ROS Node Configuration - Final Proposal #3288

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

kaspermeck-arm on Feb 24, 2023 Collaborator edited

The Open AD Kit WG is working on improving the process of improving how to configure the ROS node parameters. Gathered from discussions and meetings the following final proposal has been made. We will discuss this in the Open AD Kit meeting on Wed March 1st, so if you cannot attend, please comment here. Thanks!

DevOps Dojo: ROS Node Configuration - Final 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 file

The ROS parameter 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 file which avoids the uncertainty of where a parameter is declared. The parameter file will:

- configure the ROS nodes
- be copied to Autoware Launch repository
- 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.

Category

Design

Labels

None yet

6 participants

Naming convention and file path

- 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*

Attributes

All parameters have the following attributes:

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

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

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.

```
/**:
  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*.

ROS node declare parameter

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.

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>`
`PARAMETER_DOUBLE_ARRAY`	`std::vector<double>`
`PARAMETER_STRING_ARRAY`	`std::vector<std::string>`

Which can be used in the code with the following pattern:

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

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.

Notice that we need to add `.defalt_value` in the `declare_parameter(...)` function in order to match the new YAML layout of the parameter file.

Edit: `bounds` replaced with `validation` under parameter attributes.
Edit 2: `version: '1.0'` removed from the parameter file layout, as it isn't compatible with launching the node.
Edit 3: `int` replaced with `int64_t` in example parameter file to be comply with the allowed parameter types.

↑ 4 👍 3

7 comments · 24 replies

Oldest Newest Top

kenji-miyake on Feb 27, 2023

> parameter file path: autoware.universe/.../config/*.param.yaml

There is a related discussion:
https://github.com/orgs/autowarefoundation/discussions/3281
It suggests changing the directory name from `config` to `default_config` or something.

↑ 1 2 replies

kaspermeck-arm on Feb 28, 2023 Collaborator Author

@kenji-miyake

I think having `default` in the name of the parameter file would make more sense, as there might be multiple parameter configurations for a single node, i.e., more than one parameter file. Adding `default` would make the parameter file name even longer and I don't know how much value this would actually add. An alternative is that the parameter file which has the name of the node is the default one.

kenji-miyake on Feb 28, 2023

@kasperornmeck Thank you!
Would it be possible for you to write the comment also in the linked discussion thread, with some examples like the format of https://github.com/orgs/autowarefoundation/discussions/3281#discussioncomment-5072139?

kenji-miyake on Feb 27, 2023

> Please note, the library as a whole will not be adopted as this is quite an invasive change.

@kasperornmeck Do you mean like this by this sentence? In that case, it may be a good point of compromise.

↑ 1 👀 1 10 replies

Show 5 previous replies

kenji-miyake on Mar 1, 2023 edited

Ah, you just meant like removing default values from `declare_parameter(...)`? If so, I misunderstood that you meant adding modifications to the internal implementation of `declare_parameter`.
But in that case, you need to consider how to load parameters from the PickNik's format file. I believe it's not supported yet. So you need either offline parameter conversion or extending the launch action.

mitsudome-r on Mar 1, 2023 Maintainer

@kenji-miyake We had extra discussion in Open AD Kit.

> But in that case, you need to consider how to load parameters from the PickNik's format file. I believe it's not supported yet. So you need either offline parameter conversion or extending the launch action.

I think you have some misunderstanding.
Kasper's intention is not to use Picniks' format directly, but to use a similar format which can be consumed as normal ROS parameter as shown in the "Layout" section in the proposal. However, only `default_value` is actually read from `declare_parameter` and "description", "type", and "validation" would be ignored. He doesn't want to make it a comment but as a readable value in YAML format so that we can also use it to generate a table in README.md

mitsudome-r on Mar 1, 2023 Maintainer

I personally feel uncomfortable to add parameters that won't be used in a ROS node, but it is less intrusive than make modification to declare_parameter() function or forcing a dependency to external package.

kenji-miyake on Mar 1, 2023 edited

@mitsudome-r I see. Thank you for the additional information. It's clear to me now.
If it works correctly, it seems to be okay. But I'm curious about how/when to validate the parameters. I'd like to discuss it at a later date. (Probably it's during generating the documentation?)

kaspermeck-arm on Mar 2, 2023 Collaborator Author

@mitsudome-r **@kenji-miyake**

There is a full section on validation from PickNikRobotic's found here: https://github.com/PickNikRobotics/generate_parameter_library#parameter-definition

I have updated the proposal to exchange `bounds` with `validation`. After talking to **@ambroise-arm** today, not all parameters require validation, e.g., the range of `int64_t` could be bounds for a parameter and this would then be determined by the `type`.

kenji-miyake on Mar 2, 2023

@kasperornmeck With the information from **@mitsudome-r** in https://github.com/orgs/autowarefoundation/discussions/3288#discussioncomment-5168427, I was able to clear my misunderstanding.

However, I have several concerns then.

1. Parameters in autoware_launch aren't validated.

I understood that your proposal only loads the parameters online, not validates them.
Regarding the parameter files in `autoware.universe`, we can validate them, for example during automatic document generation.
But it's difficult to validate parameter files in `autoware_launch` because it's located in a different repository.

It can be permissible for a while, but in the future, we need some validations to ensure system quality.
For example, online validation, or an external tool that compares the parameter file in `autoware_launch` with the parameter definition in `autoware.universe`.

2. Parameters in autoware_launch contain unnecessary information.

When we copy parameter files from `autoware.universe` to `autoware_launch`, I think the fields other than `default_value` are redundant.

```
version: '1.0' // Not used
ros__parameters:
  range: {
    type: int, // Not used
    default_value: 90,
    description: "The range of the 2D grid with respect to the or
    validation: {
      bounds<>: [MIN_VALUE, MAX_VALUE] // Not used
    }
  }
```

If you write as follows, it's not so redundant but a little weird.
Also, it would be difficult to maintain the parameter files because you need to modify the content after copying the file. (But this can be automated if we develop a tool.)

```
ros__parameters:
  range: {
    default_value: 90,
  }
```

Therefore, we'd like to consider other alternatives as well.
TIER IV will discuss it next week and write ideas here if we could come up with any good ideas.

↑ 1 3 replies

ambroise-arm on Mar 2, 2023 Collaborator

> in the future, we need some validations to ensure system quality.

Exactly, but this is only a proposal for the first step of reworking the configuration files. Having the validation would be a second step. By having the same yaml format as the one used by PickNikRobotics, we don't close the door to using it in the future if we want to. Or if we don't want to, we can develop our own checker, but the fundamental information would still be there, ready to be used.

> I think the fields other than `default_value` are redundant

The fields other than `default_value` are not used in the Autoware code **at the moment**, but they provide information that resides nowhere else. So they are not redundant. Except for the type, which could be inferred from the default_value.

I personally don't see a problem with having information in the yaml file that is not used in the code, as it is still valuable information for the user. But I understand the concern.

👍 4

doganulus on Mar 6, 2023 Collaborator edited

Parameter files discussed here and the proposal are conceptually more general than the currently used ones. This is the main pain point and misunderstanding between the AWSG and OpenADKit groups.

This is a declarative *specification* of your nodes and their parameters. And it is desired to make it executable by allowing it to be loaded at launch as well as generating documentation. Therefore, the concept we talk about here is an *executable specification.* Parameter validation is another application that uses this specification. We also need this (default) specification to be overridden at upper levels as many modern software systems implement. See this is a powerful concept and relatively easy to adopt.

On the other hand, the current concept only deals with loading parameter values from an external source. Hence, it may seem redundant if you look at only that angle.

And do not be confused by both concepts having been implemented by yaml files. Previously you had a single use case and writing yaml files for that. Now you write a yaml specification and use it for multiple use cases.

Hope I was able to clarify.

kaspermeck-arm on Mar 7, 2023 Collaborator Author

@doganulus

Can you come with a concrete example of what you think the parameter file should look like?

kenji-miyake on Mar 7, 2023

@kasperornmeck cc **@mitsudome-r** **@xmfcx**
TIER IV has discussed this today, and the result was that there might be a better alternative solution for this purpose. I'll explain it below.

First, I think your primary purposes are the following two.

1. Automatically generate documentation from the parameter definition file.
2. Remove default_value from declare_parameter, and ensure every parameter is listed in the parameter file.

For 1, it's enough if we put a parameter definition file aside from the existing normal parameter files.
Regarding the consistency between the parameter file and its definition, we can develop a simple tool if necessary. (But ideally, it would be better to be validated online.)

For 2, it can also be done with other solutions such as autowarefoundation/autoware.universe#2945 or using PickNik's library.

Therefore, we'd like to propose another approach: start just newly placing PickNik's parameter definition files. (No need to use the generated parameter library code.)

The approach has several advantages.

- There is no change to the existing parameter files; we can reuse them. (No need to add "default_value" to all files.)
  - Actually, changing the parameter format has a big impact on third-party tools that depend on the current format. (for example, a parameter management/distribution service.)
- Those who want to use PickNik's library can use it easily.

Of course, it's possible to generate documentation from the definition files if we develop a tool.

That's the overview of our proposal.
Since it's a little complicated, we'd like to explain this to you in more detail interactively. How about discussing it on a meeting of Open AD Kit WG or another separate meeting?
Thank you!

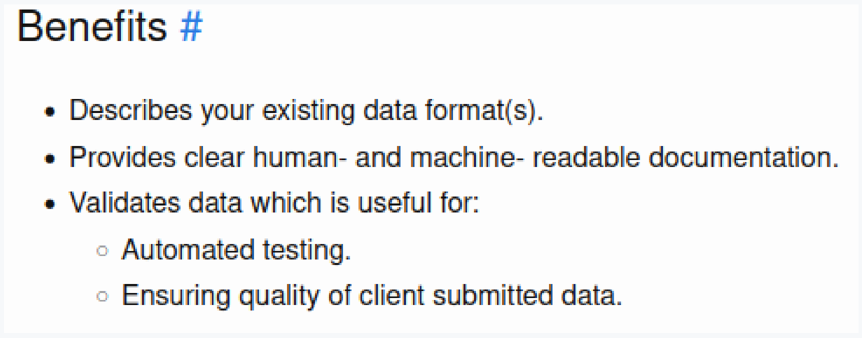
↑ 1 👍 1 7 replies

Show 2 previous replies

kaspermeck-arm on Mar 7, 2023 Collaborator Author

@kenji-miyake, I'm not quite sure what you are suggesting. Do you want to use `JSON` format instead of `yaml` format?

From https://json-schema.org/#benefits



> ## Benefits #
>
> - Describes your existing data format(s).
> - Provides clear human- and machine- readable documentation.
> - Validates data which is useful for:
>   - Automated testing.
>   - Ensuring quality of client submitted data.

Which aligns with what we're trying to achieve.

```
/**:
  ros__parameters:
    range: {
      type: int64_t,
      default_value: 90,
      description: "The range of the 2D grid with respect
to the origin",
      validation: {
        bounds<>: [MIN_VALUE, MAX_VALUE]
      }
    }
```

In attempt to make this into a `JSON` schema.

```
"type": "object",
  "properties": {
    "type": {"type", "string"},
    "default_value": {"type", "number"},
    "description": {"type": "string"},
    "validation": {"type": "array"}
  }
```

Is this what you mean? If not, can you give a concrete example?

From https://json-schema.org/understanding-json-schema/about.html:

> You may have noticed that the JSON Schema itself is written in JSON. It is data itself, not a computer program. It's just a declarative format for "describing the structure of other data". This is both its strength and its weakness (which it shares with other similar schema languages). It is easy to concisely describe the surface structure of data, and automate validating data against it. However, since a JSON Schema can't contain arbitrary code, there are certain constraints on the relationships between data elements that can't be expressed. Any "validation tool" for a sufficiently complex data format, therefore, will likely have two phases of validation: one at the schema (or structural) level, and one at the semantic level. The latter check will likely need to be implemented using a more general-purpose programming language.

Specifically, **It's just a declarative format for "describing the structure of other data".**. So if we create a `JSON` schema, the feature we get is to ensure that all parameter files are structured correctly. Then, **The latter check will likely need to be implemented using a more general-purpose programming language.** referring to the validation.

doganulus on Mar 7, 2023 Collaborator edited

Hi @kasperornmeck, I have suggested using JSON Schema if the intention is to go with a separate parameter definition file (suggested by TierIV). JSON Schema can validate both JSON and YAML documents. In this case, schemas would be used to define and validate existing `*.param.yaml` files.

Below is an example.

```yaml
# ROS parameter file to be validated
node_name:
  ros__parameters:
    range: 90
```

And this is a JSON Schema for `node_name.param.yaml`

```json
{
  "$id": "https://www.autoware.org/schemas/node_name.sch[illegible]js
  "$schema": "https://json-schema.org/draft/2020-12/schema",
  "title": "NodeName Schema",
  "type": "object",
  "properties": {
    "node_name": {
      "type": "object",
      "properties": {
        "ros__parameters": {
          "type": "object",
          "properties": {
            "range": {
              "type": "integer",
              "minimum": 80,
              "maximum": 100,
              "default": 90
```

```
                }
              }
            }
          }
        }
      }
    }
```

kenji-miyake on Mar 7, 2023

> I'm not quite sure what you are suggesting. Do you want to use JSON format instead of yaml format?

No, but I think JSON schema can be supported in addition to YAML format.
(But it's not a big issue now.)

> Is this what you mean? If not, can you give a concrete example?

No, my proposal is just to use PickNik's format.
Your proposal is slightly different from this format.
I believe it's good to separate the definition (PickNik's file) and the real parameters (the existing parameter files).

@kasperornmeck

kaspermeck-arm on Mar 7, 2023 Collaborator Author

@kenji-miyake

It would really help me if you put together a complete counter-proposal with exactly what you mean, similar to the one I created at the top of this discussion.

Currently I don't understand.

- What is meant by the real parameters?
- Do you want to have two different files for the parameters?

kenji-miyake on Mar 7, 2023

> It would really help me if you put together a complete counter-proposal with exactly what you mean, similar to the one I created at the top of this discussion.

I will describe it later.

> What is meant by the real parameters?

@kasperornmeck Mm, I'm sorry for my bad word choice. Maybe "actual" is a better word? 🤔
I meant the existing parameters that are loaded to nodes like https://github.com/autowarefoundation/autoware.universe/blob/b2a43c72979969dc07b015be1a2b9004479f7fd5/planning/behavior_path_planner/config/behavior_path_planner.param.yaml.

> Do you want to have two different files for the parameters?

> Yes, the definition and the actual parameters.
> It's a standard way, PickNik also supports it.

kaspermeck-arm on Mar 7, 2023 Collaborator Author edited

Edit: accidentally put the a comment as a new reply.

↑ 1 0 replies

kenji-miyake on Mar 8, 2023

@kasperornmeck This is our proposal. Is it clear to you?

Our purposes:

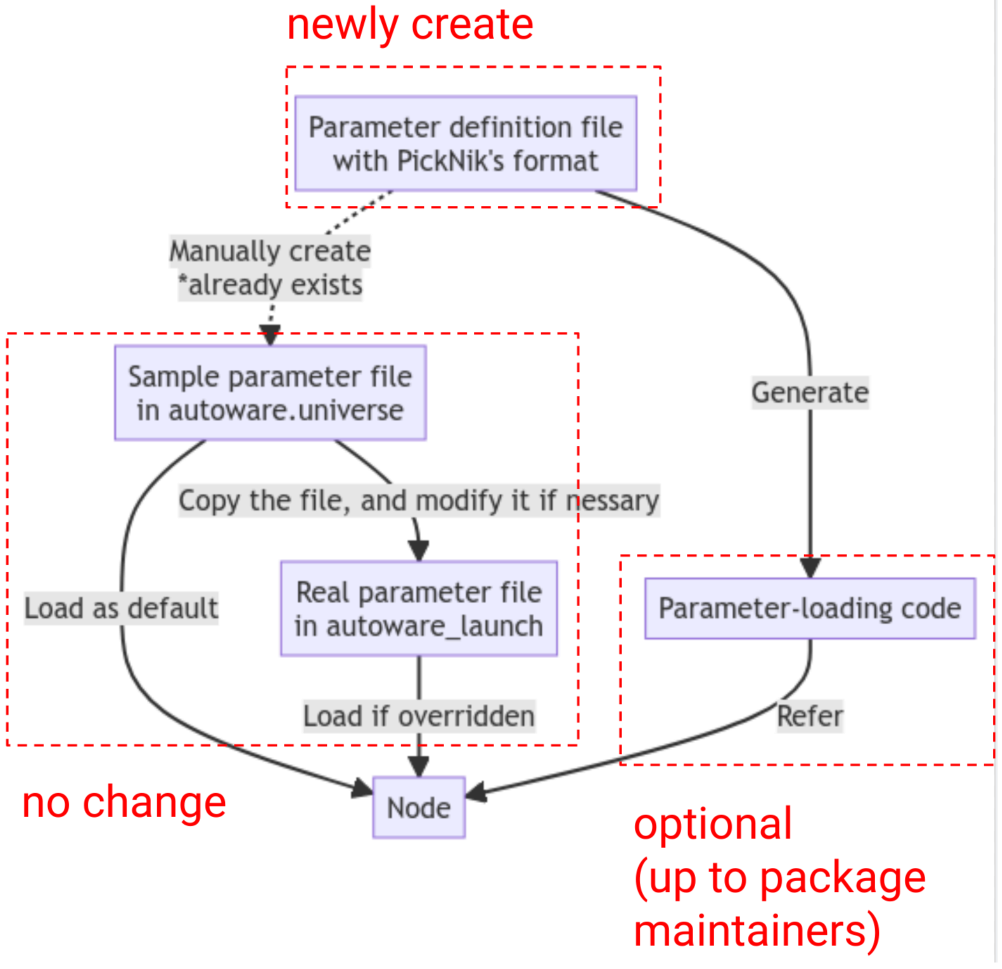
- Validate parameter files.
- Remove duplicates in code.
  - declare_parameter
  - Dynamic parameter update
  - Definition of parameter structures
- Generate parameter documents automatically.
- Accomplish the above with minimal changes and minimal effort
  - Having no API break is preferred.

TIER IV's proposal:

- Keep the existing parameter files and reuse them.
- Newly create parameter definition (schema) files.
  - Either PickNik or JSON Schema is okay, but considering Autoware is ROS-based, PickNik would be better for now.
- Those who want to use PickNik use it.
  - It can make the code around parameters efficient.
- Those who don't want to use PickNik just remove default values like refactor(obstacle_collision_checker): delete default values autoware.universe#2945.

What we need to do for the proposal:

- Package maintainers will create the parameter definition files of nodes.
- Package maintainers will do either:
  - Apply PickNik.
  - Remove default values from declare_parameter.
- Tool developers will develop a document generation tool.
  - Optionally, they can also develop an offline parameter validation tool, etc.


newly create
Parameter definition file with PickNik's format
Manually create *already exists
Generate
Sample parameter file in autoware.universe
Copy the file, and modify it if nessary
Load as default
Real parameter file in autoware_launch
Parameter-loading code
Load if overridden
Refer
Node
no change
optional (up to package maintainers)

↑ 1 👍 2 2 replies

kaspermeck-arm on Mar 8, 2023 Collaborator Author

@kenji-miyake

Thanks for the explanation.

How does your proposal address the documentation requirement?

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

kenji-miyake on Mar 8, 2023

@kasperornmeck I'm not sure if I understand your question correctly, but since the format is almost the same as your proposal, I believe our proposal can do what your proposal can do.

kaspermeck-arm on Mar 8, 2023 Collaborator Author

New updated discussion #3325.

↑ 1 0 replies