UNSW
AUSTRALIA

School of Computer Science and Engineering
Faculty of Engineering
The University of New South Wales

A Modern IEEE 2030.5 Client Implementation

by Ethan Dickson

Supervised by Jawad Ahmed
Assessed by Nadeem Ahmed

Thesis C Report
Submitted November 2023

Thesis submitted as a requirement for the degree of
Bachelor of Engineering in Software Engineering

Abstract

IEEE 2030.5 is a standardised application-layer communications protocol between end-user energy devices, and electric utility management systems. In recent years, the protocol has seen unanimous adoption, and also proposed adoption by DNSP's in Australia, and the United States of America. This thesis will discuss how we have leveraged modern programming techniques, and the Rust programming language, to produce a safe, secure, robust, reliable, and open-source implementation of a IEEE 2030.5 client library that can be used to develop software for use in the smart grid ecosystem.

Acknowledgements

I would like to thank Jawad Ahmed and Nadeem Ahmed for their guidance as supervisor and assessor, respectively.

I would also like to thank Neel Bhaskar for his work on an IEEE 2030.5 server implementation as part of research at UNSW previously.

I would also like to acknowledge the efforts of all contributors to free and open-source software, especially in the Rust ecosystem. This thesis simply wouldn't have been possible without them.

Finally, I would like to thank my friends and family for their support throughout my degree, and this thesis.

Abbreviations

API Application Programming Interface

ARENA Australian Renewable Energy Agency

BSGIP Battery Storage and Grid Integration Program

CPUC California Public Utilities Commission

CPU Central Processing Unit

CSE Computer Science and Engineering

CSIP Common Smart Inverter Profile

CSIP-AUS Common Smart Inverter Profile Austrai

CVE Common Vulnerabilities and Exposures

DER Distributed Energy Resources

DNS Domain Name System

DNSP Distribution Network Service Provider

DNS-SD Domain Name System - Service Discovery

ECC Electric Curve Cryptography

EPRI Electric Power Research Institute

EXI Efficient XML Interchange

FS Function Set

HAN Home Area Network

HTTP Hypertext Transfer Protocol

IEEE Institute of Electrical and Electronics Engineers

I/O Input / Output

IoT Internet of Things

IP Internet Protocol

OOP Object-Oriented Programming

OS Operating System

REST Representational State Transfer

RSA Rivest-Shamir-Adleman

SEP Smart Energy Profile

SIP Smart Inverter Profile

TCP Transmission Control Protocol

TLS Transport Layer Security

UDP User Datagram Protocol

UNSW University of New South Wales

XML Extensible Markup Language

XSD XML Schema Definition

Contents

List of Figures

Chapter 1

Introduction

Society faces a growing need for reliable, sustainable and affordable electricity. One such way we have attempted to address this problem is via the invention of the 'smart grid', an electric grid assisted by computers, where by communication is enabled between electric utilities and end-users via a computer network, such as the Internet.

The portion of the smart grid that exists in the end-user environment are end-user energy devices. This category of end-user energy devices further encompasses the category of "Distributed Energy Resources", devices that deliver AC power to be consumed in the residence and/or export AC power back to the electric grid. Examples include solar inverters, household solar batteries, and biomass generators. [oEE18]

Through the use of distributed energy resources, fossil fuel based energy generation can be more readily replaced with clean, renewable energy, the need for which is of growing importance as we seek to address the threat of global climate change. Of great importance to the success of Distributed Energy Resources is their integration, and ongoing management as part of the broader electric grid.

End-user energy devices may require communication with electric utilities for the purpose of managing electric supply and demand, monitoring usage, and ensuring end-users are compensated, and charged, for their energy supply and demand, respectively.

The 2030.5 protocol is an IEEE standardised communication protocol purpose built for securely integrating end-user energy devices, and therefore Distributed Energy Resources into the wider electric grid. Since it's inception in 2013, the protocol has seen both minor and major revisions, and has seen unanimous adoption by DNSPs in both Australia, and the United States of America - in the Australian Common Smart Inverter Protocol (CSIP-AUS) [Age23] and Californian Smart Inverter Protocol [Lum16], respectively.

Thus, the goal of this thesis is to implement a safe, secure, reliable and performant framework for developing IEEE 2030.5 clients for use in the smart grid ecosystem.

Chapter 2

Context

IEEE 2030.5 is but one protocol designed for communication between end-user energy devices and electric utilities. Although IEEE 2030.5 is a modern protocol, it is not wholly new or original, rather it is the product of historically successful protocols, designed with the same aim. In this section we'll examine the predecessors to IEEE 2030.5, and their influence on the standard.

2.1 Smart Energy Profile 1.x

Developed by ZigBee Alliance, and published in 2008, Smart Energy Profile 1.x is a specification for an application-layer communication protocol between end-user energy devices and electric utilities. The specification called for the usage of the "ZigBee" communication protocol, based off the IEEE 802.15.4 specification for physical layer communication. [All13,]

The specification was adopted by utilities worldwide, including the Southern California Edison Company, who purchased usage of the system for $400 million USD. [Hei08,]

According to SunSpec in 2019, over 60 million smart meters are still deployed under ZigBee Smart Energy 1.x, with 550 certified SEP 1.x products. [All17,]

2.2 SunSpec Modbus

Referenced in the specification as the foundations for IEEE 2030.5 is the Sunspec Alliance Inverter Control Model, which encompasses the SunSpec Modbus Protocol. Sunspec Modbus is an extension of the Modbus

communication protocol, also designed for end-user energy devices, and was published, yet not standardised, in 2010. The protocol set out to accomplish many of the same goals as IEEE 2030.5 does today. Tom Tansy, chairman of the SunSpec Alliance pins the goal of the protocol as to create a 'common language that all distributed energy component manufacturers could use to enable communication interoperability'. [Tan21,]

2.3 IEEE 2030

IEEE 2030 was a guide, published in 2011, to help standardise smart grid communication and interoperability, and describe how potential solutions could be evaluated. A major goal of these potential communication protocols is that, by their nature of existing in the end-user environment, they were to prioritise the security of all data stored and transmitted, such that communication between electric utilities cannot be intercepted, monitored or tampered by unauthorised users.

Furthermore, the protocol was to ensure that an electric grid denial of service cannot be brought about by attacks on smart grid communication infrastructure. [SVC+20] [oEE11,]

2.4 Producing IEEE 2030.5

In the interest of interoperability with a future standard, the SunSpec alliance donated their SunSpec Modbus protocol to form IEEE 2030.5. Simultaneously, ZigBee Alliance was looking to develop Smart Energy Profile 2.0, which would use TCP/IP. At this point the SunSpec Alliance formed a partnership with ZigBee Alliance, and IEEE 2030.5 was created as a TCP/IP communication protocol that is both SEP 2.0, and interoperable with SunSpec Modbus. [Tan21,]

Likewise, IEEE 2030.5 works to improve the security of smart grid communication protocols, and the guiding principles put forward by IEEE 2030.

The extensibility of the protocol was also considered in it's design. The specification provides a standard method for extending it's functionality, such that legislative, state-specific, and proprietary extensions of the standard can be developed whilst retaining the protocol's core design. Examples of this include the later discussed CSIP and CSIP-AUS, where both allow for clients & servers to be deployed under a different model from that describe in the specification, whilst CSIP-AUS extends the possible structured data that can be communicated between clients & servers to better fit the requirements of electric utilities in Australia.

With IEEE 2030.5 extending and combining these existing, widely used, protocols, we're reassured it actually solves the problems faced by utilities and smart grid device manufacturers alike, and wasn't created in a vacuum, unaware of real world requirements, or the needs of device manufacturers and electric utilities.

Chapter 3

Background

3.1 High-Level Architecture

The IEEE 2030.5 protocol follows a REST API architecture, and as such, adopts a client-server model.

Transmitted between clients and servers are 'Resources', all of which are defined in a standardised schema, an XSD. Despite the client-server model, the IEEE 2030.5 specification purposefully does not make distinctions between clients and servers, as to avoid resources having differing behaviours on each. Rather, a server simply exposes resources to clients, and clients retrieve, update, create and delete resources on servers. Servers communicate with many clients, and when following a set of requirements detailed in the specification, clients can communicate with multiple servers.

Being the product of existing technologies, the IEEE 2030.5 resources cover a wide range of applications, and as such, the specification logically groups resources into discrete 'function sets', of which there are twenty-five. Device manufacturers or electric utilities implementing IEEE 2030.5 need only communicate resources from function sets relevant to the purpose of the device.

The specification defines two methods by which clients retrieve resources from server. The default method has clients 'poll' servers for the latest versions of resources on a timed interval. The second, more modern, and more scalable method, has clients 'subscribe' to a resource, after which they will be sent notifications containing any changes to the subscribed resource from the server, without needing to poll.

Despite this, whether a resource can be subscribed to can be further refined by the server exposing the resource.

For that reason, it is often required that clients employ both polling, and subscriptions when maintaining the latest instance of a resource. [Wei21] [oEE18]

3.2 Protocol Design

Resources are transmitted between clients and servers using HTTP/1.1, over TCP/IP, optionally using TLS. As a result, the protocol employs the HTTP request methods of GET, POST, PUT and DELETE for retrieving, updating, creating and deleting resources, respectively.

The specification requires that TLS certificates be signed, and all encryption done, using ECC cipher suites, with the ability to use RSA cipher suites as a fallback.

The standardised resource schema is defined in a XSD, as all transmitted resources are represented using either XML, or EXI, with the HTTP/1.1 `Content-Type` header set to `sep+xml` or `sep-exi`, respectively.

In order to connect to servers, clients must be able to resolve hostnames to IP addresses using DNS. Similarly, the specification permits the ability for clients to discover servers on a local network using DNS-SD.

3.3 Usage

The IEEE 2030.5 function sets cover a wide range of applications and uses, aiming to support as many end-user energy devices as possible. A subset of these possible use cases are as follows:

- (Smart) Electricity Meters can use the 'Metering' function set to 'exchange commodity measurement information' using 2030.5 resources. [oEE18]
- Electric Vehicle chargers may wish to have their power usage behaviour influenced by resources belonging to the 'Demand Response and Load Control' function set.
- Solar Inverters may use the 'Distributed Energy Resources' function set such that their energy output into the wider grid can be controlled by the utility, as to avoid strain on the grid.

Chapter 4

Adoption

Further proving that the IEEE 2030.5 protocol is worth implementing is it's adoption by electric utilities, as well as the tariffs and guidelines created by government energy bodies mandating it's use. Consequently, many implementations of the protocol exist already, with the vast majority of them proprietary, or implementing proprietary extensions of the standard. In this section, we will examine both, and discuss how they may influence our open-source implementation.

4.1 California Public Utilities Commission

'Electric Rule 21' is a tariff put forward by CPUC. Within it are a set of requirements concerning the connection of end-user energy production and storage to the grid. In this tariff, it is explicitly clear that "The default application-level protocol shall be IEEE 2030.5, as defined in the California IEEE 2030.5 implementation guide" [GC23,]. Given that the state of California was among the first to make these considerations to the protocol, it's likely that future writers of legislation or tariffs will be influenced by Rule 21, particularly how they have extended the protocol to achieve scale in the realm of smart inverters. For that reason, we let the implementation models for the Californian IEEE 2030.5 implementation guide influence our own development of the protocol, whilst of course still adhering to the specification.

Relating directly to use of the IEEE 2030.5 protocol at scale are the high level architecture models defined in the California SIP implementation guide.

Individual/Direct Model

Under this model there is a direct communication between an IEEE 2030.5 compliant client, in this case a solar inverter, and a IEEE 2030.5 compliant server, hosted by the electric utility. This model alone does not impose any additional restrictions over those already existing in Rule 21. It requires the inverter to be a 2030.5 Client, and be managed individually by the server.

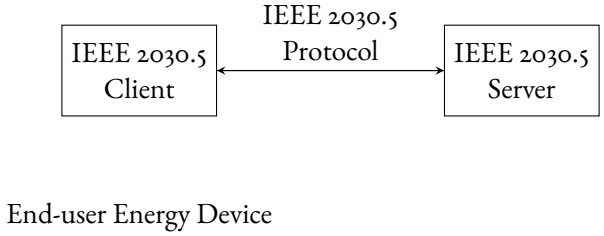

Figure 4.1: The Individual/Direct IEEE 2030.5 Model, as defined by California SIP

Aggregated Clients Model

The aggregated clients model, outlined in the implementation guide, is one preferred for use by electric utilities. Under this model, the 2030.5 client is but an aggregator communicating with multiple smart inverters, acting on their behalf. The rationale behind this model is to allow utilities to manage entire geographical areas, or a specific model of end-user energy device as though it were a single entity.

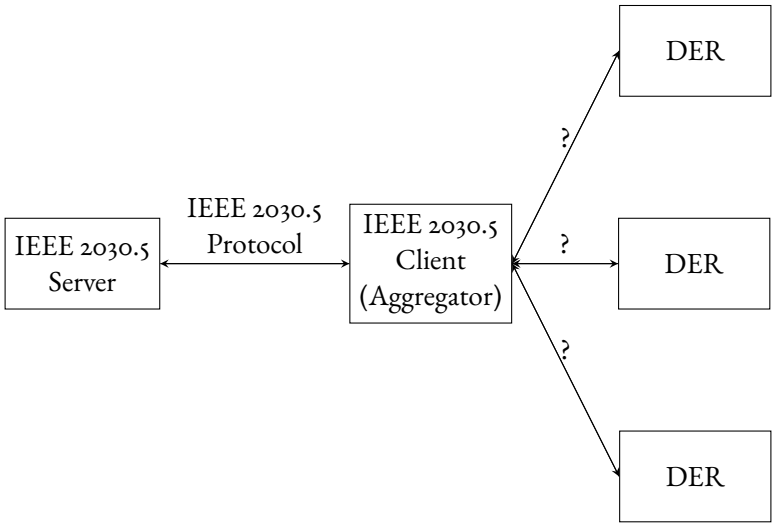

Figure 4.2: The Aggregated Clients IEEE 2030.5 Model, as defined by California SIP

The IEEE 2030.5 server is not aware of this aggregation, as the chosen communication protocol between an aggregator client and an end-user energy device is unspecified and out of scope of the model, as indicated in Figure 4.2. Under this model, aggregators may be communicating with thousands of IEEE 2030.5 compliant clients. For this reason, the California SIP mandates the subscription/notification retrieval method be used by clients, rather than polling. This is mandated in order to reduce network traffic, and of course, allow for use of the protocol at scale.

Given the circumstances of this model, the aggregator IEEE 2030.5 client is likely to be hosted in the cloud, or on some form of dedicated server.

4.2 Australian Renewable Energy Agency

"Common Smart Inverter Profile" (Common SIP) (CSIP-AUS) is an implementation guide developed by the "DER Integration API Technical Working Group" in order to "promote interoperability amongst DER and DNSPs in Australia". The implementation guide has DER adhere to the IEEE 2030.5 spec, whilst further leveraging the aforementioned CPUC California SIP, including support for use of the client aggregator model, and the mandated use of subscription/notification retrieval by those aggregator clients. [Age23,]

Most importantly, the Australian Common SIP extends upon existing IEEE 2030.5 resources to support `Dynamic Operating Envelopes`, and it does so whilst still adhering to the IEEE 2030.5 specification. As per the specification, resource extensions are to be made under a different XML namespace, in this case `https://csipaus.org/ns`, where extension specified fields are to be appropriately prefixed with `csipaus`. [Age23,]

4.3 SunSpec Alliance

As of present, the specification behind the aforementioned SunSpec Modbus is still available and distributed, as it's "semantically identical and thus fully interoperable with IEEE 2030.5". The primary motivation for implementing SunSpec Modbus is it's compliance with IEEE 1547, the standard for the interconnection of DER into the grid. [All,]

4.4 Open-source implementation

Despite the protocol's prevalence and wide-spread adoption, the vast majority of implementations of the standard are proprietary, and thus cannot be distributed, modified, used, or audited by those other than the rightsholder, with the rightsholder conditionally providing commercial licenses for a monetary fee.

For that reason, any and all open-source contributions involving IEEE 2030.5 actively work to lower the cost of developing software in the smart grid ecosystem, and are all necessary in ensuring the protocol can be as widely adopted as possible, as to then incorporate as many end-user energy devices into the smart grid as possible.

In this section we will discuss existing open-source implementations of the standard, and identify the tradeoffs in each.

4.4.1 Electric Power Research Institute Client Library: IEEE 2030.5 Client

One of the more immediately relevant adoptions of the protocol is the mostly compliant implementation of a client library by EPRI, in the United States of America. Released under the BSD-3 license and written in the C programming language, the implementation comes in at just under twenty-thousand lines of C code. Given that a IEEE 2030.5 client developed using this library would require extension by a device manufacturer, as to integrate it with the logic of the device itself, EPRI distributed header files as an interface to the codebase. For the purpose of demonstration and testing, they also bundled a client binary that can be built and executed, running as per user input.

The C codebase includes a program that parses the IEEE 2030.5 XSD and converts it into C data types (structs) with documentation. This is then built with the remainder of the client library.

The implementation targets the Linux operating system, however, for the sake of portability, EPRI defined a set of header file interfaces that contained Linux specific API calls, such that they could be replaced for some other operating system.

These replaceable interfaces include those for networking, TCP and UDP, and event-based architecture, using the Linux `epoll` `syscall`, among others.

The IEEE 2030.5 Client implementation by EPRI states, in it's User's Manual, that it almost perfectly conforms to the IEEE 2030.5 specification according to tests written by QualityLogic. The one exception to this is that the implementation does not support the subscription/notification mechanism for resource retrieval, as of this

report. This is particularly unusual given that the California SIP mandates the use of subscription/notification under the client aggregator model, a model of which this implementation was targeted for use in.

One potential pain point for developers utilising this library is the ergonomics of the interface provided. The C programming language, whilst universal, lacks many features present in more modern programming languages that can be used to build more ergonomic and safe interfaces. For instance, the codebase's forced usage of global mutable state for storing retrieved Resources, goes against modern design principles, and could be easily avoided in a more modern programming language.

Furthermore, being written in a language without polymorphism, be it via monomorphisation, dynamic dispatch or tagged unions, C forgoes a great deal of type checking that could be used to make invalid inputs to the interface compile-time errors, instead of run-time errors. For example, due to the lack of tagged unions (or algebraic sum types) in C, many functions exposed to users accept a typeless pointer, which is then cast to a specific type at runtime, providing no compile-time guarantees that that type conversion is possible, or that the underlying input is interpreted correctly, or even that the given pointer points to memory that it the process is capable of reading and/or modifying.

In contrast to this, the library is sufficiently modular, providing interfaces across multiple C headers, where users of the library need only compile code that is relevant to their use-case. For example, developers building IEEE 2030.5 clients that need not handle DER function set event resources are not required to compile and work with the code responsible for managing them.

Additionally, the usage of the `epoll` `syscall` and the library's state-machine centric design lends itself to the scalability of the client, allowing it to handle operations asynchronously, and better perform it's role operating under the client aggregation model. [Ins18,]

4.4.2 Battery Storage and Grid Integration Program: envoy-client

A newer implementation of the protocol is `envoy-client`, developed by BSGIP, an initiative of The Australian National University. Of note is that this client has been open-sourced ahead of the release of the IEEE 2030.5 Server library implementation `envoy`, and provides a very bare implementation of core IEEE 2030.5 Client functionality written in Python, and therefore provides a far more modern interface than that of the EPRI library. [BSG21]

The library was released publicly in 2021, and has seen virtually no updates since. It is possible the client will

see a major update when the `envoy` library is released, as improvements to 2030.5 test tooling were indicated as being developed. [Cut22]

Despite being written in Python, a programming language with support for asynchronous programming, async await syntax is not present in the codebase. For that reason, it's likely the user of the library will be required to wrap the provided codebase with async python in order for it to scale as a client aggregator. In theory, it's also likely that the Python Global Interpreter Lock would impact the ability for the client to take advantage of multiple threads, and potentially lead to performance issues at scale.

Despite the library's current incomplete state, a dynamically typed programming language lends itself well to the nature of 2030.5 client-server resource communication, as it allows for deserialisation of XML resources to dynamically typed Python dictionaries, where each key and value can have a different type, using the python "Any" type. This lack of type-checking on resources leads to faster development times, and is part of BGSIP's justification for the client & server being implemented in Python.

Under this design, the checking of XML attributes and elements is left to the user of the library - they must ensure that the given XML resource is of the same type as expected, and that it contains the expected fields.

By the very nature of dynamic typing, Python provides no guarantees that parts of resources accessed are present, where unchecked accesses to data may lead to runtime errors.

With minimal dependencies, and without an interface for TLS, the library is as portable as Python itself.

Under these circumstances, the library is an ideal tool for quickly testing an IEEE 2030.5 Server implementation, but would likely struggle in real-world use, aggregating on behalf of clients.

Chapter 5

Implementation Design

Bibliography

[Age23] Australian Renewable Energy Agency. Common smart inverter profile - australia. https://arena.gov.au/assets/2021/09/common-smart-inverter-profile-australia.pdf, January 2023.

[All] SunSpec Alliance. Sunspec modbus. https://sunspec.org/sunspec-modbus-specifications/, 2021.

[All13] ZigBee Alliance. Zigbee specification faq. https://web.archive.org/web/20130627172453/http://www.zigbee.org/Specifications/ZigBee/FAQ.aspx, June 2013.

[All17] SunSpec Alliance. Ieee 2030.5/ca rule 21 foundational workshop. https://sunspec.org/wp-content/uploads/2019/08/IEEE2030.5workshop.pdf, June 2017.

[BSG21] BSGIP. envoy-client. https://github.com/bsgip/envoy-client, 2021.

[Cut22] CutlerMerz. Review of dynamic operating envelope adoption by dnsps. https://arena.gov.au/assets/2022/07/review-of-dynamic-operating-envelopes-from-dnsps.pdf, August 2022.

[GC23] Pacific Gas and Electric Company. Electric rule no.21. https://www.pge.com/tariffs/assets/pdf/tariffbook/ELEC_RULES_21.pdf, February 2023.

[Hei08] Bob Heile. Zigbee smart energy. https://www.altenergymag.com/article/2008/10/zigbee-smart-energy/468/, January 2008.

[Ins18] Electric Power Research Institute. Electric power research institute ieee 2030.5 client user's manual. https://www.epri.com/research/products/000000003002014087, July 2018.

[Lum16] Gordon Lum. California use case for ieee2030.5 for distributed energy renewables. https://smartgrid.ieee.org/bulletins/december-2016/california-use-case-for-ieee2030-5-for-distributed-energy-renewables, December 2016.

[oEE11] Institute of Electrical and Electronics Engineers. Ieee guide for smart grid interoperability of energy technology and information technology operation with the electric power system (eps), end-use applications, and loads, 2011.

[oEE18] Institute of Electrical and Electronics Engineers. Smart Energy Profile Application Protocol (2030.5-2018), December 2018.

[SVC+20] Partha S. Sarker, V. Venkataramanan, D. Sebastian Cardenas, A. Srivastava, A. Hahn, and B. Miller. Cyber-physical security and resiliency analysis testbed for critical microgrids with ieee 2030.5, 2020.

[Tan21] Tom Tansy. Get on the modbus: An explanation of ieee 1547 and why it matters for solar installers. https://solarbuildermag.com/news/get-on-the-modbus-an-explanation-of-ieee-1547-and-why-it-matters-for-solar-installe March 2021.

[Wei21] Ben Weise. On the implementation and publishing of operating envelopes. https://arena.gov.au/assets/2021/04/evolve-on-the-implementation-and-publishing-of-operating-envelopes.pdf, april 2021.