And you're contradicting yourself by first agreeing with him ("should not modify name during encoding") and later saying that wireEncode *can* modify the name...

#37 - 06/17/2019 04:36 AM - Junxiao Shi

> you're contradicting yourself by first agreeing with him ("should not modify name during encoding") and later saying that wireEncode *can* modify the name...

wireEncode cannot modify the name in an externally visible way. Every name modification appears as if it's performed by setName or a method that affects an element covered by the digest, not wireEncode.
What wireEncode does internally is an implementation detail.

#38 - 06/29/2019 10:34 AM - Alex Afanasyev

Let me suggest a combination approach, based on what Junxiao suggested, but without modifications within const functions:

- In constructor or setName (same as in comment #35). With the exception that we make a requirement/assumption that not having parameters implies params-sha256=0000000000000000000000000000000000000000000000000000000000000000.
- getName does not do any modification of the name ever
- wireEncode does not do any modification of the name ever
- Any method that sets or updates parameters (AppParameters or any field afterward) triggers updating of the digest name component. We may do it multiple times, but I prefer paying this price compared to modifying memory in const methods. These methods should do throwing similar to what described in comment #35)

#39 - 06/29/2019 10:46 AM - Davide Pesavento

> With the exception that we make a requirement/assumption that not having parameters implies params-sha256=0000000000000000000000000000000000000000000000000000000000000000

I don't understand this sentence. Do you mean that if there are parameters (or will be added later), the user can set a name *without* params-sha256= component and we automatically append one? (that would be my preference I think)

> We may do it multiple times, but I prefer paying this price compared to modifying memory in const methods.

Yes, this is the intrinsic tradeoff of eager vs lazy computation of the digest. I have no preference either way. With the current Interest format, recalculation would only happen in rare or contrived cases, so I'm not worried. I will adopt your approach.

#40 - 06/30/2019 09:47 PM - Davide Pesavento

- Status changed from In Progress to Code review

- % Done changed from 50 to 80

https://gerrit.named-data.net/c/ndn-cxx/+/5457

#41 - 07/01/2019 08:10 AM - Alex Afanasyev

Davide Pesavento wrote:

> > With the exception that we make a requirement/assumption that not having parameters implies params-sha256=0000000000000000000000000000000000000000000000000000000000000000
>
> I don't understand this sentence. Do you mean that if there are parameters (or will be added later), the user can set a name *without* params-sha256= component and we automatically append one? (that would be my preference I think)

No. I meant that if app plans parameters, it must add a placeholder params component (with params-sha256=0000... value).

#42 - 07/01/2019 12:37 PM - Davide Pesavento

Oh. I kinda disagree with that actually. It seems slightly simpler, from the app point of view, to have the Interest class append a ParametersSha256DigestComponent to the name as needed. Of course if the app protocol requires the digest component to be in a specific position within the name, the app can put a placeholder component at the required position. Same for removing the component automatically in unsetApplicationParameters(). Otherwise the app has to do the following, which is unnecessarily verbose and repetitive:

```
interest.unsetApplicationParameters();
auto name = interest.getName();
// find and erase the ParametersSha256DigestComponent from name (we don't even have the API for this in Name)
interest.setName(name);
```

Moreover, an all-zero SHA256 hash value is not special, it's just one of many possible outputs. So it would be quite ugly to designate params-sha256=00000... as a special placeholder component.

#43 - 07/01/2019 12:46 PM - Davide Pesavento

Davide Pesavento wrote:

Of course if the app protocol requires the digest component to be in a specific position within the name, the app can put a placeholder component at the required position.

And it would be good to have a nicer API for this, instead of writing long and ugly names with .../params-sha256=... in the code. But this is minor.

#44 - 07/13/2019 10:49 AM - Davide Pesavento

- Status changed from Code review to Closed

- % Done changed from 80 to 100