

**CableLabs®**  
**Proactive Network Maintenance**

**Proactive Operations Platform, Application User Guide  
and Technical Report**

**CL-TR-ProOps-V01-190925**

**RELEASED**

**Notice**

**Some content from this technical report are reprinted with permission of SCTE•ISBE and the NCTA, the Cable-Tec Expo® 2019. Some content is copied and reused verbatim, and other content is edited for this technical report's needs. Additional material is added to this technical report for completeness.**

This CableLabs' technical report is the result of a cooperative effort undertaken at the direction of Cable Television Laboratories, Inc. for the benefit of the cable industry and its customers. You may download, copy, distribute, and reference the documents herein only for the purpose of developing products or services in accordance with such documents, and educational use. Except as granted by CableLabs in a separate written license agreement, no license is granted to modify the documents herein (except via the Engineering Change process), or to use, copy, modify or distribute the documents for any other purpose.

This document may contain references to other documents not owned or controlled by CableLabs. Use and understanding of this document may require access to such other documents. Designing, manufacturing, distributing, using, selling, or servicing products, or providing services, based on this document may require intellectual property licenses from third parties for technology referenced in this document. To the extent this document contains or refers to documents of third parties, you agree to abide by the terms of any licenses associated with such third-party documents, including open source licenses, if any.

© Cable Television Laboratories, Inc. 2019

## DISCLAIMER

This document is furnished on an "AS IS" basis and neither CableLabs nor its members provides any representation or warranty, express or implied, regarding the accuracy, completeness, noninfringement, or fitness for a particular purpose of this document, or any document referenced herein. Any use or reliance on the information or opinion in this document is at the risk of the user, and CableLabs and its members shall not be liable for any damage or injury incurred by any person arising out of the completeness, accuracy, or utility of any information or opinion contained in the document.

CableLabs reserves the right to revise this document for any reason including, but not limited to, changes in laws, regulations, or standards promulgated by various entities, technology advances, or changes in equipment design, manufacturing techniques, or operating procedures described, or referred to, herein.

This document is not to be construed to suggest that any company modify or change any of its products or procedures, nor does this document represent a commitment by CableLabs or any of its members to purchase any product whether or not it meets the characteristics described in the document. Unless granted in a separate written agreement from CableLabs, nothing contained herein shall be construed to confer any license or right to any intellectual property. This document is not to be construed as an endorsement of any product or company or as the adoption or promulgation of any guidelines, standards, or recommendations.

## Document Status Sheet

<b>Document Control Number:</b>	CL-TR-ProOps-V01-190925			
<b>Document Title:</b>	Proactive Operations Platform, Application User Guide and Technical Report			
<b>Revision History:</b>	V01 – 09/25/19			
<b>Date:</b>	September 25, 2019			
<b>Status:</b>	<del>Work in Progress</del>	<del>Draft</del>	<b>Released</b>	<del>Closed</del>
<b>Distribution Restrictions:</b>	<del>Author Only</del>	<del>CL/Member</del>	<del>CL/Member/Vendor</del>	<b>Public</b>

### Trademarks

CableLabs® is a registered trademark of Cable Television Laboratories, Inc. Other CableLabs marks are listed at <http://www.cablelabs.com/certqual/trademarks>. All other marks are the property of their respective owners.

# Contents

<b>1</b>	<b>SCOPE.....</b>	<b>7</b>
1.1	Introduction .....	7
1.2	Purpose .....	7
<b>2</b>	<b>INFORMATIVE REFERENCES .....</b>	<b>8</b>
2.1	Reference Acquisition.....	8
<b>3</b>	<b>TERMS AND DEFINITIONS .....</b>	<b>8</b>
<b>4</b>	<b>ABBREVIATIONS AND ACRONYMS.....</b>	<b>9</b>
<b>5</b>	<b>THE PROACTIVE OPERATIONS PLATFORM.....</b>	<b>10</b>
5.1	Control & Schedule Worker .....	12
5.2	Work Queues .....	12
5.3	Workers and Workflow Description.....	13
5.4	Config Files .....	14
5.5	Data Stores.....	14
5.6	Scheduling Considerations .....	15
5.7	Additional Notes about ProOps .....	15
<b>6</b>	<b>PNM EXAMPLE APPLICATION FOR PROOPS .....</b>	<b>16</b>
6.1	Base data polling (Pollers).....	16
6.2	Triggers (Analyzers).....	16
6.3	Actions.....	17
6.4	Modules .....	17
<b>7</b>	<b>CONFIGURING AN APPLICATION .....</b>	<b>19</b>
<b>8</b>	<b>ENVISIONED USE CASES .....</b>	<b>28</b>
<b>9</b>	<b>SYSTEM ARCHITECTURE AND WORKFLOWS .....</b>	<b>30</b>
9.1	Introduction .....	30
9.2	ProOps and CCF .....	30
9.2.1	Component Descriptions.....	30
<b>10</b>	<b>WORKERS AND WORKFLOWS .....</b>	<b>32</b>
10.1	Data Collection Worker .....	32
10.1.1	Functionality Description .....	32
10.1.2	Workflow.....	32
10.1.3	Job Description Syntax and Format .....	33
10.2	Downstream RxMER Statistics Worker .....	33
10.2.1	Functionality Description .....	33
10.2.2	Workflow.....	34
10.2.3	Job Description Syntax and Format .....	34
10.3	Downstream Channel Estimation Statistics Worker .....	35
10.3.1	Functionality Description .....	35
10.3.2	Workflow.....	35
10.3.3	Job Description Syntax and Format .....	35
10.4	Downstream Constellation Display Statistics Worker.....	36
10.4.1	Functionality Description .....	36
10.4.2	Workflow.....	36
10.4.3	Job Description Syntax and Format .....	37
10.5	CM Result Trigger Analyzer .....	37
10.5.1	Functionality Description.....	37

10.5.2	Workflow.....	38
10.5.3	Job Description Syntax and Format .....	38
10.6	Result DB Worker .....	40
10.6.1	Functionality Description .....	40
10.6.2	Workflow.....	40
10.6.3	Job Description Syntax and Format .....	41
10.7	ProOps Configuration Server.....	41
10.7.1	Functionality Description .....	41
10.7.2	RESTful APIs .....	41
10.8	ProOps Data Service.....	42
10.8.1	Functionality Description .....	42
10.8.2	RESTful APIs .....	42
11	CONCLUSION .....	47
APPENDIX I ACKNOWLEDGEMENTS .....		48

## Figures

Figure 1 – Depiction of ProOps Platform on Top of CCF and a Network .....	10
Figure 2 – Depiction of ProOps with OODA overlaid .....	12
Figure 3 – Depiction of the Example Application that comes with ProOps today .....	16
Figure 4 – Configuration Management Server Web GUI Index Page (task queue configurations).....	19
Figure 5 – Configuration Management Server – Defined Workers .....	20
Figure 6 – Job Scheduling .....	20
Figure 7 – Configuring Job Scheduling.....	21
Figure 8 – Configuring for T3 and T4 Device Event Log Errors .....	22
Figure 9 – Configuring the RxMER Statistics Worker.....	22
Figure 10 – Configuring a Data Collection Worker .....	23
Figure 11 – Channel Estimation Statistics Worker.....	24
Figure 12 – Top-level System Monitoring .....	24
Figure 13 – Detailed Worker System Monitoring .....	25
Figure 14 – System Monitoring Graph.....	26
Figure 15 – System Monitoring Graph of Data Collection.....	27
Figure 16 – ProOps with Data Store and CCF.....	30
Figure 17 – Data Collection Worker’s Workflow .....	32
Figure 18 – Downstream RxMER per Subcarrier Statistics Worker’s Workflow .....	34
Figure 19 – Downstream Channel Estimation Statistics Worker’s Workflow .....	35
Figure 20 – Downstream Constellation Display Statistics Worker’s Workflow .....	37
Figure 21 – CM Result Trigger Analyzer.....	38
Figure 22 – Result DB Worker’s Workflow.....	41

Tables

Table 1 – Data Types.....32

Table 2 – Data Collection Worker Job Description.....33

Table 3 – Downstream RxMER Per Subcarrier Statistics Worker Job Description .....34

Table 4 – Downstream Channel Estimation Statistics Worker Job Description.....35

Table 5 – Downstream Constellation Display Statistics Worker Job Description.....37

Table 6 – CM Result Trigger Analyzer Worker Job Description .....40

Table 7 – Result DB Worker Job Description .....41

# 1 SCOPE

## 1.1 Introduction

As part of its long-standing proactive network maintenance (PNM) project, CableLabs® has been assessing the needs of operators and vendors in the area of PNM with the goal of reducing adoption friction for members and vendors. We identified a few main issues, an important one of which is addressed by the work presented in this technical report: the Proactive Operations (ProOps) platform.

ProOps is a platform (environment, framework) for turning data into operations action. That includes proaction, when the data allow it. PNM data does, so we built an example application that comes with ProOps, which serves multiple purposes: as an example to show how to use ProOps, as a starting point for trying basic network data-driven PNM and reactive operations, and as a launch point for implementing and sharing PNM best practices.

To turn data into action, most operators rely on engineering and technician expertise. It is common to simply gather and plot the data, then look at the output. CableLabs built the cable modem validation application (CMVA) for that latter purpose (as well as for cable modem (CM) certification test automation and sharing). But without a human expert sifting through the data, not much can be done with it. CMVA is great for developing PNM ideas, but it still requires experts to do the next step, and developers to build solutions to try. That requires investment risk that we surmise is a roadblock to implementation of PNM. But for many operators, and some vendors, there just aren't enough available experts to do the work manually. The industry needs help getting over the hurdle of turning the data we exposed into action we can take with confidence.

ProOps was built to facilitate the automation of turning data into operations action. Generally, we identified the steps to accomplish that task as 1) data extraction (observe), 2) analysis across time and network elements (orient), 3) correlating problems and measuring severity (decide), and 4) defining work items that are worthy of attention (act). The steps can be labeled as observe, orient, decide, and act (OODA) to roughly follow the OODA process or OODA loop, which is a cyclic process developed by US Air Force Colonel John Boyd [Boyd], [OODA Loop]. Combat operations resemble network operations more than we care to admit perhaps, so the labels fit. Boyd systematized the combat operations process as a rapid cycle of the OODA loop. Likewise, network operations follows a similar process, and the concept helps explain how ProOps works.

In the future, we expect to release applications and modules to enhance existing applications, in the ProOps environment. Once ProOps is installed, new applications will work like updates to ProOps, making the operations impact even less. New applications and modules will interwork with existing applications in the same deployment of ProOps, or parallel deployments utilizing the same common collection framework (CCF) instance are possible too. Multiple deployment models are available today, and more to come as vendor and operator members request.

ProOps is currently available for use by CableLabs members and vendors under nondisclosure agreement (NDA) and intellectual property rights (IPR), with the additional common code collection (C3) community agreement.

## 1.2 Purpose

This technical report provides architecture and user guide information for ProOps users.

In the rest of this technical report, we explain in-depth the structure and function of ProOps, and the example application that comes with it today. This will lead us to explain some of the basic ways you can configure and build your own solutions in ProOps to support your operations improvement experiments and inventions. We also cover use cases for ProOps that we envision, which hopefully will interest you or spur your own imagination to invent use cases we haven't thought of yet.

One purpose of the user guide portion of this document is to help ProOps users quickly understand ProOps's installation, definition of ProOps's required data APIs, and deployment scenarios.

## 2 INFORMATIVE REFERENCES

This technical report uses the following informative references. References are either specific (identified by date of publication, edition number, version number, etc.) or non-specific. For a non-specific reference, the latest version applies.

[Boyd]	<a href="https://danford.net/boyd/">https://danford.net/boyd/</a>
[CM-OSSiv3.1]	DOCSIS 3.1 Cable Modem Operations Support System Interface Specification, CM-SP-CM-OSSiv3.1-I16-190917, September 17, 2019, Cable Television Laboratories, Inc.
[OODA Loop]	<i>Boyd's OODA Loop and the Infantry Company Commander</i> , A. Bazin, Infantry Magazine, 2005.
[ProOps Blog]	<a href="https://www.cablelabs.com/cable-network-reliability-proops-platform-for-pnm-and-more">https://www.cablelabs.com/cable-network-reliability-proops-platform-for-pnm-and-more</a>
[ProOps Code]	<a href="https://code.cablelabs.com/proactive-operations-platform">https://code.cablelabs.com/proactive-operations-platform</a>
[XCCF-PNM]	CableLabs Proactive Network Maintenance Combined Common Collection Framework Architecture Technical Report, CL-TR-XCCF-PNM-V01-180814, August 14, 2018, Cable Television Laboratories, Inc.

### 2.1 Reference Acquisition

- Cable Television Laboratories, Inc., 858 Coal Creek Circle, Louisville, CO 80027; Phone +1-303-661-9100; Fax +1-303-661-9199; <http://www.cablelabs.com>

## 3 TERMS AND DEFINITIONS

This document uses the following terms:

<b>Proactive Operations Platform</b>	A software platform that allows plug-and-run style Proactive Network Maintenance tool development and deployment at scale.
--------------------------------------	--



## 4 ABBREVIATIONS AND ACRONYMS

This document uses the following abbreviations:

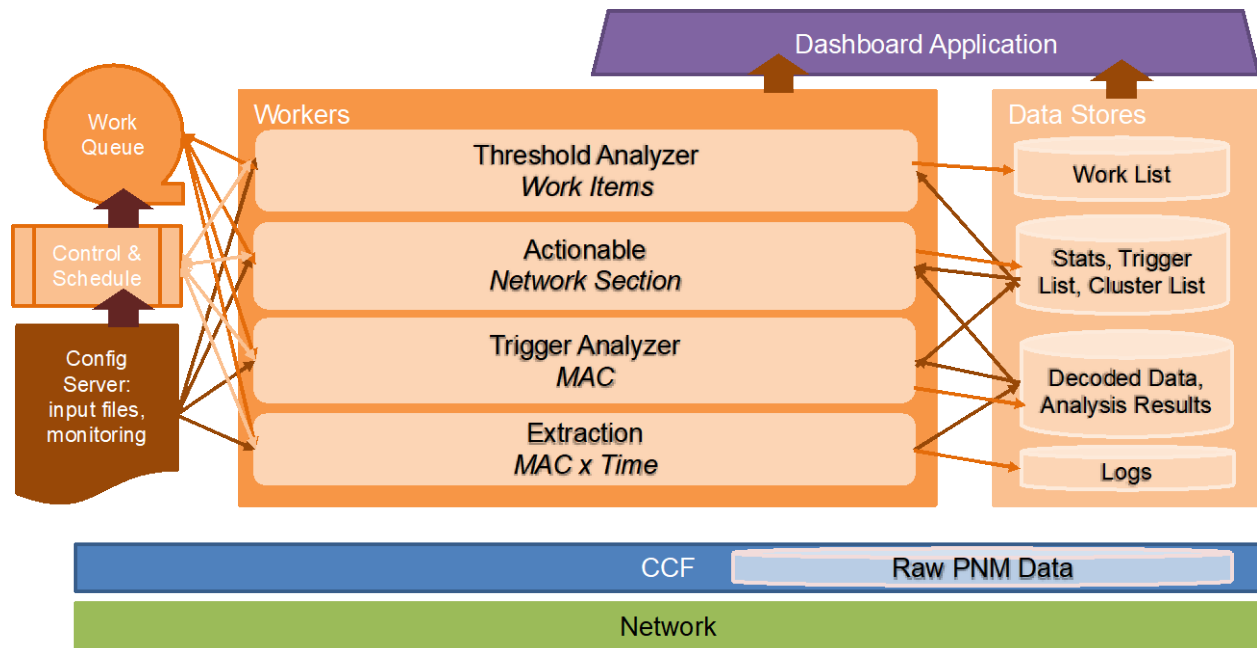
<b>API</b>	application programming interface
<b>C3</b>	common code collection
<b>CableLabs</b>	Cable Television Laboratories, Inc.
<b>CCF</b>	common collection framework
<b>ChEst</b>	channel estimation (coefficient)
<b>CM</b>	cable modem
<b>CMTS</b>	cable modem termination system
<b>CMVA</b>	cable modem validation application
<b>dB</b>	decibel
<b>DOCSIS</b>	Data-Over-Cable Service Interface Specifications
<b>EWMA</b>	exponentially weighted moving average
<b>FEC</b>	forward error correction
<b>FIFO</b>	first in first out
<b>GIS</b>	geographic information system
<b>GUI</b>	graphical user interface
<b>IP</b>	Internet protocol
<b>IPR</b>	intellectual property rights
<b>ISBE</b>	International Society of Broadband Experts
<b>IT</b>	information technologies (information technology)
<b>MAC</b>	media access control
<b>MER</b>	Modulation Error Ratio
<b>MIB</b>	management information base
<b>OFDM</b>	Orthogonal Frequency Division Multiplexing
<b>OODA</b>	observe, orient, decide, and act
<b>PNM</b>	proactive network maintenance
<b>ProOps</b>	Proactive Operations
<b>REST</b>	Representational State Transfer
<b>RxMER</b>	receive modulation error ratio
<b>SCTE</b>	Society of Cable Telecommunications Engineers
<b>TFTP</b>	trivial file transfer protocol
<b>VM</b>	virtual machine

## 5 THE PROACTIVE OPERATIONS PLATFORM

The ProOps platform is a framework for constructing applications that turn network data into action. ProOps is simpler to understand in the context of an application, so we later explain it in the context of the simple but complete example application that comes with ProOps. As a foundation, we first explain the ProOps application environment as it stands alone, before an application has been constructed. We also explain briefly here the elements of the environment, covering the details later in the application context.

An important class of components of ProOps is the worker, which is a module of code which can be tasked by the system. Some workers are a part of and come with ProOps, and some workers function for the application. A worker is a module of code which is scheduled to execute according to the configuration instructions, taking inputs as directed, and sending its output where directed. In software development terms, it is like a microservice, but subscribed to a task queue instead of waiting for an application programming interface (API) call. An application in ProOps is configured from workers that execute the needed instructions.

Figure 1 shows the elements of the PNM application environment as a raw environment, without hosting a working application, or any workers; the workers box represents just the worker organization without the workers that make up the application.



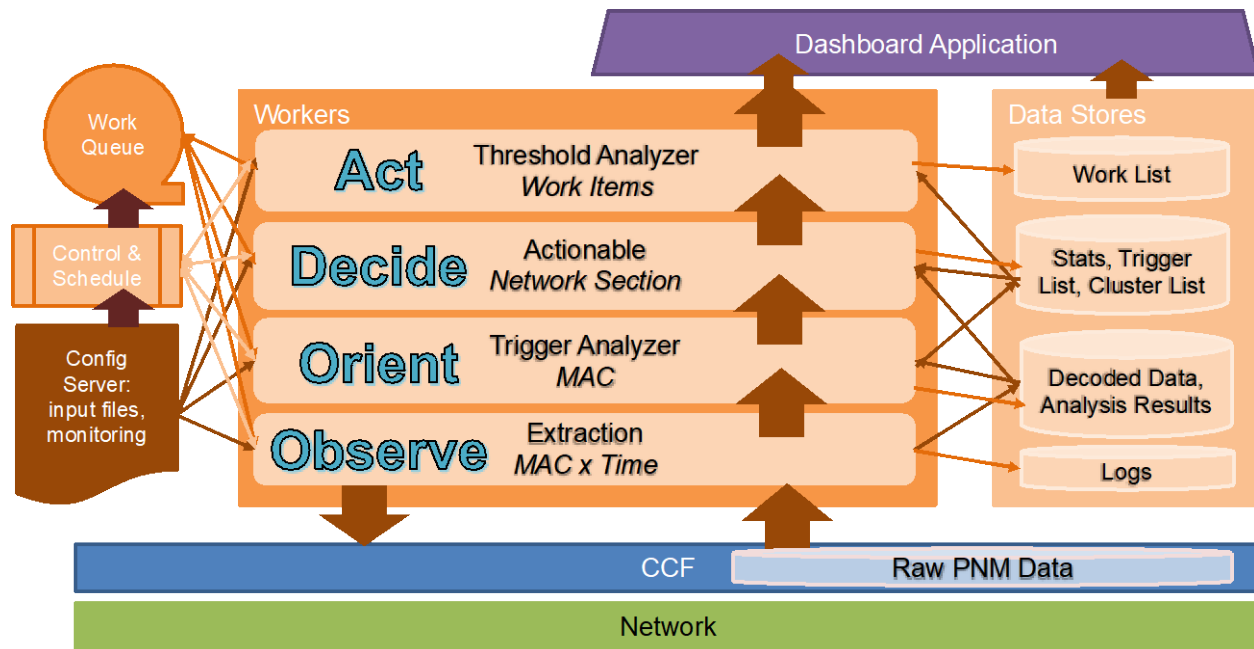
**Figure 1 – Depiction of ProOps Platform on Top of CCF and a Network**

We begin at the bottom of ProOps and work our way up. The descriptions of the network or CCF are outside the scope of this technical report. For the latter, we refer the reader to the Common Collection Framework architecture document [XCCF-PNM]. The remaining elements are all a part of ProOps, so are described briefly here.

- The configuration server (config server) hosts the configuration data which define the messaging mechanisms and necessary additional configurations for workers. It describes the workers to be connected, and any changes that affect the work they do. The user can control applications through this interface for the most part, as the configuration is what defines the application from a pool of workers. When workers start up, they request configurations from the configuration server using RESTful APIs.
- The control and schedule section handles the scheduling of workers, and controls task creation and ending as needed to handle the processing according to the configuration server. The tasks describe the data that each worker is fed, the data that each worker outputs, where the outputs are held, where the inputs are obtained from, and any additional information for specific workers such as thresholds, Internet protocol (IP) addresses of cable modem termination systems (CMTSs) and CCFs, etc.

- The work queues are a number of first-in, first-out (FIFO) queues of tasks that need to be done by the workers according to the schedule dictated by the configuration server.
- The workers are held in the worker environment in the center of the figure. These workers simply exist in a pool and are organized as dictated by the configuration server, but we recognize that for PNM we will need the data from the network to be translated into actionable work. Therefore, we have identified reasonable steps we expect are sufficient for most any application envisioned. These steps are part of the design advantage of ProOps, but not a constraint to the solution possibilities. Those steps are briefly described as follows. Note how they align to the well-known OODA loop or OODA process.
  - **Extraction** takes place periodically as dictated by the initial schedule outlined in the configuration. Because the same media access control (MAC) addresses are collected multiple times on a(n) (approximate) schedule, we note that this step conducts data collection of MAC by time (**MAC x Time**). The output at this layer goes to the log file, but also decoded data and analysis results are held, too. Because this layer is focused on data extraction, we align this layer to the **Observe** step of the OODA process.
  - **Trigger Analyzer** will analyze the data from the extraction and determine whether a change of action is triggered, such as obtaining new data elements, taking data elements on a different schedule, collecting data from related network elements, and placing some of these MAC addresses on a list that identifies further action. The analysis on this layer likely evaluates data on a single MAC over time. For this reason, we note that this step conducts data analysis by **MAC**. The output from this step consists of statistics, and a determination of which MACs are triggered. Because this layer establishes some context to the information, we align this layer to the **Orient** step of the OODA process.
  - The **Actionable** step takes the MAC addresses that are flagged by the trigger analyzer and determines the network section relevant to the identified MACs. For example, clustering of MACs is possible here. But to make these actionable, we also have to introduce a measure of performance and comparison for these MACs or clusters of MACs, so that decisions can be made to bring the network section to actionable work. Therefore, we note that this step conducts analysis on the **network sections**. The output from this step can be clusters of MAC addresses, or single MAC addresses representing network points or CMs, but will definitely be additional statistics to support decisions. Because this layer makes our oriented view of the data something we can act on, it is really the layer in which decisions are made. Therefore, we align this layer to the **Decide** step of the OODA process.
  - The **Threshold Analyzer** will look at the measure of performance and comparison, introduce further analysis of other MACs in the cluster, look at historical information, and decide which of the identified actionable network sections are good choices to act on based on information provided in the configuration server. This step therefore conducts analysis on **work items**, and provides output to the work list. Because this layer considers the decision made that there is something that can be acted upon, and either selects or allows the selection of work that will be acted on, we align this layer to the **Act** step of the OODA process.
- The data stores hold the data elements needed for the application as dictated by the configuration server. The following elements are expected to be needed in many applications.
  - Logs will hold output from the extraction layer.
  - Decoded data and analysis results will be held in a data lake or database depending on the needs of the application.
  - Statistics, triggered lists, and clustering lists are captured in a database or tables as needed by the application.
  - The work list is likewise captured in a list or database as needed.
- Finally, a dashboard application is fed by the entire set of identified data, so that geographic information system (GIS) maps, graphs, and detailed work packages can be assembled as needed by whatever system this will feed. This approach allows most any operator to take the output from the various steps of an application in this environment and tie it easily into its existing ticketing systems and other operations needs.

The previous Figure 1 is revised below in Figure 2 to show the flow of information and how the layers reflect the OODA steps, with arrows showing the process flow.



**Figure 2 – Depiction of ProOps with OODA overlaid**

Next, we discuss some of the detail in the sections of the ProOps architecture. In the explanation of the architecture, we put **workers in bold**, *data in italics*, and configuration to help keep it all straight.

## 5.1 Control & Schedule Worker

**Control & schedule** is a special worker that interacts at the system level to control the other workers, follow the application and flow defined in the configuration file, and schedule work according to the task description or conditions discovered from the other workers or their data output. The tasks are posted periodically by the schedule worker to the task queue(s) and assigned to workers by subscription. If a configuration instructs data collection to happen every  $x$  hours, then this worker will schedule work for the queue based on this schedule. Further, if a result from a data poll and analysis indicates triggering a rule given in the config file, then this worker will update the schedule it follows for those impacted CMs accordingly, and therefore change the schedule that dictates how it places work in the work queue. It must therefore hold state, but can always regain a lost state by reading the config data and the information in the data stores, as those are sufficient to define the schedule.

Note that if this worker is responsible for monitoring a trigger condition to instantiate another worker, such as a **ranker worker**, then it must have access to the data or worker output that triggers the follow up work. If instead it is to act on a schedule only, then maintaining the time against that schedule is important, but not catastrophic if state is lost. The *log file* could be used to capture time and a backup of **control & schedule** state to avoid a loss of state at failure.

This special worker also regulates the system so to not tax the network too much, and to poll and archive data based on scheduling requirements from the application configuration. **The control & schedule worker** reads the config data, builds a schedule for how other workers are started for data collection, then creates the needed polling workers at the appropriate time.

Note that the **control & schedule worker** is the one worker type that keeps track of timing; hence, it needs to maintain state information.

## 5.2 Work Queues

This entity is a number of queues of tasks to be handled by different workers in the system. The **control & schedule** worker will place tasks on the queues for other workers, and likewise another worker can generate a task for the

**control & schedule** worker to handle. The work queues manage tasks and assignments, holding the tasks to do like a pipeline connecting tasks to workers.

### 5.3 Workers and Workflow Description

In this architecture, workers are created to handle tasks, and the work to be handled arrives to a queue. A special **control & schedule** worker handles all timing for scheduling, and assigns work to the other workers. Based on this concept, the following workers need to be defined, as illustrated in Figure 2. The previous figures only show in some cases the worker classes, so more detail is described here than can be shown in the figures. Further, other types of workers can be defined at will, and nothing precludes the addition of layers, or the extension of function of identified layers, in this architecture. The layering is simply a framework that facilitates action to support a network and services. The configuration file, described later, is what enforces the construct of the overall system, and therefore the workflows or applications built in the environment.

- Observe = Extraction [**polling worker, translator, linear calculator**]: This is a pool of workers which form a linear processing of data from the CCF. The new **polling workers** interact with the CCF to poll the data. The data received are then translated by **translator workers** if and when necessary, and finally processed by **linear calculator workers**. The linear calculator worker is a special calculator worker that is always applied to data right off of the CCF. The outputs are cataloged into the *decoded data and analysis results data lake or database* by a **catalog worker** (if needed).
- Orient = Trigger Analyzer [**calculator workers, anomaly detector workers, machine learning workers, trigger analyzer worker**]: This is a pool of workers who evaluate data as it shows up in the *decoded data and analysis results*, against the analyses invoked and configured by the config file, placing statistics and a list of triggered MAC addresses into the *Stats, Trigger List, Cluster list database*. The **control & schedule worker** will monitor the database for new data which triggers the need for a specific worker type, or monitor a clock to trigger, based on the config file. Then it will schedule the appropriate workers to complete the configured tasks. **Calculator workers** will simply calculate statistics from the raw data and place the statistics into the database. **Anomaly detector workers** will search a series of data (over frequency, time, etc.), indicate the data included in a detected anomaly, and in some cases indicate the class of anomaly detected. **Machine learning workers** will apply other machine learning techniques to data and statistics. Both the latter worker types may contain or use calculator workers, for example, too. A **trigger analyzer** worker will assess these outputs against the triggers indicated in the config file. The output from these analyses are tasks that can be to sample new data, or sample some data sources for specific MACs more frequently, or to re-analyze older data, for example.

A note is warranted here about the role of the config file versus the changes decided: The config file holds information about the frequency of data collection, but only for the default data frequency; it also holds information about the triggers and actions from triggers regarding data polling frequency, but only as conditions, not as system state. Changes in system state are held in memory; or, in the case of MAC addresses subject to frequent polling, in the *stats, trigger list, cluster list database*.

This means workers can relate to all other workers in a nested, series, parallel, or other configuration relationship. More worker types can be defined to fit into this pool as needed by specific applications.

- Decide = Actionable Layer [**severity calculator worker, ranker worker, cluster worker**]: This worker class likely only contains a few worker types (depending on the nature of the application configured), tasked with translating the trigger list and collected statistics and data for triggered MAC addresses, and providing the necessary performance measures with which to rank the MAC addresses. It can work exclusively with the *stats, trigger list database* as it reads the information, processes, then adds an updated measure of performance to the data, and provides an updated ranking. That assumes the configuration file is set up to provide the needed data to supply the measures needed for action. The work here can be done periodically, or triggered by a condition such as a number of data updates on the triggered list, or a number of new entries to the triggered list. The **control & schedule worker** will trigger action as directed by the config file, and trigger a **severity calculator worker** based on the information in the config file which specifies what triggers the calculation. Once the calculation is updated, the **ranker worker** will evaluate the updated measures of performance and rank the MAC addresses by severity, placing the performance measures and the ranked list back into the database. The **cluster worker** will evaluate the information relating to the MAC addresses and attempt to cluster them by similarities in the statistics and performance measures. Workers on this layer may need access to the raw data or

decoded data or analysis results in some cases too. Either on a schedule or as triggered by conditions, as specified in the config file, the **cluster worker** is started to evaluate triggered MAC addresses and their performance and other needed information, conduct clustering based on multiple dimensions of data, and output clusters of MAC addresses that are likely experiencing the same problem.

- **Act = Threshold Analyzer:** The threshold analyzer will examine the trigger clusters (which are assumed to include clusters defined by single MAC addresses in many cases) and decide which of the clusters are actionable. This can be done through the config file which holds the rules that trigger the action based on an external financial model, operations rules, workload, number of potential jobs on the list, etc. The **control & schedule worker** will read the configuration file to determine when or under what condition to trigger a sorting of the opportunities in the *trigger cluster database* (and potentially the *trigger list*). The sorting is done by the **threshold analyzer sorter worker** which will read the config file to determine what measures to use in the sorting, and then gather the information it needs, and finally sort the opportunities into a *dispatch list*. An output of this group of workers is also a measure of performance that is used for ranking, but potentially also additional measures for ranking or consideration including but not limited to an estimate of the benefit achieved by completing the associated work package. This analyzer layer can also call on external information such as available technicians in an area, day of week, likelihood of access to the network elements necessary, etc.

## 5.4 Config Files

One or more configuration files (config files) are necessary to define the application from the ProOps elements, and to control how the application functions day to day. There are several types of configuration information needed for this workflow, and that information could be stored in a single file, or multiple files, or in a database, for example. It could be edited in a file editor, or through a graphical user interface (GUI) defined for the application. The files could be separate or together or organized in any way. As developed currently, there is an interface with organization to configure workers and general parts of an application. We assume in this description that configuration will be read-in from a file, but will consider each configuration type as a separate file for the sake of explanation.

- Extraction Config File: This config file specifies the frequency of polling for each data element, and which data elements are polled as part of the application. It may also contain instructions for how to handle missing data, timeouts, and other problems that may be encountered in data polling.
- Trigger Analyzer Config File: This config file contains instructions of which analysis workers are included in the application or workflow, and how they are started (such as when data appears in the *decoded data and analysis results* database or data lake). It also includes the rules by which CMs are triggered for action to the next level, and flagged for ranking, plus what workers need to do to support the action layer.
- Action Config File: This config file contains the details of the model for the measure of performance for CMs based on their statistics and trigger state. The model may be complex or simple, depending on how data over time or missing data are treated, and whether a nonlinear model is needed. It also specifies what to use for ranking. This config file also may provide information for how to cluster CMs, essentially controlling the model for clustering, but also specifying the worker that is used.
- Threshold Analyzer Config File: This config file provides the rules by which opportunities (clusters or individual CMs) are ranked and sorted, included or excluded, on the dispatch list. It also specifies how often the list is revisited and updated, which means how often the clusters and individual CMs are analyzed.

## 5.5 Data Stores

Several different data stores are needed, though they need not be distinct. However, as the nature of what they store is different, it is likely advantageous to keep them in solutions suited best to their forms, and therefore desirable to be distinct in at least some way.

- **Raw PNM Data:** The PNM data can be kept in the file system of the CCF, or in the trivial file transfer protocol (TFTP) server used by the operator, or in a separate data lake. As the data held here are disparate, it is not likely that a database is an appropriate solution. Therefore, we refer to the (Raw) PNM Data store as a data lake.

- *Decoded data, analysis results*: The decoded data and analysis results can go into a data lake or database, as long as the relations are held between MAC, time, and data to analysis. The data here can be contained in a relational database as it can contain decoded data and statistics of multiple types associated with individual CMs by type, time of day, etc.
- *Stats, trigger list, cluster list*: The data here can be contained in a relational database as it can contain statistics of multiple types associated with individual CMs by type, time of day, etc., and a separate related list of the CMs that meet the threshold for triggering based on the rules set in the [config file](#). The cluster information can be held in a list, a database, or other simple form as it only needs to be a list of how the CMs are clustered in common groups for PNM work opportunities. But all these separate elements are related, so a relational database is a good candidate.
- *Work (dispatch) list*: This data store is simply a list of CMs or CM clusters which are opportunities for work. Each item on the list has a performance measure that indicates how critical the problem is, or how large the proactive opportunity is, so may even be an expected benefit in dollars, for example. There should be links back to the supporting evidence for the decision to add this work to a dispatch list, and to help with troubleshooting.
- *Logs*: The system will keep a list of system logs for overall system health. This may be a method to maintain state under failure too.

## 5.6 Scheduling Considerations

- A CM can usually only respond to one request at a time, or very few. But if a CM doesn't respond at the time data were requested, and there is reason to expect it should, then a retry may be in order.
- CCF jobs have three states: accepted, complete, and failed. Some complete jobs may have incorrect, errored, or incomplete data responses as well, so a complete job may still be a failed job.
- The parsing of the data output may result in a new fourth state of complete but errored.
- Thus, the state of a completed job from the CCF will create subsequent requests to be scheduled based on the final job status.
  - If the job status is complete, and parsing yields a correct output, then schedule the next job according to the instructions in the [config file](#).
  - If the job status is complete but errored, or failed, then schedule a new request immediately, or after a short amount of time as specified by the [config file](#).
  - The [config file](#) should also state whether subsequent jobs are scheduled based on the completion time of the last job, or based on a system clock. In other words, if a data type is requested six times a day, and the first one is delayed by an hour, then does the next data request happen in three hours or four hours? Both may be possible, but configurable in the [config file](#).
- The **control & schedule worker** will need to sort jobs on a common system clock. The control worker handles this task.

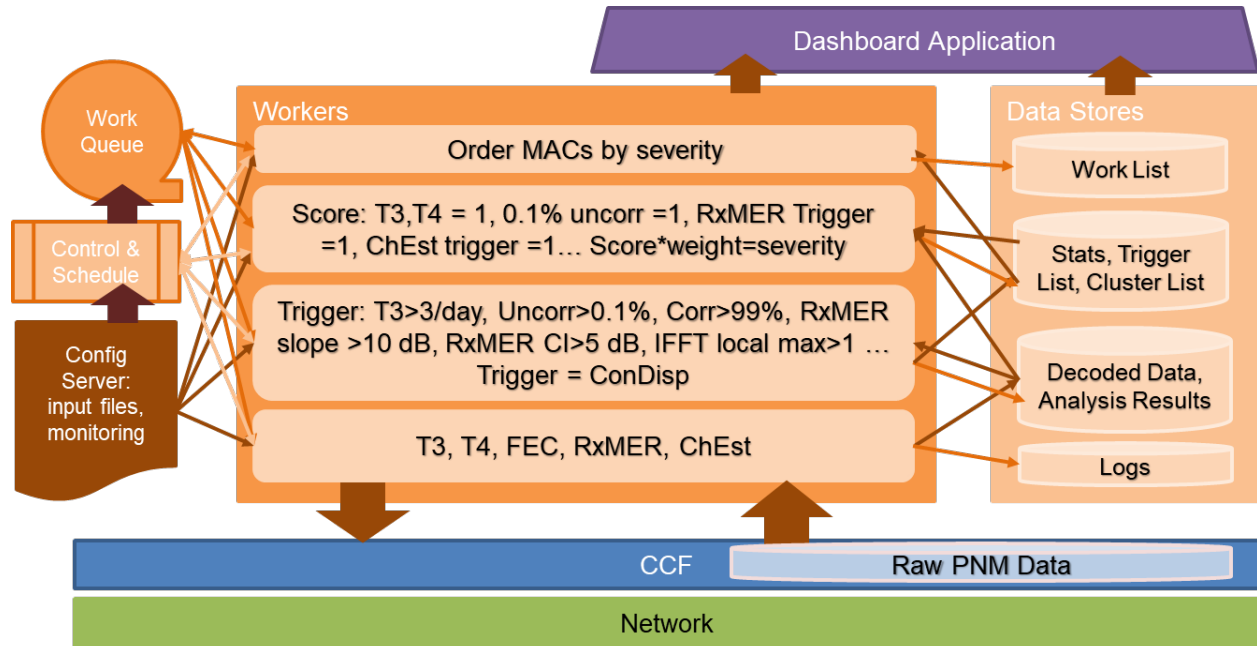
## 5.7 Additional Notes about ProOps

- The [config file](#) must account for missing data in the way we analyze and calculate performance measures. A non-responsive CM may end up on a separate list.
- RabbitMQ is our choice for the first version of ProOps for managing the worker queue.
- We stagger polling to get the data we need and manage the poller resources.

## 6 PNM EXAMPLE APPLICATION FOR PROOPS

Here we define the example application that comes with ProOps. Out of the box, very little configuration is necessary to get the system running with the example application. However, the utility of the example application is highly dependent on the user's ability to configure it for their use case specifically.

The first version of the example application that comes with ProOps does not do clustering; the identified CMs will go directly to the threshold analyzer as single CMs. This can be reflected as each CM is its own cluster, and the cluster algorithm doing no work, but still following the diagram in Figure 3.



**Figure 3 – Depiction of the Example Application that comes with ProOps today**

### 6.1 Base data polling (Pollers)

- Pull T3 and T4 errors from CM logs every hour (configurable).
- Pull forward error correction (FEC) stats every hour (configurable).
- Pull receive modulation error ratio (RxMER) every four hours (configurable).
- Pull channel estimation every four hours (configurable) at same time.

### 6.2 Triggers (Analyzers)

- Repeating errors at a configurable frequency, say more than three per day, configurable.
- Uncorrectables higher than 0.1%, correctables at 99% or higher for over an hour, all configurable.
- RxMER and SpecAn results with a slope greater than 10 dB, configurable. Also, trigger any RxMER with a 95% confidence interval greater than 5 dB (calculated from percentile responses), configurable. Also look for ripples in either or both by inverse fast Fourier transform (IFFT) with an energy spike. Take the largest continuous segment of data from either and transform (IFFT) it into time domain, then search for local maximum. Each result with more than one local maximum is triggered (this removes the high frequency main energy). Transform to distance and severity of echo tunnel, trigger anything higher than 10 dB, configurable.



- ChEst indicating a range greater than 10 dB, configurable. Also look for ripples in either or both using same method as above. Transform to distance and severity of echo tunnel, trigger anything higher than 10 dB, configurable.
- ConDisp would be extracted after other issues were found, such as FEC, T3, T4, RxMER issues. For now, just plot and calculate stats of mean, variance, and third moment.

### 6.3 Actions

- A CM must be triggered for an action to apply.
- We apply a weight to each trigger type (management information base (MIB), and trigger, so that a given measure could have more than one trigger).
- Score: Create a score for each measure based on the following rules.
  - 1 point for each error or boot issue
  - 1 point for each percent uncorrectable
  - 1 point for each trigger (RxMER, ConDisp), each time
  - 1 point for each trigger in ChEst
- Each day, multiply the score by the weight (score\*weight), and sum the results for each CM each day. Use an exponentially weighted moving average (EWMA) scheme to get the CMs their final scores over time. The EWMA approach is a worker that can be exchanged with other approaches.
- Each CM triggered will have its frequency of data collection doubled, configurable for each measurement. Only after having a 0 score\*weight for three full days, configurable, will a CM be removed from the action list. The frequency of data collection will be configurable by trigger type, and may cross measures. For example, triggering FEC could result in increasing data collection on FEC by five times the rate, and ConDisp each time FEC is collected, or RxMER at twice the rate.
- All CMs on the action list are prioritized by their score\*weight, summed daily over the duration they are on the list (each day's score\*weight summed over the contiguous days each CM is on the list, for each CM), or using the EWMA approach as we will for this example application.
- The list is reported for the CMTS, and the CMTS is scored accordingly by summing the measures for the CMs on the list.

### 6.4 Modules

The example application that comes with ProOps has several modules, following the structure of ProOps explained previously.

- Control & schedule: A polling scheduler to schedule the data requests in a manner to help the CMTS not get overloaded.
- A data store of results collected, statistics, and scores for each CM, over time. Age off the raw data.
- Translator workers: A translator for each data type pulled.
- Linear calculators: Rapid versions of other calculators or anomaly detectors or other entities which are always applied to data pulled, by data type, before holding in a data store.
- Calculator workers: A calculator or more for each data type pulled.
- Anomaly detector: An anomaly detector for the spectrum and RxMER data. This is just a simple statistical calculator for the first release. A true anomaly detector will be made available later.
- Trigger analyzer: An evaluator to decide on the triggering of a CM, and its weight.
- Severity calculator Worker: A calculator that calculates the severity of the opportunities.

- Cluster Worker: A clustering algorithm that handles work assignments. This is defined for a later release.
- Ranker worker: A calculator and ranker for clusters and CMs together to calculate the measure of severity and rank the opportunities.
- Threshold analyzer sorter worker: A threshold analyzer to indicate dispatchable work, based on rules configured.
- A configuration file to allow changing all the various default values of the application.
- A mechanism (GUI later, via a management interface or the dashboard application) for accessing the list of CMs (and clusters) to act on, and a way to compare their severity.
- Work queue: The queue of work to be performed, organized in a FIFO manner.

The example application is untested in live plant, so the initial configuration described may be far off from a useful combination of settings. The settings will allow verification of function, and be instructive for configuration and developing more functionality, as we now explain.

## 7 CONFIGURING AN APPLICATION

Applications are defined by how the configuration connects the workers. Thus you can define a new application from an existing one by changing the configuration, or adding new workers to the worker environment and configuring their use, or a combination of these actions. Adding a new worker is outside the scope of this technical report, but we will explain how to configure an application using existing workers as it further explains how to make use of ProOps for PNM uses.

ProOps has several base workers that must be configured for an application to work, as opposed to optional workers that can be configured or not depending on need.

Each PNM measurement can be configured on multiple dimensions, aside from the obvious parameters such as network scope for the deployment and other factors controlled by the context of the deployment, or settings for specific PNM measurements like sampling period for FEC statistics. Some of the settings specific to ProOps that should be evaluated for suitability of default settings include the following.

- Time between data requests to a network element for a given data type.
- Time to wait for a time out on a data request.
- Triggering based on range, standard deviation, linear or curve fitting regression parameters, maximum, minimum, mean, median, mode, third or fourth moments, or other user-defined statistics including machine learning or artificial intelligence analysis outcomes.
- Weight given to a performance measure as the result of a trigger, such as the range of a measure times a constant factor, or the square of the sum of the uncorrectable errors in the last hour.
- Which worker or driver to use for a given request, and where to send the output of a worker.
- Various queue management settings to control workers assigned to work.

The screenshot shows the 'Edit Config' page for task queue configurations. The interface has a blue header with the CableLabs logo and navigation links: 'PNM Management Server', 'Config Management', 'Job Scheduling', and 'System Monitor'. The main content area is divided into two sections. On the left, there is a table with one row and two columns, labeled '1' and '0'. On the right, there is a 'Required Fields' section with several input fields: 'Worker Identifier (Unique Key)', 'Task Queue Name', 'Task Queue Username', 'Task Queue Password', 'Task Queue IP', 'Task Queue Exchange', and 'Task Queue Routing Key'. A blue 'CREATE/UPDATE' button is located at the bottom right of the form.

**Figure 4 – Configuration Management Server Web GUI Index Page (task queue configurations)**

See Figure 4. On the index page of the configuration server, the task queue information can be configured. The task queue information is essential for all internal communications. When workers start, they request task queue configuration along with other configurations from the configuration server through RESTful APIs.

Show 10 entries Search:

Worker Identifier (Unique Key)	Task Queue Name	Task Queue Username	Task Queue IP	Task Queue Exchange	Task Queue Routing Key	Remove Config
result_db_worker	pro_ops_results	testing	10.70.35.120:5672	pro_ops_results	pro_ops_results	
pro_ops_job_scheduling	pro_ops_jobs	testing	10.70.35.120:5672	pro_ops_jobs.all	pro_ops_jobs	
data_collector_3	pro_ops_cm_data_collection_jobs	testing	10.70.35.120:5672	pro_ops_jobs.cmdatacollection	pro_ops_cm_data_collection_jobs	
data_collector	pro_ops_cm_data_collection_jobs	testing	10.70.35.120:5672	pro_ops_jobs.cmdatacollection	pro_ops_cm_data_collection_jobs	
cm_nmer_stats	pro_ops_nmer_stats_jobs	testing	10.70.35.120:5672	pro_ops_jobs.nmer_stats	pro_ops_nmer_stats_jobs	
cm_result_threshold_trigger	pro_ops_result_trigger_jobs	testing	10.70.35.120:5672	pro_ops_jobs.result_trigger	pro_ops_result_trigger_jobs	
cm_condisp_stats	pro_ops_condisp_stats_jobs	testing	10.70.35.120:5672	pro_ops_jobs.condisp_stats	pro_ops_condisp_stats_jobs	
cm_channel_estimation_stats	pro_ops_channel_estimation_stats_jobs	testing	10.70.35.120:5672	pro_ops_jobs.chest_stats	pro_ops_channel_estimation_stats_jobs	

Showing 1 to 8 of 8 entries Previous 1 Next

**Figure 5 – Configuration Management Server – Defined Workers**

Defined workers are listed at the configuration server's index page, shown in Figure 5. When a new worker starts up, the configuration server uses the worker identifier to determine which configuration the worker uses and replies to the RESTful API call with the worker's configuration. The worker identifier is a unique key or name that each worker owns. The worker configuration can be removed by clicking the "REMOVE" button.

CableLabs PNM Management Server Config Management Job Scheduling System Monitor testing

Required Fields

Worker Name (Unique Key)

Worker Identifier

data\_collector

Job Interval (Seconds)

Edit Job Message

1 0

Show 10 entries Search:

Worker Name (Unique Key)	Worker Identifier	Job Interval (Seconds)	Last Run	Status Control	Remove Schedule
trigger	cm_result_threshold_trigger	300	2019-07-16 01:16:09		
nmer_stats	cm_nmer_stats	600	2019-07-16 01:16:09		
data_collector_04	data_collector	30			
data_collector_03	data_collector	30			
data_collector_02	data_collector	30	2019-07-16 01:18:35		
data_collector_01	data_collector	600	2019-07-16 01:09:38		
condisp_stats	cm_condisp_stats	300	2019-07-16 01:17:44		
chest_stats	cm_channel_estimation_stats	300	2019-06-26 22:16:33		

Showing 1 to 8 of 8 entries Previous 1 Next

© PNM Management Server Version 0.1 - 2019 CableLabs

**Figure 6 – Job Scheduling**

The job scheduling page, shown in Figure 6, displays all defined job schedules. When a new scheduling is started, the configuration server sends a message to the task queue to which the job scheduling worker is subscribed. The job scheduling worker will then add the job to its scheduling queue and send task messages to the task queue periodically based on the job interval.

**Required Fields**

Worker Name (Unique Key)  
trigger

Worker Identifier  
cm\_result\_threshold\_trigger

Job Interval (Seconds)  
300

**Edit Job Message**

```

1- {
2-   "cm_rxmer": {},
3-   "data_service_address": "http://10.70.35.120:10001",
4-   "data_service_store_api": "/api/triggerResults/",
5-   "cm_rxmer_stats": {
6-     "data_service_api": "/cmRxmerStats",
7-     "data_service_address": "http://10.70.35.120:10001",
8-     "duration": 86400,
9-     "trigger_field_config": [
10-      {
11-        "field": "range",
12-        "threshold": {
13-          "greater_than": 2
14-        }
15-      },
16-      {
17-        "field": "std",
18-        "operation": "OK",
19-        "threshold": {
20-          "greater_than": 0
21-        }
22-      },
23-      {
24-        "field": "ordinary_linear_regression_m",
25-        "operation": "OK",
26-        "threshold": {
27-          "not_range": [-0.0001, 0.0001]
28-        }
29-      }
30-    ]
31- },
32-   "cm_event": {
33-     "data_service_api": "/getEventsData",
34-     "data_service_address": "http://10.70.35.120:10001",
35-     "duration": 86400,
36-     "trigger_field_config": [
37-      {
38-        "field": "T3_T4_count",
39-        "threshold": {
40-          "greater_than": 0

```

**Job Schedules Table**

Worker Name (Unique Key)	Worker Identifier	Job Interval (Seconds)	Last Run	Status Control	Remove Schedule
trigger	cm_result_threshold_trigger	300	2019-07-16 01:16:09	<a href="#">START</a>	<a href="#">REMOVE</a>
rxmer_stats	cm_rxmer_stats	600	2019-07-16 01:16:09	<a href="#">STOP</a>	<a href="#">REMOVE</a>
data_collector_04	data_collector	30		<a href="#">START</a>	<a href="#">REMOVE</a>
data_collector_03	data_collector	30		<a href="#">START</a>	<a href="#">REMOVE</a>
data_collector_02	data_collector	30	2019-07-16 01:18:35	<a href="#">STOP</a>	<a href="#">REMOVE</a>
data_collector_01	data_collector	600	2019-07-16 01:09:38	<a href="#">START</a>	<a href="#">REMOVE</a>
condisp_stats	cm_condisp_stats	300	2019-07-16 01:17:44	<a href="#">STOP</a>	<a href="#">REMOVE</a>
chest_stats	cm_channel_estimation_stats	300	2019-09-26 22:16:33	<a href="#">START</a>	<a href="#">REMOVE</a>

Showing 1 to 8 of 8 entries

**Figure 7 – Configuring Job Scheduling**

Here in Figure 7 is an example of what can be added to the job scheduling configuration. Other than job interval configuration, we can also add configurations such as data service address and data store API to let the worker know where to retrieve data and where to store results. For example, by configuring the “cm\_rxmer\_stats” configuration field, the worker will retrieve data from <http://10.70.35.120:10001/cmRxmerStats> with a backlog duration of 86400 seconds (a day), and find all CMs that have an RxMER range (max - min) greater than 2 dB, or standard deviation greater than 0 (which is set artificially small just for demonstration purposes), or an *m* value (slope in ordinary linear regression) that is smaller than -0.0001 or greater than 0.0001 (this is also set to a very small range as an example). Figure 8 shows an example where we set a greater than 0 threshold for T3 and T4 device event log errors.

CableLabs PNM Management Server Config Management Job Scheduling System Monitor testing

CREATE/UPDATE

Edit Job Message

```

16 {
17   "field": "std",
18   "operation": "OK",
19   "threshold": {
20     "greater_than": 0
21   }
22 },
23 {
24   "field": "ordinary_linear_regression_m",
25   "operation": "OK",
26   "threshold": {
27     "out_range": [-0.0001, 0.0001]
28   }
29 },
30 },
31 },
32 {
33   "data_service_api": "/getEventsData",
34   "data_service_address": "http://10.70.35.120:10001",
35   "duration": 86400,
36   "trigger_field_config": [
37     {
38       "field": "T3_T4_count",
39       "threshold": {
40         "greater_than": 0
41       }
42     }
43   ]
44 },
45 },
46 {
47   "data_service_api": "/getDefectStatsData",
48   "data_service_address": "http://10.70.35.120:10001",
49   "duration": 86400,
50   "trigger_field_config": [
51     {
52       "field": "uncorrectable_codewords",
53       "weight": 1.0,
54       "threshold": {
55         "greater_than": -1
56       }
57     }
58   ]
59 },
60 {
61   "cm_channel_estimation": {
62     "data_service_api": "/cmChannelEstimationStats",
63     "data_service_address": "http://10.70.35.120:10001",
64     "duration": 86400,
65     "trigger_field_config": [
66       {
67         "field": "tilt_db_per_mhz",
68         "weight": 1.0,
69         "threshold": {
70           "out_range": [-0.0001, 0.0001]
71         }
72       }
73     ]
74 }

```

Worker Name (Unique Key)	Worker Identifier	Job Interval (Seconds)	Last Run	Status Control	Remove Schedule
rxmer_stats	cm_rxmer_stats	600	2019-07-16 01:16:09	STOP	REMOVE
data_collector_04	data_collector	30		START	REMOVE
data_collector_03	data_collector	30		START	REMOVE
data_collector_02	data_collector	30	2019-07-16 01:18:35	STOP	REMOVE
data_collector_01	data_collector	600	2019-07-16 01:09:38	START	REMOVE
condisp_stats	cm_condisp_stats	300	2019-07-16 01:17:44	STOP	REMOVE
chest_stats	cm_channel_estimation_stats	300	2019-06-26 22:16:33	START	REMOVE

Showing 1 to 8 of 8 entries

Previous 1 Next

© PNM Management Server Version 5.1 - 2019 CableLabs

Figure 8 – Configuring for T3 and T4 Device Event Log Errors

CableLabs PNM Management Server Config Management Job Scheduling System Monitor testing

Required Fields

Worker Name (Unique Key)

Worker Identifier

Job Interval (Seconds)

CREATE/UPDATE

Edit Job Message

```

1 {
2   "data_service_api": "/getCableMerData",
3   "data_service_address": "http://10.70.35.120:10001",
4   "data_service_store_api": "/cmRxMerStats",
5   "cm_mac": [],
6   "duration": 86400
7 }

```

Show 10 entries

Search:

Worker Name (Unique Key)	Worker Identifier	Job Interval (Seconds)	Last Run	Status Control	Remove Schedule
trigger	cm_result_threshold_trigger	300	2019-07-16 01:16:09	START	REMOVE
rxmer_stats	cm_rxmer_stats	600	2019-07-16 01:16:09	STOP	REMOVE
data_collector_04	data_collector	30		START	REMOVE
data_collector_03	data_collector	30		START	REMOVE
data_collector_02	data_collector	30	2019-07-16 01:18:35	STOP	REMOVE
data_collector_01	data_collector	600	2019-07-16 01:09:38	START	REMOVE
condisp_stats	cm_condisp_stats	300	2019-07-16 01:17:44	STOP	REMOVE
chest_stats	cm_channel_estimation_stats	300	2019-06-26 22:16:33	START	REMOVE

Showing 1 to 8 of 8 entries

Previous 1 Next

© PNM Management Server Version 5.1 - 2019 CableLabs

Figure 9 – Configuring the RxMER Statistics Worker

The job scheduling of the RxMER statistics worker (see Figure 9) is another example of how to configure a worker. The RxMER statistics worker only requires brief configurations that include job interval, data service API, data

service address, duration, and cm\_mac list. The job interval defines how frequently the job is scheduled. The data service API and data service address define where the RxMER statistics worker retrieves RxMER data. Duration defines the data retrieving window from the moment the data is retrieved. The cm\_mac list helps filter interesting cable modem MAC addresses. When the list is empty, all CM data will be retrieved; otherwise, only CM RxMER data from cable modems in the cm\_mac list will be retrieved. The output from the RxMER statistics worker includes "range", "std", "max", "min", "mean", "ordinary\_linear\_regression\_m", and "ordinary\_linear\_regression\_c".

**Required Fields**

Worker Name (Unique Key)  
data\_collector\_01

Worker Identifier  
data\_collector

Job Interval (Seconds)  
600

**Edit Job Message**

```

1 {
2   "cmts_ip": "10.32.40.68",
3   "decf_ip": "10.95.254.82:8888",
4   "db_server_address_and_api": "http://10.70.35.120:10001/pubData",
5   "namespace": "decf",
6   "cm_ip_to_filter": "",
7   "cmts_community_string": "CableLabs50MP",
8   "cm_community_string": "private",
9   "drivers": ["cmPNMDsRxMer"],
10  "drivers_actual_name": ["cmPNMDsRxMer"],
11  "timeouts": [290]
12 }

```

**Show 10 entries**

Worker Name (Unique Key)	Worker Identifier	Job Interval (Seconds)	Last Run	Status Control	Remove Schedule
trigger	cm_resut_threshold_trigger	300	2019-07-16 01:16:09	<a href="#">START</a>	<a href="#">REMOVE</a>
rxmer_stats	cm_rxmer_stats	600	2019-07-16 01:16:09	<a href="#">STOP</a>	<a href="#">REMOVE</a>
data_collector_04	data_collector	30		<a href="#">START</a>	<a href="#">REMOVE</a>
data_collector_03	data_collector	30		<a href="#">START</a>	<a href="#">REMOVE</a>
data_collector_02	data_collector	30	2019-07-16 01:18:35	<a href="#">STOP</a>	<a href="#">REMOVE</a>
data_collector_01	data_collector	600	2019-07-16 01:09:38	<a href="#">START</a>	<a href="#">REMOVE</a>
condisp_stats	cm_condisp_stats	300	2019-07-16 01:17:44	<a href="#">STOP</a>	<a href="#">REMOVE</a>
chest_stats	cm_channel_estimation_stats	300	2019-06-26 22:16:33	<a href="#">START</a>	<a href="#">REMOVE</a>

Showing 1 to 8 of 8 entries

**Figure 10 – Configuring a Data Collection Worker**

Shown in Figure 10, the data collection worker is a special worker that interacts with the CCF. In this example, based on the configuration, the data collection worker is configured to run every 600 seconds. It points to a CCF instance with an IP address of 10.95.254.82:8888, and it collects cmPNMDsRxMer data from CMTS 10.32.40.68. The collected data are automatically decoded and stored using the API defined in "db\_server\_address\_and\_api". The cmts\_ip field can be a list, and can contain multiple CMTS IP addresses.

CableLabs PNM Management Server Config Management Job Scheduling System Monitor testing

Required Fields

Worker Name (Unique Key)  
chest\_stats

Worker Identifier  
cm\_channel\_estimation\_stats

Job Interval (Seconds)  
300

CREATE/UPDATE

Edit Job Message

```

1 {
2   "data_service_api": "/getChannelEstimationData/",
3   "data_service_address": "http://10.70.35.120:10001",
4   "data_service_store_api": "/cmChannelEstimationStats/",
5   "cm_mac": [],
6   "duration": 86400
7 }

```

Show 10 entries Search:

Worker Name (Unique Key)	Worker Identifier	Job Interval (Seconds)	Last Run	Status Control	Remove Schedule
trigger	cm_result_threshold_trigger	300	2019-07-16 01:16:09	START	REMOVE
nmer_stats	cm_nmer_stats	600	2019-07-16 01:16:09	STOP	REMOVE
data_collector_04	data_collector	30		START	REMOVE
data_collector_03	data_collector	30		START	REMOVE
data_collector_02	data_collector	30	2019-07-16 01:18:35	STOP	REMOVE
data_collector_01	data_collector	600	2019-07-16 01:09:38	START	REMOVE
condisp_stats	cm_condisp_stats	300	2019-07-16 01:17:44	STOP	REMOVE
chest_stats	cm_channel_estimation_stats	300	2019-06-26 22:16:33	START	REMOVE

Showing 1 to 8 of 8 entries

Previous 1 Next

© PNM Management Server Version 5.1 - 2019 CableLabs

Figure 11 – Channel Estimation Statistics Worker

Figure 11 shows the CM downstream channel estimation worker, which has a configuration that is similar to CM RxMER statistics worker's configuration. The output includes calculation results on the tilt of the channel estimation and inferences on potential echo distances.

CableLabs PNM Management Server Config Management Job Scheduling System Monitor testing

Online Hosts

Show 10 entries Search:

Host IP	CPU Usage	Memory Usage	Running Workers	Num Worker Processes	Last Stats Update
192.168.64.9	63.0	71.8	cm_channel_estimation_stats	2	2019-07-16 01:46:00
192.168.64.7	62.3	71.8	data_collector	2	2019-07-16 01:46:00
192.168.64.15	65.4	71.8	result_db_worker	2	2019-07-16 01:45:58
192.168.64.14	62.7	71.8	cm_result_threshold_trigger	2	2019-07-16 01:46:00
192.168.64.13	65.4	71.8	pro_ops_job_scheduling	1	2019-07-16 01:45:57
192.168.64.12	63.5	71.8	cm_nmer_stats	2	2019-07-16 01:46:00
192.168.64.11	63.1	71.8	cm_condisp_stats	2	2019-07-16 01:46:00
172.18.0.5	1.1	52.0	cm_nmer_stats	2	2019-07-16 01:46:00
172.18.0.3	1.5	52.0	result_db_worker	2	2019-07-16 01:45:58
172.18.0.14	1.2	52.0	cm_result_threshold_trigger	2	2019-07-16 01:45:59

Showing 1 to 10 of 14 entries

Previous 1 2 Next

Online Workers

Show 10 entries Search:

Worker Identifier	Process ID	Host IP	Last Stats Update	Task Received	Task Started	Task Succeeded	Task Failed	Task Sent	Show Graphs
-------------------	------------	---------	-------------------	---------------	--------------	----------------	-------------	-----------	-------------

© PNM Management Server Version 5.1 - 2019 CableLabs

Figure 12 – Top-level System Monitoring



Online Workers

Show 10 entries

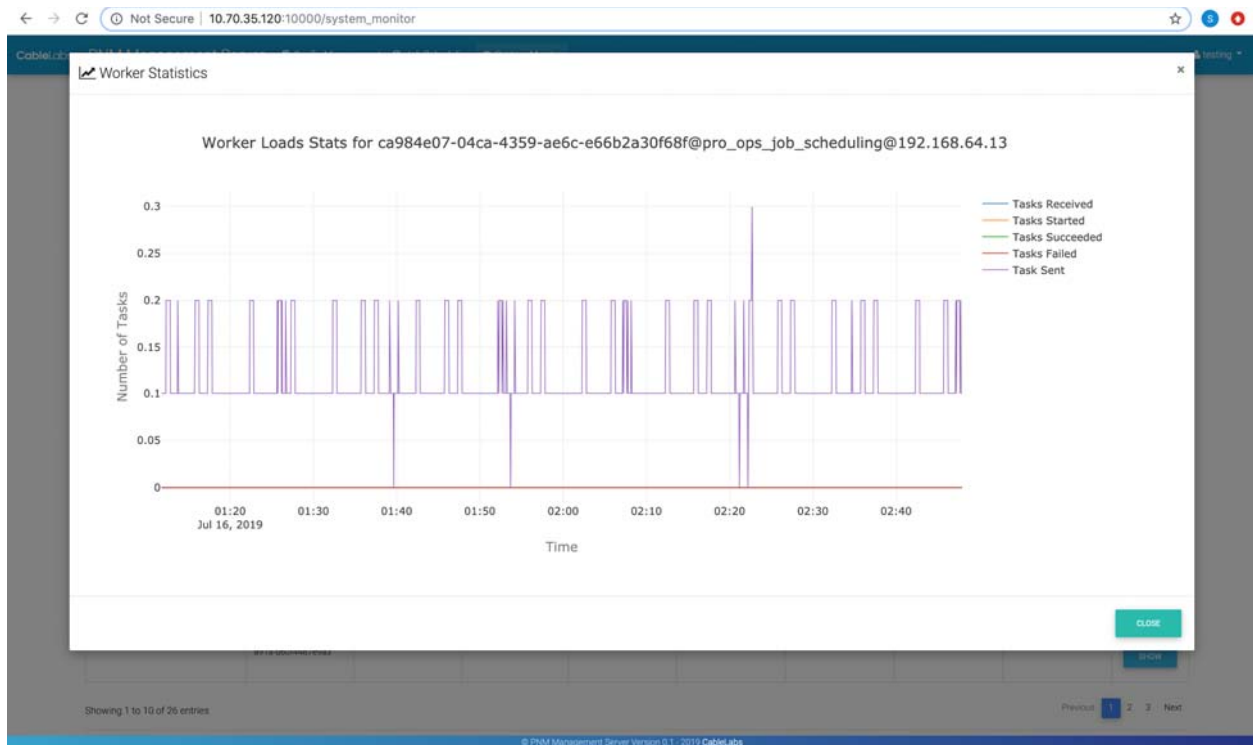
Search:

Worker Identifier	Process ID	Host IP	Last Status Update	Task Received	Task Started	Task Succeeded	Task Failed	Task Sent	Show Graphs
result_db_worker	771af0ce-9294-4806-9f04-6921815c1366	172.18.0.3	2019-07-16 01:49:25	36	36	36	0	0	
result_db_worker	db82d590-14a5-42e9-bb63-73a0bc9523dc	172.18.0.3	2019-07-16 01:49:22	36	36	36	0	0	
result_db_worker	e40c42f6-18a7-425c-a53a-e79f910a1a67	192.168.64.15	2019-07-16 01:49:23	6130	6130	6130	0	0	
result_db_worker	e7c4335e-1479-462e-8f8e-b85e661c2999	192.168.64.15	2019-07-16 01:49:22	6131	6131	6131	0	0	
pro_ops_job_scheduling	80c524ac-0986-445f-9a99-63518e42b51	172.18.0.11	2019-07-16 01:49:22	0	0	0	0	0	
pro_ops_job_scheduling	ca984a07-04ca-4359-ae6c-e6602a30f68f	192.168.64.13	2019-07-16 01:49:23	67	67	67	0	69790	
data_collector	12c7ab2f-446e-4c07-b911-e19c6d4f999b	172.18.0.13	2019-07-16 01:49:23	245	245	0	245	0	
data_collector	3ab309e5-8721-499e-b352-4c811439ca29	172.18.0.13	2019-07-16 01:49:25	246	246	0	246	0	
data_collector	63764412-3b2c-44b7-9775-9c8cc0c3982a	192.168.64.7	2019-07-16 01:49:20	29445	29445	10673	18772	0	

© ProOps Management Server Version 0.1 - 2019 CableLabs

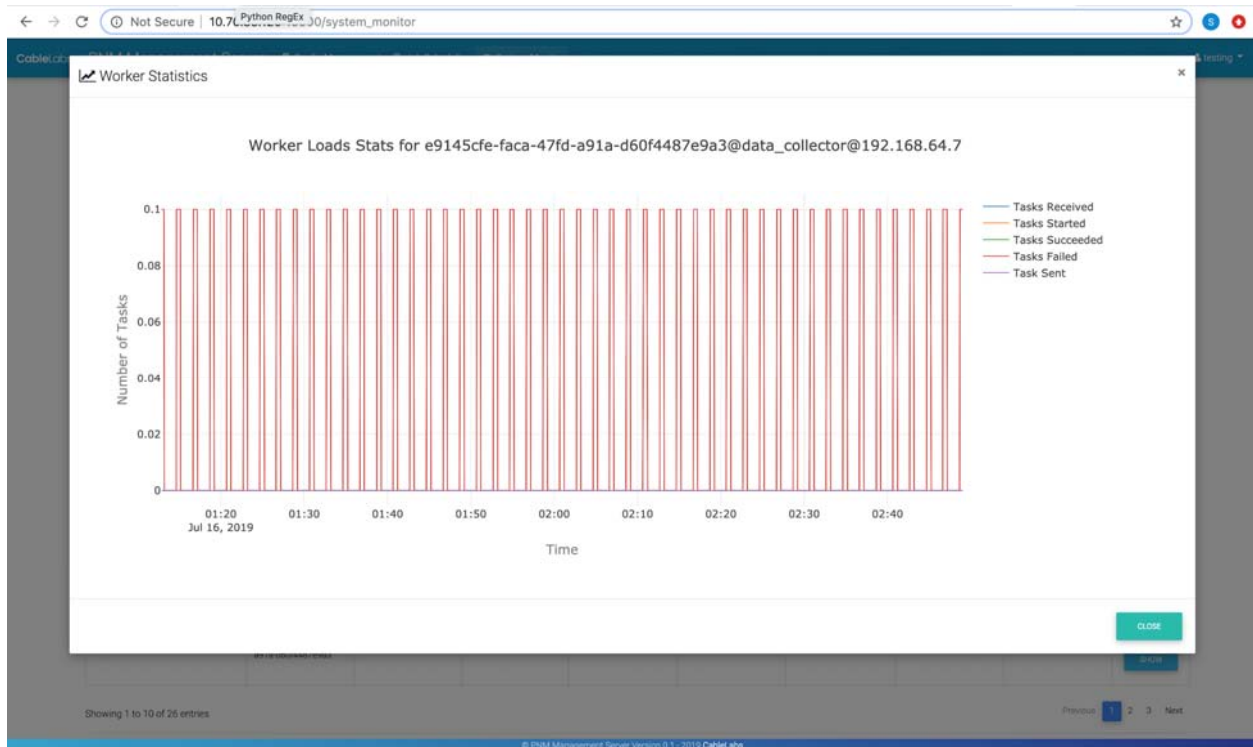
**Figure 13 – Detailed Worker System Monitoring**

Figure 12 and Figure 13 show the system monitoring capabilities that come with ProOps. The configuration server has a page that displays worker running statistics and host resources. In Figure 12, workers are running in separate docker containers and these containers show up as different hosts. The system monitoring page shown in Figure 13 also has a table that displays more detailed worker statistics. The statistics are calculated on a per-worker process basis, and include how many tasks are received, started, succeeded, failed, and sent. As an example, some data collection tasks failed here because of lab equipment changes and wiring changes during collection.



**Figure 14 – System Monitoring Graph**

Here in Figure 14 is an example visualization of how the job scheduling worker is handling tasks every few seconds. The graph shows average numbers for each counter over time. This output provides a quick and easy way to see whether system resources are sufficient, or ProOps is properly configured to perform well and as intended for the application configured.



**Figure 15 – System Monitoring Graph of Data Collection**

See Figure 15. Here is an example visualization of how the data collection worker is handling tasks every few seconds. The graph shows average numbers of each counter over time. This is useful for making sure the data collection resources are not overtaxed, or the network is not overtaxed as well.

## 8 ENVISIONED USE CASES

We created ProOps to support several use cases for both operators and vendors. Because of this, we had to create it to support several broad advantages.

- **Scalability** – The architecture of ProOps extends CCF in a modular, layered way. As such, we have full options of ways to package the software, and scale it to any use case, from a desktop deployment to a data center to a cloud architecture.
- **Low adoption friction** – Existing deployments of CCF can easily be extended to host ProOps. Once ProOps is installed, applications are simple updates to ProOps as additional, optional software modules that link into it. Each additional module installed within ProOps behaves like an update.
- **Flexibility and configurability** – Because ProOps allows anyone to add workers to the environment and configure rules for their interaction, each deployment of ProOps can be configured independently and in a broad range of ways. That flexibility provides full control to the user, which means it can be configured poorly, but because it is flexible it can be easily corrected and adjusted to fit changing needs too.
- **Actionable output** – The use of layers of workers as explained previously will allow ProOps to be configured so that the output can be acted on with confidence. If conditions change and the repair work that it suggests is no longer as desired, then adjustments to the configuration can correct that easily.
- **Interworkability** – Use of CCF allows one data source to support many applications. Likewise, ProOps is a platform that can support many applications in the same platform, allowing them to share information throughout the process of turning data into action. New applications built in ProOps can run with other existing applications in the same platform deployment. Hardware and virtual machine (VM) resources are the main consideration only.
- **Modular** – ProOps is modular, so you can replace elements like the worker queue, databases, and worker modules with your own creations or to fit our own corporate information technologies (IT) policies.
- **Speed** – CableLabs has and will continue to improve the speed of execution of ProOps, but what is important to note is that an application can be created in ProOps rapidly, so it brings speed of execution and deployment together. As we develop more basic workers for the environment, much more will be possible with just a simple reconfiguration of the platform to utilize existing workers in different ways.

Several operator use cases can be supported by ProOps.

- **Experiment with PNM** – You can connect ProOps to a small test network and test its capabilities against your expectations, and tune it for your own needs. Then depending on the outcome of your experiment, you can take the solution and deploy it in your network in many different ways, whichever suits your situation best.
- **Build and test a new PNM idea or solution** – As ProOps is a flexible platform, you can create your own PNM method and test it in an example or live network as you see fit, and even tune the platform to your new solution before full deployment.
- **Development environment** – Because of its flexibility and open architecture, you can develop prototype solutions inside of ProOps, essentially making it your team's development environment for operations solutions.
- **Network sampling** – ProOps was created to schedule work, so you can schedule network sampling for any operations need, including creating requirements for PNM, or to build a business case for a conceived PNM solution.
- **Grow your own PNM program** – Because you can use ProOps for the entire chain of PNM solutions, you can experiment, build, and deploy what you come up with, and support it yourself if you wish. Each development in ProOps works with the previous, so you can grow the entire solution and keep it updated as network needs change, keeping your solution up to date, and tailored for specific problems, architectures, or needs in an area.
- **Gather requirements for vendor supported solutions** – Because you can use it as an experiment platform or to develop a business case for a PNM effort, it can be used to support a vendor-supported PNM solution too.

Vendors also can use ProOps to their advantage.

- Rapid prototype new potential solutions – Vendors can use ProOps to develop their new PNM products or services, and
- A framework to support products and services – Vendors can use ProOps as a vehicle to deploy their solutions, to ease adoption and reduce the operations impact of the deployment, and to work in harmony with home-grown or other vendor solutions in an integrated manner.
- Free sample solutions – Vendors can deploy their proprietary solutions as example offerings with limited capabilities so operators can test their solution before purchasing the full enterprise supported version.
- Network sampling for developing operator specific solutions – Vendors can deploy a network sampling version of ProOps in an operator's network to help them determine the benefits of implementing a specific PNM or operations solution, to help get over the business case uncertainties that may hinder purchase decisions.
- Rapid, flexible data collection for consultation – Vendors who work as consultants to operators can use ProOps to collect network data to look for a specific problem, or to support a specific network issue they are trying to resolve.

ProOps was made to turn data into action, so we expect network operations and engineering personnel will discover new ways to make use of it that we have not yet defined.

## 9 SYSTEM ARCHITECTURE AND WORKFLOWS

### 9.1 Introduction

This section of the technical report explains the current version of ProOps with the reference application contained within. The Proactive Operations (ProOps) platform is a framework for constructing applications that turn network data into action. It relies on and connects to a network data collection framework. We will explain ProOps in terms of the Common Collection Framework (CCF) connected to it, and the workers contained within it, some of which are native to ProOps while others make the application run. We will also explain how to add your own workers to the environment to make ProOps bring life to your own application ideas.

### 9.2 ProOps and CCF

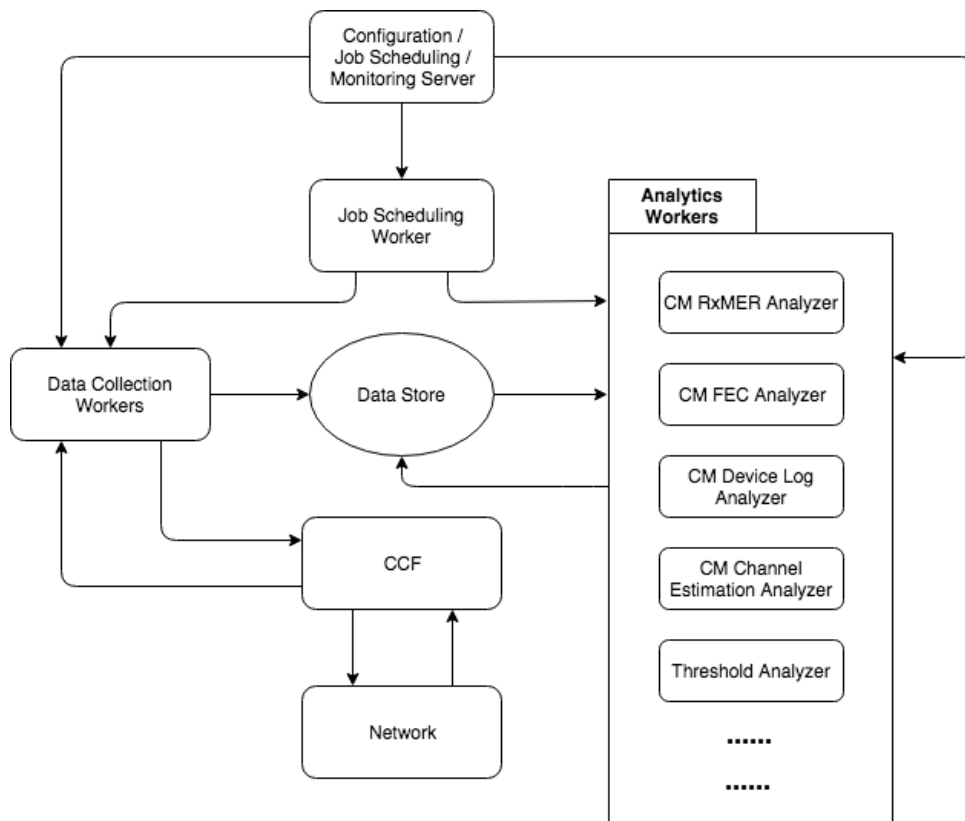


Figure 16 – ProOps with Data Store and CCF

#### 9.2.1 Component Descriptions

- Configuration/Job Scheduling/Monitoring server hosts all information related to worker initial configuration and scheduling. Workers request initial configurations including task queue information from the configuration server through pre-defined RESTful APIs on start up, and report worker statistics to the configuration server.
- Job Scheduling worker is a special worker that handles periodic job queueing and polling frequency changes when a job is triggered to be run more frequently. The Job Scheduling worker subscribes to a task queue in order to connect to and get updates from the configuration server. It distributes tasks to various workers based on the job scheduling and configuration information.
- Data Store is a software component that stores all the collected data and calculation results from workers. Currently, the implementation uses PostgreSQL with RESTful APIs built on top to serve the needs of workers.

- Data Collection Workers are a set of workers which periodically collect data by calling CCF APIs and store decoded data into the Data Store.
- CCF is the Common Collection Framework developed by CableLabs.
- Analytics Workers are a set of workers which handle PNM related calculations and tasks in a distributed, multi-processed way by utilizing FIFO task queues.

## 10 WORKERS AND WORKFLOWS

### 10.1 Data Collection Worker

#### 10.1.1 Functionality Description

Data collection worker is a type of worker that handles all data collection activities by calling CCF APIs. The data collection tasks are handled in multi-processed and multi-threaded way by the data collection worker and the collected data is decoded and stored in the data service that is specified in the job description. The following data types are supported:

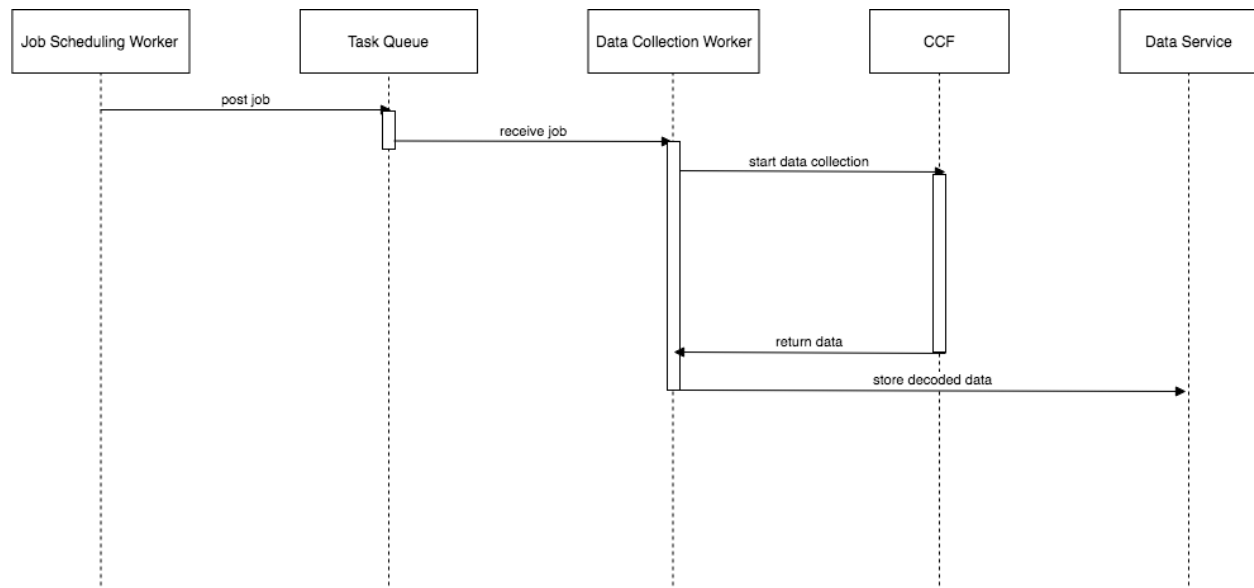
**Table 1 – Data Types**

Data Type Name	Description
cmPNMDsRxMer	RxMER data measured by the CM on downstream OFDM channel(s), the data is encoded in binary file based on DOCSIS 3.1 OSSI specification defined format.
cmDsFecStats	FEC statistics measured by the CM on downstream OFDM channel(s), the data is read from CM's MIBs.
cmPNMDsConDisp	Constellation diagram measured by the CM on downstream OFDM channel(s), the data is encoded in binary file based on DOCSIS 3.1 OSSI specification defined format.
cmPNMFecSum	FEC summary data measured by the CM on downstream OFDM channel(s), the data is encoded in binary file based on DOCSIS 3.1 OSSI specification defined format.
cmPNMUsPreEq	Pre-equalization I and Q data that the CM is using for upstream OFDMA channel(s), the data is encoded in binary file based on DOCSIS 3.1 OSSI specification defined format.
cmPNMSpecAnData	Spectrum analysis data measured by the CM on downstream, the data is encoded in binary file based on DOCSIS 3.1 OSSI specification defined format.
cmEvents	CM device event logs, the data is read from CM's MIBs.
cmOfdmChanTopo	OFDM channel ifIndex-based logical topology.

More data types will be supported as CCF drivers and applications require, as specifications support.

#### 10.1.2 Workflow

The data collection worker subscribes to a task queue and performs data collection based on the job description. The collected data is decoded, formatted, and stored into the data service address and API described in the job description. The following diagram shows the workflow of the data collection worker.



**Figure 17 – Data Collection Worker's Workflow**



### 10.1.3 Job Description Syntax and Format

A typical job description for the data collection worker is shown below.

```
{
  "cmts_ip": "10.32.40.68",
  "dccf_ip": "10.95.254.82:8888",
  "db_server_address_and_api": "http://10.70.35.120:10001/putData/",
  "namespace": "dccf",
  "cm_ip_to_filter": "",
  "cmts_community_string": "CableLabsSNMP",
  "cm_community_string": "private",
  "drivers": ["cmPNMDsRxMer"],
  "drivers_actual_name": ["cmPNMDsRxMer"],
  "timeouts": [300]
}
```

The table below describes each field in the job description.

**Table 2 – Data Collection Worker Job Description**

Field Name	Description
cmts_ip	The CMTS(s) IP address(s) that the data collection worker is collecting data from. This field's value can be a single IP address in string or a list of IP address. When this field's value is a list of IP addresses, the job scheduling worker will generate separate tasks for each CMTS.
dccf_ip	The IP address of the CCF instance that the data collection worker is going to call.
db_server_address_and_api	The IP address, port, and API that the data collection worker will use to store collected data into the data service.
namespace	The namespace of the CCF drivers in the job description. For example, cmPNMDsRxMer belongs to "dccf" namespace.
cm_ip_to_filter	The key string or a list of strings for the data collection worker to target specific CMs for data collection. The value of this field can be a string or a list of strings. When the value is an empty string, the data collection worker collects data from all CMs in DOCSIS 3.1 mode that are discovered by CCF.
cmts_community_string	The community string that CCF can use to read MIB values from the CMTS.
drivers	The names of data collection drivers that the data collection worker is going to call.
drivers_actual_name	The actual names of data collection drivers that the data collection worker will use to generate CCF API calls. This field is used to compromise that custom drivers use different strings as their names and APIs.
timeouts	A list of timeout values that are going to be used by each driver.

## 10.2 Downstream RxMER Statistics Worker

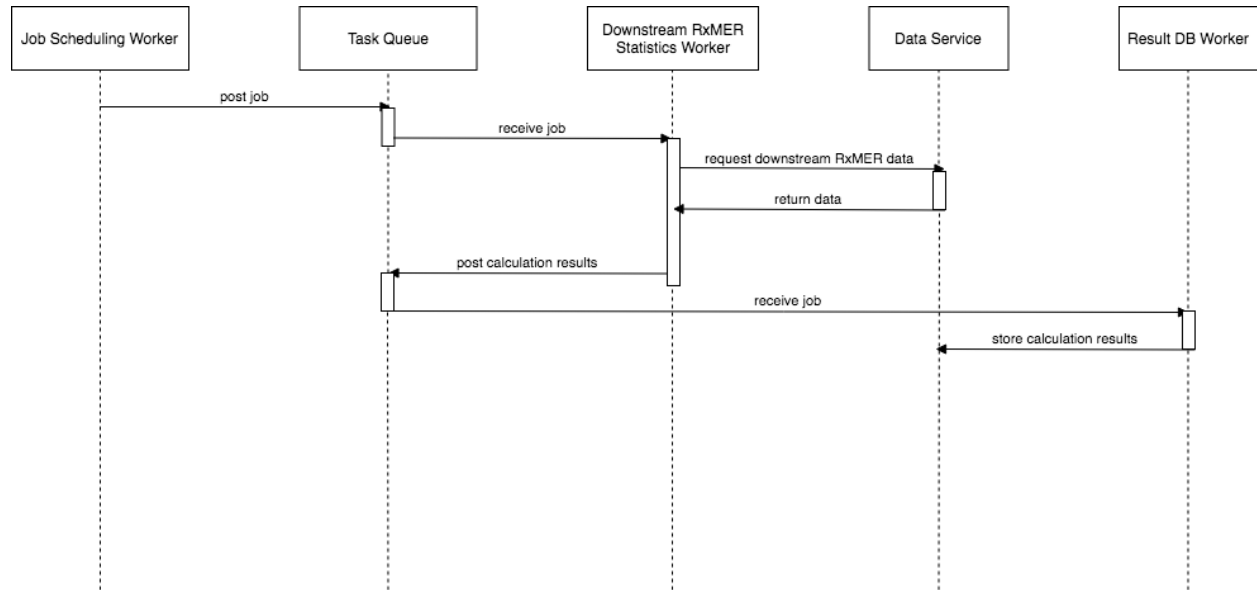
### 10.2.1 Functionality Description

Downstream RxMER statistics worker is a type of worker that handles downstream RxMER data statistical calculations. The following calculations are performed on each RxMER sample:

- mean value of all MER values
- standard deviation of all MER values,
- max value from of all MER values
- min value from of all MER values
- range – the absolute difference between max value and min value
- linear regression of MER values over frequency
- quantiles of all MER values (5%, 50%, 95%)
- confidence interval to the mean value (with confidence of 0.95)

### 10.2.2 Workflow

The downstream RxMER statistics worker subscribes to a task queue and performs RxMER statistical calculations based on the job description. The worker retrieves RxMER data from the data service and publishes calculation results to a result task queue. The results in the result task queue are then collected by the result DB worker and stored into the data service. The following diagram shows the workflow of the downstream RxMER statistics worker.



**Figure 18 – Downstream RxMER per Subcarrier Statistics Worker’s Workflow**

### 10.2.3 Job Description Syntax and Format

A typical job description for the downstream RxMER statistics worker is shown below.

```

{
  "data_service_api": "/getCmRxMerData/",
  "data_service_address": "http://10.70.35.120:10001",
  "data_service_store_api": "/cmRxMerStats/",
  "cm_mac": [],
  "duration": 86400
}
  
```

The table below describes each field in the job description.

**Table 3 – Downstream RxMER Per Subcarrier Statistics Worker Job Description**

Field Name	Description
data_service_api	The data service API that the worker will retrieve RxMER data from.
data_service_address	The IP address data service that worker will retrieve RxMER data from.
data_service_store_api	The data service API that the worker will specify in the result for the result DB worker to use to store data.
cm_mac	A list of specific CM MAC addresses that the worker will perform data retrieving and calculation on. If the list is empty, the worker will retrieve all available samples without limitation on MAC addresses.
duration	The time duration that the worker will perform calculations on. The duration uses second as the unit. For example, if 86400 is used as the value of this field, the worker will retrieve RxMER samples collected and stored from 24 hours before to current.

## 10.3 Downstream Channel Estimation Statistics Worker

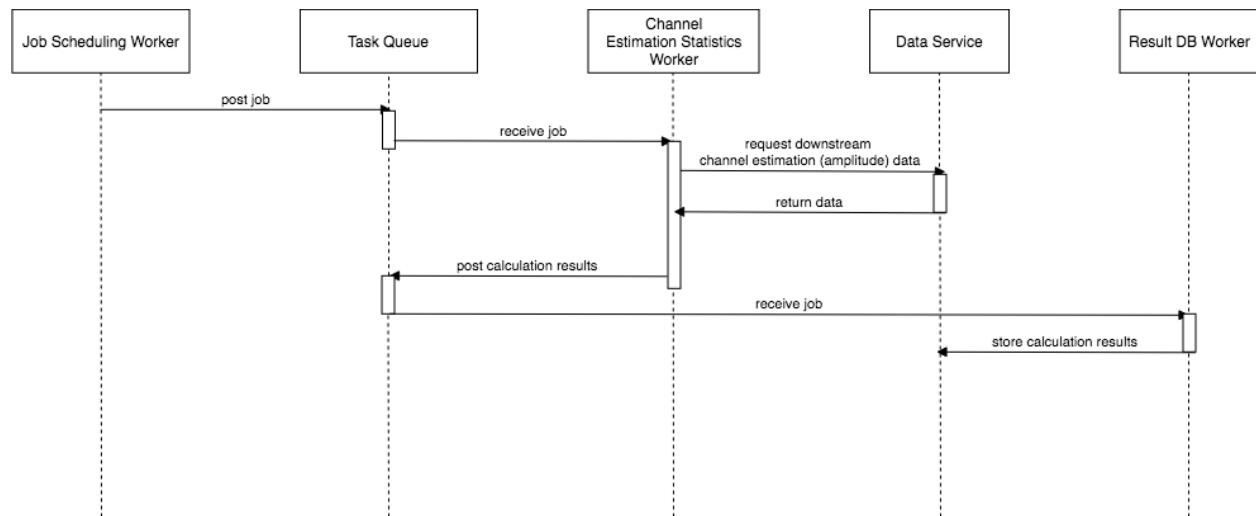
### 10.3.1 Functionality Description

Downstream channel estimation statistics worker is a type of worker that performs statistical calculations on downstream channel estimation's amplitude values. The following calculations are performed on each downstream channel estimation sample.

- tilt over frequency (dB/MHz)
- potential echoes and inferred distances of each echo (unit: foot)

### 10.3.2 Workflow

The downstream channel estimation statistics worker subscribes to a task queue and performs statistical calculations on channel estimation's amplitude data based on the job description. The worker retrieves channel estimation data from the data service and publishes calculation results to a result task queue. The results in the result task queue are then collected by the result DB worker and stored into the data service. The following diagram shows the workflow of the downstream channel estimation statistics worker.



**Figure 19 – Downstream Channel Estimation Statistics Worker's Workflow**

### 10.3.3 Job Description Syntax and Format

A typical job description for the downstream channel estimation statistics worker is shown below.

```

{
  "data_service_api": "/getCmChannelEstimationData/",
  "data_service_address": "http://10.70.35.120:10001",
  "data_service_store_api": "/cmChannelEstimationStats/",
  "cm_mac": [],
  "duration": 86400
}
  
```

The table below describes each field in the job description.

**Table 4 – Downstream Channel Estimation Statistics Worker Job Description**

Field Name	Description
data_service_api	The data service API that the worker will retrieve channel estimation data from.
data_service_address	The IP address data service that worker will retrieve channel estimation data from.
data_service_store_api	The data service API that the worker will specify in the result for the result DB worker to use to store data.

Field Name	Description
cm_mac	A list of specific CM MAC addresses that the worker will perform data retrieving and calculation on. If the list is empty, the worker will retrieve all available samples without limitation on MAC addresses.
duration	The time duration that the worker will perform calculations on. The duration uses second as the unit. For example, if 86400 is used as the value of this field, the worker will retrieve channel estimation samples collected and stored from 24 hours before to current.

## 10.4 Downstream Constellation Display Statistics Worker

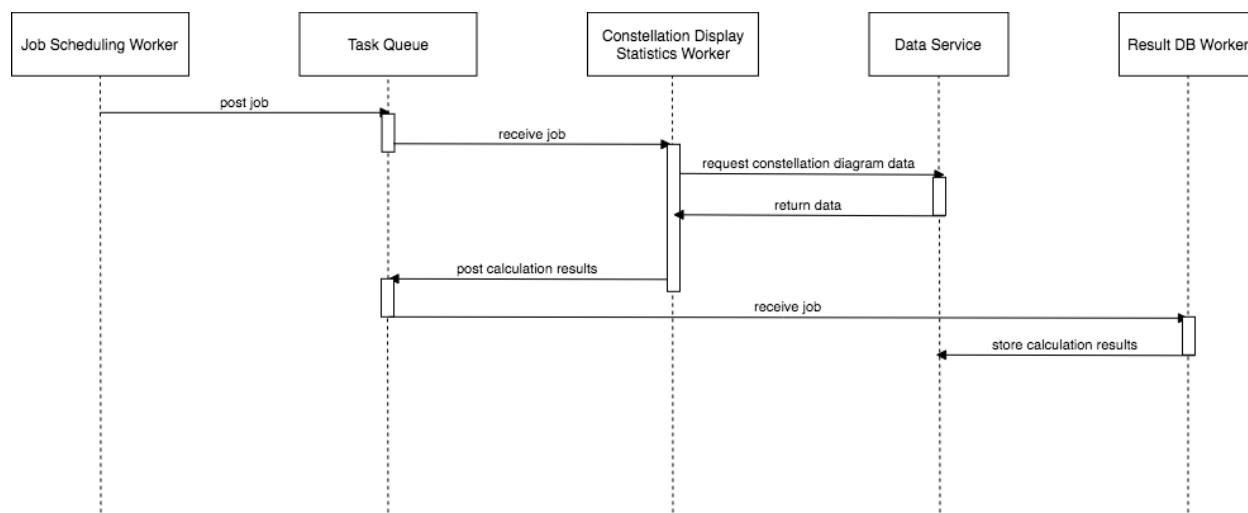
### 10.4.1 Functionality Description

Downstream constellation display statistics worker is a type of worker that performs statistical calculations on downstream constellation diagram data. The following calculations are performed:

- for each target, calculate the following of all closest points
  - max value (on I, Q values and the square root of  $I^2 + Q^2$ )
  - min value (on I, Q values and the square root of  $I^2 + Q^2$ )
  - mean value (on I, Q values and the square root of  $I^2 + Q^2$ )
  - standard deviation (on I, Q values and the square root of  $I^2 + Q^2$ )
  - median value (on I, Q values and the square root of  $I^2 + Q^2$ )
  - 95, 75, 60, 25, 5 percentiles (on I, Q values and the square root of  $I^2 + Q^2$ )
  - Rayleigh fit (location and scale on the square root of  $I^2 + Q^2$ )
  - Anderson Darling test (k statistic and p value on the square root of  $I^2 + Q^2$ )
- combine all retrieved constellation diagram samples and calculate
  - max value (on the square root of  $I^2 + Q^2$ )
  - min value (on the square root of  $I^2 + Q^2$ )
  - mean value (on the square root of  $I^2 + Q^2$ )
  - standard deviation (on the square root of  $I^2 + Q^2$ )
  - median value (on the square root of  $I^2 + Q^2$ )
  - 95, 75, 60, 25, 5 percentiles (on the square root of  $I^2 + Q^2$ )
  - Rayleigh fit (location and scale on the square root of  $I^2 + Q^2$ )
  - Anderson Darling test (k statistic and p value on the square root of  $I^2 + Q^2$ )

### 10.4.2 Workflow

The downstream constellation display statistics worker subscribes to a task queue and performs statistical calculations on constellation diagram data based on the job description. The worker retrieves constellation diagram data from the data service and publishes calculation results to a result task queue. The results in the result task queue are then collected by the result DB worker and stored into the data service. The following diagram shows the workflow of the downstream constellation display statistics worker:



**Figure 20 – Downstream Constellation Display Statistics Worker’s Workflow**

### 10.4.3 Job Description Syntax and Format

A typical job description for the downstream channel estimation statistics worker is shown below.

```

{
  "data_service_api": "/getCmConDispData/",
  "data_service_address": "http://10.70.35.120:10001",
  "data_service_store_api": "/putCmConDispStats/",
  "cm_mac": [],
  "duration": 86400
}
  
```

The table below describes each field in the job description.

**Table 5 – Downstream Constellation Display Statistics Worker Job Description**

Field Name	Description
data_service_api	The data service API that the worker will retrieve constellation diagram data from.
data_service_address	The IP address data service that worker will retrieve constellation diagram data from.
data_service_store_api	The data service API that the worker will specify in the result for the result DB worker to use to store data.
cm_mac	A list of specific CM MAC addresses that the worker will perform data retrieving and calculation on. If the list is empty, the worker will retrieve all available samples without limitation on MAC addresses.
duration	The time duration that the worker will perform calculations on. The duration uses second as the unit. For example, if 86400 is used as the value of this field, the worker will retrieve channel estimation samples collected and stored from 24 hours before to current.

## 10.5 CM Result Trigger Analyzer

### 10.5.1 Functionality Description

The CM result trigger analyzer calculate scores on events that are noted in the job description. The scores are used by the job scheduling worker to determine any changes to make on data collection frequency. The following calculations are performed by the CM result trigger analyzer:

- for each data type that can be analyzed by the trigger analyzer
  - calculate score based on interested fields, thresholds and weights, threshold types include “greater\_than”, “smaller\_than”, “in\_range”, and “out\_range”

- apply operations that are specified in the job description to the above results
  - “OR” operation updates the previous scores from the current field calculation result
  - “AND” operation keeps intersected fields and adds their values and removes all non-intersected fields from the current result

### 10.5.2 Workflow

The CM result trigger analyzer subscribes to a task queue and performs score calculations on data types and thresholds described in the job description. The worker retrieves data from the data service and publishes calculation results to a result task queue. The results in the result task queue are then collected by the result DB worker and stored into the data service. The following diagram shows the workflow of the CM result trigger analyzer.

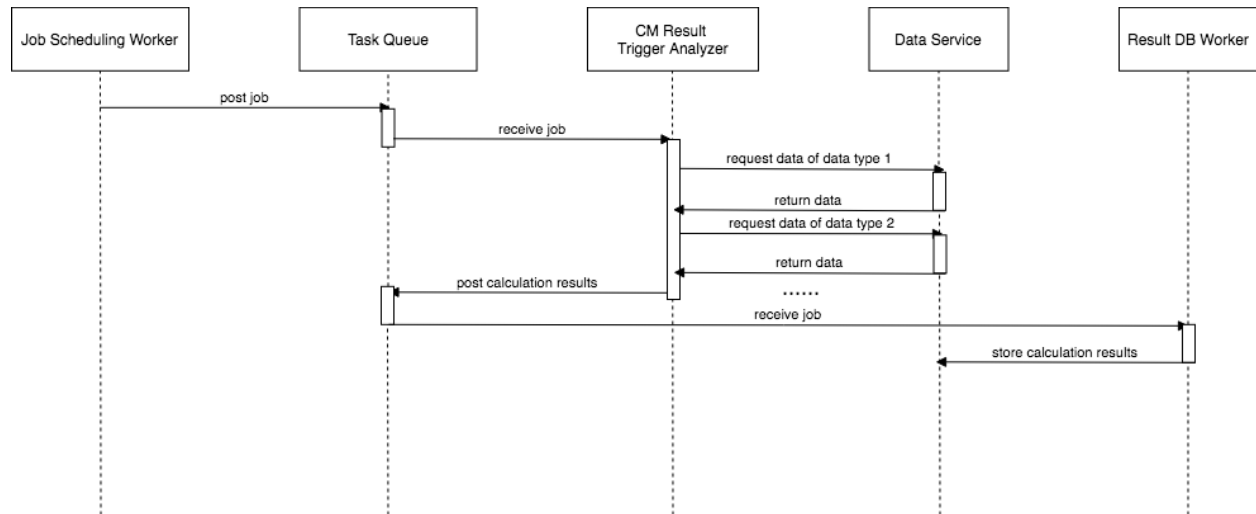


Figure 21 – CM Result Trigger Analyzer

### 10.5.3 Job Description Syntax and Format

A typical job description for the downstream channel estimation statistics worker is shown below.

```

{
  "cm_mac": [],
  "data_service_address": "http://10.70.35.120:10001",
  "data_service_store_api": "/cmTriggerResults/",
  "cm_rxmer_stats": {
    "data_service_api": "/cmRxMerStats/",
    "data_service_address": "http://10.70.35.120:10001",
    "duration": 86400,
    "trigger_field_config": [
      {
        "field": "range",
        "weight": 1.0,
        "threshold": {
          "greater_than": 2
        }
      },
      {
        "field": "std",
        "weight": 1.0,
        "operation": "AND",
        "threshold": {
          "greater_than": 1.5
        }
      }
    ]
  }
}
  
```

```

        {
            "field": "ordinary_linear_regression_m",
            "weight": 1.0,
            "operation": "AND",
            "threshold": {
                "out_range": [-0.1, 0.1]
            }
        }
    ]
},
"cm_event":{
    "data_service_api": "/getCmEventsData/",
    "data_service_address": "http://10.70.35.120:10001",
    "duration": 86400,
    "trigger_field_config": [
        {
            "field": "T3_T4_count",
            "weight": 1.0,
            "threshold": {
                "greater_than": 3
            }
        }
    ]
},
"cm_fec_stats":{
    "data_service_api": "/getCmFecStatsData/",
    "data_service_address": "http://10.70.35.120:10001",
    "duration": 86400,
    "trigger_field_config": [
        {
            "field": "uncorrectable_codewords",
            "weight": 1.0,
            "threshold": {
                "greater_than": 100
            }
        }
    ]
},
"cm_channel_estimation":{
    "data_service_api": "/cmChannelEstimationStats/",
    "data_service_address": "http://10.70.35.120:10001",
    "duration": 86400,
    "trigger_field_config": [
        {
            "field": "tilt_db_per_mhz",
            "weight": 1.0,
            "threshold": {
                "out_range": [-0.1, 0.1]
            }
        }
    ]
}
}

```

The table below describes each field in the job description.

**Table 6 – CM Result Trigger Analyzer Worker Job Description**

Field Name	Description
cm_mac	A list of specific CM MAC addresses that the worker will perform data retrieving and calculation on. If the list is empty, the worker will retrieve all available samples without limitation on MAC addresses.
data_service_address	The IP address data service that worker will specify in the result for the result DB worker to use to store data.
data_service_store_api	The data service API that the worker will specify in the result for the result DB worker to use to store data.
cm_rxmer_stats	If the cm_rxmer_stats field exists, the trigger analyzer will retrieve data and perform analysis on CM downstream RxMER statistical calculation results based on this field's configurations.
cm_event	If the cm_event field exists, the trigger analyzer will retrieve data and perform analysis on CM device T3 and T4 event data based on this field's configurations.
cm_fec_stats	If the cm_fec_stats field exists, the trigger analyzer will retrieve data and perform analysis on CM FEC codeword data based on this field's configurations.
cm_channel_estimation	If the cm_channel_estimation field exists, the trigger analyzer will retrieve data and perform analysis on CM downstream channel estimation data based on this field's configurations.
data_service_api	The API that the trigger analyzer will use to retrieve data.
data_service_address	The IP address that the trigger analyzer will use to retrieve data.
duration	The time duration that the worker will perform calculations on. The duration uses second as the unit. For example, if 86400 is used as the value of this field, the worker will retrieve data collected and stored from 24 hours before to current.
trigger_field_config	An ordered list of configurations that describe thresholds and weights of different target fields. For example, the following field configuration specifies that the trigger analyzer finds all channel estimation data samples that have tilt_db_per_mhz greater than 0.1 or smaller than -0.1, and calculates the score using weight 1.0: <pre> {   "field": "tilt_db_per_mhz",   "weight": 1.0,   "threshold": {     "out_range": [-0.1, 0.1]   } } </pre>

## 10.6 Result DB Worker

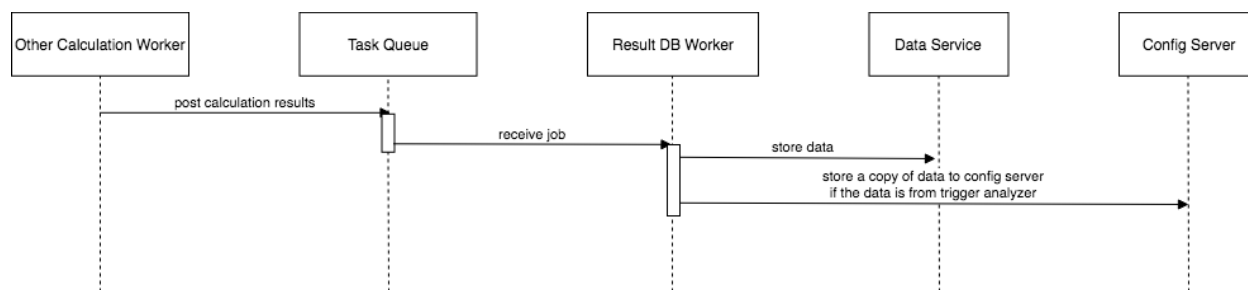
### 10.6.1 Functionality Description

The result DB worker serves the purpose of separating calculation and DB insertion work. By taking the information in the job description posted by other calculation workers, the result DB worker points to data services and stores calculation results to databases. The separation of calculation and DB insertion allows the calculation workers to move on to the next task immediately after finishing the current task without waiting on DB bottlenecks. The result DB worker also sends a copy of trigger results to the configuration server as the configuration server and the job scheduling worker manage the data collection frequency changes.

### 10.6.2 Workflow

The result DB worker subscribes to a task queue and stores data from the queue to data services based on the job description. The following diagram shows the workflow of the downstream constellation display statistics worker.





**Figure 22 – Result DB Worker's Workflow**

### 10.6.3 Job Description Syntax and Format

A typical job description for the downstream channel estimation statistics worker is shown below.

```

{
  "worker_identifier": <the worker identifier of the calculation worker>,
  "worker_job_description": <the current calculation worker's original job description>,
  "result": <the calculation result from the calculation worker>
}

```

The table below describes each field in the job description.

**Table 7 – Result DB Worker Job Description**

Field Name	Description
worker_identifier	The worker identifier of the calculation worker that is publishing results. This field helps the result DB worker to find out what type of the result is.
worker_job_description	The job description that the current calculation worker received. The information in the job description helps the result DB worker to point to the data service where the result is going to be stored.
result	The calculation result from the calculation worker.

## 10.7 ProOps Configuration Server

### 10.7.1 Functionality Description

The configuration server (config server) hosts the configuration data which define the messaging mechanisms and necessary additional configurations for workers. It describes the workers to be connected, and any changes that affect the work they do. The user can control applications through this interface for the most part, as the configuration is what defines the application from a pool of workers. When workers start up, they request configurations from the configuration server using RESTful APIs.

### 10.7.2 RESTful APIs

#### 10.7.2.1 Retrieve worker configuration

<b>Title</b>	<b>Retrieve worker configuration</b> Retrieve basic configurations of workers based on worker identifiers.
<b>URL</b>	/worker_portal?worker_identifier=<WORKER_IDENTIFIER>
<b>Method</b>	GET
<b>URL Params</b>	Required: worker_identifier

<b>Sample Response</b>	<pre>{   "task_queue_name": &lt;task_queue_name&gt;,   "task_queue_username": &lt;task_queue_username&gt;,   "task_queue_password": &lt;task_queue_password&gt;,   "task_queue_ip": &lt;task_queue_ip&gt;,   "task_queue_exchange": &lt;task_queue_exchange&gt;,   "task_queue_routing_key": &lt;task_queue_routing_key&gt;,   "cm_trigger_result_api": "cm_trigger_results",   "&lt;optional_config_field&gt;": &lt;optional_config_field_value&gt;,   ..... other optional config fields and values ..... }</pre>
<b>Sample Call</b>	curl -X GET http://10.70.35.120:10000/worker_portal?worker_identifier=cm_channel_estimation_stats

### 10.7.2.2 Update worker stats

<b>Title</b>	<b>Update worker stats</b> Retrieves the MAC address list of Cable Modems that are on the specified slot/port/channel.
<b>URL</b>	/worker_portal
<b>Method</b>	POST
<b>Example Data with API Call</b>	<pre>{   "worker_identifier": &lt;worker_identifier&gt;,   "host_ip": &lt;worker_host_ip&gt;,   "process_id": &lt;worker_process_id&gt;,   "task_received": &lt;number_of_tasks_received&gt;,   "task_started": &lt;number_of_tasks_started&gt;,   "task_succeeded": &lt;number_of_tasks_succeeded&gt;,   "task_failed": &lt;number_of_tasks_failed&gt;,   "task_sent": &lt;number_of_tasks_sent&gt;,   "host_cpu_usage": &lt;cpu_usage_on_host&gt;,   "host_mem_usage": &lt;memory_usage_on_host&gt; }</pre>

## 10.8 ProOps Data Service

### 10.8.1 Functionality Description

The data service stores data from workers in the system. It stores decoded and formatted data from the data collection worker, calculation results from calculation workers, and trigger analysis result from trigger analyzer. It also implements pre-defined APIs to serve all the components and workers with the data they need. The reference implementation of ProOps data service uses PostgreSQL database. It also has visualization capabilities for downstream RxMER data.

### 10.8.2 RESTful APIs

#### 10.8.2.1 Retrieve CM RxMER data

<b>Title</b>	<b>Retrieve CM RxMER data</b> Retrieve CM RxMER data stored in the data service.
<b>URL</b>	/getCmRxMerData
<b>Method</b>	GET
<b>URL Params</b>	<b>Optional:</b> cm_mac: specifies the CM MAC address where the RxMER data was collected from. Default: "ALL". duration: specifies the amount of data will be returned based on their timestamps, all data stored before current_time - duration will be returned. Default: 0

<b>Sample Response</b>	<pre>{   "0": {     "cm_mac": "0050F112E00E",     "cmts_ip": "10.32.40.68",     "timestamp": 1560353317.0,     "spacing": "50",     "first_active_subcarrier_index": "1160",     "first_active_subcarrier_frequency": "303600000",     "start_freq": "245600000",     "channel_id": "159",     "mer_data": "52.50,51.50,52.50,50.75....."   },   "1": {     .....   }   ..... }</pre>
<b>Sample Call</b>	curl -X GET http://10.70.35.120:10001/getCmRxMerData/?cm_mac=0050F112E00E&duration=86400

### 10.8.2.2 Retrieve CM RxMER statistics data

<b>Title</b>	<b>Retrieve CM RxMER statistics data</b> Retrieve CM RxMER statistics data stored in the data service.
<b>URL</b>	/getCmRxMerStats
<b>Method</b>	GET
<b>URL Params</b>	<b>Optional:</b> <ul style="list-style-type: none"> <li>cm_mac: specifies the CM MAC address where the RxMER stats was calculated from. Default: "ALL".</li> <li>duration: specifies the amount of data will be returned based on their timestamps, all data stored before current_time - duration will be returned. Default: 0</li> </ul>
<b>Sample Response</b>	<pre>{   "0": {     "cm_mac": "0050F112E00E",     "cmts_ip": "10.32.40.68",     "timestamp": 1560353317.0,     "std": 1.5,     "max": 50.5,     "min": 45.5,     "range": 5.0,     "ordinary_linear_regression_m": 0.001,     "quantiles": "{\"95\": 50.0, .....}",     "confidence_interval_to_mean": .....   },   "1": {     .....   }   ..... }</pre>
<b>Sample Call</b>	curl -X GET http://10.70.35.120:10001/cmRxMerStats/?cm_mac=0050F112E00E&duration=86400

### 10.8.2.3 Retrieve CM downstream constellation display data

<b>Title</b>	<b>Retrieve CM downstream constellation display data</b> Retrieve CM downstream constellation display data from data service.
<b>URL</b>	/getCmConDispData
<b>Method</b>	GET

<b>URL Params</b>	<b>Optional:</b> <ul style="list-style-type: none"> <li>cm_mac: specifies the CM MAC address where the constellation display data was collected from. Default: "ALL".</li> <li>duration: specifies the amount of data will be returned based on their timestamps, all data stored before current_time - duration will be returned. Default: 0</li> </ul>
<b>Sample Response</b>	<pre>{   "0": {     "cm_mac": "0050F112E00E",     "cmts_ip": "10.32.40.68",     "timestamp": 1560353317.0,     "spacing": "50",     "subcarrier_zero_frequency": "245600000",     "modulation_type": "7",     "number_of_sample_symbols": "8192",     "channel_id": "33",     "data_length": "16384",     "i_part_data": "0.1,0.1.....",     "q_part_data": "0.1,0.1....."   },   "1": {     .....   }   ..... }</pre>
<b>Sample Call</b>	<pre>curl -X GET http://10.70.35.120:10001/getCmConDispData/?cm_mac=0050F112E00E&amp;duration=86400</pre>

#### 10.8.2.4 Retrieve CM events data

<b>Title</b>	<b>Retrieve CM events data</b> Retrieve CM events data from data service.
<b>URL</b>	/getCmEventsData
<b>Method</b>	GET
<b>URL Params</b>	<b>Optional:</b> <ul style="list-style-type: none"> <li>cm_mac: specifies the CM MAC address where the CM events data was calculated from. Default: "ALL".</li> <li>duration: specifies the amount of data will be returned based on their timestamps, all data stored before current_time - duration will be returned. Default: 0</li> </ul>
<b>Sample Response</b>	<pre>{   "0": {     "cm_mac": "0050F112E00E",     "cmts_ip": "10.32.40.68",     "timestamp": 1560353317.0,     "cm_events": ".....",   },   "1": {     .....   }   ..... }</pre>
<b>Sample Call</b>	<pre>curl -X GET http://10.70.35.120:10001/getCmEventsData/?cm_mac=0050F112E00E&amp;duration=86400</pre>

#### 10.8.2.5 Retrieve CM DOCSIS 3.1 downstream OFDM FEC statistics data

<b>Title</b>	<b>Retrieve CM DOCSIS 3.1 downstream OFDM FEC statistics data</b> Retrieve CM DOCSIS 3.1 downstream OFDM FEC statistics data from data service.
--------------	--

<b>URL</b>	/getCmFecStatsData
<b>Method</b>	GET
<b>URL Params</b>	<b>Optional:</b> <ul style="list-style-type: none"> <li>cm_mac: specifies the CM MAC address where the CM FEC statistics data was collected from. Default: "ALL".</li> <li>duration: specifies the amount of data will be returned based on their timestamps, all data stored before current_time - duration will be returned. Default: 0</li> </ul>
<b>Sample Response</b>	<pre>{   "0": {     "cm_mac": "0050F112E00E",     "cmts_ip": "10.32.40.68",     "timestamp": 1560353317.0,     "total_codewords": "1000000",     "corrected_codewords": "200000",     "uncorrectable_codewords": "0"   },   "1": {     .....   }   ..... }</pre>
<b>Sample Call</b>	<pre>curl -X GET http://10.70.35.120:10001/getCmFecStatsData/?cm_mac=0050F112E00E&amp;duration=86400</pre>

#### 10.8.2.6 Retrieve CM DOCSIS 3.1 downstream OFDM channel estimation data

<b>Title</b>	<b>Retrieve CM DOCSIS 3.1 downstream OFDM channel estimation data</b> Retrieve CM DOCSIS 3.1 downstream OFDM channel estimation data from data service.
<b>URL</b>	/getCmChannelEstimationData
<b>Method</b>	GET
<b>URL Params</b>	<b>Optional:</b> <ul style="list-style-type: none"> <li>cm_mac: specifies the CM MAC address where the CM channel estimation data was collected from. Default: "ALL".</li> <li>duration: specifies the amount of data will be returned based on their timestamps, all data stored before current_time - duration will be returned. Default: 0</li> </ul>
<b>Sample Response</b>	<pre>{   "0": {     "cm_mac": "0050F112E00E",     "cmts_ip": "10.32.40.68",     "timestamp": 1560353317.0,     "spacing": "50",     "first_active_subcarrier_index": "1160",     "first_active_subcarrier_frequency": "303600000",     "start_freq": "245600000",     "channel_id": "159",     "chest_data": "0.55,1.52,0.98,1.0....."   },   "1": {     .....   }   ..... }</pre>
<b>Sample Call</b>	<pre>curl -X GET http://10.70.35.120:10001/getCmChannelEstimationData/?cm_mac=0050F112E00E&amp;duration=86400</pre>

**10.8.2.7 Retrieve CM DOCSIS 3.1 downstream OFDM channel estimation statistics**

<b>Title</b>	<b>Retrieve CM DOCSIS 3.1 downstream OFDM channel estimation statistics</b> Retrieve CM DOCSIS 3.1 downstream OFDM channel estimation statistics from data service.
<b>URL</b>	/getCmChannelEstimationStats
<b>Method</b>	GET
<b>URL Params</b>	<b>Optional:</b> <ul style="list-style-type: none"> <li>cm_mac: specifies the CM MAC address where the CM channel estimation statistics was calculated from. Default: "ALL".</li> <li>duration: specifies the amount of data will be returned based on their timestamps, all data stored before current_time - duration will be returned. Default: 0</li> </ul>
<b>Sample Response</b>	<pre>{   "0": {     "cm_mac": "0050F112E00E",     "cmts_ip": "10.32.40.68",     "timestamp": 1560353317.0,     "distances": ".....",     "tilt_db_per_mhz": 0.01   },   "1": {     .....   }   ..... }</pre>
<b>Sample Call</b>	curl -X GET http://10.70.35.120:10001/getCmChannelEstimationStats/?cm_mac=0050F112E00E&duration=86400

**10.8.2.8 Retrieve CM trigger results**

<b>Title</b>	<b>Retrieve CM trigger results</b> Retrieve CM trigger results.
<b>URL</b>	/getCmTriggerResults
<b>Method</b>	GET
<b>URL Params</b>	<b>Optional:</b> <ul style="list-style-type: none"> <li>cm_mac: specifies the CM MAC address where the CM trigger results was calculated from. Default: "ALL".</li> <li>duration: specifies the amount of data will be returned based on their timestamps, all data stored before current_time - duration will be returned. Default: 0</li> </ul>
<b>Sample Response</b>	<pre>{   "0": {     "cm_mac": "0050F112E00E",     "cmts_ip": "10.32.40.68",     "timestamp": 1560353317.0,     "total_weight": 100.0,     "cm_event_weight": 20.0,     "cm_rxmer_stats_weight": 30.0,     "cm_fec_stats_weight": 20.0,     "cm_channel_estimation_weight": 30.0   },   "1": {     .....   }   ..... }</pre>
<b>Sample Call</b>	curl -X GET http://10.70.35.120:10001/getCmTriggerResults/?cm_mac=0050F112E00E&duration=86400

## 11 CONCLUSION

ProOps is free to use by CableLabs' vendors and operator members. Contact the authors of this technical report to obtain a copy of the software, and to get help configuring it for your needs. We encourage everyone who can access ProOps to use it in any way they envision, from reviewing the architecture to taking those advantages in their own developments on up to full use and deployment of the code base.

ProOps is a platform constructed for turning data and information into action. It is built around well-defined, general steps that facilitate making decisions automatically, but while keeping control fully in the hand of the users. Rather than relying on assumed experts at hand to review network data to determine what needs to be done, ProOps provides an environment to automate that work. This means limited expert resources can be shared through ProOps, thus extending the effectiveness of expertise. For example, CableLabs expects to work with the PNM community to build into ProOps much of what will be documented in the forthcoming DOCSIS® 3.1 PNM Best Practices document.

ProOps is well-suited for PNM, but it can be used for turning any data into action, really. With CCF being fully flexible to gather most any network or system data source through creation of a driver, ProOps likewise can collect that data, analyze it, add context, translate results into potential work, then select the work that is most important to do. Network operations efficiency and improving service reliability are the intended goals of ProOps, but only our imaginations will limit what it can do.

We hope for, and fully support, operators and vendors contributing code to the C3 repository for sharing solutions, guiding the industry to solve problems, and sharing ideas. CableLabs intends to build workers with new capabilities and share them in ProOps as workers with which members can build solutions, and for CableLabs to build other proof-of-concept applications. In the months ahead, the power of ProOps will increase due to the contributions from the entire community. We hope capable operators and interested vendors will work with us to develop workers and applications that solve important PNM needs so that both operators and vendors may take advantage from the PNM capabilities that CableLabs provides.

## **Appendix I      Acknowledgements**

We wish to thank the following participants contributing directly to this document.

John Phillips

Jason Rupe

Jay Zhu

Also, we thank Christie Poland for the fine editing.

\* \* \*