Name Server Requirements and Design for a Hybrid Control System

Beau Harrison

High-Level Goals

- Centralized Source of Truth
  - Serve as a single, reliable source for all device definitions, property configurations, and associated metadata.
- Comprehensive and Scalable Design
  - Support ACNET, EPICS, and future control systems, ensuring flexibility for evolving architectures.
- Enhanced User Empowerment
  - Provide users and application developers with introspection capabilities for metadata and functionality of devices, properties, and their capabilities.
- Interoperability
  - Leverage gRPC as the protocol for service-to-service communication.
    - This matches the architecture model for the rest of the ACORN project.
  - Use RBAC via HTTP/gRPC to adhere to our zero-trust security model.

Requirements: Functional Requirements

1. The Name Server shall store metadata describing the information available in the control system.
   - The Name Server is a repository of all the metadata necessary to describe the information and access methods for the information available within the control system. The nameserver is not responsible for putting in place any configuration or setting on an IOC or ACNET Front-end.
2. The Name Server shall not perform any external actions.
   - The Name Server is purely a metadata repository and does not directly interact with or configure external systems.
3. The Name Server shall support a device model.
   - A device is a container that includes:
     - Properties or data channels (e.g., EPICS PVs).
     - Alarm configurations.
     - Node information (physical location, hostname, IP address).
     - Access controls (RBAC) per property.
     - Property transforms.

Requirements: Functional Requirements (cont.)

4. The Name Server shall map one-to-many EPICS Process Variables to a device.
5. The Name Server shall map one-to-many transforms to each EPICS Process Variable.
   - A transform is a string providing the instructions on how to convert the raw data value into units usable by code or by a user.
6. The Name Server shall store metadata describing transforms.
7. The Name Server shall store rendering metadata for a property.
8. The Name Server shall map a node to a device.
9. The Name Server shall store metadata describing a node.
   - Metadata includes the IP address and hostname of the node.
10. The Name Server shall map location information to a node.
11. The Name Server shall provide efficient search capability by **device** name.
12. The Name Server shall provide efficient search capability by **node** name.
13. The Name Server shall provide efficient search capability by **IP** address.

Requirements: Functional Requirements (cont.)

14. The Name Server shall provide efficient search capability by property name.
15. The Name Server shall map alarm configuration to properties.
    - Valid alarm configurations shall include the EPICS alarm values and the Analog/Digital alarm values for ACNET.

Requirements: Non-Functional Requirements

1. Scalability
   - Efficiently handle devices numbering in the tens of thousands with low-latency gRPC queries.
2. Caching
   - Use in-memory caching (e.g., Redis or similar) to minimize query times for frequently accessed data. While LOOKUP provides a caching solution, it is written in C++ with only consideration for the legacy ACNET system. Due to the implementation constraints of the existing system, there is likely not useful logic to save. Redis or a similar in-memory solution could be used to align with the new architecture and provide a distributed cache for scalable workloads. Integration should ensure consistency between the cache and the database, with mechanisms to synchronize updates effectively.
3. Usability
   - Leverage human-centered design principles to align with user workflows.
   - Provide robust introspection capabilities to surface data structure functionality and metadata.

Requirements: Non-Functional Requirements (cont.)

4. Reliability
   - Ensure high availability through fault-tolerant architecture and robust backup mechanisms.
5. Future-proofing
   - Build with extensibility in mind to support evolving data structures, field systems, and control system architectures.

Updated System Design: Core Components

1. Device Registry
   - Central repository for device definitions, properties, and metadata.
   - Incorporate device class hierarchy inspired by CERN.
2. API Layer
   - gRPC as the primary protocol for service communication.
3. Management Interface
   - Flutter Web app for intuitive GUI-based operations.
   - CLI for efficient bulk operations.
     - Requirement needs verification as it could be mitigated through GUI design.
   - Visualization tools to represent device hierarchies and relationships.
4. Database
   - Use PostgreSQL for relational storage.
   - Design tables for:
     - Devices
     - Properties
     - Classes
     - Metadata (e.g., alarm configurations, rendering information, and transformation logic)
     - Node configurations (e.g., IP, protocol)
5. Caching Layer
   - In-memory caching for commonly queried fields to ensure high performance.

Updated System Design: Core Components (cont.)

6. Security Layer
   - Centralized Keycloak integration for authentication and RBAC.
7. Version Control
   - Manage versions of device definitions and classes with rollback capability.
8. Introspection and Metadata System
   - Provide users with visibility into device and property capabilities.
   - Include rich metadata about alarms, data rendering, and transformation logic.

Updated System Design: Next Steps

1. Data Modeling
   - Start with a strawman data model, incorporating device aggregation, property mappings, and metadata storage.
2. Functional Prototype
   - Build a small-scale prototype integrating a subset of ACNET and EPICS devices for testing.
3. Collaboration
   - Include INL’s Human Factors team for user testing and workflow analysis. Their insights will be incorporated by conducting workflow observations, gathering user feedback during prototyping phases, and validating usability improvements. INL's recommendations can inform interface design, visualization choices, and the organization of data to ensure the system meets user expectations and minimizes friction in their workflows.
   - Reevaluate ANL’s role in the project; they may contribute to EPICS integration but lack expertise in broader Computer Science and data architecture needs.
4. User Feedback
   - Solicit feedback from a small group of stakeholders on the introspection system and data models.