# GEISA Specification 0.1.0

*Release 0.1.0*

**Grid Edge Interoperability and Security Alliance**

**Jun 17, 2025**

# CONTENTS:

# CHAPTER ONE

# ABSTRACT

The GEISA Specification describes the operating environment for a utility-focused grid edge device with sufficent detail so that two vendors working from the specification alone can create an implementation such that a GEISA compliant application can run on both without modification.

While the initial focus is smart meters, the GEISA Execution Environment (EE) runs on a variety of devivce types.

CHAPTER

TWO

# KEYWORDS

TODO

## CHAPTER THREE

# NOTICE AND LEGAL INFORMATION

TO DO Patent policy statements

# CHAPTER FOUR

# INTRODUCTION

The GEISA Specification describes the operating environment for a utility-focused grid edge device with sufficent detail so that two vendors working from the specification alone can create an implementation such that a GEISA compliant application can run on both without modification.

The specification outline is as follows:

- Design Principles
- Envisioned System Architecture
- Hardware Requirements
- Operating System
- Base Libraries
- Core Services
- Applicaiton Isolation
- Application Management
- Applicaiton Programmingn Interfaces (APIs)
- Security

CHAPTER

FIVE

# NORMATIVE REFERENCES

TODO: Offical documents references that are essential to this spec

CHAPTER

SIX

# DEFINITIONS, ACRYONYMS, AND APPRVIATIONS

**ADC**
Analog-to-digital conversion

**API**
Applications Programming Interface

**COAP**
Constrained Applications Protocol

**FAN**
Field Area Network

**EE**
Execution Environment

**GEISA**
Grid Edge Interoperability and Security Alliance

**GPIO**
General Purpose Input/Output

**HAN**
Home Area Network

**LAN**
Local Area Network

**MQTT**
Message Queueing Telephany Transport

**OS**
Operating System

**RMS**
Root Mean Square

**SPI**
Serial Peripheral Interface

CHAPTER

SEVEN

# DESIGN PRINCIPLES

The GEISA design principles are:

- Interoperability
- Constrained Environment
- Minimal Implementation
- Core Specification w/Extensions
- Security

Each design principle is described in more detail below.

## 7.1 Interoperability

The EE must enable **source-code interoperable** applications, isolating them from the underlying hardware implementation.

All GEISA applications must use a single, consistent **GEISA API** to access resources such as sensors, actuators, persisten storage, networking, and more.

All GIESA application must be assureed that they will have the resources they need based on their **requirements manifest**.

## 7.2 Constrained Environment

The GEISA EE is a minimal resource environment, having contraints on CPU, RAM, storage, and networking.

As much as possible. resources should be reserved for GEISA applications, not consumed by the underlying operating system.

> **Note**
>
> The original text in the spec was for the overall platform, but this is for the GIESA EE. The idea is GEISA EE cares about how much available for EE and not total on the platform.

While a GEISA compliant EE may offer more, the minimal GEISA EE has **TBD of RAM** and **TBD of storage** reserved for GEISA applications.

Efficiency is critical. The EE shall provide only those services which are so widely required that it would be less effience to *not* provide them.

## 7.3 Minimal Implementation

The less there is, the less there is to maintain and the less there is to attack.

Keeping the system up-to-date is challenging, both for utilities and for vendors.

The GEISA EE favors the minimal implementation. If there is an option that is not needed, then turn it off. If there is an unwanted feature, leave it out.

The GEISA should include only what is required.

## 7.4 Core Specification w/Extensions

This initial GEISA specification covers the GEISA Core EE, that is, the EE that will be provided by all GEISA compliant systems.

In the future, additional extensions to the GEISA specification will be added as the comunity determines that they are needed.

## 7.5 Security

GEISA security is at the same level of importance as GEISA interpoerability.

At every level, from minimizing the attack space, to harding of the APIs and all services, GEISA security is front and center.

CHAPTER

EIGHT

# ENVISIONED SYSTEM ARCHITECTURE

This section shows verious architectural diagrams showing all the GEISA components.

Overall diagram

- One or more GEISA Compliant Applications
- The GEISA Compliant EE
- The underlaying OS
- The underlying Hardware

CHAPTER

NINE

# HARDWARE REQUIREMENTS

**Note**

The original text specified an ARMv7 platform with 512MB or RAM and 1GB of flash. However the text below focused on the GEISA EE environment instead of the lower-layer hardware platform.

As a **source-code interoperabiilty** specification, the underlying hardware must ensure that that it supports the GEISA EE.

The vision is that GEISA EE runs on a wide range of hardware platforms with various capabilities:

- Device type such as smart meter, load switch, EV charger, etc.
- Processor architectures such as ARM32, ARM64, RV32, RV64, AMR64, etc.
- Single core and multi core
- RAM sizes from 512MB and higher
- Storage sizes from 1GB and higher
- Networking Interfaces such as mesh, Wi-Fi, celluar, etc.
- Metrology Interfaces to provide voltage, current, etc.
- Hardware Interfaces such as GPIO, ADC, SPI, etc.
- Hardware Watchdog

A given hardware platform must provide a tool-chain for buidling the GEISA EE, along with the necessary hardware resources.

## 9.1 Metrology Hardware

One of the primary targets for the GEISA EE is on smart electric meters, hence the need for a standardize interface to metrology information.

**Note**

A GEISA EE environment is not expected to provide metrology info when used on a non-metrology device.

Minimal **static** metrology information:

- Number of Phases

- Meter Rating
- Meter Form
- Base Frequency
- Serial Number

Minimal **dynamic RMS values** information:

- Voltage Reading, RMS
- Current Reading, RMS
- Frequency

Minimal **dynamic waveform** information:

- Voltage Reading, ADC
- Voltage Scaling, ADC
- Current Reading, ADC
- Current Scaling, ADC
- Samples/Cycle, 16, 64, 128, 256, 512, etc.
- Resolution per sample, e.g. 16-bits
- Wall Clock Time
- Monotinic Time

## 9.2 Sensor Hardware

A GEISA EE compliant platform can optionally provide one or more sensors.

Some example sensors that may be provided include:

- Temperature Sensor
- Humidity Sensor
- Switch Sensor
- GPS

## 9.3 Hardware Enumeration

The GEISA EE provides an API so that GIESA compliant applications can query the hardware resources available:

- Device Info - returns device info, e.g. meter, ev charger, etc.
- CPU Info - return CPU info such as arch, number of cores, etc.
- RAM Info - return memory available to GEISA EE applciation
- Persistent Storage Info - return persistent storage info for GEISA EE application
- Non-Persistent Storage Info - return non-persistent storage info for GEISA EE application
- Network Info
- Metrology Info

CHAPTER

TEN

# OPERATING SYSTEM

GEISA shall use Linux as the core operating system, based, in part, due to the requirment for application isolation.

**Note**

It is not clear that GEISA 1.0.0 should specify an exact kernel since existing vendors/utilities are using various versions of the Linux kernel.

If the focus on GEISA is on the EE, one can argue that the EE should be isolated from a given version of the kernel.

GEISA will use a Linux 6.x kernel (exact version TBD).

The GEISA Linux kernel must minimize the attack surface and size. This means removing all unnecessary components and extranous modules.

Whether the GEISA Linux kernel supports loadable kernel modules or not is an implementation decision that is invisible to a GEISA application.

There is no requirement for the underlying Linux kernel to support real-time features, and no real-time features are exposed to GEISA applications.

# CHAPTER ELEVEN

# BASE FILESYSTEM

A GEISA container provides the follwing minimum filesystem.

- bin
- dev
- etc
- lib
- proc
- sbin
- sys
- tmp
- usr
- var

CHAPTER

# TWELVE

# BASE LIBRARIES

Applicaiton isolation is assumed to be provided through containers. Containers are able to inherit from the host operating system. To facilitate clean and regular updates, and to minimze container sizes, GEISA applications are encouraged to take advantagge of the libraries provided by the base GEISA environment.

This does not prevent GEISA applications from including their own libraries is the GEISA environment does not provide the needed library

There are to major groups of libraries:

- C Language Runtime Libraries (e.g. gcc, uclibc, musl)
- Other C Libraries

## 12.1 C Langauge Runtime Libraries

- libc - Core runtime support
- libgcc - GNU C Compiler Collection (low-level runtime support)
- libstdc++

## 12.2 Other C Libraries (Minimum)

- libasyncns.so - Asynchronouse Name Service
- libatomic - Atomic operations
- libcap.so - POSIC capabilities
- libcrypt - Password hashing (MD5, SHA-255)
- libcrypto - OpenSSL (hashing, encryption, digital signatures, random numbers, certs/keys)
- libdl - Dynamic loading
- libm - Math library
- libnsl - Network Services Library
- libpthread - POSIX Threads
- libresolv - DNS resolution and name services
- librt - Real-time
- libutil - Users, group,s pseudo-ttys (pty), etc.

## 12.3 Other C Libraries (Additional)

**NOTE** Expand this list

- libpcap
- libseccomp
- libsqlite3
- libssl
- libstdc++
- libtinycbor.so
- libz.so

Also to be included:

- ZeroMQ
- MQTT
- Dbus
- Others

CHAPTER

# THIRTEEN

# CORE SERVICES

TODO

CHAPTER

FOURTEEN

# APPLICATION ISOLATION

GEISA applications shall be isolated from each other for the following reasons:

- To ensure that one application cannot impact another application.
- To ensure that one application cannot see the artifacts of another application.

**Note**

Other than specifiying that application that **isolation** shall be provided, it is not clear that the GEISA 1.0 specification should specify the isolation technology.

One vendor/utility may have an existing implmentation with Linux Containers (LXC), while another may have an existing implementation with systemd.

It may be useful to make this an "after 1.0" GEISA release requirement, or a future GEISA Extension.

The most immediate need is for GEISA apps to run on multiple platforms from multiple vendors/utilities, perhaps using isolation techniques that are already used by existting vendor.

Reguardless of container implmeentation (e.g. LXC, systemd, etc.), an **authenticated appllication manifest** shall control access to the following:

- Permissions
- Performance
- System Control
- Networking Control
- API Control

## 14.1 Permissions

Most if not all of the bullets below are achieved by having each app in it's own isolated conainer.

- Apps should run in independent processes
- Apps must run with least privilege
- Apps access permissions should be deny by default
- App-to-app communication will be denied by default (**NOTE** perhaps hold off on app communication until after GEISA 1.0.0)
- Apps cannot access other apps memory or other resources

- Apps should not know about other apps unless explicitly informed
- Apps access to the local file-system will be isolated/restricted

## 14.2 Performance

GEISA isolation must meet these performance requirements:

- Apps cannot impact the performance of the system
- Apps cannot impact the stability of the system
- Apps cannot impact the performance of other apps
- Apps cannot impact the stability of other apps
- Apps cannot create denial-of-service situations

## 14.3 System Control

The system (the container) shall control every aspect of an GEIA app, including:

- CPU (how much CPU allowed)
- Memory (how much RAM memory allowed)
- Persistent Storage (how mush Flash storae allowed)
- Non-Persistent Storage (how much tmpfs allowed)
- Networking (what networking interfaces and ports allowed)

Addition system control:

- App-to-app communication (**NOTE** perhaps hold off on app communication until after GEISA 1.0.0)
- Inspect and control application packet flows

## 14.4 Networking Control

These permissions relate to controlling app access to the network.

By default, apps are not given any network access.

- Which network interfaces (none by default)
- Which network ports (none by default)
- Instantaneous Bandwidth
- Average network volume over a period (e.g. 1 hour, 24 hours)
- Direct versus Indirect Access
- Allowed destination addresses
- Allowed destination ports

## 14.5 API Control

These permissions realte to controlling app access to the network.

- Metrology Acccess (e.g. 1-second RMS, Waveforms)
- Actuator Access (if present)
- Sensor Access (if present)
- Hardware Access (GPIO, I2C, SPI, etc.)
- Inter-App Communication (**NOTE** perhaps hold off on app communcation until after GEISA 1.0.0)
- Device-to-Device Communcation (**NOTE** hold off until after GEISA 1.0.0)

## 14.6 Container Resource Management

Container resource limits shall include the following:

- CPU limit (% of CPU)
- Memory Limit (in 1K units)
- Persistent Storage Limit (in 1K units)
- Non-Persisten Storage Limit (in 1K units)
- Allowed Network Bandwidth (in 1K units)
  - Ongoing Limit Outbound
  - Ongoing Limit Inbound
  - Burst Limit Outbound
- Allowed networking interfaces
  - HAN
  - LAN
  - FAN
- Defin Container Access Levels
  - Level 0 - Read and Control - Core Features - Immutable
  - Level 1 - Read and Control - Utility
  - Level 2 - Read only

CHAPTER

FIFTEEN

# APPLICATION MANAGEMENT

Application management is the process of deplying, activating, deactivating, and decommissioning applications in the GEISA EE.

## 15.1 Metadata

GEISA's application management system follows a model similar to Amazon IoT Greengrass or Microsoft Azure IoT in this applications have a recipe, manifest, or other set of meta-data describing the requirements and dependencies.

GEISA application meta-data shall include:

- Name
- Description
- Version
- **Hash of the application image**
    - The GEISA EE shall not activate an application unless the hash of the image matches the hash in the meta-data
- System Resources Required
- **External Dependencies**
    - GEISA applicaitons should be as self-contained as possible,

  with all necessary dependencies contained with the application artifacts
- **Application Configuration**
    - GEISA appliciations may need basic information to initialize

  such as the URL of a server. The system operator should be able to change the configuration information without needing to redeploy the application.
- **Launch Stragety**
    - Includes details such as whether the application should

  automatically be restarted if it fails, and how manhy failures with a given period of time constitues a permanent failure.

Here is an example of meta-data.

```
{
  "geisa": {
    "com.example.HelloWorld": {
```

(continues on next page)

(continued from previous page)

```
      "author": "Some Company",
      "version": "1.0.0",
      "arch" : "arm32",
      "clib" : "musl",
      "artifiacts": {
        "overlay" : "zip"
      }
    }
  },
  "geisa_sdk_version": "1.0.0"
}
```

## 15.2 Application Artifacts

The GEISA Application artifacts includes all parts of the application that are installed within the GEISA EE, including:

- Executables
- Libraries
- Configuration

## 15.3 Application Management Protocol

**Note**

It is not clear this should be part of the GEISA 1.0 specification, especially given that a vendor/utility may already have protocols for app management.

It may be useful ot make this an “after 1.0” GEISA release requirement, or a future GEISA Extension.

The most immediate need is for GEISA apps to run on multiple platforms from multiple vendors/utilities, perhaps using deployment techniques that are already used by vendor/utilty.

GEISA needs to select a specific application management protocol to enable interoperability. Give the scale of GEISA (millions of devices), it is critical that the protocol be as lightweight as possible.

GEISA will use the OMA Specworks LWM2M specification.

LWM2M originated with COAP, and has been expanded to other protocols including HTTP and MQTT.

CHAPTER

SIXTEEN

# APPLICATION PROGRAMMING INTERFACE (API)

For GEISA applications to be **source portable** between multiple platforms, the GEISA standard establishes a well define set of APIs between the application and the common operating environment.

TODO: Copy info from Confluence

CHAPTER

SEVENTEEN

# SECURITY

Security is fundamental to the GEISA specification, so much so that it is included in the name itself: Grid Edge Interoperability and **Security** Allicance.

The following are the minimum criteria needed success of the GEISA specification and for the implmeentor.

- Clear expections must be illustrated outlining what responsibilities fall to the GEISA team vs the entities or individuals implementing the specification. This will be ferred to as the "Shared Responsiblity Model".
- The specification must affort an implementor the ability(ies) to comply with applicable standards, frameworks, certifications, etc. such as IEC 62433 without an undue amount of overhead

## 17.1 GEISA Responsibilities

TODO: Copy from Confluence pages

## 17.2 Implementer Responsibilities

TODO: Copy from Confluence pages

# CHAPTER EIGHTEEN

# REVISION HISTORY

Version 0.1.0 - 2025-06-17 - Early (and incomplete) Draft