SCTE CABLE-TEC EXPO 22
SEPTEMBER 19-22 • PHILADELPHIA, PA

2022 Fall Technical Forum
SCTE • CABLELABS • NCTA

Robust and Resilient Service Assurance System Design with Observability to Improve Enterprise Customer Experience

A Technical Paper prepared for SCTE by

Anil Mohan
Principal Engineer, Product Development Engineering
Comcast Cable
1800 Bishops Gate Blvd, Mt. Laurel, NJ 08054
anil_mohan@cable.comcast.com

Xin Huang
Sr. Principal Engineer, Product Development Engineering
Comcast Cable
1800 Bishops Gate Blvd, Mt. Laurel, NJ 08054
xin_huang@cable.comcast.com

Table of Contents

Title **Page Number**

List of Figures

Title **Page Number**

List of Tables

Title **Page Number**

1. Introduction

Service assurance (SA), as a subset of the operational support system (OSS), plays an important role in the internet service provider (ISP) ecosystem. However, the rapidly evolving internet service provider (ISP) technologies, enterprise services offerings, and customer expectations bring great challenges to the modern service assurance system design. This paper discusses several general design principles and best practices that are essential to building a robust and resilient service assurance system with observability and awareness that could stay ahead of these fast-paced industry transformations.

First, collecting telemetry from multiple sources helps to avoid single point of failure (SPOF) and improve confidence and accuracy of alerting customers of network/service issues. Second, introducing a unified mediation layer provides flexibility to isolate vendor-specific implementations and prevent bugs from negatively impacting customer experience. Third, making use of cross product correlation and leveraging machine learning (ML) for data analytics, trending and anomaly detections to prevent service interruptions and guarantee accurate customer alerting.

This paper reflects years of SDN-based centralized service assurance system integration design, development, and customer support experience. In this paper, the authors will share ways in which these principles and techniques are applied in our enterprise service product to support business values and keep our customers happy.

2. High-level Service Assurance System Design and Challenges

This section describes the initial service assurance system design and challenges and next section will describe how we resolved these challenges using the advanced service assurance architecture design. Figure 1 below shows the different layers and functional blocks of the initial service assurance architecture design with the key functional roles played by each layer. Figure 2 below shows the different challenges faced at the different layers in the service assurance architecture.

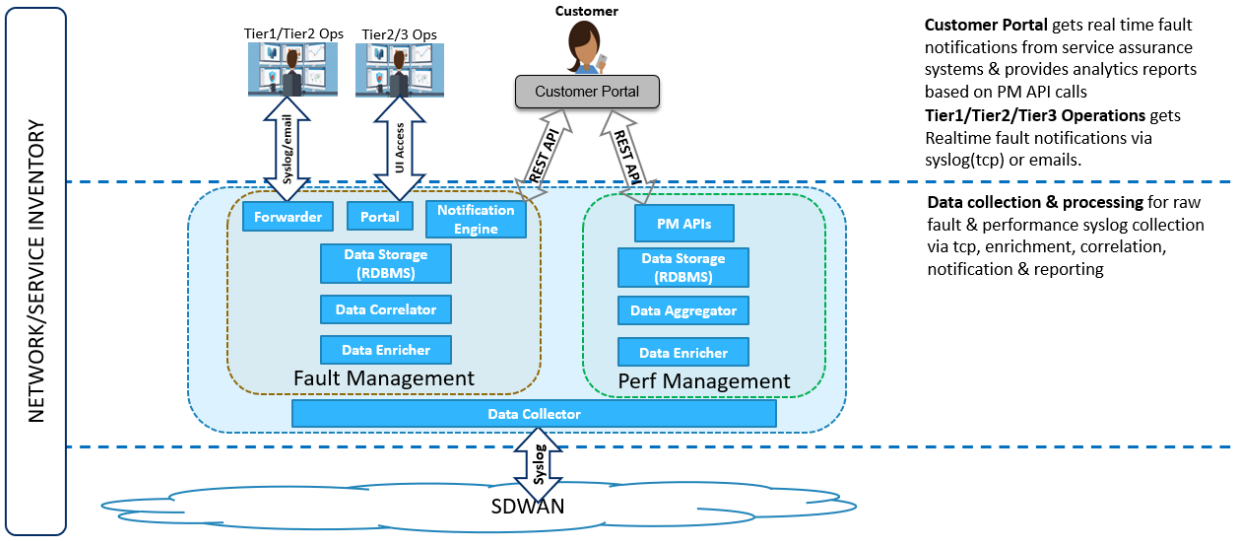


Figure 1 – Initial Service Assurance Architecture

As illustrated in Figure 2, the key challenges faced in the initial service assurance architecture design are:

- Out of sync device or port status between systems;

- High response time and timeouts for data presentations;
- Lack of mediation layer to standardize the multi-vendor log/event formats;
- Least scalable with fewer integration options;
- Lack of advanced correlation between faults and performance metrics to provide meaningful insights;
- Lack of visibility to different layers of network service; and
- Lack of advanced troubleshooting capabilities.

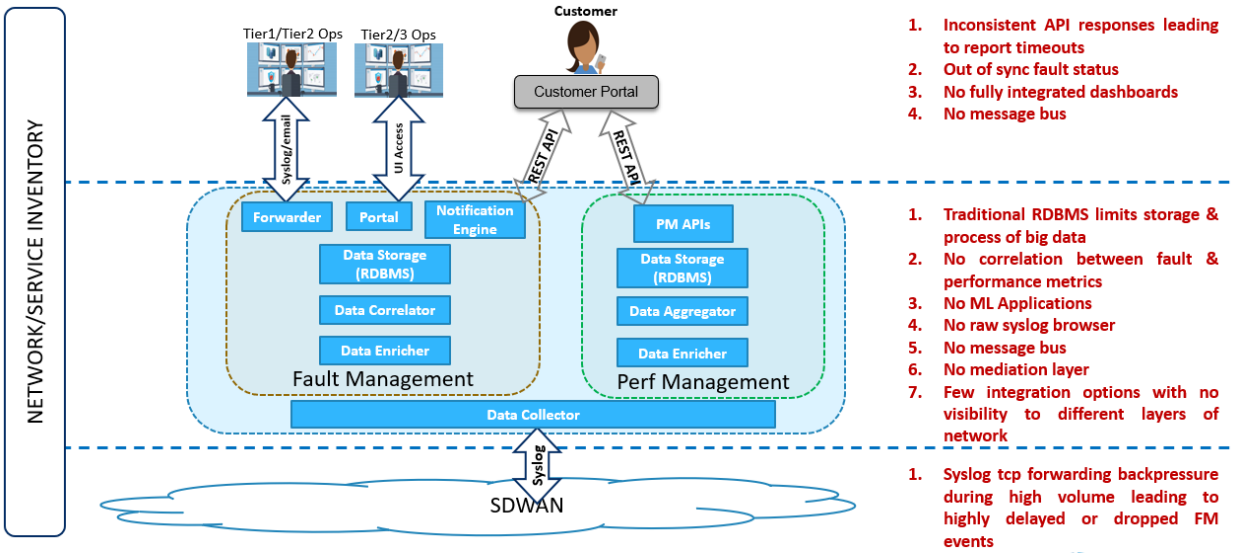


Figure 2 – Initial Service Assurance Architecture Challenges

2.1. Out of Sync Device or Port Status between Systems

With the initial service assurance system, there were some cases of device or port statuses going out of sync between systems. Customer experience may have been impacted by these out-of-sync issues.

Below listed some of the reasons for statuses to go out of sync:

1. When vendor element managers fail to send set or clear event due to bugs.
2. When set and clear events/messages come at the same time.
3. When there is no dedicated path for faults, fault event may be delayed or dropped due to congestion in the analytics data pipeline to service assurance systems.
4. When the vendor does not provide single fault event for a problem and the service assurance system does not have the capability to cross-validate against performance metrics.
5. When producing or consuming systems fail to process the fault events.

2.2. High Response Time and Timeouts for Data Presentations

The initial service assurance architecture design had the potential to cause delays and timeouts in loading top reports for customers.

Below are listed some of the reasons for such high response time or timeouts for the reports:

1. The granular customer data for the top reports were pulled for the selected time and aggregations were run on the fly at the portal reporting platform.

2. Every user click for a similar report triggers corresponding database pulls for the granular customer data. This causes a lot of unnecessary database transactions which is not a normalized approach.
3. The granular customer data was being pulled from non-big data storage systems which caused delays when the customer data were high.
4. The report data were tied with inventory cache where the inventory cache was not up to date all the time, leading to missing or inconsistent data.

2.3. Lack of Mediation Layer to Standardize the Multi-Vendor Log/Event Formats

In the initial service assurance architecture, there was no mediation layer to standardize, filter out, modify, or convert log/event formats to meet the customer needs. This led to higher time to roll out changes because of the need to accommodate log/event format changes.

Below listed some of the reasons for requiring a proper mediation layer in service assurance architecture:

1. Log/event format changes due to vendor upgrades or bug fixes.
2. Filter out log/event due to bugs in existing vendor versions
3. Standardize multiple logs/events into same format for better processing and reporting
4. Define higher priority pipelines for fault events compared to performance metrics
5. Enable tagging and distribution of logs/events

2.4. Least Scalable with Fewer Integration Options

Initial service assurance architecture had scalability limitations at all functional layers, including but not limited to data collection, data storage, data integration with external systems etc. This would become a bottleneck as the network grew and caused delays in processing incoming or outgoing messages leading to bad customer experiences.

Below are listed some of the reasons for requiring a more scalable and highly interfacing architecture:

1. Number of logs/events per customer device is not static but rather dynamic based on customer traffic. The service assurance systems should be scalable enough to accommodate changing volumes of customer device traffic and usage.
2. To achieve a better predictive and reliable service assurance system, there is always the need to integrate multiple datasets from different platforms. This requires multiple integrations / interfacing points.
3. For faster rollout of features, service assurance architecture should be able to consume / collect multiple data types using different protocols, and be able to produce / export data to different interfacing systems.

2.5. Lack of Advanced Correlation between Faults and Performance Metrics to Provide Meaningful Insights

Initial service assurance systems did not have the capability to correlate across fault and performance metrics. Correlating across fault gives meaningful insights into the data along with accuracy and better ML predictions. Alternate options within initial service assurance systems limited the full capabilities one gets from marrying the fault and performance metrics.

Below listed some of the benefits for correlating between faults and performance metrics:

1. For meaningful insights into customer data.

2. Good ML predictions.
3. Better accuracy of fault events.

2.6. Less Visibility to Different Layers of Network Service

The initial service assurance systems were not flexible enough to provide visibility to different layers of network service since there were minimum integration options and no correlation between fault and performance metrics. Visibility to different layers of network service can help in identifying the root cause of any issue, provide better representation of the service to operations or customer, and create a better fit for proactive monitoring of the network service.

Below listed some of the reasons for having better visibility to different layers of network service:

1. Better root cause analysis (RCA).
2. Better visualization of the service.
3. Better proactive monitoring of the service.

2.7. Lack of Advanced Troubleshooting Capabilities

The initial service assurance systems did not have good visualization capabilities to create on-the-fly dashboards on fault, performance metrics or both fault and performance metrics.

Below are listed some of the factors that impaired the troubleshooting capabilities:

1. Lack of visibility to different layers of network services.
2. Lack of correlation between fault and performance metrics.
3. Separate clients for viewing faults and performance metrics, rather than a single pane of glass view.

3. Advanced Service Assurance System Design with Observability and Awareness

This section describes the key enhancements / features of the advanced service assurance architecture that can overcome each of the challenges explained in Section 2 by implementing Architectural design changes or methodology changes.

Figure 3 and Figure 4 describe the high-level service assurance architecture providing observability and awareness.

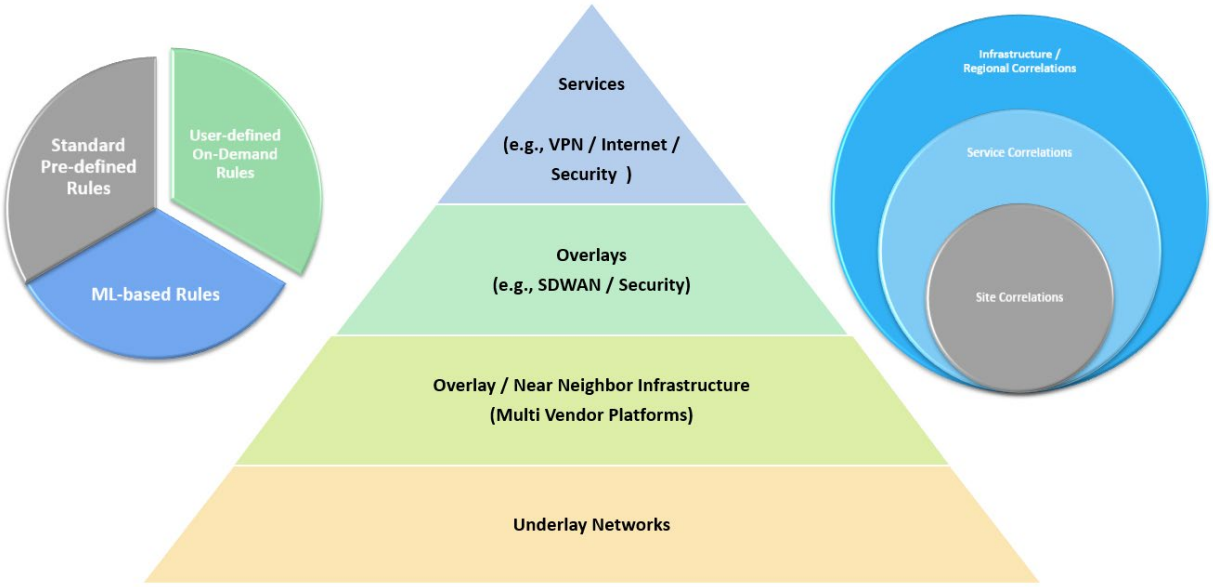


Figure 3 – Advanced Service Assurance Functional Blocks

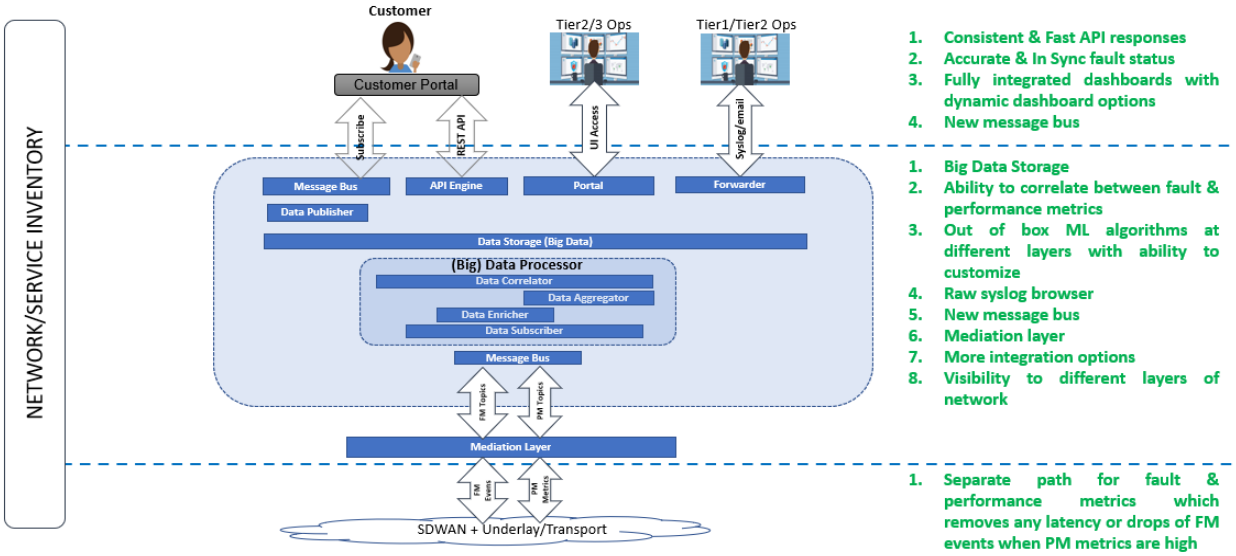


Figure 4 – Advanced Service Assurance Architecture

3.1. New Mediation Layer Added

A new mediation layer added between the network source and the service assurance collection layer as shown in Figure 4Figure 3 above. This layer adds the functionalities below to help in creating a more robust service assurance architecture by eliminating some of the challenges listed in section 2. For example:

1. Filtering out messages.
2. Format changes of messages.
3. Supports multiple input and output format for message transfers between systems.
4. Message tagging.
5. Distribution of messages to multiple destinations.

3.2. Separate Dedicated Path for Fault and Performance Events

As part of the advanced service assurance architecture, separate dedicated paths were added for fault and performance events from the source to mediation layer to data collectors in assurance systems. This avoids any latencies on fault events whenever the volume of performance events becomes too high and causes backpressure on those paths.

3.3. Additional Sources for Alarm Validations

As part of the advanced service assurance architecture, more sources were added to fault management to increase the confidence level of the alarm. The below sub-sections explain the 5 major additional sources added to improve the accuracy of device/port/path status. This helped to achieve more consistent and accurate operational statuses.

3.3.1. Polling the Device / NMS

Polling the device/NMS via different supported protocols (e.g., REST API, SNMP, ICMP, etc.) always helps to sync the right operational status within service assurance systems.

3.3.2. Adding Heartbeat Events from Device / NMS

Heartbeat events from device / NMS provide a recurring event on the health of the device every 'x' minutes. This helps to keep the service assurance systems in sync on the device health status. This also helps to minimize the dependency on polling since polling device / NMS sometimes can be expensive depending on the device / NMS providers.

3.3.3. Correlating Events/Alarms against Performance Metrics

Device / port / path level performance metrics that generated every 'x' mins from device / NMS can be considered as heartbeat messages. The presence of performance metrics can be considered as 'UP' heartbeat while absence of performance metrics from a previously present metric can be considered as 'DOWN' heartbeat. Correlating the events/alarms against these performance metrics will help in improving the confidence level of the corresponding event/alarm. This also helps in minimizing the dependency on polling. Below Figure 5 shows cross-correlation between alarms and performance metrics for finalizing device status.

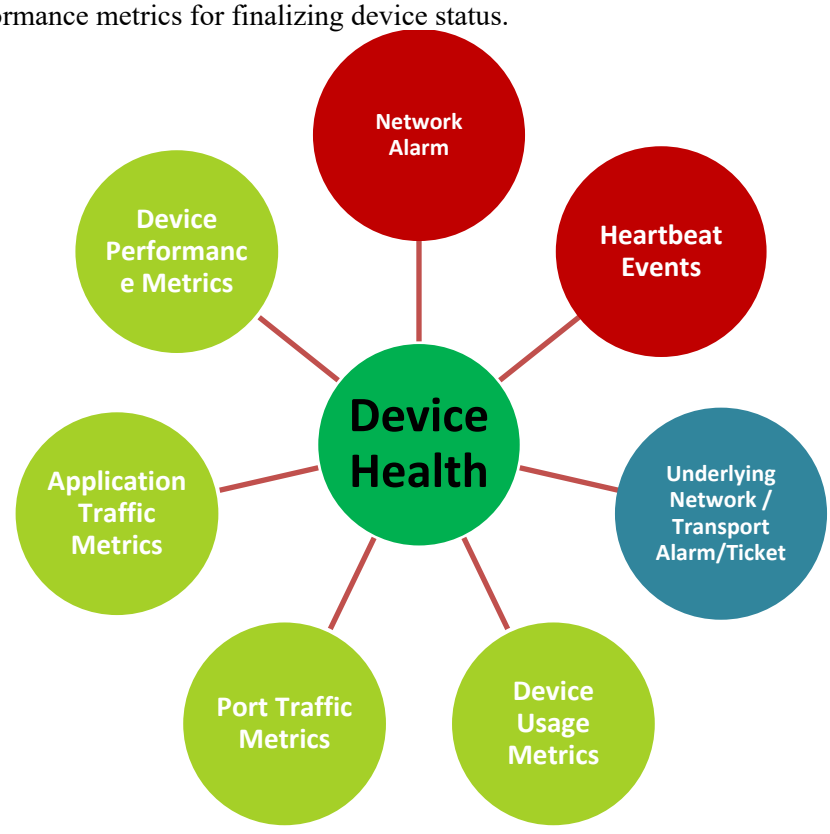


Figure 5 – Cross-Correlation between Alarms and Performance Metrics

3.3.4. Correlating against Underlying Network/Transport Layer Alarms / Tickets

Correlating fault events against underlying network / transport layer alarms / tickets always gives better insights into the root cause of the fault/event. This also aids in creating a visual representation of network service and giving the operations team advanced troubleshooting capabilities.

3.4. New Data Delivery Methodology for Faster and Consistent API Responses

In the initial service assurance architecture approach of providing APIs for performance metrics, we have seen multiple challenges leading to delays / timeouts in API responses. In this new advanced service assurance architecture, based on the learnings, we have come up with a new data delivery methodology for performance metrics in reports to customers. The new methodology is much faster and provides consistent API responses.

Below are the key steps for implementing this new data delivery methodology:

1. Identify the different report types to be delivered to customers.
2. Identify the different groupings needed for each of the report, like customer-level, device-level, port-level etc.
3. Identify the pre-defined time buckets with granularity needed.
4. For each type of the above report, pre-stage the data for the report by scheduling these granular data from the database at regular intervals based on the time buckets.
5. Create API request definitions which pick up the latest pre-staged data every time a user clicks a report.

The key benefits of this approach include:

1. No on-the-fly aggregations, leading to faster response times for loading reports.
2. Normalized approach with minimum database calls for reports. No unnecessary database transactions.
3. Removal of the dependency on the inventory cache, thereby resolving any missing or inconsistent data.

3.5. Hybrid Notification Approach

Table 1 - Hybrid Notification Approach (with Underlying Transport Issue Identified)

Category	Status	Realtime / Soaked	Source	Notification Severity	Notification Message
Device / Port / Path Health Notifications	Down	Realtime	Network Set Alarms	Major	Instead of reporting as down, should say something like "there seems to be an issue with the device / port / path. Hold on for further updates"
Underlying Transport Ticket Notifications for the Device / Port / Path Health notifications	Down	Soaked ('X' mins)	PM Metrics / Transport Alarms / Tickets	Critical ↑	Change severity to Critical and say the device / port / path as down.
Device / Port / Path Health Notifications	Up	Realtime	Network Clear Alarms	Major ↓	Change severity from Critical to Major, should say something like "issue seems to be resolved. Hold on for further updates"
Underlying Transport Ticket Resolution Notification for the Device / Port / Path Health notifications	Up	Soaked ('X' mins)	Transport Alarms / Tickets	Cleared ↓	Clear the status of device / port / path

All "Real time" and "Soaked" notifications will go to separate message bus topics. So, consumers can choose to subscribe to specific message topics for each notification type

3.6. New Message Bus Added

Based on the challenges observed in the initial service assurance architecture, message bus is a better fit in the advanced service assurance architecture. Message bus provides a highly reliable, scalable data collector as well as an asynchronous communication platform with decoupling for internal and external data movements. It comes with data persistence and fault tolerance that allows the service assurance system to continue processing data even when different parts of the system fail. Most of the message bus platforms available in market come with a lot of additional features like filtering, enrichment, correlation, and more, providing more intelligence at each layer of the architecture.

3.7. Data Storage Changed from Traditional RDBMS to Big Data system

There are 3 major challenges for traditional RDBMS Storage – data too large, data too complex (multiple data types), and data too fast. As the network grows, the analytics data also grows and as we add more features, there will always be new data types/formats to be processed which makes the analytics data complex with different types/formats. As data grows, data ingestion or retrieval times cannot be compromised as it will negatively affect the customer experience. With these challenges and the added benefits of big data systems, it was an easy decision to move away from the traditional RDBMS to big data.

3.8. New Raw Log/Event/Message Browser

Raw log / event / message browser is a good add-on to any service assurance architecture since it provides the user / operations the ability to skim through historical / real-time alarms / events / performance metrics for troubleshooting purposes / generating on the fly reports / statistics.

4. Improvements for Customers and Operation Teams Experiences

4.1. Out-of-Sync Device or Port Status between Systems Improvement

Figure 6 shows the percentage of occurrences of out of sync device / port alarm status before and after moving to advanced service assurance with the key enhancements or features described in Section 3.

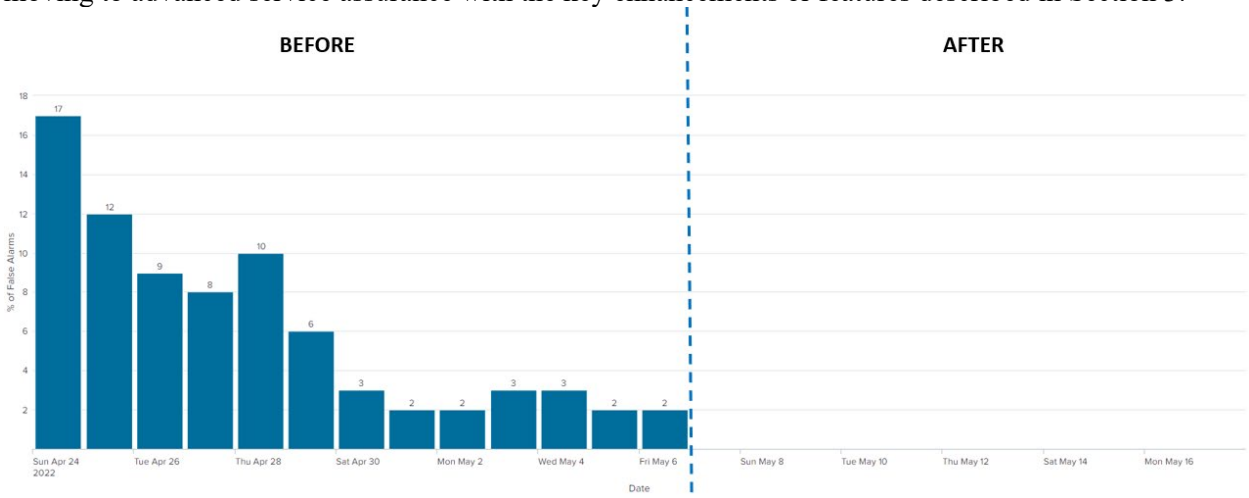


Figure 6 – Out of Sync Issue Before and After

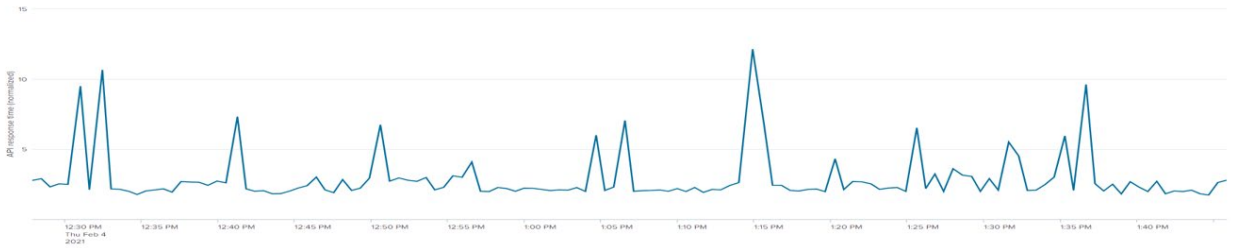
Figure 6 clearly shows the improvements in out of sync issues before (using initial service assurance systems) and after (using the advanced service assurance systems). Before the percentage of out-of-sync statuses varied from 2% - 17% while after there were no such out-of-sync occurrences observed based on 30-day data between April and May 2022

4.2. Data Presentations Response Time Improvement

Below graphs show the API response times at customer / single site / multiple site levels for analytics data consumed by systems external to service assurance systems before and after moving to Advanced Service Assurance with the key enhancements or features described in Section 3.

4.2.1. Customer Level:

Response Times (BEFORE):



Response Times (AFTER):

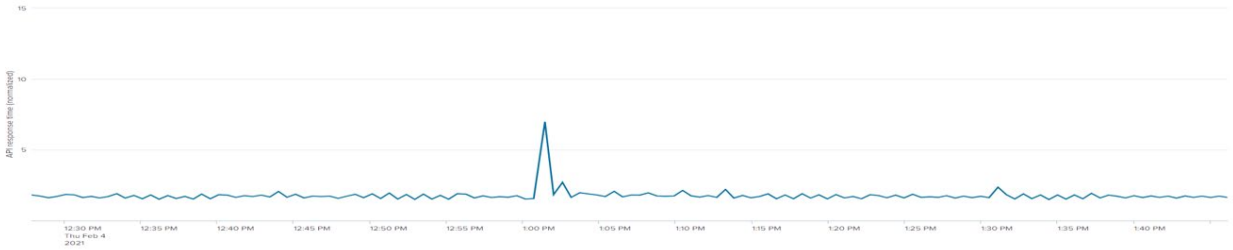
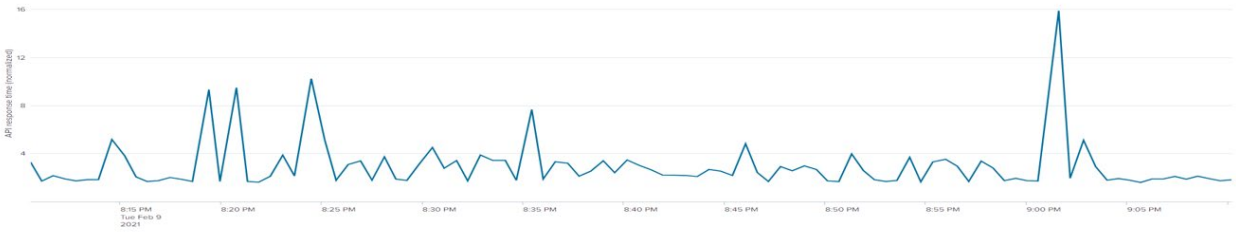


Figure 7 – API Response Times for Customer Level Before and After

4.2.2. Site Level:

Response Times (BEFORE):



Response Times (AFTER):

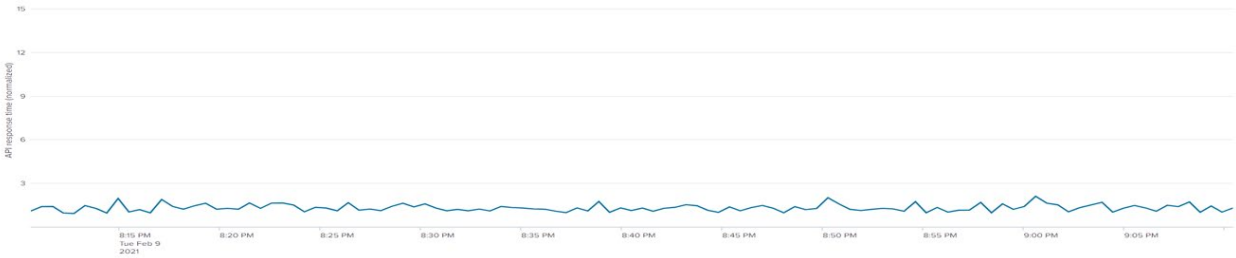


Figure 8 – API Response Times for Site Level Before and After

4.2.3. Multiple Sites Level:

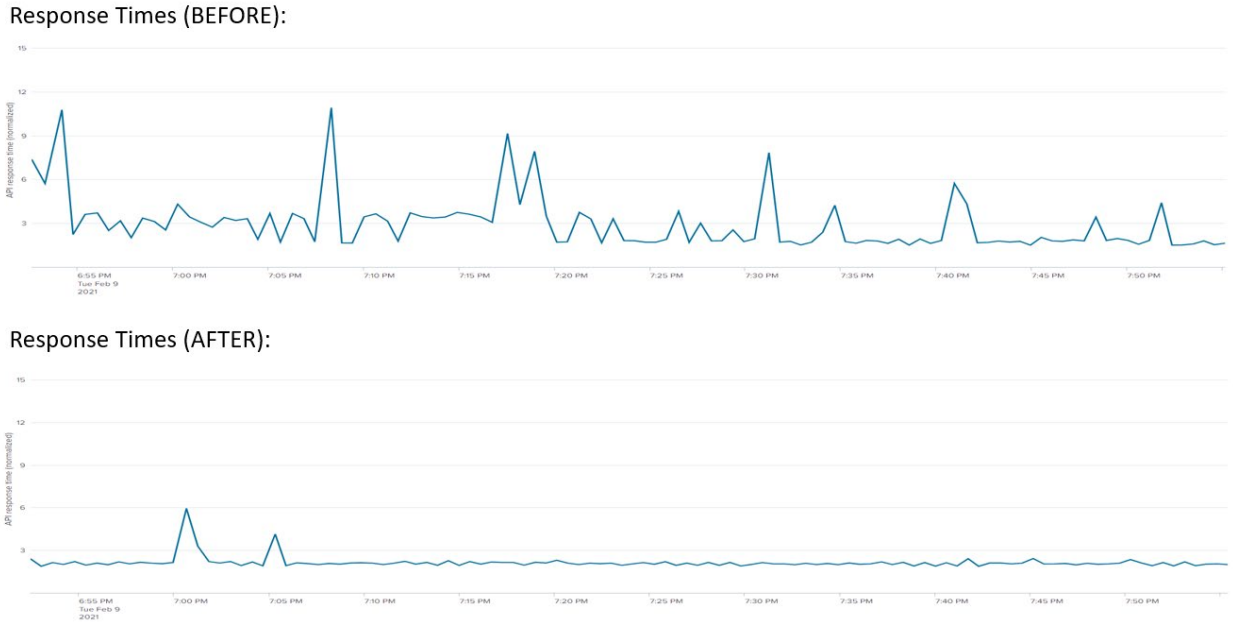


Figure 9 – API Response Times for Multiple Sites Level Before and After

Figure 7, Figure 8, and Figure 9 clearly show the improvements in API response times at customer / single site / multiple site level for analytics data consumed by systems external to service assurance systems. Below are the key highlights:

- API response times are consistent and lower.
- More aggregation types are supported.
- More flexible filtering with ease-of-use APIs

4.3. Mediation Layer Improvements

The below graph shows the percentage of occurrences of alarms with missing device names for fault events before and after moving to advanced service assurance with the key enhancements or features described in the previous section.

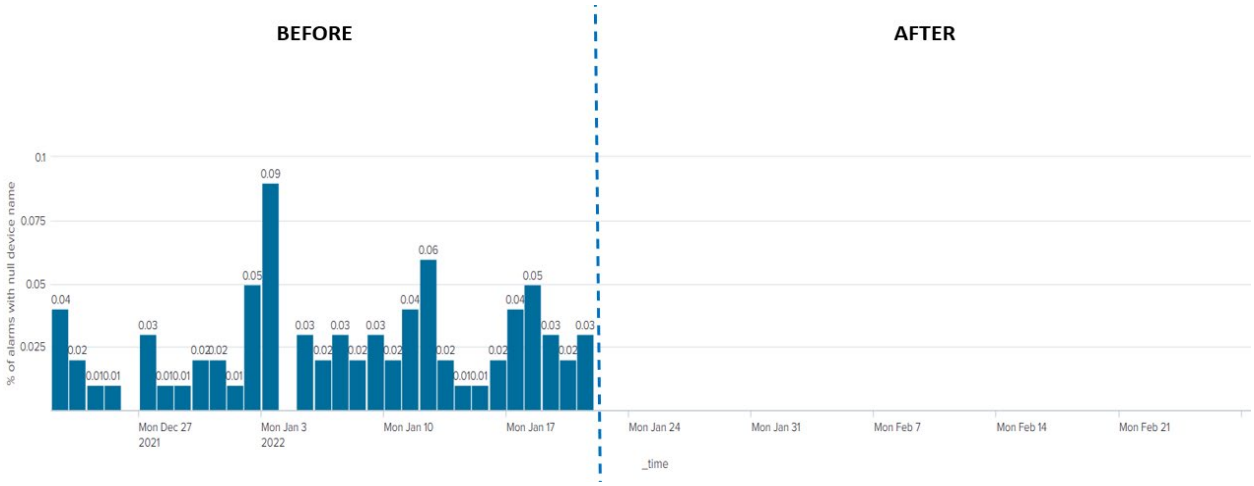


Figure 10 – Null Device Name Issues Before and After

Figure 10 clearly shows the improvements in occurrences of alarms with missing device names for fault events before (using initial service assurance systems) and after (using the advanced service assurance systems). Before the percentage of such missing device names in fault events was very low, varying from 0.01% - 0.09%, while after there were no such fault events with missing device name occurrences based on data collected between Dec 2021 and Feb 2022.

5. Conclusion and Future Work

The initial service assurance architecture design, which faced multiple technical challenges, was changed to include a mediation layer, separate dedicated FM and PM paths, and additional sources of alarm validations, along with a new data delivery methodology for faster and consistent API responses, etc.

This re-architecture has shown a lot of improvements. For example:

- Minimum out of sync issues (reduced from around 15% errors to 0% based on 30-day data between April and May 2022).
- Faster API response times and no time outs for API responses (80% reduction).
- Better integration capabilities.

This new robust and resilient service assurance architecture with observability and awareness has enabled to add more advanced correlations with capability to use ML algorithms to build data models for better prediction, trending, and forecasting.

Abbreviations

API	application programming interface
FM	fault management
ICMP	Internet Control Message Protocol
ISP	internet service provider
ML	machine learning

NMS	network management systems
OSS	operational support system
PM	performance management
RCA	root cause analysis
RDBMS	relational database management system
REST	representational state transfer
SA	service assurance
SCTE	Society of Cable Telecommunications Engineers
SDN	software-defined networks
SDWAN	software defined wide area network
SNMP	Simple Network Management Protocol
SPOF	single point of failure
Syslog	System Logging Protocol
TCP	Transmission Control Protocol
UI	user interface
VPN	virtual private network

Bibliography & References

M. Casado, N. McKeown, and S. Shenker, “From ethane to SDN and beyond”, in ACM SIGCOMM Computer Communication Review, vol. 49, issue 5, Oct. 2019, pp 92-95.

L. L. Peterson, C. Cascone, and B.S. Davie, “Software-Defined Networks: A System Approach”; System Approach, LLC.

J. Halvorsen, J. Waite and A. Hahn, "Evaluating the Observability of Network Security Monitoring Strategies With TOMATO," in IEEE Access, vol. 7, pp. 108304-108315, 2019, doi: 10.1109/ACCESS.2019.2933415.

Deep Learning: A Visual Approach, Andrew Glassner; No Starch Press

G. Xu, Y. Cao, Y. Ren, X. Li and Z. Feng, "Network Security Situation Awareness Based on Semantic Ontology and User-Defined Rules for Internet of Things," in IEEE Access, vol. 5, pp. 21046-21056, 2017, doi: 10.1109/ACCESS.2017.2734681.

S. Chickerur, A. Goudar and A. Kinnerkar, "Comparison of Relational Database with Document-Oriented Database (MongoDB) for Big Data Applications," 2015 8th International Conference on Advanced Software Engineering & Its Applications (ASEA), 2015, pp. 41-47, doi: 10.1109/ASEA.2015.19.

N. Kumar, A. Leventer and A. Matatyaou, "Monitoring and Troubleshooting at Scale with Advanced Analytics", SCTE Cable-Tec Expo 2021.

