ROS Node Configuration - Final2 Proposal #3325

Closed Locked **kaspermeck-arm** started this conversation in **Design**

kaspermeck-arm on Mar 8, 2023 Collaborator edited

The Open AD Kit WG is working on improving the process of how to configure the ROS node parameters. Gathered from discussions and meetings the following Final2 Proposal has been made. Thanks!

DevOps Dojo: ROS Node Configuration - Final2 Proposal

Background

DevOps: Ros Node Configuration addresses the topic of how ROS nodes are configured. Guidelines, documentation and changes to the parameter files and ROS nodes will be made, based on best-practices from cloud-native and software-defined development methodologies.

How to configure ROS nodes is non-differentiating, and creating alignment in the AWF community will allow developers and users of Autoware to become more productive as less time will be spent on trivial tasks.

ROS parameter definition file

The ROS parameter definition file layout which will be adopted in Autoware is inspired by PickNikRobotics generate parameter library. Please note, the library as a whole will not be adopted as this is quite an invasive change. At the same time, by adopting their parameter file layout, no doors are being closed.

Each ROS node has a single-source parameter description file which avoids the uncertainty of where a parameter is declared. The parameter description file will:

- generate the ROS configuration file into `config` folder during build
  - also copied to Autoware Launch repository

Category

Design

Labels

None yet

5 participants

- be rendered as a table to be used in the web documentation, similiar to NDT Scan Matcher

There needs to be a 1-to-1 match between declarations and parameters. All parameters listed in ROS configuration file must be declared in the ROS node, and no parameters may be listed in the parameter file which aren't declared in the ROS node.

Naming convention and file path

- parameter definition file path: *autoware.universe/.../src/*.param.def.yaml*
    - "..." is the feature/function and package
    - "*" is the name of the node
- autoware.universe/localization/ndt_scan_matcher/src/ is using the correct naming convention and file path
    - the feature/function is *localization*
    - the package is *ndt_scan_matcher*
    - the name of the node is *ndt_scan_matcher*

Attributes

All parameters have the following attributes:

- name
- type
    - see ParameterType for allowed types
- default_value
- description
- validation
    - only when applicable
    - see Validators for possible validators

These attributes should be populated for all parameters in the parameter file.

Parameter types

The parameter types handled by `declare_parameter` origin from the definitions in ParameterValue.msg. Those map as:

ParameterType enum	C++ type
`PARAMETER_BOOL`	`bool`
`PARAMETER_INTEGER`	`int64_t`
`PARAMETER_DOUBLE`	`double`
`PARAMETER_STRING`	`std::string`
`PARAMETER_BYTE_ARRAY`	`std::vector<unsigned char>`
`PARAMETER_BOOL_ARRAY`	`std::vector<bool>`
`PARAMETER_INTEGER_ARRAY`	`std::vector<int64_t>`

ParameterType enum	C++ type
`PARAMETER_DOUBLE_ARRAY`	`std::vector<double>`
`PARAMETER_STRING_ARRAY`	`std::vector<std::string>`

Layout

We will use lidar_apollo_segmentation_tvm_nodes as an example and as a base for our modifications. In addition to the attributes listed in the previous section, the parameter file should be version-controlled.

```
lidar_apollo_segmentation_tvm_nodes:
  ros_parameters:
    range: {
      type: int64_t,
      default_value: 90,
      description: "The range of the 2D grid with respect to the or
      validation: {
        bounds<>: [MIN_VALUE, MAX_VALUE]
      }
    }
  ...
```

To see the changes made, view the original test.param.yaml #L4. Note that only *range* has been moved to the new format and that the proper file name would be *lidar_apollo_segmentation_tvm.param.yaml*.

Generated parameter file

The parameter description file will be used to generate the parameter file. The layout below will be used, which is the same layout which is currently used, e.g., see Lidar Apollo Segmentation TVM Nodes Parameter File.

```
/**:
  ros__parameters:
    range: 90
    ...
```

Please note that this parameter file is automatically generated and only contains `name` and its corresponding `default_value`. The parameter file path is:

- parameter file path: *autoware.universe/.../config/*.param.yaml*
  - "..." is the feature/function and package
  - "*" is the name of the node
- autoware.universe/localization/ndt_scan_matcher/config/ndt_scan_matcher.param.yaml is using the correct naming convention and file path
  - the feature/function is *localization*
  - the package is *ndt_scan_matcher*
  - the name of the node is *ndt_scan_matcher*

ROS node declare parameter function

The new parameter file layout requires minor modifications to how *declare_parameter(...)* is used today. If declare_parameter(...) has no *default_value*. It throws an exception, which is desirable as it enforces the parameter file to contain the declared parameter.

We'll be using lidar_apollo_segmentation_tvm_node.cpp #L43 as the example to show the required change, which can be used in the code with the following pattern:

```
declare_parameter<int64_t>("range");
```

For clarity, it is important to stick to using only one of those predefined types in the template. Although using a different type compiles just fine (for example, `rclcpp` correctly infers that `int32_t` should map to `PARAMETER_INTEGER`, but the returned value is still an `int64_t`), it is misleading and could lead developers to make assumptions that result in unexpected runtime behaviors.

Edit: Added link to https://github.com/PickNikRobotics/generate_parameter_library#built-in-validators for possible validators

↑ 1 👍 1

5 comments · 41 replies

Oldest Newest Top

kenji-miyake on Mar 9, 2023

@kasperornmeck Thank you for summarizing it.
It generally looks good to me!

I have one small favor for you.
Although code generation isn't in the scope yet, related to the purpose "allow developers and users of Autoware to become more productive as less time will be spent on trivial tasks." that you mentioned, there is a possibility that TIER IV will introduce code generation by ourselves as another activity aside from the Open AD Kit WG.
(Open AD Kit doesn't need to work on the task.)

This is because any improvement should be allowed as long as it doesn't interfere with the freedom of Open AD Kit WG and any others.

I ask for your understanding of that. 🙇
Thank you!

↑ 2 👍 2 2 replies

kaspermeck-arm on Mar 9, 2023 Collaborator Author

@kenji-miyake

I don't see any issues with further improving the parameter configuration process through code generation. Is it the PickNikRobotic's code generation library you want to introduce? (If that's the case, the work done in this DevOps Dojo would be a stepping stone)

👍 1

kenji-miyake on Mar 9, 2023

> Is it the PickNikRobotic's code generation library you want to introduce?

Yes. I've tried it and felt it is almost a perfect solution for us.

👍 1

doganulus on Mar 9, 2023 Collaborator edited

I also carry my suggestion using JSON Schema here under this proposal.

If we use JSON Schema rather than modified Picknik format, we can also use a large set of tooling without much effort:

- Editor support such as https://marketplace.visualstudio.com/items?itemName=redhat.vscode-yaml
- CI actions such as (https://github.com/marketplace/actions/yaml-file-validation-using-a-remote-json-schema
- Many validator implementations in different languages such as https://pypi.org/project/jsonschema/

Those are all excellent practices for cloud-native development. Using a well-adopted format and suggesting it to the ROS community would be better for the cloud-native development of robotics, which aligns with Open AD Kit objectives. I think this is what we are interested in under this working group.

↑ 1 19 replies

Show 14 previous replies

doganulus on Mar 11, 2023 Collaborator

@kenji-miyake Thank you for confirming my post above. If the Open AD Kit and Autoware have a goal to reach different communities and organizations outside of the ROS community, the Autoware community must endure and address similar criticism that I gave above. My choice of words was careful and addressed the tooling technology, which could be wasteful, objectively bad, and archaic. Tooling technology has no feelings. So I don't see any problem with those words. Again thank you for your understanding.

I still don't understand why criticism of tooling is taken so personally. After all, this is a working group to discuss cloud-native tooling. If we make everything sugar-coated and pursue incremental changes here, then again what is the purpose of the group? Things I wish might not happen immediately, or there might better alternatives, or we cannot have the resources to implement those. But I am not happy with the direction. It does not go towards the goal of cloud-native development and testing.

Finally, a simple configuration file discussion should not be that long but I want to ensure the long-term goals of the working group. And I let you know my honest thinking and intention.

mitsudome-r on Mar 11, 2023 Maintainer edited

If everyone's okay with this proposal, I think Kapser and others could start defining the parameter definitions using JSON schema. (We still don't forbid to use PickNik library though)

doganulus on Mar 11, 2023 Collaborator

I think I must go first after all the talk. Starting a new comment below.

kenji-miyake on Mar 11, 2023

> So I don't see any problem with those words.

@doganulus If you think so, it's fine.
The last couple of things that I want to tell you are the following:

- The way people receive words is different from each other.
- In order to accomplish a big goal, we need to find mates with a common purpose.

Therefore, if you want some help or agreement, you must try to understand other people well. Otherwise, you'll get objections or even be ignored.

I really don't feel the way you communicate with other people goes well.
However, if it's okay for you, it's your liberty unless you violate the rules of the community.

Although here I'll stop convincing you or having discussions with you because I have no obligation to them, I hope for your success in the future.
Thank you.

doganulus on Mar 11, 2023 Collaborator edited

@kenji-miyake We will be friends later, you will see better later why I do not want to give any compromise in complex system design. If you have time, read the great speech by my favorite scientist, Claude Shannon.

doganulus on Mar 11, 2023 Collaborator edited

There are a few style details for writing schemas.

Please find an example param+schema configuration for `ndt_scan_matcher` node here:
https://github.com/doganulus/autoware-schema/tree/main/localization/ndt_scan_matcher/config

Some style choices I have made:

- Write the node name `ndt_scan_matcher` explicitly instead of `/**`. Problem? Explicit is better?
- Use schema version `draft-07`. This is the most recent one that has a large tooling support.
- Use schema definitions and references to avoid the nesting caused by `namespace` and `ros__parameters`.
- Not set `additionalProperties: false` to restrict extra keys. Shall we set?
- Test `redhat.vscode-yaml` extension for VSCode. This extension adds validation support in the editor, parameter explanation based on parameter descriptions, and code completion based on parameter names and default values.

↑ 1 11 replies

Show 6 previous replies

doganulus on Mar 23, 2023 Collaborator

@xmfcx The mapping between the YAML file and its schema has been done explicitly in the `.vscode` settings as here. The names may be different but it is good to keep them the same, I think.

```
{
    "yaml.schemas": {
        "./localization/ndt_scan_matcher/config/ndt_scan_matcl
            "ndt_scan_matcher.param.yaml"
        ],
    }
}
```

The tooling supports remote locations as well as local system locations. In the future, Autoware Foundation may want to publish these schemas online and we can refer to their remote locations as `https://autoware.org/schemas/core/v1/localization/ndt_scan_matcher.schema.json`. That would be a good feature to enhance the cloud-native claim of Autoware.

Also, I will not do it but you may want to suggest PickNik to generate their `cpp` files from schemas. That way it feels more natural. They would not need to maintain a custom format when there is already a standard format if they want to generate code. Being slightly less verbose doesn't justify having a competing format in my eyes.

doganulus on Mar 23, 2023 Collaborator edited

@kasperornmeck There is an active proposal to standardize the extensions `.json` and `.schema.json` for JSON Schema documents (both allowed). Not using any extension is also common but I don't think we would go for that.

If you want to read the proposal, it is here, see Section 2.2.5.

kaspermeck-arm on Mar 23, 2023 Collaborator Author

@doganulus
I think I understand where the confusion lays... from my experience, a schema is used to validate a data file to ensure that, e.g., bounds are met. Is it common practice to define the actual value for the parameter in the JSON Schema?
See https://restfulapi.net/json-schema/ under "JSON Schema Validation Example" as an example.

kaspermeck-arm on Mar 23, 2023 Collaborator Author

@doganulus
I followed this guide https://medium.com/@joshuaavalon/intellisense-for-json-and-yaml-in-vs-code-f626fc733426 to understand how JSON Schema works in VSC. I added the following in my `settings.json` file found through `Ctrl + ,` under `JSON: Schemas`. I added these lines (replacing USER):

```
"json.schemas": [
  {
    "fileMatch": ["data.json"],
    "url": "/home/USER/fun/project/address.schema.json"
  }
]
```

and created `data.json`. Hovering, autocomplete, etc works great!

As we'll have a unique schema for each ROS node, there will be a lot of schemas (100+). Do you have any idea how to do this at scale? Do we need 100+ entries in the `settings.json` file like the one I added above?

doganulus on Mar 24, 2023 Collaborator

> Yes, the original proposal has proposed to include actual parameters and definition (description/validation) fields together. This isn't a common practice but we could enable online validation (now discarded under this work). Once TierIV suggested separating parameter values and parameter definition, then things exactly turned into the common practice of schemas and offline validation. So we could apply JSON schema without much effort thanks to the existing tooling.
>
> Another confusion may stem from the fact that JSON Schema documents are valid JSON documents. So we can write a JSON schema for JSON Schema versions, which is known as meta-schemas. The versions, or dialects, like `draft-07` are meta schemas. And `$schema` field represents the dialect of the schema document.
>
> Regarding scalability, each ROS node is different so there is an inherent complexity to match parameter files and schemas here. We may leave it to the developer Dojo. But, given that these declarations wouldn't change often, I don't find having 100+ declarations unmanageable in settings. Alternatively, it may be possible to merge all these schemas to make one giant schema and publish it online together with releases. Perhaps this could be done only for the core repository and only for users.

kaspermeck-arm on Mar 22, 2023 Collaborator Author

@kenji-miyake @mitsudome-r @doganulus

I've been catching up on the discussion and I thank everyone involved for their engagement!

Below I've listed the scope for the parameter definition file.

- the parameter definition file **will** be used for the following:
  - generate the parameter file
  - markdown table for web documentation
- the parameter definition file **might** be used for the following:
  - parameter validation
  - code generation

I think the focus when determining the file format should be on how suitable it is in the context in which it'll be used. The ROS community uses `python`, `XML` and `YAML` formats for node configuration/launch, so the community is already familiar with the `YAML` format, see Launch-file-different-formats. Using the `YAML` format leaves the door open for code generation. It is because of these two reasons which I am suggesting use the `YAML` format.

I would like for us to move on to the execution phase and start working on other *dojos*, we have lots of fun tasks to get done! I hope and wish that we can come to an agreement soon.

↑ 1 👍 1 9 replies

Show 4 previous replies

kenji-miyake on Mar 24, 2023

@xmfcx Thank you for the summary. I totally agree with your proposal.

❤️ 1

doganulus on Mar 24, 2023 Collaborator

@xmfcx @kenji-miyake I also agree with this summary. Thanks for your efforts!

❤️ 1

kaspermeck-arm on Mar 24, 2023 Collaborator Author

@xmfcx @kenji-miyake @doganulus
Thanks for the discussion and capturing the suggested changes.

I think there is only one aspect left to decide on. Should the JSON Schema also act as the JSON document? I.e., do we want to include the `default_value` for each parameter in the JSON Schema file? Discussion points:

- if we don't include a `default_value`, we cannot generate the `node.param.yaml` file from the `node.schema.json` file
- common definition (https://www.mongodb.com/basics/json-schema-examples) of a schema
    - > JSON Schema is a model that represents the format and structure of a common group of JSON documents.
- pros/cons with separation of schema and document; what's best practice?

I will create a Final3 proposal to ensure we are aligned once we've determined how we wish to do with the schema/document topic.

doganulus on Mar 26, 2023 Collaborator

> Do we want to include the default value for each parameter in the JSON Schema file? If we don't include a default_value, we cannot generate the `node.param.yaml` file from the `node.schema.json` file

We may not need to. Generation can be done based on types if `default` is missing in the schema. Say, booleans get `false`, numbers get zero, and strings/arrays/objects get the empty. Alternatively, just generate all with `null` values. That won't be valid, of course.

> common definition of a schema, pros/cons with separation of schema and document; what's best practice?

I would prefer the word *specification* here. It is the specification of parameter files. Technically it falls under the category of formal specifications, which is considered as a good practice for safety-critical system design (see ISO 26262 for example). Separating *specification* and *verification* is another good practice in general.

kaspermeck-arm on Mar 27, 2023 Collaborator Author

> > Do we want to include the default value for each parameter in the JSON Schema file? If we don't include a default_value, we cannot generate the `node.param.yaml` file from the `node.schema.json` file
>
> We may not need to. Generation can be done based on types if `default` is missing in the schema. Say, booleans get `false`, numbers get zero, and strings/arrays/objects get the empty. Alternatively, just generate all with `null` values. That won't be valid, of course.

I think the default value should be a valid and tested value for the ROS node.

> > common definition of a schema, pros/cons with separation of schema and document; what's best practice?
>
> I would prefer the word *specification* here. It is the specification of parameter files. Technically it falls under the category of formal specifications, which is considered as a good practice for safety-critical system design (see ISO 26262 for example). Separating *specification* and *verification* is another good practice in general.

In Final3 Proposal I'll add terminology to align the community. When you say to separate *specification* from *validation*, does that mean we shouldn't include, e.g., `exclusiveMinimum` from Draft 07 Schema in the JSON Schema specification?

kaspermeck-arm on Mar 27, 2023 Collaborator Author

New proposal found here: https://github.com/orgs/autowarefoundation/discussions/3371.

↑ 1 0 replies