# Enterprise Logging

The purpose of the Enterprise Logging within any organization is to provide critical visibility into organization’s services in order to diagnose and debug efficiently and to detect violations, such as security vulnerabilities. As a large government organization, VA is also under scrutiny of many various regulations, so having an enterprise logging solution helps prevent compliance violation of such regulatory requirements.

A logging management process covers the following steps:

1. Collect – collect the logs from various platforms
2. Alert – alert on events logged into the logging platform
3. Report – provide reports on logged events and raised alerts
4. Store – store historical and real-time logs in the logging platform and archive in offline storage solution after the retention age
5. Search – provide ability for users and admins to search and filter logs by key attributes, such as event name, message type, log timestamp, originating platform, application name, source IP address, etc.
6. Share – finally distribute the logs via logging services and solutions within the organization



### Logging Types

There are three major logging types:

* **Application Logging** – targeted to log application events and messages emitted from custom applications and commercial products. An example application log event is to log a start and end of a custom function or routine execution and to log exceptions raised by the application code.
* **API Logging** – focused on REST and SOAP API logging, with a primary focus on logging service request/response patterns, custom API events and exceptions. An example API logged event is to log an HTTP request payload and query string parameters, followed by logged response from the backend platform.
* **Infrastructure Logging** – used to log network and server events, with a focus on logging infrastructure events for further monitoring and troubleshooting. An example infrastructure
* Security Logging – security logging is targeting events used by the security teams to quickly audit applications and platforms for any security vulnerabilities and malicious uses.

SIEM (Security Information and Event Management) logging platform provides specialized logging services which comply with the SIEM security standards. A SIEM-compliant platform would have these capabilities:

* The aggregation, analysis, and reporting of log output from networks, operating systems, databases, and applications.
* Applications that verify identities and manage access.
* Vulnerability management and forensic analysis.
* Policy compliance.
* External threat notifications.
* Customizable dashboards.

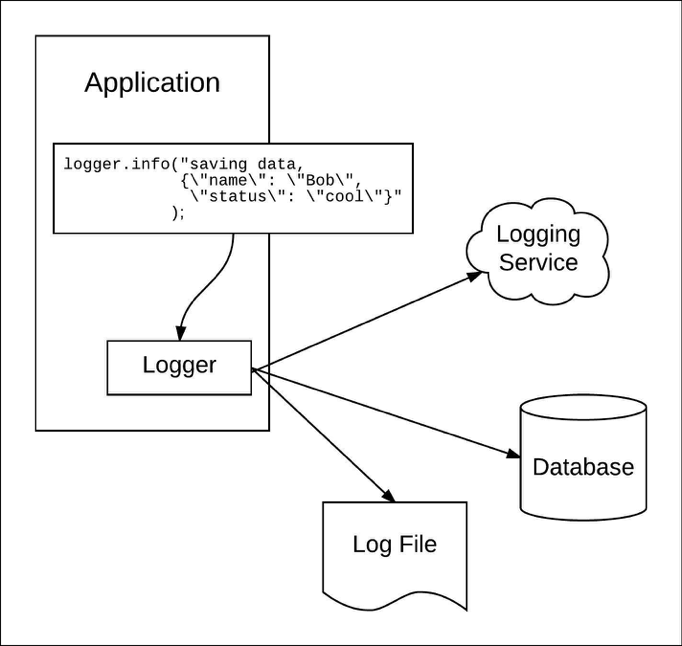
### Logging Targets

A good enterprise logging platform should have capabilities to log events and messages to various targets, which include:

* Log File on Disk/Network Drive
* Console
* Logging Service (e.g., Splunk)
* Big Data Store (e.g., Hadoop)
* Relational Database (e.g., Oracle or SQL Server)
* No-SQL Database (e.g., MongoDB)

Depending on the structure and volume of logged events, an organization may prefer to log all events to a No-SQL or Relational Database, or to a Logging Service such as Splunk.

A diagram below provides an example of how a logged message can be distributed across multiple targets, including a logging service (such as Splunk logging aggregator service), a database, and a log file.



## Preferred Logging Tools and Platforms

As applications and infrastructure components get added to the VA’s data centers and private cloud instances, the amount of log data collected grows enormously. Therefore, having appropriate logging solutions to aggregate all of the logged data, store and make this data searchable and auditable is critical for VA. When selecting enterprise logging platform and designing an enterprise-wide logging services, there is a number of factors to consider:

1. **Security** – because logs contain data from various applications, infrastructure, and network devices, they may contain sensitive PII, PHII and PCI information. To prevent an operator or network intruder from breaching that information, the enterprise logs have to secured *at rest* and *in transit*. Sensitive information such as EIN, SSN, Tax ID, Account Number, Card Number, Test Results are to be either masked or encrypted, to prevent an intruder from obtaining this information in the logs.
2. **High Availability** –when transmitting log data to the required backend for compliance or debugging, missing portions of logs create a discrepancy in the enterprise history. With such discrepancies and gaps in the logs, it would be impossible to determine which application failed, what network device streamed malicious content, or what source IP address hacked the security firewall. High availability of the logging infrastructure means having a secondary logging pipeline that can route and audit critical logs if the primary pipeline is offline.
3. **Processing and Filtering at the Edge** – instead of waiting until all the data is in the logging platform to begin analysis, the logging pipeline should do some preliminary work to augment streams of data with relevant information or distribute processing across multiple nodes. Sometimes, logs produce a bunch of noise rather than store crucial information. The logging pipeline should be able to filter on a variety of fields, including the source of the log, severity, content inside the message and log time.
4. **Application-independent Logging** – when the logging pipeline is tightly coupled with the application, the application needs to be re-deployed every time there is a change required to the logging pipeline. Having an application-independent logging is critical to avoid application dependencies on the logging services and platform.
5. **Reliability** – the logging backend platform should be always up and running, accepting logs from various applications across the enterprise. If the logging backend becomes unavailable, the system should try to recover and get back online, or switch to the backup logging backend until the primary platform is being recovered from an error or hardware failure.
6. **Monitoring** – logging platform can be down or unavailable for various reasons, preventing critical logs to be shipped and saved via the logging pipeline in the logging backend platform. Therefore, monitoring the logging platform is critical to quickly determine if the logs are no longer being generated.
7. **Logging Agent Weight** – the logging agent deployed next to the application should be lightweight, performant, and reliable. Using a heave logging agent can impact the overall performance of the applications.

Commonly used logging platforms include on-premise and cloud-based logging service providers. A widely-popular platform that supports both on-premise and cloud-based deployments is Splunk, followed by other cloud players such as Loggly, LogDNA, Sumo Logic, Logz.io, and a few dozen similar solutions. Other logging solutions include IAAS platforms with built-in logging services, such as AWS Event Logging, Azure LogAnalytics, and Google Cloud Logging.

With all of these platforms, besides the seven factors listed above, three other key factors when selecting a logging platform should include ability to manage both application and security/SIEM logging solutions, cost of storage of large log volumes, and the retention policy (number of days for storing historical logs before discarding or archival).

For API-level logging solution, it is important to consider how the API Gateways and API Management platforms within the organization can easily integrate with the logging platform of choice. Splunk, for example, provides various integration options, using Splunk APIs, using Log4j Appenders, or using Splunk Forwarders installed directly on the servers. In addition, a number of API Gateway vendors have built-in integrations with Splunk via connectors, plugins, and policies, which helps the API development teams to avoid unnecessary coding of the logging integration.

## Leveraging Enterprise Logging for Troubleshooting

Enterprise logging use cases vary across organizations, but most of the users of the logging platforms require certain ability to use the logging platform to troubleshoot issues and exceptions for a given application or an API. A logging platform should provide ability to quickly filter and select an application name, timestamp, and a keyword such as “Error” or “Exception”.

There are certain best practices for API and application development which help with troubleshooting issues with that API or application in the future. One key best practice is to use a correlation id or tracking id that can be passed across application components or across API layers to easily trace a single request or user operation. Such correlation id or tracking id is usually a generated GUID (using a UUID function or other GUID generator). A correlation id is generated at the outer most layer, such as a User Interface used by the end customer or the experience API layer. This correlation id is then passed from layer to layer all the way to the final backend platform to be returned back to the invoker in response. This correlation id gets logged at each layer and operation, to be easily filtered and searched on during troubleshooting of any issues.

A screenshot from Splunk below provides an example dashboard with an ability to quickly add search criteria and filter logged messages by a certain keyword such as “Exception”:

