Executive summary

Network Operations teams need a network automation solution suited for the day-to-day challenges of operating today’s complex and rapidly evolving networks. Key to this challenge is the need to reduce MTTR. This change from traditional paradigms involves both a mentality as well as a technology shift. This whitepaper outlines a practical, scalable approach to NetOps automation for the purpose of continuous MTTR reduction.
Table of Contents

INTRODUCTION ........................................................................................................................................... 2

THE TRANSFORMATION OF NETOPS .............................................................................................................. 2
  Keeping the Lights On ......................................................................................................................................... 2
  The Legacy NetOps Model: A Human-Centric Approach .................................................................................. 3
  The NetOps Transformation Model: Built for Automation ................................................................................ 3
  MTTR Reduction: The Ultimate Value of NetOps Automation ......................................................................... 4

THE CHALLENGE TO REDUCE MTTR .............................................................................................................. 4
  A Review of MTTR ........................................................................................................................................... 4
  Traditional Approaches to Reduce MTTR ......................................................................................................... 6
  An Incident Response Framework .................................................................................................................. 7

A NETWORK AUTOMATION ARCHITECTURE TO CONTINUOUSLY REDUCE MTTR ................................. 9
  Triggered Automation: Automate First Response ............................................................................................ 10
  Interactive Automation: Augment Engineering Talent .................................................................................... 11
  Proactive Automation: Do Better Next Time ................................................................................................... 12

LIFECYCLE OF AN INCIDENT: USE CASE OF NETOPS AUTOMATION ..................................................... 14
  A NetOps Automation Implementation ........................................................................................................... 15
  A Walkthrough of an Incident Response Case Study ....................................................................................... 15
  The NetOps Automation Maturity Model ........................................................................................................ 19

CONCLUSION .................................................................................................................................................. 19
Introduction

Today’s enterprises are undergoing rapid digital transformations as business leaders are asking that the network become more agile, flexible, and secure. The modern network is increasingly more distributed, with more services, connected devices, and complexity, driving fundamental changes across IT. In parallel, the value of the network to the business is growing, as well as the pressure to maintain and secure it.

When there is a problem within the IT infrastructure, or a critical service is down or impacted, the cost can be astounding. A Gartner study estimates the average cost of network downtime to be $5,600 per minute, extrapolated to over $300,000 per hour. At the high end this number may double. Despite the enormous expense of downtime, IT organizations have struggled to find reliable methods to reduce this cost. The need is urgent and the challenge seemingly insurmountable.

This whitepaper will review the obstacles in reducing incident response times and their associated operational costs in this era of rapid change. Read further to learn:

- How the hybrid of physical, virtual, software-defined, and public cloud networks has made IT operations significantly more challenging
- How network automation can help enterprises achieve continuous MTTR reduction
- Why “proactive automation” has a much higher ROI than “reactive automation”
- How “shifting workloads to the left” is critical to saving costs and reducing MTTR
- A sample implementation of the proposed architecture

The Transformation of NetOps

Imagine a world where network incidents are detected, diagnosed, and root cause is determined, in a fraction of the time it takes with manual efforts? In such a world, an automated network management fabric would help to isolate the cause, enable a fix to be safely and quickly pushed out, and verify that the problem has cleared. The system to achieve this would have post-incident tuning capabilities so that resolution can occur even quicker next time. Operations teams, already under increasingly overwhelming workloads, would no longer exist day to day just fighting fires.

Is such a concept feasible? Is it possible to implement a scalable network troubleshooting automation framework that can trend MTTR downwards over time?

Keeping the Lights On

NetOps teams provide two functions – to deliver on services required by the business, and to ensure downtime is minimum.

The first is dominated by projects such as new data centers, public cloud migration, or implementing QoS for a new voice and video service. The second is arguably the most critical in terms of impact to a company’s revenue and reputation.
Ensuring minimal downtime can be further split into two categories –

1. Prevent outages from happening
2. Resolve outages as soon as possible

These two categories have spawned two KPIs: Mean Time Between Failures (MTBF) and Mean Time to Repair (MTTR).

Top of mind for technology leaders is the need to reduce network downtime and improve service reliability, while at the same time reducing the operational cost for the organization. While NetOps has many duties, keeping the lights on is priority number one.

The Legacy NetOps Model: A Human-Centric Approach

According to Cisco’s 2019 Global Networking Trends Survey, 33% of respondents said Identifying and troubleshooting network issues is the most time-consuming task. A close second place at 30% of respondents is Detecting and remediating security threats or attacks on the network.

Despite this, rigid processes, manual data collection mechanics, inefficient collaboration, and a lack of knowledge management dominate the operational paradigms used by today’s NetOps teams.

Further exacerbating these shortcomings is a reluctance to implement automation within day-to-day network operations.

Manual diagnostic activities performed by Tier-1 or service desk teams are inefficient and tedious. Escalations to senior engineers are frequent and cumbersome and result in inconsistent outcomes. Technical know-how is tribal, and incident post-mortem lessons learned are often not operationalized.

The need is clear – NetOps must consider a new model.

The NetOps Transformation Model: Built for Automation

A NetOps transformational journey requires new methodologies and processes to accommodate the global shift to digital technologies. Network operations teams can no longer efficiently manage the network with tactical approaches using legacy mechanisms to build, operate, and troubleshoot. New, strategic approaches are required as the line between NetOps and DevOps blurs.

This new model requires closed-loop mechanisms to achieve continuous improvement and self-documenting workflow automation. This is a shift to a business-centric and intent-based mindset that is automation-friendly, analytical, and most importantly, proactive. Network diagnostic work would move from sequential, CLI-focused methods to multi-threaded, integrated automation.
NetOps transformation is therefore both a mentality shift as well as a technology shift. Gartner refers to the need to move NetOps into the new paradigm as “NetOps 2.0”, with a key mantra of “automation by default, manual by exception”.

**MTTR Reduction: The Ultimate Value of NetOps Automation**

Historically, automation within NetOps has largely fallen in administrative tasks or failure prevention monitoring such as redundancy verification, device hardening verifications, or compliance audits. These automations mostly serve to support MTBF and to prevent the inherent risks within networks that can cause outages. Typically, the network engineering and architecture teams are the main users of this use case, where their jobs are to deliver new services and deliver redundancy and reduce inherent risks.

Reducing MTTR arguably has an equal if not greater impact on the overall target of reducing downtime. However, thus far the industry has not provided sufficient automation solutions to enable infrastructure teams to become more efficient in this role.

Compounding this is the added complexity of new technologies, the sheer volume of devices, and fragmentation of subject matter expertise, all leading to longer troubleshooting times. The gap in the industry is an automation solution that augments Network Operations and improves MTTR.

**The Challenge to Reduce MTTR**

For decades, the IT industry has attempted to solve the MTTR dilemma. These approaches have focused on partial solutions and have not solved the problem from a holistic perspective.

Network operations and incident response is not a technology, design, or protocol that can be readily manipulated or orchestrated. Rather, it is a process and workflow that is not easily automated. We must also consider today’s evolving network landscape and the difficulties it brings in applying practical, scalable automation for network operations.

**A Review of MTTR**

MTTR is defined by three operational phases: Detect, Identify, and Fix. Each phase poses its own challenges. Understanding how to apply automation to incident response requires a thorough analysis of MTTR and the incident response workflow.
Detect

Fault detection is handled by existing monitoring and event management solutions, which are commonly deployed in enterprise environments. While increased device telemetry has produced considerably more noise and false positives, today’s event correlation solutions and SIEM products have been developed to help reduce this flood.

The delay in MTTR is therefore usually not within this phase itself but instead occurs during the transition from the Detect phase to the Identify phase. This is commonly a cause of initial delays in incident resolution.

Additionally, the information coming from the monitoring systems generally lacks details, often providing little actionable intelligence.

Identify

Large potential delays and unreliable variability exist in the MTTR Identify phase. This is the elephant in the room and the problem which requires the most effort to resolve.

Unpredictable, this phase has the largest impact on the cost of an outage. Without a means to methodically tackle this variability, we cannot measurably improve the biggest portion of MTTR. Hence, the biggest reduction in MTTR will come from Mean Time to Identify (MTTI). The automation strategy must enable teams to obtain and analyze data faster, to isolate root cause.

Fix

While the fix phase can be very brief, efforts to reduce the inherent risk of pushing a change and to integrate this phase into a full incident response workflow is desired.
Proactive

We must also consider a theoretical fourth phase of MTTR – the **Proactive Phase**. The incident is resolved, but what if a similar event recurs later? Can we record the lessons learned and make them executable to do better next time?

In the below illustration of the typical NetOps workflow for incident response, the challenges of this process are highlighted. Important to our discussion here is the post-mortem tasks that must be part of the MTTR workflow.

### Traditional Approaches to Reduce MTTR

The industry has offered many solutions in attempts to drive down MTTR. Many of the solutions focus on reducing Mean Time Between Failures (MTBF), and most only provide meager benefits for MTTR.

Today within the incident response process, the lack of automation manifests in unpredictable, risky, and manual network operations. This results in longer MTTR times and higher costs for the organization.

The below diagram illustrates today’s levels of automation relative to desired levels of automation.
Traditional approaches focus on improving monitoring, data correlation, documentation, change management, and even incident response processes. All are desirable and should be implemented.

The takeaway is that none of these approaches do much to address the biggest time gaps in the MTTR process. A solution that scales diagnostic troubleshooting and incorporates workflow integration and targets the biggest MTTR time gaps is necessary.

An Incident Response Framework

MTTR represents a series of connected but discrete operations merged into a single metric. Understanding how to improve this metric requires that we analyze MTTR from the perspective of an IT organization’s incident response workflow.

MTTR reduction requires a systematic approach to address discrete areas of improvement within each step of the MTTR process flow. The solution must weave together the MTTR pieces in a manner that increases speed, minimizes risk, and codifies lessons learned for similar events in the future.

To visualize how the MTTR workflow applies to an incident response methodology, we can overlay the MTTR process on top of a typical network operations workflow.
An analysis of MTTR from this view reveals the operational areas needing improvement.

1. **Fault Detected.** To understand this stage, let’s look at what is happening. First, a network monitoring tool will detect a fault and some automated event correlation may occur. Next, the monitoring system may automatically generate an incident ticket. Now, an investigation must begin to determine root cause. While fault detection is largely automated, this transition from detection to the action of investigating is typically not automated and is a cause of delay.

2. **Idle Time.** This is the wait period, after an event is ongoing but before an incident response investigation has begun. A ticket may sit idly for an hour or more, while potentially critical diagnostic information vanishes.

3. **First Response.** This is often the most time-consuming stage and therefore where MTTR can be reduced the furthest. Critical here is having the right data and the right know-how. Hugely variable, this stage can potentially take several hours.

4. **Escalation.** When the first engineer is unable to resolve the issue, escalation is needed. The common flaw at this step is duplication of effort. The escalation engineer will inevitably end up repeating the work of the first engineer before moving on to more advanced diagnostic.

5. **Remediation.** This is the fix. The goal here is to ensure that we push safe changes, that we do no additional harm, and that we verify that the fix was successful. Automation is the safest way to push out such changes during the high stress of an incident response.

6. **Post-mortem.** This is an often neglected or poorly implemented stage of incident response. Implementing lessons learned so that we can be better next time is critical and yet exceedingly
challenging to enact successfully. A well-designed knowledge management system is required. To improve MTTR continuously, this is the most critical point to understand and address.

Hence, MTTR = \( \sum (1 + 2 + 3 + 4 + 5) \). Stage 6 is crucial for improving resolution times of future incidents.

A Network Automation Architecture to Continuously Reduce MTTR

Traditionally, the movement between MTTR stages, as well as the diagnostic tasks during the investigation, are manual. MTTR reduction therefore has a large dependency on people.

Improving MTTR without automation needs either more people or a better network, both of which are difficult to achieve.

A scalable approach is through better automation across each step of the MTTR workflow.

Put simply, MTTR reduction can be achieved through two methods:

- **Increase automation** at every stage of the MTTR process
- **Develop proactive automation** at the post-mortem stage, following every incident

NetOps automation can be categorized into these three categories:

- Triggered Automation – occurring the moment an incident is detected
- Interactive Automation – to assist network engineers in their diagnosis
- Proactive Automation – to make incident response more effective in the future

Each category will have its own mechanisms to drive incident response and improve MTTR.

Figure 5 - Categories of NetOps Automation
Triggered Automation: Automate First Response

Core to the philosophy of NetOps automation is that no person should touch a trouble ticket before automation has done its work. When a fault occurs within the network, the first challenge is the resulting Idle Time.

The ticket sits unworked and potential diagnostic data may even clear before an investigation can begin. Software must augment this process and initiate the diagnosis of the event. Triggered automation closes the gap between the detection of the fault and the action of investigating.

For triggered automation to be successful, full workflow integration is required. The event detection system or ITSM must communicate with the NetOps automation system to automate a diagnosis.

We also have an important concept at this stage -- the known-problem library. As part of the initial triage of an incident, an engineer will want to know, “is this a known issue”? Known issues have known causes and known solutions. Once an incident is resolved and the solution determined, that solution becomes part of the known-problem library.

Hence, with each new incident there is a need to execute a known problem diagnosis. As the known-problem library grows and their associated operational runbooks are absorbed by the automation framework, the automation system gains the ability to quickly parse through all historical known issues to determine (1) is there already a known solution for this new incident, or (2) to rule out the list of known issues to reduce troubleshooting time.

The full triggered automation systems architecture would look something like this:

![Figure 6 - Triggered Automation Architecture](image)

Automation can augment the Detect phase in three ways: 1. Automatically gather additional telemetry to help in problem classification and diagnosis, and 2. Reduce transition delays between the Detect and Identify phases, and 3. Execute diagnosis of known problems.
Interactive Automation: Augment Engineering Talent

Automation is designed to augment people. Rather than sequentially parsing through CLI outputs of network equipment, the engineer would leverage pre-built operational runbooks and retrieve contextual diagnostic data at the click of a button. This helps to provide repeatable, predictable outcomes, ensures that the relevant data is accurately retrieved, and greatly reduces the time taken for the diagnostic process.

Scalable Diagnostics with Interactive Automation

Once the First Engineer responds to the incident and begins the initial triage and investigation, the priority will be to obtain the right data quickly and to perform accurate, efficient analysis.

The method typically involves manual command line digging. The goal is to accelerate this diagnosis using automation. Knowing what data to get, retrieving it rapidly, and leveraging expert know-how to analyze this data is required.

Automation should also provide:

1. Data analytics to enable historic data comparisons to know, “what has changed”.
2. Baseline analysis to understand, “is this normal”. Beyond traditional device configuration baselines, this also includes telemetry baselines and network and operating state baselines.
3. Context-aware knowledge library (know-how).
4. Executable knowledge (operational runbooks).

These functions enable an engineer to obtain the correct data and perform the analysis much faster.

Collaboration with Interactive Automation

Some issues can be resolved by the first level of support. Many will require escalation. Collaboration often fails during incident response, with data not properly conveyed to the next engineer or diagnostics not captured or saved.

The escalation engineer typically will duplicate the work of the first engineer before moving on to more advanced diagnosis. The network automation solution should record the collected diagnostics and troubleshooting notes and enable everyone to work on the same problem with the same data.

Collaboration often fails during incident response, with data not properly conveyed or diagnostics not captured or saved.
Safely Fix and Verify with Interactive Automation

When it comes to the fix, the goal is to safely push out the change and verify that the fix did indeed resolve the issue. A well-designed change automation system is needed to ensure that this is successful and that the solution automates the full mitigation sequence. This includes the change deployment, the before and after quality assurance, and the validation that the problem has cleared.

Hence, the automation solution needs to ensure that mitigation can be safely executed, assurance is provided that no additional harm was done, and that reliable post-fix verification is performed.

Proactive Automation: Do Better Next Time

To see continual improvement year-after-year requires that more and more issues be near-instantly diagnosed with their root cause identified. In other words, the automation strategy should focus on moving increasingly more issues to near-zero time to resolution.

As more issues occur with proper post-mortem reviews, a NetOps team would classify those issue types into a “known problem” category, adding each to the previously described known-problem library leveraged by Triggered Automation.

As more and more of these operational runbooks are fed to the machine, more and more known problems can have their diagnosis fully automated. This process continuously pushes MTTR lower.

A NetOps automation strategy must improve knowledge management and replace the “hero culture” and tribal knowledge concepts that are so prevalent within engineering teams. With proactive automation, we convert lessons learned into repeatable and executable diagnostic automation tasks. More than just documenting that lesson, the goal is to implement an automated diagnostic that checks for this problem next time there is an incident.

To achieve these proactive automation goals, the automation platform must provide:

- Knowledge – Executable and Reusable
- Drive better known-problem diagnosis
- Self-documenting workflow

Shift Knowledge to the Left

When designing a knowledge management framework and network automation strategy, the objective is to enable junior engineers to leverage the organization’s senior level expertise.

From the view of an escalation chain, the goal will be to shift knowledge from senior staff, logically residing on the right side of the operational flow, towards the first responders on the left side of the workflow.
flow. This effectively “shifts knowledge to the left”. Along with this knowledge movement, the associated workloads also shift with the benefit of reducing operational costs.

This downstream flow of knowledge enables the diagnostic work previously performed by a Tier-1 engineer to be handled by the machine. Advanced work previously performed by escalation engineers can now be handled by the Tier-1 team.

By shifting knowledge to the left, we achieve the following key benefits.

✓ Reduced ticket escalations
✓ Expanded team knowledge
✓ Reduced operational costs
✓ Reduced MTTR

Build NetOps Runbook Automation

There are two critical times when knowledge should be fed back into the automation platform:

Operational Handoff – Once your team has implemented a new network design (e.g. MPLS), does everyone on the team know how to troubleshoot the new environment? Required is a reliable, consistent, easy to follow method for documenting operational procedures related to new designs or new technology. The building of an operational runbook for the new design is considered part of the necessary handoff from the architect to the operator.

Following an Incident – After an incident is resolved, the team will typically get together for a post-mortem review. The goal is, to do better next time. But does that happen? Are the lessons learned actionable and executable and will they truly be utilized during a recurrence of a similar incident? This is necessary to reduce MTTR.

The described process creates a closed loop mechanism for continual improvement, capturing knowledge at these two critical and common moments. In combining knowledge management with runbook automation, we can achieve continuous MTTR reduction. This feedback mechanism defines the concept of Proactive Automation.

Build Known-Problem library

An important value of the building of NetOps runbooks after each incident is the construction of the known-problem library. As mentioned earlier in this paper, this becomes the database of known issues with their associated solutions. As this library grows, so does the ability to quickly isolate root cause for known problems, or alternatively, rule out a swath of potential causes.

For each new incident, the proposed automation framework must be able to quickly parse through all known problems in the library and run associated diagnoses. As new problems are found, the framework must accommodate adding the new operational runbooks to the known-problem library.

Automating this known problem diagnosis has a considerable positive impact on MTTR and is one of the cornerstones of both the Triggered Automation and the Proactive Automation mechanisms.
Continuous and Real-time MTTR Measurement

Success is validated by measurement. Network automation can assist with this validation. In using automation to augment incident response and MTTR reduction, we achieve a flurry of useful metrics – the amount of time spent at each stage, delays in transition, and so on. A side effect of this automation and associated metrics is that reporting can now be provided to track and understand how incident response is handled from an operational perspective.

The automation framework should enable such reporting and the measurement of MTTR. In making available these metrics, we enhance the continual tuning of the proactive feedback mechanism and know where to further tune our automation system and processes to make improvements.

Lifecycle of an Incident: Use Case of NetOps Automation

Reducing MTTR ultimately relies on the transformation of traditional NetOps into a new model of NetOps/DevOps hybrid, with the support of a modern automation platform and a knowledge management framework.

To solidify the points made in this paper, we’ll review a case-study of a hypothetical implementation of a network automation deployment for MTTR reduction. The goal of such an implementation should be to increasingly augment an engineering team with automation.

The diagram below summarizes this full incident response framework with the applied automations for each stage.

![Figure 7 – An Incident Response Automation Framework with Dynamic Map and Executable Runbooks](image-url)
A Network Automation Architecture to Continuously Reduce MTTR

A NetOps Automation Implementation

For our use-case review, we’ll use an industry-leading solution for NetOps automation and MTTR reduction, the NetBrain Automation platform.

NetBrain’s automation approach utilizes two core automation technologies – Dynamic Map and Executable Runbook, powered by an underlying model of the network.

To build the model, NetBrain performs a deep discovery of the network’s control plane logic which serves as the foundation for NetBrain Automation. A patented neighbor-walking algorithm leverages CLI automation, SNMP, and APIs to decode thousands of data variables per device, to create a “digital twin” of the network.

This discovery process populates the automation database, enabling the visualization of this data with NetBrain’s Dynamic Map and providing repeatable automation capabilities with NetBrain’s Executable Runbook technology.

This automation implementation provides a comprehensive NetOps platform for incident response and knowledge management and offers the following advantages:

- **Dynamic network mapping** for visualization and as the user interface for automation
- **Runbook Automation** for rapid analysis
- **Tool ecosystem integration**, for end-to-end analysis on one page
- **Centralized elastic knowledge base** for codified know-how
- **Event-Triggered Automation**, for instant diagnostic and mapping automation of the problem
- **Management network abstraction** with the creation of the data model and a conceptual management network “fabric”

A Walkthrough of an Incident Response Case Study

With NetOps Automation, there are generally two types of users: **consumers** of executable knowledge and **creators** of executable knowledge. To understand how this solves the challenges presented in this whitepaper, let us examine this using an example network incident.

**With NetOps Automation, there are generally two types of users:** **consumers** of executable knowledge and **creators** of executable knowledge.

Case Study of Triggered Automation

In our example enterprise, the monitoring infrastructure has detected a poor video quality issue between the Boston and New York site locations.

The network team’s application performance monitor notifies their ServiceNow ITSM and generates a new trouble ticket. Here, workflow integration comes into play.
To simplify the integration, NetBrain has developed a ServiceNow-certified application available on the ServiceNow App Store. Using the ServiceNow GUI, the NetBrain administrators in this enterprise have previously self-configured this plug-and-play API, enabling rapid integration with their ServiceNow ITSM.

This ITSM integration enables (1) the creation of a contextual NetBrain Dynamic Map of the problem area at the time of ticket creation, and (2) enriches the trouble ticket with diagnostic data obtained at the time of the event – *Just in Time Automation.*

In our example video quality incident, the Dynamic Map is used to visualize any relevant data about the network – topology data, configuration and design data, baseline data across thousands of data points, and even data from integrated third-party solutions.

During incident response, this map provides instant visualizations of the problem area.
Triggered Automation has now occurred, and valuable data has been gathered at the instant of the event start using the Executable Runbook. Below, we see that our first response engineer has reviewed these automated diagnostics. The data retrieved includes basic device health, QoS parameters, access-control lists, and other relevant collected logs.

Figure 10 - Triggered Automation Diagnostics

What was a manual effort is now a Zero-Human-Touch mechanism. This ensures that every ticket is automatically enriched with a contextual map and diagnostic data.

Case Study of Interactive Automation

Continuing with our example poor video quality issue, our engineer has reviewed the map of the problem and the collected diagnostics, but still needs to drill down further to determine root cause. For this, the scalability of the automation platform is critical. Additional diagnostics or more advanced design reviews may be needed to determine root cause.

Our engineer now leverages the automated drill down capabilities of the NetBrain Automation platform to do further analysis, historical comparisons, and to compare this data with previous network baselines composed of over 10,000 discoverable network data points.
The Network Operations team’s expert know-how and operational procedures have been previously converted into Executable Runbooks, allowing large swaths of contextual data to be pulled, parsed, analyzed, and displayed on the console at the push of a button by any engineer on the team.

Case Study of Proactive Automation

At this point, our NetOps team has identified the issue to be a misconfigured QoS parameter on a router. The misconfiguration has been successfully remediated with a configuration fix using the same NetBrain automation platform.

Should this problem recur, we now know what data is needed to identify the problem and what is required to fix it. By adding this issue to known-problem library, the team can ensure that they can identify and remediate much faster should this happen again.

Using NetBrain’s automation mechanisms, the additional diagnostic commands are added to the existing Executable Runbook. Should the event recur, the system will trigger automated diagnosis and the root cause will be determined instantly, with a near-zero Time to Repair for this repeat occurrence.

This also helps to automatically rule out possible known issues in unrelated incidents. This is the continual loop feedback mechanism discussed previously in this paper. The more known problems and scenarios for which an Executable Runbook is built, the further MTTR will reduce.

In summary, the proper post-mortem for proactive automation requires:

1. Adding the new problem, once resolved, to the known-problem library.
2. Adding or modify associated network baseline data for quicker known-problem detection should this issue recur.
3. Adding any necessary additional operations to the Executable Runbook so that the same problem can be identified more quickly next time.
The NetOps Automation Maturity Model

The above case study walked through a full workflow for incident response, providing real world examples of each automation touch point.

The below illustration outlines an expected maturity model for this deployment.

![Levels of NetOps Automation](image)

Rapid data collection, contextual data correlation, and executable know-how come together to greatly reduce the time needed to root cause an IT incident.

Out of the box, this automation architecture will provide immense value to an engineering team. The real value however is in the Proactive piece, with the steady building of the knowledge management system in the form of codified operational runbooks.

Conclusion

This whitepaper reviewed areas within MTTR that can lead to reduced time to resolve. Within NetOps, workflow automation must be applied as part of network automation. Triggered automation, interactive automation, and proactive automation all become part of a bigger NetOps automation methodology.

The automation platform must also provide Knowledge Management functionality for the network team, with the experts within the organization building the database of operational runbooks for every known problem encountered and resolved.

As a scalable platform, this enables teams to write automation procedures once to be used by everyone, to apply automation across any network and any workflow, and to leverage automation technologies as the primary mechanism for augmenting network troubleshooting. As shown, the solution must:

- Increase Automation at Every Level of MTTR Process
• Add Proactive Automation at each Post-mortem Stage

By incorporating NetOps automation into existing end-to-end workflows, incident response times are dramatically reduced. This enables an organization to continually improve MTTR year-after-year.

This is a transformational automation journey for Network Operations that provides immense value out of the box for Day-2 operations, with continually increasing value over time as the platform’s knowledge base grows. Applying this strategy enables the transformation of NetOps for the next generation network.