Deprecation and Archival

Approved by with TSC consent on 4/6/22

From time to time, advances in the project will dictate that modules, services, documentation, code repository or other artifact of the project are no longer to be used going forward.

Deprecation

When a module, service or similar artifact is determined to be no longer part of a future release, it is “deprecated”. Deprecation of any module, service, etc. requires the approval vote by the majority of the TSC. When a module, service, etc. is voted to be deprecated by the TSC,

- The TSC should forecast the eventual archive of the item (either by release or triggering event such as the creation of a substitute service).
- A decision record in the EdgeX Design Decisions Project Board is recorded
- If the programming language used to create the module, service, etc. allows it, the deprecated item should be labelled deprecated by the language idiom
    - Examples
        - Using "// Deprecated: followed by some information about the deprecation" to start the GoDoc comment of a public function, struct field, type, etc. (see https://rakyll.org/deprecated/ for good practices of deprecation in Go)
        - Using "@Deprecated(since=version)" annotation in Java
    - In the code, provide (by language idiom or comment) information about what to use instead if there is an alternative
- A note indicating the deprecation of the module, service, etc. will be placed on the cover Readme.md page of any repository containing code or documentation for the deprecated item. The note should indicate when the deprecation started (which release) and when the artifact will be archived. Note: some repositories contain code/documentation for multiple modules or services so the note should be explicit with regard to what is being deprecated, when deprecation takes effect (which release) and when archival is expected.
- When an endpoint, parameter or schema has been deprecated in the REST API, the Swagger documentation should be flagged accordingly (see https://swagger.io/specification/)
- The EdgeX documentation (docs.edgexfoundry.org) should contain a deprecation note when a feature is deprecated until it is removed (archived).

Deprecation is a warning to adopters that the module, service, etc. is nearing its end of life.

Deprecation does not mean that the artifact cannot be worked on. Specifically, bug fixes, security issues, etc. may require the code of a deprecated artifact to continue to receive new issues/commits/PRs/etc. Significant change, such as adding new features, of a deprecated item will generally be rejected. Deprecation is an indication to the EdgeX community and adopters that the artifact is slated for eventual archival. Continued use of, certainly continued reliance on, the artifact is discouraged.

Archival

When a module, service or similar artifact is officially retired and no longer to be used as part of the project, it is archived. In most cases, a module, service, etc. is archived only after being deprecated for a period of time.

Archival of any module, service, etc. requires the approval vote by the majority of the TSC. When a module, service, etc. is voted to be deprecated by the TSC,

- A decision record in the EdgeX Design Decisions Project Board is recorded
- A note indicating the archival of the module, service, etc. will be placed on the cover Readme.md page of any repository containing code or documentation for the archived item. The note should point to any replacement module, service, etc. when one exists.
- A request to the Linux Foundation service team is made to archive the associated repository. Note, if the repository holding the archived module, service, etc. still contains non-archived code/documentation, then the repository must not be archived and the Readme update will suffice to indicate archival.

Work on archived modules, services, etc. is prohibited. An artifact that is part of the current LTS release cannot be archived until the support period expires.