Contributor's Guide

Welcome! We really hope you are here because you are seriously thinking about adding to the EdgeX Foundry IoT platform. We can't wait to receive your contribution! Before you make your first submission, this page has been set up to help guide you through some information you need to know before working on and contributing code to the EdgeX project.

Code Style Guides

The EdgeX project technical community has adopted the Google Style Guides for all programming languages. As most of the EdgeX code is in Java today, you will find the Java guide helpful: https://google.github.io/styleguide/javaguide.html.

If using Eclipse as your IDE, you will find the Eclipse style guide configuration file at https://github.com/google/styleguide and instructions for how to setup Eclipse to use it here: https://github.com/HPI-Information-Systems/Metanome/wiki/Installing-the-google-styleguide-settings-in-intellij-and-eclipse.

The EdgeX project does use checkstyles during the build process. So code that does not conform to the style guide will result in a request by the project to address issues before being accepted.

Line Feeds

Line endings in code files are different for *nix versus Windows. In *nix, lines end with “\n”. In Windows, lines end with “\r\n” (carriage return, line feed). This creates issues when pulling/working with code created in different environments.

The *nix line feed is the desired line ending for EdgeX Foundry source code. Windows developers need to configure tools to use and apply the appropriate line endings. Thanks to Gorka Garcia from Cavium, here are some examples of how to configure some popular tools to deal with this issue.

- For Eclipse (Version: Oxygen.2 Release (4.7.2)), go to “Window > Preferences > General > Workspace” settings and find “New text file line delimiter” and set it to “Other: Unix”, this will set the line endings for new files. There is also “File > Convert Line Delimiters To > Unix” option for converting files already with windows line endings.
- For Microsoft Visual Studio Code (v1.19.2) you need to add the following to the user configurations:

  "files.eol": "\n"

- For Notepad++ (v7.5.1) go to "Settings > Preferences > New Document/Default Directory" then select "Unix/OSX"

Submission DCO

As a potential contributor of code to the project, you are highly encouraged to read the Technical Charter and in particular note section 12 - Intellectual Property Policy. Contributors are advised to understand that submitting code to the project indicates you agree with the the Developer Certificate of Origin. See the Committing Code Code Guidelines for details on how to sign your code contributions to indicate your agreement with the DCO. See http://elinux.org/Developer_Certificate_Of_Origin for a description of how the Linux kernel implements its DCO.

You can add the DCO signoff to your Git commit by adding the **--signoff** flag to your git command.

If you've already committed your code to your local branch without the DCO signoff, you can add it with the following commands:

```
git reset --soft HEAD^

git commit --signoff -m "<your original commit message>"
```

If you've already pushed your code branch to Github and created a pull request, you will need to forcibly override that branch:

```
git push --force <your remote> <your branch>
```

Submission License and Copyrights

All code submitted to the project must be made available under Apache License, Version 2.0.

A copyright header should appear at the top of all project artifacts (code files, configuration files, documentation, etc.). The copyright header should contain:

- Apache License 2.0 statement
- Copyright of the contributing entity (person or corporation)

Note: when subsequent contributors make substantive changes to a file they may also optionally add a copyright line to the header for themselves (person or corporation), but should preserve existing copyright statements and license.

Existing source from the project can be used to exemplify this header.

Example Project Artifact Header

```
/*******************************************************************************
 * Copyright 2016-2017 Dell Inc.
 *
 * Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except
 * in compliance with the License. You may obtain a copy of the License at
 *
 * http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software distributed under the License
 * is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express
 * or implied. See the License for the specific language governing permissions and limitations under
 * the License.
 *
 *******************************************************************************/
```

Code Styles and Checkstyles

The EdgeX project has adopted the Google Style Guides (https://github.com/google/styleguide) for use in all EdgeX coding.

In support, of keeping the styles applied to projects, developers are encouraged to use the tools at their disposal to apply the styles to their code and to address issues before checking code into the project's repositories.

Any new Java repositories should use the elements (highlighted in RED) below in their project pom.xml to apply the Maven Checkstyle Plugin – as most of the current code is in Java at this time these elements are already added to existing Java repository poms. Please note that this checkstyle will produce warnings in the console and a file, but will only fail the build in the result of a violation ERROR. The check can be skipped in development by changing the <skip> to true.

```
 <properties>
     <checkstyle.plugin.version>2.17</checkstyle.plugin.version>
  </properties>

  <build>
     <plugins>
        <plugin>
           <groupId>org.apache.maven.plugins</groupId>
           <artifactId>maven-checkstyle-plugin</artifactId>
           <version>${checkstyle.plugin.version}</version>
           <executions>
              <execution>
                 <id>validate</id>
                 <phase>validate</phase>
```

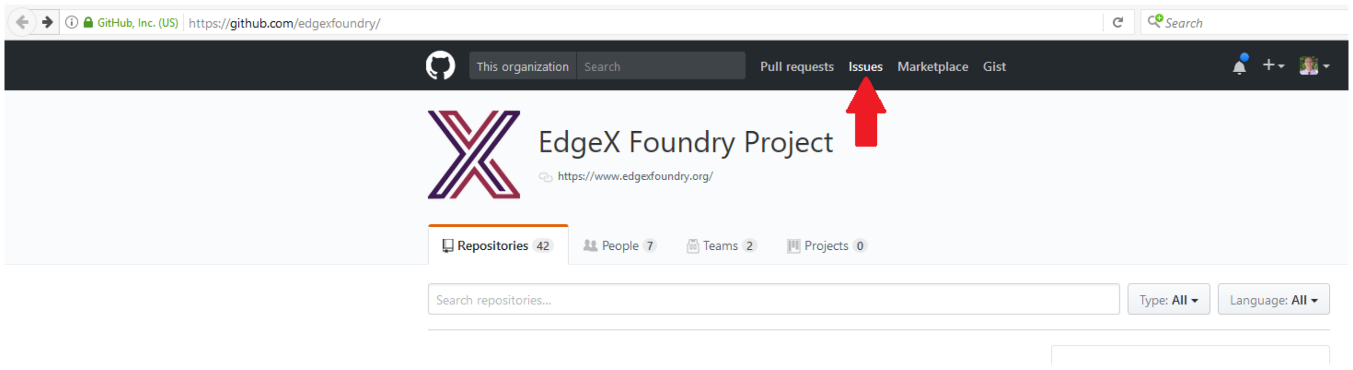
```
          <configuration>
            <configLocation>google_checks.xml</configLocation>
            <consoleOutput>true</consoleOutput>
            <violationSeverity>error</violationSeverity>
            <skip>false</skip>
            <linkXRef>false</linkXRef>
            <includeTestSourceDirectory>true</includeTestSourceDirectory>
            <outputFile>${project.build.directory}/edgex-checkstyles-result.xml</outputFile>
          </configuration>
          <goals>
            <goal>check</goal>
          </goals>
        </execution>
      </executions>
    </plugin>
  </plugins>

  <pluginManagement>
    <plugins>
      <!--This plugin's configuration is used to store Eclipse m2e settings
        only. It has no influence on the Maven build itself. -->
      <plugin>
        <groupId>org.eclipse.m2e</groupId>
        <artifactId>lifecycle-mapping</artifactId>
        <version>1.0.0</version>
        <configuration>
          <lifecycleMappingMetadata>
              <pluginExecution>
                <pluginExecutionFilter>
                  <groupId>org.apache.maven.plugins</groupId>
                  <artifactId>maven-checkstyle-plugin</artifactId>
                  <versionRange>[2.17,)</versionRange>
                  <goals>
                    <goal>check</goal>
                  </goals>
                </pluginExecutionFilter>
                <action>
                  <ignore></ignore>
                </action>
              </pluginExecution>
            </pluginExecutions>
          </lifecycleMappingMetadata>
        </configuration>
      </plugin>
    </plugins>
  </pluginManagement>
</build>
```

Bug Tracking

EdgeX uses GitHub Issues to submit and track bugs.

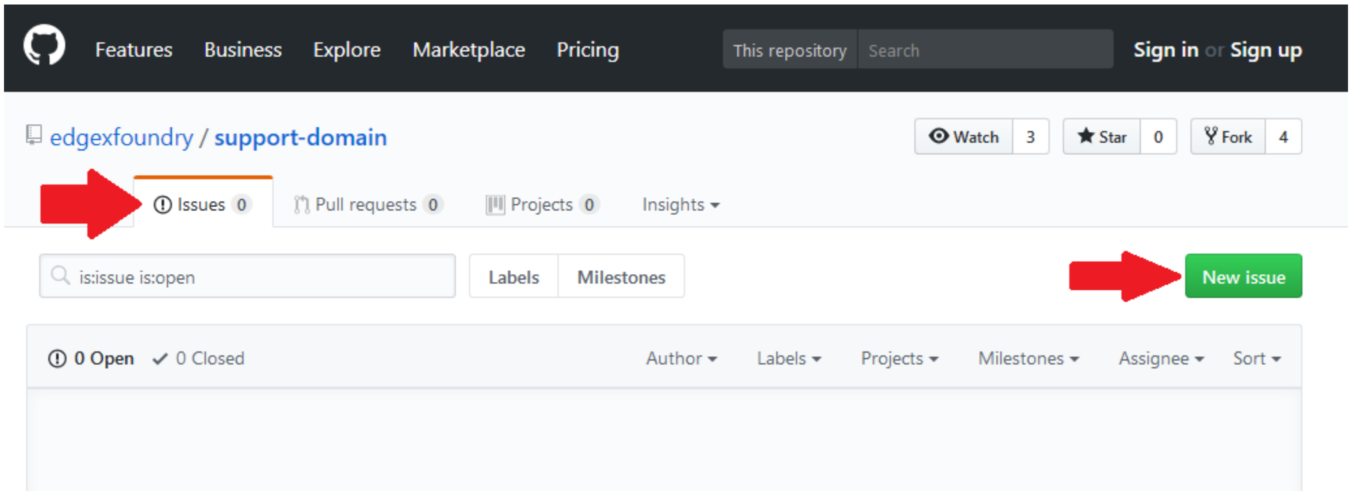
Reporting Issues

This is a great way to contribute. Before reporting an issue, please review current open issues to see if there are any matches. To see the current list of EdgeX issues, go to the project's GitHub site and click on the Issue link at the top of the page.



How to Report an Issue

In the GitHub repository, anyone can create a new issue for any of the project's repositories - no sign in is required! Locate the repository you believe contains the bug. Then click on the Issues tab of the repository, and then click on the green "New Issue" button to create and submit a new bug.



When reporting an issue, please provide as much detail as possible about how to reproduce it. Details are key, please include the following:

- OS version
- EdgeX version
- Environment details (virtual, physical, configuration details etc.)
- Steps to reproduce
- Actual results
- Expected results

If you would like, you could also bring up the issue on Slack for initial feedback before submitting the issue in GitHub Issues .

Versioning and Versioning Scheme

EdgeX has adopted a significance-based approach to versioning. This versus a semantic versioning approach (see reference #1 below).

Instead of tightly coupling the version number updates to changes in the REST API(s) exposed by a microservice, EdgeX updates the version number when there has been a significance in the microservice as a whole. Similar to the semantic version scheme, the significance of change scheme also uses a three-part version with numbers representing major, minor, and patch changes. The latter can be left off until actually needed.

A microservice starts its life as version 0.1, and each new release bumps the second number. The move from 0.x to 1.x in EdgeX signifies that the microservice is transitioning from a development to a production quality release. The decision to make the transition to a production release rests with the responsible Working Group chair with approval from the EdgeX Technical Steering Committee. Further jumps in the major number typically indicate major new releases of the project which *could* signify API breakage, but could also signify major new functionality as well.

If/when a maintenance release of a particular microservice is warranted, a third number gets introduced. So, if 0.5 is released, and a serious bug is found and fixed immediately, 0.5.1 could be released right away, while development continues as normal on master toward 0.6. For example, you can see the historical version numbers used by the bluez project (Linux Bluetooth stack) (see reference #2 below) are just major.minor, as they usually rely on distros to release maintenance releases. An example of a project that uses three-digit versions is snapd (see reference #3 below), the daemon at the heart of the snap ecosystem.

[1] https://en.wikipedia.org/wiki/Software_versioning

[2] https://git.kernel.org/pub/scm/network/ofono/ofono.git/refs/tags

[3] https://github.com/snapcore/snapd/tags