30-06-2023

White Paper:
Argus OAV Architecture Analysis

Grant Agreement No.:	101100680
Work Package:	WP6
Task Item:	Tasks 3 and 4
Nature of Document:	White Paper
Dissemination Level:	PU (Public)
Lead Partner:	Sikt
Document ID:	GN5-1-23-3BDA1F
Authors:	Morten Brekkevold (Sikt), Vidar Faltinsen (Sikt), Ivana Golub (PSNC), Aleksandra Dedinec (UKIM/MARnet), Maria Isabel Gandía (CSUC/RedIRIS)

Abstract

Argus is an open-source tool for Network Operation Centres and service desks to aggregate incidents from all their monitoring applications into a single, unified dashboard and notification system. This document analyses the mapping of the Argus architecture to the TM Forum's Open Digital Architecture, aiming to provide a standardised view of the components and implementations of orchestration, automation and virtualisation within the Argus deployment at Sikt, the Norwegian NREN.

Table of Contents

Table of Figures

Executive Summary

This document analyses the mapping of the Argus architecture to the TM Forum's Open Digital Architecture (ODA), aiming to provide a standardised view of the components and implementations of orchestration, automation and virtualisation (OAV) within the Argus deployment at the Norwegian National Research and Education Network (NREN), Sikt.

Argus is an open-source tool for Network Operation Centres (NOCs) and service desks to aggregate incidents from all their monitoring applications into a single, unified dashboard and notification system. As most NOCs use a myriad of applications to monitor their infrastructure and services, very often more than one of these applications are configured to raise alarms in the event of unexpected and unwanted situations. For operational teams, the need to monitor several applications at the same time in search for active alarms is neither practical nor a very efficient solution. Argus mitigates these scenarios by providing the NOC with a singular overview of actionable incidents, and by providing a single point of notification configuration.

As is the case with many network management systems (NMSs), each of them might incorporate different tools for different functionalities. Even though such an architecture is easily understood by the team that is using the system, it might not be clear to another team, either within the same organisation or from another organisation. This is where the Open Digital Architecture as a reference architecture comes into play. Mapping existing digital architectures to a reference architecture helps with a better understanding of a system's architecture, which can then help with interoperability, integration and joint operations.

1 Introduction

1.1 Argus

Argus [ARGUS] is an open-source tool for Network Operation Centres (NOCs) and service desks to aggregate incidents from all their monitoring applications into a single, unified dashboard and notification system. Most NOCs will, out of necessity, use a myriad of applications to monitor their infrastructure and services. In turn, they need to contend with manually managing notification profiles and monitoring dashboards in each individual application. Argus mitigates these scenarios by providing the NOC with a singular overview of actionable incidents, and by providing a single point of notification configuration and visualisation.

Argus is agnostic with regard to the details of each monitoring application, but instead provides a REST API to report new incidents, and to search, fetch or update the status of already registered incidents. Glue services for several source applications already exist, and more can be easily written using the documented API and/or existing API client libraries for Python. Incidents are associated with a source application and can be tagged with arbitrary metadata from the source application, including URLs to drill down into incident details in the source application. Metadata can be used to make arbitrary incident filters, which can be applied both in the dashboard UI and in notification profiles. Mechanisms also exist to add acknowledgements to incidents, and to link incidents with tickets in the NOC's ticketing application. The data model even supports registering inter-relationships between incidents. Notifications via email, SMS and Microsoft Teams are supported, while more mediums are planned (such as Slack).

Argus has been proven useful in providing a distinct interface between individual product teams and the organisation's NOC when it comes to defining which monitoring alerts should be handled by the NOC, and which ones the product teams will handle themselves.

Argus is open source and is mainly developed by resources from Sikt (the Norwegian NREN), including through participation in and financing from the Network Technologies and Services Development Work Package (WP6) of the GN4-3 project and the Network Development Work Package (WP6) of the GN5-1 project. Argus is currently in production use at Sikt and SUNET (the Swedish NREN).

1.2 Open Digital Architecture

Argus's orchestration, automation and virtualisation (OAV) architecture analysis has been conducted using the TM Forum Open Digital Architecture (ODA) [ODA] functional blocks as a reference point. The TM Forum ODA is promoted as a blueprint for new digital industry architectures, and the rationale for its selection as a reference model by the GN5-1 WP6 Network eAcademy team is given in the GN4-3 Deliverable D6.6 *Transforming Services with Orchestration and Automation* [GN4-3_D6.6]. The whole set of ODA documentation provides common terminology, a minimum set of core design principles, and groups of decoupled functionalities. Together they define the requirements for the implementation of an agile model-driven service management architecture that incorporates orchestration and automated operations, as well as virtualised or hybrid environments.

The main idea behind ODA is that of decoupling and integration of components, which enables the independent selection of solutions for each component while at the same time maintaining a unified overall approach that

supports the full end-to-end service lifecycle (including interoperability). The high-level ODA functional architecture maps the main components by their capabilities into the ODA functional blocks (see Figure 1.1).

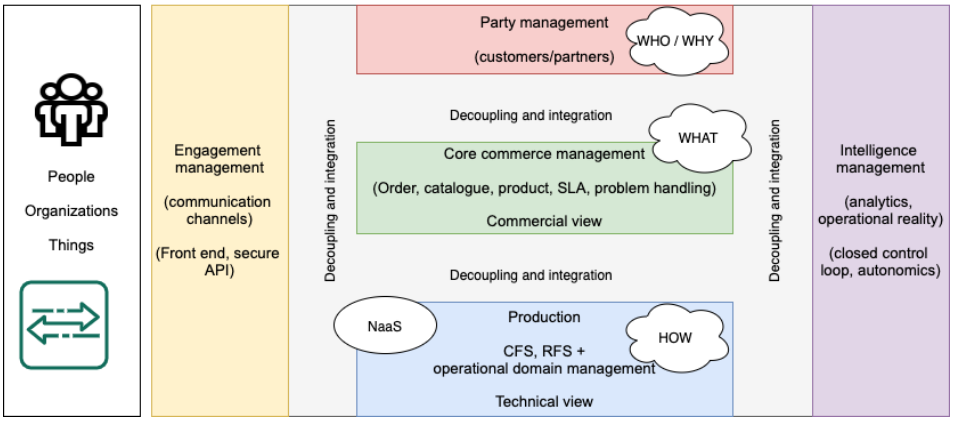

Figure 1.1: The TM Forum ODA functional architecture

In a nutshell:

- The **Engagement Management** functional block focuses on the engagement with the end users (people and systems) that can interact via multiple channels.
- The **Party Management** functional block handles the processes that are related to all parties that interact with the organisation and defines their roles and relationships.
- The **Intelligence Management** functional block is in charge of the implementation of data analytics processes, and, based on the analysis, provides closed control loops for full automation wherever possible.
- The **Core Commerce Management** functional block focuses on the placement of products and services to the customers, and manages the product lifecycle.
- The **Production** functional block manages the delivery and lifecycle of all customer-facing and resource-facing services; these can be based on different technologies or might be a combination of multiple operational domains, including multi-domain services provided with the cooperation of other parties.
- **Decoupling and Integration** accentuates the separation of concerns between the functional blocks and serves as the glue between the functional blocks, supporting interoperability.

2 Architecture Analysis

2.1 Sikt, the SSC and Argus: High-Level OAV Approach

Sikt [SIKT] is a large organisation, the result of a merger between three formerly separate government agencies (the Norwegain NREN Uninett, the Norwegian Centre for Research Data (NSD) and the Directorate for ICT and Joint Services in Higher Education and Research (Unit)), where the division for Data and Infrastructure (DI) largely encompasses the NREN activities previously managed by Uninett. Within just DI alone, a multitude of applications are deployed to monitor the various services and products operated by the division, even before the introduction of Campus Network as a Service (CNaaS). With CNaaS, the number of monitoring application deployments grows with every new customer.

The Sikt Service Centre (SSC) is a combination of a traditional Network Operation Centre (NOC) and a service desk and is a continuously evolving part of Sikt. The SSC monitors Sikt's IT infrastructure, identifying and resolving problems before they impact users, and handles user requests, troubleshoots problems, and escalates more complex issues to higher-level support teams, inspired by the ITIL principles.

The SSC's base of operations is reminiscent of a war room, with large screens occupying one of its walls. The management of screen real estate is crucial to ensure that the most important and relevant information is easily visible and accessible to the team. While there may in principle be 15 different "dashboards" to screen for alerts for CNaaS alone, the relevant incidents detected by these tools are aggregated into a single dashboard using Argus, which has become a crucial tool for the management of CNaaS incidents at Sikt.

Argus itself is developed as open-source software by the Sikt CNaaS developer team, supported by the GÉANT project. The challenges and requirements of day-to-day operations at the SSC constitute most of the requirements used to build Argus. Making Argus easy to integrate with other systems, such as monitoring applications, ticketing systems and messaging platforms, has been paramount in making Argus a viable solution also for other NOCs and service desks and is a key reason why Argus development has been supported by the GÉANT project.

Sikt's Argus production deployment is managed by the CNaaS developer team through the use of Sikt's preferred deployment platform Platon. Platon, also termed Platform as a Service (PaaS), consists of an AWS-hosted Kubernetes cluster, where application deployments are easily managed through GitLab repositories and automated continuous integration / continuous delivery (CI/CD) pipelines.

The CNaaS network engineering team (the team supporting campuses in managing their local area network infrastructure) is responsible for the design, procurement, installation and day-to-day remote operation of CNaaS network deployments. The team employs various monitoring applications for each deployment and is responsible for establishing service level agreements (SLAs) with the SSC. They train SSC personnel when necessary, and they ensure that incidents covered by the SLA get reported from their monitoring applications into the Argus dashboard seen by the SSC.

While the SSC maintains its overview of the current IT infrastructure through visual observation of its screens, it only operates during typical working hours. Outside of working hours, Sikt maintains an on-call duty roster. The person on call will not continuously monitor the dashboards but will typically look them over before writing their daily reports. It is therefore important that incidents of a severity that warrants immediate response outside working hours are communicated to the on-call person through SMS messages sent to the on-call mobile phone (a phone that is physically relayed to the next person on the duty roster during on-call rotation). The filtering and notification dispatch for these severe incidents are managed using Argus.

2.2 Mapping to ODA Functional Architecture

When put into the context of the TM Forum ODA functional representation, the Argus architecture components can be mapped as shown in Figure 2.1. The grey boxes in the diagram represent the Sikt Argus deployment architecture components, and their placement within the ODA functional blocks is defined based on their main functionalities. If the components provide complex, multifaceted functionalities, they are broken down into sub-components which are placed in the respective functional block. The mapping is explained in more detail below.

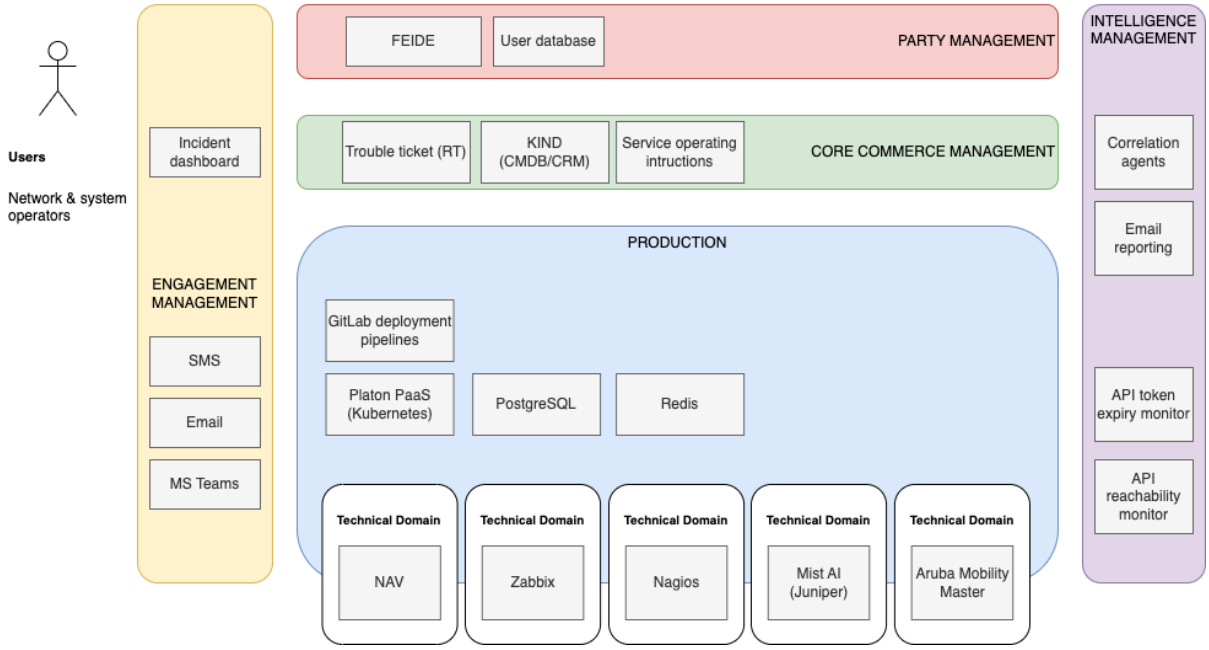

Figure 2.1: Sikt's Argus deployment components mapped to the TM Forum ODA

2.2.1 Engagement Management

There are two primary methods of user engagement. One is the Argus Incident Dashboard, used by the SSC in their base of operations, but also displayed on an office wall screen where the CNaaS network engineers are seated. The second is dispatching incident notifications through various channels, such as email, SMS, MS Teams, or any other medium for which a plugin has been written. This is used by whoever is on call, but to some extent, it is also used by the CNaaS network engineers (with the aim of delivering the best customer network experience possible).

2.2.2 Party Management

Party management is primarily about who is allowed to access Argus and the data stored within it.

While Argus does feature a local user database, it is used to create system- or role-based accounts. Examples of these include the following:

- The on-call role has an account, used to define a notification profile for the on-call duty roster.
- Every monitoring application that integrates with Argus needs a system account, so that API access tokens can be issued for it.

Accounts and access for physical persons are managed through single sign-on with the national identity provider FEIDE.

2.2.3 Core Commerce Management

Core commerce management components are all about how the SSC and service operators interact with each other and with affected customers during the handling of incidents.

Any time Argus reports an incident that warrants a response from Sikt, the SSC creates a trouble ticket in the Request Tracker (RT) ticketing system, and links the two together. This ticket can be viewed as a log of exchanges/transactions between the SSC, the service operators and/or the customer, and serves as an audit trail of what occurred during the incident response.

How to respond to a given incident depends on which service is affected. A service must have published a set of agreed-upon operating instructions before the SSC will commit to monitoring and responding to its incidents. Any incident posted to Argus must be explicitly tagged with a service identifier, which can be used to identify the correct operating instructions.

Incidents are normally also tagged with a customer identification, and with the affected physical components. The inventory database, KIND, will be consulted by the SSC. This is where the information about physical components is kept, with information such as which service subscriptions and customers they serve, and who are the technical contacts, both at Sikt and at the customer site, etc. Together with the operating instructions, this informs the SSC about whom the trouble ticket should be directed at.

2.2.4 Production

The production components of the Argus architecture are chiefly divided between two groups:

- The components that constitute the deployment of the Argus software itself.
- The various components that comprise the monitoring services that report to Argus (the technical domains).

Deployment of Argus in Sikt is fully automated by GitLab CI/CD pipelines that integrate with Sikt's Kubernetes infrastructure (colloquially named Platon in-house). Some of the important aspects of these pipelines are:

- The back-end and front-end components of Argus are deployed in tandem by the same pipeline.
- To upgrade to a latest version of either the back-end or front-end components, all that is needed is to push a Git commit that changes the variable defining the software version tag to deploy.
- All deployments are automatically made in a staging environment that is separate from the production environment. If needed/wanted, the staging environment can be manually tested and verified. The production environment is only replaced by the staging environment when one of the devops team manually pushes the deploy button in GitLab.
- Any number of "review" sites can be created at will. If there is a need to test experimental Argus functionality in a production-like environment, all that is needed is to push a new Git branch to GitLab, and a separate environment for that Argus deployment is created. This environment will be deployed with a snapshot of the Argus production database, so it can be freely altered and experimented with, without affecting the production environment.

The two main external services on which each Argus instance depends are:

- **Redis** [REDIS]: open-source, in-memory data store used for caching and message passing internally between Argus components during runtime. Each Argus instance deploys its own Redis container as part of its Kubernetes deployment.
- **PostgreSQL** [PostgreSQL]: the RDBMS where Argus stores its persistent data about incidents, users, notification profiles, source systems and so on. For the production environment, Argus uses a hosted PostgreSQL service managed by AWS on behalf of Sikt. Review instances will deploy temporary PostgreSQL containers that are provisioned with a snapshot copy of the production database.

2.2.5 Technical Domains

The technical domains consist of all the various monitoring applications in use at Sikt. These are set up and maintained by the operators of each service that uses them. To integrate any of them with Argus, a glue service for that application must first be installed (or written from scratch if none exists). Once a glue service exists, the application instance must be registered as a "source system" in the Argus administration panel, and an API token generated for it. The glue service needs to be configured with the necessary details to contact the Argus API.

Some of the domains that integrate with Argus today are:

- **Network Administration Visualised (NAV)** [NAV]: the main open-source NMS software used to monitor campus networks. Each CNaaS customer has a separate instance of NAV, and each instance is registered in Argus as a "source system".
- **Zabbix** [ZABBIX]: the main service monitoring tool employed by Sikt.
- **Nagios** [NAGIOS]: the main service monitoring tool employed by SUNET. Although it is not used by Sikt, Nagios is included in the architecture diagram since SUNET has written and published an Argus glue service for it.
- **Mist AI** [MIST_AI]: a cloud-based Wi-Fi control plane for access points produced by Juniper, which are used for delivering Wi-Fi services to some CNaaS customers. Mist can deliver incidents as webhooks, so a single webhook-based glue service is deployed to convert Mist webhooks for all CNaaS customers into Argus incidents.
- **Aruba Mobility Master** [ARUBA_MM] (now known as Mobility Conductor): a cloud-based Wi-Fi control plane for access points produced by Aruba, which are used for delivering Wi-Fi services to some CNaaS customers. As with Mist, only a single glue service instance is needed to synchronise Mobility Master incidents to Argus for all customers.

2.2.6 Intelligence Management

These components monitor or interact with Argus and contribute to the smooth flow of incidents through Argus. Some of the components are still in the planning stage, and not in actual production.

The most important are:

- **API reachability monitor**: formed by Zabbix triggers that test the Argus API for reachability from the various glue services. It detects sudden ACL problems, or incorrect configuration of authentication in glue software.
- **API token expiry monitor**: for the time being, Argus API tokens cannot be renewed automatically, and need to be renewed manually by an administrator. This component ensures incidents are created for tokens that are nearing expiry, as a reminder to the team to renew the tokens. This was initially

envisioned as an external API agent but was instead implemented directly into the Argus back end as a monitor that can be scheduled as a recurring job.

- **Email reporting**: the normal SSC procedure is to acknowledge all incoming incidents as an indicator that they have been noticed and are being handled by the SSC. An external API agent creates a daily email report of incidents that occurred within the last 24 hours, but that have not been acknowledged by the SSC. This is just an extra safety measure, in the rare case where human error causes transient incidents to go unnoticed and unhandled.

  Currently, the email report is not sent to any individual, but to an MS Teams channel for the CNaaS service team.

Additionally, when the inter-incident relational data model is exposed through the Argus API in a future release, the developer team plan to build Sikt-specific automated correlation agents that can utilise cross-domain service knowledge to automatically correlate incidents.

3 Conclusions

This document has presented the Argus OAV architecture as seen through the lens of the TM Forum Open Digital Architecture to provide information on how each of the component tools fits into the ODA functional blocks.

The architecture outlined in this document has demonstrated significant benefits for incident management at Sikt. Through the implementation of containerisation, Kubernetes orchestration, and integrated yet separate components, Sikt can ensure strong fault tolerance, high availability, and efficient resource utilisation. Additionally, the Argus solution efficiently manages screen real estate to minimise incident clutter at the service desk. This architecture enables simplified deployment processes, smoother scaling, and seamless updates, all of which are essential for efficient incident management. The deployment and orchestration architecture described in this document is the optimal choice for Sikt to enhance its incident management capabilities and provide high-quality services to its customers.

Finally, it can be concluded that the Argus architecture is aligned with the TM Forum ODA core principles and design concepts. It is modular, with API-based building blocks that make it compatible with other architectures aligned with the same reference architecture.

References

[ARGUS]	https://network.geant.org/argus/
[ARUBA_MM]	https://www.arubanetworks.com/products/wireless/gateways-and-controllers/mobility-conductor/
[GN4-3_D6.6]	GN4-3 Deliverable D6.6 *Transforming Services with Orchestration and Automation* https://geant.org/wp-content/uploads/2021/11/D6.6-Transforming_Services_with_Orchestration_and_Automation.pdf
[MIST_AI]	https://www.juniper.net/gb/en/products/mist-ai.html
[NAGIOS]	https://www.nagios.org/
[NAV]	https://nav.uninett.no/
[ODA]	TM Forum, GB998 Open Digital Architecture (ODA) Concepts & Principles v2.1.0, March 2021 https://www.tmforum.org/resources/reference/gb998-open-digital-architecture-oda-concepts-principles-v2-1-0/
[PostgreSQL]	https://www.postgresql.org/
[REDIS]	https://redis.io/
[SIKT]	https://sikt.no/en/home
[ZABBIX]	https://www.zabbix.com/

Glossary

ACL	Access-Control List
AI	Artificial Intelligence
API	Application Programming Interface
AWS	Amazon Web Services
CFS	Customer-Facing Service
CI/CD	Continuous Integration / Continuous Delivery (or Deployment)
CMDB	Configuration Management Database
CNaaS	Campus Network as a Service
CRM	Customer Relationship Management system
DI	Data and Infrastructure
FEIDE	Felles Elektronisk Identitetshåndtering (Common Electronic Identity Management), the Norwegian identity provider
ICT	Information and Communication Technology
ITIL	Information Technology Infrastructure Library
KIND	Sikt inventory database
MS	Microsoft
NaaS	Network as a Service
NAV	Network Administration Visualised
NMS	Network Management System
NREN	National Research and Education Network
NOC	Network Operation Centre
NSD	Norwegian Centre for Research Data
OAV	Orchestration, Automation and Virtualisation
ODA	Open Digital Architecture
PaaS	Platform as a Service
RDBMS	Relational Database Management System
REST	Representational State Transfer
RFS	Resource-Facing Service
RT	Request Tracker
SLA	Service Level Agreement
SMS	Short Message Service
SSC	Sikt Service Centre
UI	User Interface
Unit	Directorate for ICT and joint services in higher education and research
URL	Uniform Resource Locator
WP	Work Package