Enterprise Architecture

Enterprise Architecture Technical Board

Requirements for Operational Monitoring

Editor: Greg Charest

Version: 1.0
Last Revised:
Status: Final
Document Type: Guidance

Audience Level: Technical

ETAB Workgroup Members:

Jon Saperia
Rich Borroff
Luke Sullivan
Ann Lurie
Rob Ruma

Topic Statement:

A good deal of attention has been paid to specific tools for monitoring, and in some cases, vendors have been selected. Missing is a list of the desired management capabilities required in aggregate for HUIT. Also missing is an enumeration of the instrumentation (data) that each of the managed systems in our environment should support. This work addresses these two areas.

Complete topic description:
https://confluence.huit.harvard.edu/display/ETAB/Requirements+for+Operational+Monitoring+Working+Group

Executive Summary:

Operational management environments generally support five broad functional areas: Fault, Configuration, Accounting, Performance, and Security. To achieve these functions, a management environment will have two main building blocks:

- Management instrumentation - this is the data that is provided by the managed elements in the environment about their state, for example how much CPU or disk is in use, or how much traffic a particular interface has seen from a particular source.
- Management software - this software operates on the information provided by the managed elements. This software implements functions that take the data and reports on usage, faults, configuration changes, etc.

This document defines a basic set of data elements related to operational monitoring and instrumentation that should be provided by systems in the Harvard environment whether they are on-premises, in our AWS Cloud, or a SaaS provider. It also includes a description of the basic management functions that are expected for deployed services.

Introduction:

Scope of the Instrumentation
This list is focused on general instrumentation and is not intended to be a replacement for product specific instrumentation and tooling. For example, Oracle our main database supplier, has specialized instrumentation. This instrumentation provides visibility not likely to be available via standard methods and is indispensable to providing required management functions. Where possible, we should have a preference for standard products and instrumentation, but where we have made decisions to use products that can only be managed via access to proprietary instrumentation via proprietary software we must use those systems.

Data Collection and Analysis
Absent from the description of the data elements below are issues related to aggregation, overall architecture, frequency of collection, and functions performed on the data. Frequency of collection and functions performed will be described in the section on management functions. Architecture, including aggregation points, is beyond the scope of this backlog item. That can be taken up as a separate backlog topic if desired at the conclusion of this work item.

Discussion:

The further up the 'stack' you go, the more difficult it becomes to find ***common standard*** instrumentation. For this reason, when thinking about management/monitoring of technical services and their components, it is generally required that instrumentation from other levels be configured in such a way that they provide insight into application/service behavior. This is generally better that having a custom set of proprietary instrumentation and management tools for each application deployed at the University. Examples of standard instrumentation are provided for each of the categories below. In some cases, multiple technologies are available to provide the data. That involves defining the larger architecture for management and monitoring which is not in scope for this backlog topic, but could be taken up at the conclusion of this work. This work has value as a standalone element since it will help service owners identify what is required and begin to build environments that support the required instrumentation.

Recommendations: Instrumentation Requirements:

Applications

- Components (processes) that make up an application on an instance
- Application-specific memory partition monitoring, e.g., Java JVM heap consumption and garbage collection events, or Oracle database SGA/PGA usage - note, that this may come from OS level instrumentation
- Resource utilization by each of the components for each of the resources used (e.g., CPU, Memory, Stable Storage, Interface/Network)
- Information about fault and performance characteristics of dependent services from the application perspective such as I/O wait state information.
- Key metrics of performance such as latency for certain operations
- Completion/incompletion rates and other faults of the lower layer (e.g., disk or memory limits)
- Internal failures
- Where possible utilization by consumers
- Configuration changes
- Use/access policy violations
- Application-specific health checks/reports.

Middleware and Web Servers

Instrumentation for these software elements is largely the same as for applications:

- Components (processes) that make up a middleware/web server
- Resource utilization by each of the components for each of the resources used (e.g., CPU, Memory, Stable Storage, Interface/Network)
- Information about fault and performance characteristics of dependent services not only would I/O wait state be appropriate here, but waiting on database queries and 'lower level services' where possible
- Key metrics of performance such as latency for certain operations - such as web pages returned.
- Completion/incompletion rates and other faults of the lower layer (e.g., disk or memory limits)
- Internal failures

- Where possible utilization by consumers
- Configuration changes

Databases

As noted above, our primary database vendor, Oracle tends to try to have their users manage their products with proprietary tools, though some standard instrumentation has been defined. Below is a starting list of simple information relevant to databases that would be of general interest:

- The same process and resource utilization data as described for applications and middleware/web servers.
- The same information about dependent resources (e.g., storage) as for applications and middleware/web servers.
- Where possible utilization by consumers.
- Internal failures.
- Database specific information:
    - Read and write requests
    - Transactions
    - Current transactions
    - Current connection count
    - Disk queue depth
    - Locked and Active transactions
    - Database size allocation
    - Database utilization (of allocated space). From this available space can be determined
    - Read IOPS
    - Read latency
    - Write IOPS
    - Write latency
    - Resource failures
    - Running servers
    - DB disk and logical reads and writes
    - Inbound associations (e.g., SQL Connect)
    - Completed transactions

Operating Systems and Operating/System Resources (e.g., Storage, Memory, etc.)

- Many of the requirements for instrumentation for the OS and resources it uses are the same as those collected for layers above the OS. What distinguishes the data at the higher level is that they are aggregated/segmented in different ways, for example, all the processes related to an application, database, or middleware component.
- Items to include are:
-
    - System descriptive information (e.g., identifiers, how long it has been running, its idea of time and date)
    - Attached/available storage (and how much is used). This can be physical or logical
    - Current number of processes and their details such as:
        - Process name/path and type
        - Process size
        - CPU utilization
        - Memory utilization

        - Status including time of last state change
        - I/O state
        - Data read/written
        - FDs in use/open
    - Memory size
    - Allocated and used memory
    - CPU size and utilization
    - Disk size and utilization
    - Failures related to memory, disk, or other resource
    - SSL/TLS
        - certificate monitoring (impending expiry dates especially, but hash algorithm would be useful too, for when SHA-2 is deprecated).
        - encryption algorithm monitoring (to ensure we're not allowing SSLv2 for example).

Network Interfaces

It is understood that in AWS or other cloud/SaaS environments collection of some of these data elements may be more challenging.

- Number type and capacity of network interfaces
- Standard measures of aggregate network traffic: bytes, packets, errors:
    - By interface
    - Protocol and port
    - Open connections

Traffic from the Application Perspective (sources/destinations)

The idea is to capture information that that an understanding of the state of the application(s) on an instance of a system (physical or virtual). Aggregating across multiple systems is a function of the management software application and will be covered in the section on management functions. For each application, and its elements (processes), the following information

- The same data as in the process details in the Operating System section
- Network traffic by:
    - To/from sources
    - By protocol/port
    - Time spent waiting for I/O, network, or other resources
    - Application appropriate transactions (e.g. web pages, queries, etc.)

Key Infrastructure Services like DNS

- Usage by protocol operations
- Traffic to/from servers by location
- Required DNS records for an application to function (can be used in testing/validation)

Base Infrastructure (AWS, Network, etc.)

- For load balancers (and where applicable servers)
  - HTTP error codes by type (4xx, 5xx)
  - Connection requests
  - Backend healthy host count
  - Backend healthy host count
  - Backend connection errors - the number of connections that were not successfully established between the load balancer and the backend resource. Sources of failure could include application or database misconfiguration or over-utilization.
  - Latency measures (e.g., request/response)
  - Queue depth and rejected requests due to queue size/depth or other constraints
  - Failures to back end resources
- Common network statistics available such as those listed in the network interfaces section.
- Cost management: Notification of auto-scaling events, nightly summaries of EC2 instance counts, etc.

Recommendations: Requirements for Monitoring Events/Alarms

It is strongly desired that as much of the system as is possible be user configurable consistent with the requirements specified in this document and security.

The page on instrumentation requirements details information from different resources required for a management system to perform its functions. This page list the functions the management system should be able to perform. Functions that are related have been grouped into major categories. These categories are:

1. Basic data collection, aggregation and organization
2. Basic fault, configuration, accounting, performance, and security functions
3. Management system outputs (events/alarms, visualization functions, logs, etc.)
4. Management and periodic review of system settings

Collection, Aggregation, and Organization

This section details the basic data collection functions the management system should perform, on a configurable basis, for each management element it monitors. The details of the type of information to be collected for each managed system type such as a server are detailed on the instrumentation requirements page.

Basic Data Collection Parameters

- How often specific data elements are to be collected from specific resources like servers, applications, and infrastructure elements.
- Ability for users of the system to adjust collection of data (create new templates) based on factors like time of day, week, etc.

- Conditions that would cause an increase or decrease in the frequency of collected data elements, for example, if a fault or performance problem is identified, this collection system can be told to collect more or additional data elements from one or more systems.
- Ability to use templates by resource type for what data elements are to be collected. For example, collect certain pages served information from web servers.

Aggregation

We need to be able to aggregate across a variety of dimensions. Here are some examples:

- Work that a combination of resources is doing in support of a single service. For example, if multiple web servers are supporting a single application, or if several middleware components are making database requests.
- Aggregation of traffic from multiple sources to a specific destination (often we will want to sub-total by port).
- Aggregation across different time intervals.
- Aggregation of work performed by cooperating systems delivering a service. For example, a set of Web servers supporting an application.

EXAMPLES: This type of aggregation can help us in capacity planning for services such as when will more network capacity be required or when should a database or other type of server be upgraded to the next larger size, or when an auto scale group size might be increased.

Data Reduction

- Event/alarm and data de-duplication - some systems are not able to 'throttle' events. That is while a condition is true, they will continue to send/log events. The management system must be able to identify duplicates, and based on configuration, suppress, duplicates. Control over this feature should be both by manual intervention as well as configuration/programmatically.
- Hysteresis - for example, if a route is flapping, that is, available then is not, many systems will send a notification each time the state changes. The management system should be able to rate limit based on transitions per unit of time on a configurable basis.
- Across different time periods - The further in time one moves away from a particular event, the less valuable that event may become. The idea is to allow the management system to combine/average historic data to reduce space required and improve performance.

EXAMPLES: The objective here is to assist operational personnel focus on the issues of greatest importance and if there are issues, get to the root cause as soon as possible. By 'throttling' alarms or cutting down on event duplicates, operation staff are aware of problems but not getting so much information that it distracts them from the tasks of finding the error. See the route flapping example above.

Grouping/Hierarchy

- Ability to algorithmically or manually group elements together based on topology or other factors.

Management System Change

- Logs and configurable notifications for changes to the configuration of the management system

Managed Element Enrollment

- Ability to programmatically or manually enroll a managed element for monitoring when it is created or on certain state changes
- Ability for elements to self-enroll - for example, sending a message to the management system with basic information on boot up

Basic Fault, Configuration, Accounting, Performance, and Security Functions

This section identifies the operation performed on data received by the management system that have been divided into the recognized areas of management.

Fault Identification

- Root causes - It is not always possible to get at the 'root' cause with information available. The objective of these functions is to help operations people get as close to the cause of the failure as is possible
- Changes in specific values of any kind - the change in value of some management objects can be significant. The system should be able to identify when a new value is different from the previous value
- Rate of change - The system should be able to tell when the rate of change is increasing or decreasing. For example, if disk, memory, network, CPU, or process utilization has been growing at 1% for some period of time, and suddenly increases by 5%, this 'may' be significant. The system should provide functions for the configuration of this monitoring and notification.
- Direction of change - Changes in direction may be significant. For example, if consumption has been growing for some time and it reverses, investigation may be warranted.
- Fault/performance workflows (e.g., escalation)

Configuration Monitoring

- This function does not perform the configuration of elements in the environment, it monitors/receives information about when a configuration change has taken place.
- Where possible the monitoring system should have communication with the various configuration management systems that control the monitored elements. This applies whether speaking about applications and/or their deployments, base system configurations, etc.

Performance, Utilization, and Capacity Planning

- Performance data is concerned with how well a system, service, or portion is performing. The management system should be able to collect data that indicates performance as well as conducting some synthetic transactions such as retrieval of Web pages.
- Utilization measures consumption against capacity. To perform this function, the system should be able to retrieve information from the managed element that indicates capacity such as: disk/storage capacity, memory, network capacity, etc.
- For the purpose of our system, capacity planning is simply observing utilization over time for a monitored element and projecting forward in time.

Accounting

The essential difference between accounting data and performance/utilization data is that accounting data allocates consumption by an identifiable consumer such as an application, service, network, or other identifiable entity. It is not the intention of this set of functions to replicate a billing system. The idea is to use collected usage data to inform decision making capacity planning, performance or cost management.

Security

This section refers to issues related to the management software itself:

- Security controls on the management system
- Security/controls for communication between:
    - Elements of the management system
    - The management system and the managed elements in the environment
    - Ability to create/control users and group permissions
    - Controls to prevent undesired sharing key's/tokens etc.

Management System Outputs

Previous sections have identified the type of data to be collected and the operations to be performed on that data. This section enumerates the various outputs of the system.

Visualization/Graphic Display

- Dashboards - It should be possible for users with appropriate authorization to create customized dashboards appropriate to the services they are responsible for managing.
- User profiles - allow for customization of information for individual users or groups of users.

Outputs and APIs

This set of facilities relates to how the management system can integrate with other systems in the HUIT environment. The system should provide:

- Standard logging outputs

- APIs for extraction of data/functions and integration with other systems
- Support for common management protocols for event notification (see below)

Controls for Event/Alarms/Notification and Data Collection

- Maintenance and test windows
- For data collection based on certain values
- Events/Alarms and notifications
- Based on primary secondary (backup status of systems)

Historic Reporting

- Data reporting and archival policy
- Reporting schedules (e.g., daily, weekly, monthly, ad hoc)

Events/Alarms

- Mail
- Graphical/visual display
- See table below

Requirement	Justification
Ability to override schedules	The ability to modify schedules to allow for time off is necessary for the operations workflow.
Automated rotation	The ability to automatically rotate on-call operations team members is necessary for the operations workflow.
Automated event escalation	The ability to escalate events to another operations team member is necessary for the operations workflow.
Event groups and classification	The ability to group or classify events based on pre-determined criteria is necessary for the operations workflow.
Event notification suppression expiration	The ability to suppress an event notification is necessary for the operations workflow. The idea is that if you have an event that keeps firing (or is causing an alarm to fire) you want to be able to latch until cleared.
Event history auditing and reporting	The ability to view historical events is necessary for the operations workflow.
Guaranteed alert delivery	End-to-end alert delivery is critical to the operations workflow.

International notification capabilities	The ability to send notifications internationally
Multiple region availability	High availability in multiple regions
On-call schedule history	The ability to view historical on-call participation is necessary for the operations workflow.
On-call schedules	The ability to schedule operations team members to participate in an on-call schedule is necessary for the operations workflow.
Service Element Provisioned State	If a server for example, is simply a backup or a link is a secondary/backup, a failure is less critical than a main production element. The system should be able to be configured to know about these differences. This concept applies to our different environments (e.g., DEV, PROD, etc.)
Single platform	Single configuration and administration interface
Support for authentication federation	The ability to federate with the currently preferred authentication system
Support for software integrations	The ability to integrate alerts and notifications to currently implemented reporting platforms
Support multiple event notification types	The ability to use SMS messaging, mobile app push notification, phone call or email is necessary for the operations workflow
Support Time-of-Day Alerting	The ability to set up different alerting methods and pathways for events based on the time of day
Web based and mobile application capable	The ability to use a web based dashboard or a mobile application to respond to events is necessary for the operations workflow.

Recommended Features

Requirement	Justification
Automated on-call reminder notifications	The ability to configure on-call reminder notifications is a nonessential feature for the operations workflow.
Ability to export schedules to calendar applications	The ability to export on-call schedules to a personal calendar is a nonessential feature for the operations workflow.
Ability to parse structured data	The ability to ingest data from a currently implemented Application Programming Interface

Support for an Application Programming Interface	The ability to provide an Application Programming Interface to allow alerting data ingestion

System/Settings Review Policies/Defaults

1. On a periodic basis review alarm/event/function settings to ensure optimized data collection without getting too much or falling into ignoring events/alarms.
2. If an alarm keeps firing, a review of the root cause should be conducted as soon as possible to mitigate the problem.
3. A common data collection interval for most elements is 300 seconds.