**From:** Medlin, Christina <[Christina.Medlin@gtri.gatech.edu](mailto:Christina.Medlin@gtri.gatech.edu)>  
**Sent:** Wednesday, April 16, 2025 2:39 PM  
**To:** Escobar, Katherine B CIV JS J6 (USA) <[katherine.b.escobar.civ@mail.mil](mailto:katherine.b.escobar.civ@mail.mil)>; Sullivan, Stephen M CTR JS J6 (USA) <[stephen.m.sullivan14.ctr@mail.mil](mailto:stephen.m.sullivan14.ctr@mail.mil)>  
**Subject:** Notes for the NBAC-NTAC meeting

Hi,

Something else to add to your list of tasks that GTRI does for model management is providing modeling and technical support (to committees, domains, users).

I think these links are in the previous status slides but to make sure you have them:

* Tools issue tracker for NIEM API 2.0 and NIEM Toolbox issues:  [https://github.com/orgs/niemopen/projects/4](https://urldefense.com/v3/__https:/github.com/orgs/niemopen/projects/4__;!!May37g!MyEWfKYLJDIZ2_I4jcuUBsqKBflFKP-j8xUsSIBfgttNWz9xkPmcNBo-Jby1CH9chOEyQMbkRYZhDu8SbFNaUQpkLLjyBloD$)  (Kelly updated the settings so it’s public now)
* Dev branch for API code: [https://github.com/niemopen/niem-api/tree/dev](https://urldefense.com/v3/__https:/github.com/niemopen/niem-api/tree/dev__;!!May37g!MyEWfKYLJDIZ2_I4jcuUBsqKBflFKP-j8xUsSIBfgttNWz9xkPmcNBo-Jby1CH9chOEyQMbkRYZhDu8SbFNaUQpkLIrI5vV4$)
* OpenAPI file describing the API for developers: [https://tools.niem.gov/api/v2/swagger-ui/index.html](https://urldefense.com/v3/__https:/tools.niem.gov/api/v2/swagger-ui/index.html__;!!May37g!MyEWfKYLJDIZ2_I4jcuUBsqKBflFKP-j8xUsSIBfgttNWz9xkPmcNBo-Jby1CH9chOEyQMbkRYZhDu8SbFNaUQpkLGmlqc3w$)
* Export of the content for the new API database (NIEM 1.0 – 6.0 data): [https://github.com/niemopen/niem-api-db](https://urldefense.com/v3/__https:/github.com/niemopen/niem-api-db__;!!May37g!MyEWfKYLJDIZ2_I4jcuUBsqKBflFKP-j8xUsSIBfgttNWz9xkPmcNBo-Jby1CH9chOEyQMbkRYZhDu8SbFNaUQpkLDWqJAQB$)
* Dev branch for NIEM Toolbox code:  [https://github.com/niemopen/niem-toolbox/tree/dev](https://urldefense.com/v3/__https:/github.com/niemopen/niem-toolbox/tree/dev__;!!May37g!MyEWfKYLJDIZ2_I4jcuUBsqKBflFKP-j8xUsSIBfgttNWz9xkPmcNBo-Jby1CH9chOEyQMbkRYZhDu8SbFNaUQpkLM0I-o0W$)
* Website for NIEM Toolbox:  [https://niemopen.github.io/niem-toolbox/](https://urldefense.com/v3/__https:/niemopen.github.io/niem-toolbox/__;!!May37g!MyEWfKYLJDIZ2_I4jcuUBsqKBflFKP-j8xUsSIBfgttNWz9xkPmcNBo-Jby1CH9chOEyQMbkRYZhDu8SbFNaUQpkLMuYSt0K$)

Below are notes that I copied over that I quickly wrote up for today’s topics (may have gaps) in case they are helpful for you.

--Christina

**Harmonization**

Sources

1. Someone from the domains, NBAC, NTAC, or user community reports an issue

2. The NBAC / lead developer reviews the model for overlap during major versions

* Look for similar types across difference namespaces
* Look at the content in extensions and augmentations of types across different domains
  + e.g., everything built off of PersonType across the model
* Look at different properties that share the same terms.
  + e.g., All of the properties that have location or address in the name.  Or city.  Or schedule.  And so on.
* Try to make the model more consistent
  + e.g., We have a boolean type.  We try to use that in the model consistently vs adding custom code sets here and there with yes/no enumerations.
  + Not always possible due to requirements and existing standards though

Basic steps

* Identify potential issues from sources listed above
* Log them into the model issue tracker
* Prepare background material and potential options for addressing the issues
* Review the issue in a Harmonization Subcommittee meeting
* Prepare the change request
* Approve a recommendation
* Present the full NBAC with the recommendation for approval
* Load the changes into the model
* Generate an updated draft of the schemas
* Commit the changes to the dev branch
* Review the schema changes in the Harmonization Subcommittee to confirm

**Content updates**

Current process

Currently, I have basic scripts to

* Initialize new versions of NIEM for the legacy database and tools
* Run QA on change request spreadsheets
* Load changes from change request spreadsheets into the database
* Load XML schemas into the database
* Run QA on the full model
  + It checks for some conformance issues but relies on running NDR Schematron rules for full conformance checks
  + Also checks for optional things that won't be flagged as errors but might be oversights (e.g., abstract elements without substitutions, empty types)
* Generate
  + XML schemas
  + Documentation spreadsheets
  + CSV files
  + Change logs
  + Release stats

Future plans

Future plans for the API after this current phase of providing SSGT and other legacy functionality for users is to build in this model management capability into the new tooling so other people can access it as well.

Submit changes via schemas:

* Users can already submit pull requests in GitHub to change model schemas now and could continue to do so
* Still need to have some model maintainers to be able to look over the pull request and approve or reject it
  + Make sure the updated schema passes conformance checks (possible automation via GitHub Actions)
  + General review to make sure it's not making unapproved changes to Core or other domains, etc.
  + Model maintainer would also push changes to the new tools
* The new NIEM tools will still need to support loading schemas in order for GitHub changes to propagate to them
  + Needs some special handling to maintain dependencies across different namespaces and migration rules
  + Add API support to generate diffs between current and proposed content so contributors can provide migration rules for component renames or refactoring

Submit changes via change request spreadsheet:

* QA - Add change request spreadsheet validation to the api and NIEM Toolbox
  + Lets users or other model maintainers iterate over automated QA tests and updates without having to go through me
* Load content - Add tool functionality to
  + Load a change request spreadsheet to generate updated schemas
  + Generate migration rule updates

General tasks for updating the model (via schema or CR spreadsheet):

* Update the readme with change summaries as you go along (manual)
* Update the model specification document (manual)
* Update the new tools database
* Generate schemas
* Generate JSON-LD context file
* Generate documentation spreadsheets and CSVs
* Generate the change log
* Generate release stats

**Major versions vs minor versions**

Content

* For the content side of tooling, major and minor version functionality is very similar
* We load content into the database via schemas and change request spreadsheets
* We generate schemas and documentation
* Whether or not we update the Core namespace is a content issue, not a tool issue

Modeler support

* The amount of support needed for modeling is always an unknown, major or minor
* Sometimes you have to get new people up to speed on how to model NIEM and work
* Sometimes there are large sets of new requirements, sometimes it’s smaller updates of existing content
* Sometimes contributors are experienced with NIEM and sometimes they need a lot of help

Major releases

Process-wise, for major releases vs minor releases

* We also update Core
* We do more harmonization work
* We do more code table updates

Architecture

* Major versions introduce structural changes
* There may be new conformance rules requiring content changes on the NBAC side
* There may be architecture changes that require changes to the database and to code that runs QA and loads content and generates schemas and other artifacts
* May be limited to resources available to update tools and training

**Release cycle**

* Predictability is good for the community
* If Core is locked during minor versions, it's good not to go too long before updates
* If Core isn't completely locked during minor versions, maybe we can go longer between majors
* Try for lighter-weight annual releases for the time being with tool support listed above?

**Support for frequently changing artifacts**

* Recommend we start putting out early and frequent drafts of the next version
  + So in this case, once 6.0 PS02 gets approved and finalized, we initialize 6.1
  + Then we load the latest NCIC code table updates and the updates that are ready from MilOps and Biometrics
  + Tag that as some milestone.  Alpha 1 or OASIS release 1 or a working draft or whatever.
  + Also put the draft milestone in the schema version attribute?
* We keep doing that as changes come in until we are ready for 6.1 PSD 01, especially for content that is being waited on.
* This makes ready changes available in draft form for whoever needs to use it now, with the understanding that it may change in PSD 01.
* The latest draft would be available in the new tools for search and subsets vs the current single update at the end of the process.
  + The new API support for loading schema and change request updates can make this draft content available from the tools as soon as they are loaded.
* Better community support.
  + This would let the FBI do quarterly code updates.
  + This would let MilOps move forward and others do more frequent updates as well.