Working Group Draft

NTCIP IFR-8008 v00.00.00-alpha8

National Transportation Communications for ITS Protocol

ITS Open-Source Process

March 20, 2025

Notices

Copyright

Content and Liability Disclaimer

Acknowledgements

This document was prepared though an open-source standards development process with the following active contributors:

contributors 2

Check out the full list of contributors here.

In addition, the following submitted comments during the process:

- k-vaughn

The resultant document is maintained by the NTCIP Base Standards, Profiles and Protocols (BSP2) Working Group (WG), a subdivision of the Joint Committee on the NTCIP. The Joint Committee on the NTCIP is organized under a Memorandum of Understanding among the American Association of State Highway and Transportation Officials (AASHTO), the Institute of Transportation Engineers (ITE), and the National Electrical Manufacturers Association (NEMA). The Joint Committee on the NTCIP consists of six representatives from each of the standards development organizations (SDOs) and provides guidance for NTCIP development.

March 20, 2025

Foreword

Overview

This document is an NTCIP Open-Source NTCIP Process, Control, and Information Management document provided as Interim for Field Release (IFR).

Open-source documents are developed using the ITS Open-Source Process, as defined in NTCIP IFR-8008. This process provides an open standards development process that accepts issues reported by the community and resolved by peer-reviewed contributions from the community. The open source process concludes with the resultant material being approved by the defined approval process.

IFR documents are approved through a streamlined process focused on the technical experts of the community (e.g., those participating in the open-source development process) rather than through a formal ballot of industry managers.

NTCIP Process, Control, and Information Management documents define the practices and policies used by the NTCIP Joint Committee and its working groups in developing and maintaining NTCIP publications.

This document defines the process for developing projects for the ITS community using an open-source environment (e.g., GitHub). The project can produce any type of product, such as a guide, a technical specification, a test procedure (e.g., including code), etc.

The approval process for the resultant open-source product is based on the target level of specification. For example, an IFR specification undergoes a less formal approval process than a full standard.

Approvals

IFRs are peer reviewed within the open-source process with final approval by an associated WG established by the NTCIP Joint Committee.

Approval information is provided within the online environment.

Introduction

This site defines the ITS Open-Source Process as used by several projects within the ITS standards community. The process follows general practices within the larger open-source community; however, this document:

- provides a step-by-step overview of the process, so that those unfamiliar with open-source processes can better understand the process and become contributors,
- formalizes the process (e.g., by clearly defining what are requirements), and
- tailors the process (e.g., by defining the preferred tools to be used).

This document contains one normative annex.

The following keywords apply to this document: AASHTO, ITE, NEMA, NTCIP, open-source, process.

This document uses only metric units.

October 30, 2024

Section 1 General

1.1 Scope

This document specifies the process used to produce open-source documents within the field of Intelligent Transportation Systems (ITS).

The process follows general practices within the larger open-source community; however, this document:

- provides a step-by-step overview of the process, so that those unfamiliar with open-source processes can better understand the process and become contributors,
- formalizes the process (e.g., by clearly defining what are requirements), and
- tailors the process (e.g., by defining the preferred tools to be used).

The process to approve the resultant product is defined elsewhere (e.g., NTCIP 8001).

The ITS Open-Source Process is based on the practices defined by open-sauced. However, whereas open-sauced is written as an informative guide and describes how systems can work; this document is written as a specification to define how the ITS Open-Source Process will work. While still providing a discussion of the issues; it highlights the requirements and notable options along the way by stating each in its own paragraph and boldfacing the keywords "shall" and "may" to clearly designate requirements and options. The remaining text provides further guidance and can include additional options that do not necessitate specific numbering.

Section 2 Overview

Managing an open-source project involves four major activities as described in the following clauses:

1. Establishing the project
2. Processing comments
3. Processing contributions
4. Approving releases

2.1 Establishing the Project

Figure 1 provides an overview of the process to establish a new open-source project.

```
%%{init: { 'sequence': { 'mirrorActors': false } }}%%
sequenceDiagram
  participant Proposer
  participant Committee
  participant WG as Working Group
  participant Maintainer
  participant Repo as Open-Source Project Repository

  Proposer ->> Committee: Propose project
  Committee ->> WG: Establish WG
  Committee ->> Maintainer: Assign maintainer
  Maintainer ->> Repo: Establish public repository
  Maintainer ->> Repo: Upload initial baseline
  Maintainer ->> WG: Suggest project plan
  WG -->> Maintainer: feedback
  Maintainer ->> Repo: Post project plan
  Maintainer ->> Repo: Create appropriate branches for work
```

When someone identifies a need for a new shared resource (e.g., industry standard, reusable code, etc.) within ITS, they can develop a proposal and submit it to an appropriate committee. The proposal can be relatively simple (e.g., a statement of goals and structure) or a complete prototype.

If the proposal is accepted by the committee, the committee will assign a working group and one or more maintainers who will become responsible for leading the project. This will often include the individual proposing the project. The maintainer will establish the open-source project repository on the standards development organization's open-source website (e.g., GitHub account) and upload the initial project files.

Section 3 Commenter Responsibilities

3.1 Overview

3.1.1 General

Comments on projects using the ITS Open-Source Process are always welcome, no matter how seemingly major or minor. Comments are key to improving products. The ITS Open-Source Process is designed to facilitate and encourage users to submit comments and is therefore kept simple.

Within the ITS Open-Source Process, comments can be submitted in either the discussions or issues tab of the project repository.

3.1.2 Discussions

The discussions tab provides an open forum where interested parties can discuss ideas, ask and answer questions, and formulate ideas. The discussions tab does not directly propose any change to the project but can often nurture ideas that ultimately result in refining the overall vision of the project, identify problems or ambiguities in the project contents, develop consensus on project priorities, etc.

Discussions can be started by anyone at any time. Discussions can result in refining the concept of one or more issues before submitting formally submitting them as issues.

3.1.3 Issues

Every project should follow a plan. Within the ITS Open-Source Process, the plan is documented by defining issues that are to be addressed, preferrably according to assigned priorities.

The issues tab provides an open forum where any interested party can propose specific issues that need to be addressed by project contributors. The issues can be anything from a missing comma to requesting an entirely new feature. All proposed changes to a project are supposed to be initiated by submitting an issue.

When an issue is submitted, the project maintainer is responsible for triaging the issue. Triaging includes reviewing the issue, determining if the issue fits within the project plan, potentially parsing or merging the issue to create easily managable tasks, assigning appropriate priority and tags (e.g., bug, ambiguity, editorial) to the issue, and gaining consensus on the approach. This process can involve working with others on the project team to ensure consensus on the decisions being made.

Section 4 Contributor Responsibilities

4.1 Overview

Contributions on projects using the ITS Open Source Process are always welcome, no matter how large or small. However, before contributing, it's important to familiarize yourself with the following resources of the project:

Some of this information is standardized in this document, but specific projects can extend or make exceptions to the process and will always have their own project-specific goals. Contributors are responsible for being familiar with the information contained in the following project files, as stored in the project's root directory:

- **README.md**: Provides an overview of the specific project,
- **CODE_OF_CONDUCT.md**: Identifies the code of conduct for the project, and
- **CONTRIBUTING.md**: Identifies project-specific rules for contributing.
- **LICENSE.md**: Identifies the license agreement for project files

Needs Review

We need to make sure that our standard license addresses all concerns. To date, it sounds as if CC-BY is a reasonable approach for documentation and BSD 3 clause is acceptable for code. MIBs probably need a custom license that falls in between these two and restricts the types of changes and use.

For projects following the ITS Open-Source Process, the last two files will typically only identify exceptions or extensions to the rules defined by this document.

The overall process for contributing to an ITS open-source project is shown in Figure 4-1 and described in the remainder of this section.

Section 5 Maintainer Responsibilities

5.1 Overview

The maintainer for an open-source project fulfills many responsibilities, including setting up the project, managing issues, reviewing submittals, and leading the development community. In addition, the maintainer is often a prime contributor.

5.2 Establish Repository

The maintainer **shall** work with the sponsoring SDO to establish the open-source repository for the project.

Example

NTCIP repositories are hosted athttps://github.com/ite-org/.

5.3 Configure Project Settings

5.3.1 Issues and Discussions

The maintainer **shall** ensure that the issues and discussion pages are enabled for the ITS open-source project.

Note

Within GitHub, issues are enabled by default but the discussions tab is disabled. To enable, go to the settings tab and select discussions in the general section.

5.3.2 Pages

If the project includes documentation, the maintainer **shall** ensure that GitHub pages is enabled for the project.

Section 6 WG Responsibilities

6.1 Overview

Each major stage of the open-source process is reviewed by a WG or committee to ensure a base level of consensus. The specific group that is required to provide consensus and the level of consensus required dependent upon the standardization path adopted for the project.

Example

An NTCIP experimental specification can be approved at the NTCIP WG level for all stages while an NTCIP standard requires Joint Committee approval for the project approval and release approval.

The stages within the open-source process include:

- project approval
- issue prioritization
- pull-request approval
- release approval

6.2 Project Approval

An appropriate WG or committee **shall** approve the formation of a project prior to establishing the SDO GitHub repository for the project.

The appropriate WG or committee **should** be identified in policies adopted by any SDO adopting the ITS Open-Source Process.

Note

A contributor can establish their own GitHub repository for the project before formal approval to allow WG members to gain a better idea of what is being proposed.

Annex A Contributor Covenant Code of Conduct

Each entity that participates in the development of this repository as a commenter, contributor, maintainer, or manager agrees to encourage a harassment-free environment and to act and interact in ways that contribute to an open, welcoming, and healthy community.

A.1 Scope

This Code of Conduct applies within the scope of GitHub, and also applies when an individual is officially representing the community in public forums.

A.2 Enforcement

Instances of abusive, harassing, or otherwise unacceptable behavior may be reported to the community leaders responsible for enforcement at ntcip@nema.org.

A.3 Details

For additional guidelines on the application of this code, see the Contributor Covenant.

A.4 Attribution

This Code of Conduct is adapted from the Contributor Covenant, version 2.1.

October 30, 2024

Annex B Documentation Conventions

B.1 Exceptions Allowed

Unless otherwise stated in the project-specific CONTRIBUTIONS.md file, each project based on this specification **shall** develop documentation as defined by this annex.

B.2 Development Environment

B.2.1 Overview

In addition to the development tools needed to manage and submit any contribution within the Git environment (e.g., Git, GitHub), developing project documentation requires the following tools:

- **A text editor,** which is used to create and edit markdwon and yaml files,
- **Python,** which is required to run MkDocs,
- **MkDocs,** which is an open-source tool for translating a set of markdown files into a static website, and
- **Materials for MkDocs,** which is an open-source tool that extends the markdown language to support additional features that are useful for developing the look and feel of the project's documentation.

This combination of tools has been selected because it:

- is designed to be easy to install and use,
- requires minimal setup,
- works well with Git and GitHub,
- supports search functionality,
- can produce a static website,
- when coupled with add-ons, can produce PDFs
- has an active development community

It is recommended to establish this development environment prior to making any edits. Generating the documentation website locally from a known baseline allows the contributor to verify that the development environment is working correctly prior to introducing edits to the files. Contributors are required to generate the

Annex C Coding Conventions

C.1 Python Coding Conventions

Each contributor **shall** adhere to style guidelines defined in Python Enhancement Proposals (PEP) 8 – Style Guide for Python Code.

Highlights of PEP 8

- Imports should be at the top of the file
- Imports should be grouped into three sections with a blank line between each: (1) standard library imports, (2) thirdd party library imports, and (3) local imports
- Function and variable names should be in lowercase_with_underscores
- Class names should be in UpperCamelCase
- Constants should be in ALL_CAPS_WITH_UNDERSCORES
- Do not use tabs; use four spaces for each indentation level
- Limit lines to 79 characters; or 72 characters for long comments
- Separate top-level functions and class definitions with two blank lines
- Inside functions, use one blank line to separate significant logical sections

Each contributor **should** use a linter to automatically enforce the PEP 8 rules.

Example

Pylince

October 30, 2024