Contributor Suggestions (how can you be a part of EdgeX Foundry?)

Looking for places to contribute on EdgeX Foundry? There is no lack of areas where technical and non-technical contributors (whether individuals or a small team) can help. This list provides some suggestions and needs to spur your imagination about projects that you and/or your team might find interesting. Get in touch with EdgeX TSC Chair James Butcher (james@iotechsys.com) or any member of our community if you need assistance in getting on-boarded with the EdgeX Foundry community and are ready to get started! Tune in to our EdgeX GitHub Discussion boards to learn more or ask about additional needs: https://github.com/orgs/edgexfoundry/discussions.

Technical Projects

Bug Fixes, Small Enhancements: If you are looking for a way to lend a big hand but in bite sized ways, consider helping fix an EdgeX issue or provide a requested small enhancement. A list of issues (bugs, enhancement requests, etc.) can be found with each of the EdgeX repositories in GitHub. For example, you can find a list of issues associated with the central EdgeX Go repository here: Go Core Issues List. Many of the issues are tagged with "Help Wanted" and "Good First Issues". You can also message the issue creator or a member of the EdgeX community to learn more about the issues you see and how you can help. The GitHub discussion boards are a great way to introduce yourself and seek guidance and help on any issue.

Device Service SDKs for alternate languages/platforms: the device service SDK is a tool for generating the device service scaffolding to help connect a device or sensor to EdgeX. Today, two available SDKs are written in Go and C (an older archived SDK was written in Java) and produces device services in Go and C. We want to offer SDKs in multiple languages (Python, Rust, Java, JavaScript, etc.) to make it easier for organizations to onboard their devices/sensors to EdgeX in the language of their choice that best meets use case needs. EdgeX is religious in its belief in polyglotism – that is that any micro service can and should be allowed to be written in a programming language (and with tools from that language) that makes most sense to the authors of the micro service. Multiple device service SDKs in various languages can help achieve that goal at the south end of EdgeX.

Device Services: EdgeX is about providing connectivity to devices and sensors. It's about collecting the data from those sensors/devices and actuating those devices/sensors. In EdgeX, this means providing a device service micro service that provides that connectivity. EdgeX already has some device services with more coming. But if you have a device or sensor that you would like to be able to connect to EdgeX, use the device service SDK (or choose to create it by hand given the EdgeX APIs) to create the device service that makes provisioning and using that device / sensor easy with EdgeX. The term device/sensor should be applied loosely. A device service could also be used to connect another edge platform or application (open source or proprietary) or facilitates ingestion of data from a specific protocol (like OPC-UA, Haystack, etc.).

North Bound Export: EdgeX provides some basic facilities to get device/sensor data to your enterprise or cloud provider. Need to get the EdgeX collected data to a specific cloud provider or enterprise system? Help extend the application services of EdgeX to include other distribution endpoints, transformation engines, filters, etc..

Micro service API documentation generation: today, much of the API documentation for EdgeX is manually created. This makes them hard to keep in sync with the code and error prone. Are there tools that exist that could help generate documentation from the code itself? Can you help pick better API formats, help provide better tooling, and help make them part of the EdgeX CI/build process? Could some tool or tools be used to support multiple API formats simultaneously (example RAML and Swagger)? How can we support message based protocols (MQTT, DDS, etc.) in the future? Your opinion matters with the EdgeX community, but this is an area where the opinion must be backed up with prototype projects and demonstrations. Create a solution that demonstrates how API documentation can be easily generated, easily digested by tools and easily consumed by human users.

Performance Expert: Do you know how to make Go or C hum? Are you aware of how to optimize applications to start and load faster? If you have skills to help reduce the footprint of our existing code, improve the startup time of the micro services, or otherwise optimize the micro services in terms of memory, CPU and speed, we could use your help. Our goal is to eventually provide alternate language micro services that can address performance issues with executables that are best for various use cases and platforms. Today, EdgeX is largely Go with some C and Java and help to reduce the EdgeX footprint, throughput, and speed would be very welcome.

User Interfaces: EdgeX today is generally headless - that is without graphical user interfaces. We have some simple UIs for demonstration purposes, but these can be extended and additional UIs can help provide a better user experience with EdgeX. Develop a user interface to provision a new device. Develop a user interface to add and remove rules from the rules engine. Develop a user interface to visualize data collected by Core Data. Develop a user interface for registering new export clients. The possibilities are endless.

Roadmap and Backlog: Not seeing something on this list that gets you excited about helping? Not to worry, we have plenty of additional work that might interest you. Checkout our project backlog page and the release pages (each future release provides a list up coming features and work we will need to complete).

Tech Writers

Wiki clean up: Our Wiki documentation was initially create for the Dell Fuse project. It was written by several people over two years. We'd like to have the documentation read more consistently (single-author look and feel) and we would like to make sure it is accurate (many elements have changed in the two years of the Dell Fuse to EdgeX release). If you are a tech writer that is an expert in taking what exists and making it shine, we could use your help.

Getting Started Guides: OK, we are software developers. Sometimes developers can write good documentation and sometimes we are not as clear as we should be. If you are a tech writer and can help us clean up or improve the project's documentation, we would love to have your contributions. In particular, there are many "Getting Started" guides and examples we would like to have written to ease developers and users into EdgeX properly. IoT can be a complex set of software and hardware. Help us make EdgeX easy to use and understand.

How-to Guides: We think EdgeX offers a very flexible framework from which to build edge/IoT solutions. However, it is a framework and often requires developers to do some customization, configuration or add extension pieces (like device services). Can you help create how-to guides, videos or otherwise give assistance to those just starting out with EdgeX and learning how to apply it to their use cases and systems? Blogs, Vlogs, RST documents to be added to our document pages, or just about any type and makeup of content is welcome.

Training: Do you teach? Have you created programs to help on-board new developers into a technology? EdgeX needs a set of training materials that can be used for a variety of events to help on-board and training new EdgeX users and developers.

Marketing

Evangelism: Know of an IoT event that EdgeX should be participating in? Are there blog sites that focus on use of open source technology to solve edge /fog deployments? Would you like to sponsor and run an EdgeX hackathon in the future? Help us get the word out on EdgeX.