EPICS & Phoebus Developers Meeting 2026

Documentation

List of tasks to address:
https://github.com/ControlSystemStudio/phoebus-codeathon/tree/master/documentation-tasks

Who are the users?

ChannelFinder
Roles:

Operator: Some one who is searching for channels PV names via any of the clients (web, Phoebus, etc…)

- Tutorial
    - A description of what is channel finder and how to use it, and why it exists.
- Guide
    - A list of available interfaces (Phoebus, openapi, PVInfo)
    - How to search for channels
    - How to use ChannelFinder to archive PVs

System Administrator: Deploy and configure the services, defines the tags/properties, setups recsync and other data aggregator/input pipelines

- How to
    - Installation instructions
    - How to deploy / configure receiver
- Reference
    - Configuration properties reference
- Explanation
    - Architecture explanation
        - Contains links to other services
- Guide
    - How to configure the auto-archiving feature

Developer: Uses the ChannelFinder REST API for developing various applications (clients, scripts, etc…)

- Tutorial:

        - Assumed Prerequisites: What is REST/HTTP
        - Given:
            - curl, Python requests, POSTman
        - Task to accomplish?
            - Build some application accomplishing:
                - Channel Operation
                - Property Operation
                - Tag operation
- How to Guide
    - Potentially: Q-A format, e.g. wiki.debian.org/WhereIsIt
    - I want to get the owner of channel x…
    - I want to set property y of channel x…
    - I want to know how many channels match pattern z…
    - How to configure reccaster for your IOC (guide)
- Explanation
    - Potentially more narrative information in reference
        - Use-case with reference entry? *This might be handy if you were trying to do task X in the how-to guide…*
- Reference
    - Already exists and is defined in bare form here https://channelfinder.readthedocs.io/en/latest/api.html

Contributor: who contributes code

- Guidelines:
    - What do you need to consider before contributing
    - Legal part/licensing
- Automated tests
- API changes
- Documentation
    - Rules for contributing and developing documentation
- Javadoc

Olog

Roles: migrated to the Phoebus Olog issue: https://github.com/Olog/phoebus-olog/issues/254

Alarm Services:

Roles:

Operator: Who sees the alarms, gets the notifications (emails, sms, …), is expected to respond to alarms. Search for alarms and manage the alarm history…

- Tutorials: Get an alarm, check guidance, acknowledge it, filter the table, search alarm logs

- How-to Guides: How to switch to maintenance mode, how to snooze an alarm, how to find a non latching alarm, how to ack multiple alarms
- Reference: list of all the operations that appear in the context menu
- Explanation: how are alarms generated, what is the life cycle of alarms (flow chart)

Integrator: Configures the alarms (set IOC limits, add to alarm trees, instructions & guidance), notification policies

- How-to Guides: Add/Remove/Configure an alarm to the alarm tree
- Reference: All the fields of the alarm tree that can be configured, all configuration options, all commands that can be invoked…
- Explanation: How to configure alarm limits correctly, how & which PVs to add, the Philosophy on which PVs to add, Life cycle of alarms in both IOCs and Services

System Administrator: deploys kafka, elasticsearch, alarm services, git, etc…Monitoring the health of the services. Manage data and configuration management - backup and recovery

- How-to Guide: Deploy via ansible, Deploy in containers, Deploy on a VM, How to detect and recover from health issues, how to configure alarm topics correctly in Kafka, Setup backup and recovery
- Reference: List of all deploy methods, health endpoints, server commands (import, export,)
- Explanation: The architecture of all the alarm services, how all the alarm services work together

Developer: Uses the Kafka messages and Service APIs to develop new apps

- Tutorial: simple kafka consumer which monitor alarms for an example topic
- How-to Guide: How to create a kafka consumer to the Alarm Services, How to create a report from alarm logs (elastic)
- Reference: REST API, Kafka Messages
- Explanation: The architecture of all the alarm services, how is the alarm state and config managed, how to parse the kafka messages

Contributor: contribute code to the Phoebus services in the repo

- How-to Guide: Contribution guidelines
- Reference: REST API, Kafka Messages
- Explanation:
    - The arch of the alarm micro services, which were the services split as they were, the scope of each service, alarm lifecycle…
    - javadocs

Archiver Appliance Services:

Roles:

Operator/Scientist: Anyone who wants to see/consume the history of PV values using clients like Data Browser.

- Explanation
    - What is the data you receive
    - Why data maybe missing
- Tutorial
    - Web based client
        - Fetch from a single
    - Other clients small tutorials
        - Data browser
- How to guide
        - How to use the default web based client
        - Add multiple pvs
        - Export the data
        - Change the post processor
    - Best practices for queries
- Reference
    - Want to know what gui clients there are
    - What python(other) clients there are
    - List of post processors

Integrator: Submits PVs to be archived, paused, resumed, and other management tasks. Including picking a policy.

- Tutorial
    - Submit a PV
- How to guide
    - Manage a pv (submit, configure, pause, resume)
    - Why a PV is not archiving
- Explanation
    - Good IOC/PV options on the IOC side
    - How the ‘engine’ works
    - Policy options
    - What does MONITOR vs SCAN mean

System Administrator: Deploys and configures the archiver appliance. (mariadb, storage backends, sets the policy file)
Likely also does some more difficult management tasks, such as converting PVs from one data source to another.

- Tutorial
    - Deploy the archiver
        - Simple version, i.e. docker-compose
- How to guide
    - Production ready deployment

        - Include backup strategy options
        - Backups of config database
    - Upgrading the archiver
    - Stopping and starting the wars
    - Fixing existing data
    - Troubleshooting missing data
    - Monitoring the health of the archiver
    - Migrate data
        - Failover mode
        - Consolidation
        - File formats
        - resharding
    - Writing a good policy file
    - Checking the status of a PV
    - Management of the PV
        - Rename, delete.
    - Identifying problematic PVs
    - How to customize the web ui
    - Good configuration of logs
    - Kube?
    - Redundancy
- Explanation
    - Overview of architecture on the ‘war’ level
    - Configuration database
    - Possible data errors (missing, or corrupted data)
    - Overview of the mgmt ui
        - Description of the reports
        - Description of the metric
        - Description of the storage metrics
    - Description of the picking appliance algorithm
    - ETL process
- Reference
    - Mgmt api
    - Settings file
    - The appliances xml file
    - Example Docker compose file
    - List of scripts

Developer: Develops clients to read data from the archiver, or to the management interface.

- Tutorial
    - Basic python or bash or curl script to fetch data
- Explanation
    - Pb api data stream

        - Json api
    - Reference
        - Protobuf spec
        - Mgmt api
        - Retrieval api
            - Getdata
            - Get data at time
            - Latestmetadata tag meaning

Contributor: who contributes code

- How to guide
    - Build the application
    - Run tests
    - Format code
- Explanation
    - Architecture
        - Clustering
        - Wars
        - Storage
- Reference
    - Java docs