CELTIC-NEXT
Eureka Cluster

Project Report

# CISSAN

## Collective intelligence supported by security-aware nodes

## D5.5 Tools to optimally distribute security sub-functions'

**Editor:** Lars-Göran Magnusson, Arctos Labs and Jari Partanen, Bittium

**Abstract**

This report describes the approaches and techniques used within the CISSAN project for the purpose of optimally allocating security sub-functions to devices and IoT/OT networks to enable collective intelligence. In this respect, the architecture and operation of the CISSAN Orchestrator, along with that of the optimization solver used in conjunction, are described, providing the possibility of dynamic task allocation, execution, and coordination in accordance with the capabilities of devices, the characteristics of the network, as well as disaster recovery and quality of service requirements. This report also elaborates on approaches for discovering and managing devices, performing task provisioning, data routing, and task execution.

The use of this approach allows for distributed and adaptive security operations, in which security functions can be performed by devices best suited to them, thus contributing to enhanced system security and resilience. The deployment of the CISSAN Orchestrator in the CISSAN platform and the validation of its operation in several use cases provide an example of how this technology can be successfully implemented and deployed.

**CISSAN**

**Public Report**

**April 2026**

Participants in project CISSAN are:

- University of Jyväskylä (coordinator)
- Affärsverken Karlskrona AB
- Arctos Labs Scandinavia AB
- Blekinge Tekniska Högskolan
- Bittium Biosignals Ltd
- Bittium Wireless Ltd
- Blue Science Park
- Clavister AB
- Councilbox Ltd
- Geodata ZT Gmbh
- Mattersoft
- Mint Security Ltd
- Netox Ltd
- Nodeon Ltd
- Savantic AB
- Scopesensor Ltd
- Techinova AB
- Wirepas Ltd

CISSAN-Collective intelligence supported by security aware nodes

D5.5 Tools to optimally distribute security sub-functions

Editor: Lars-Göran Magnusson, Arctos Labs and Jari Partanen, Bittium

Project coordinator: Ilgin Safak, University of Jyväskylä

CELTIC published project result



Disclaimer

# Preface

The increasing scale and complexity of Internet of Things (IoT) networks, along with their critical nature, have revolutionized the security domain, requiring security solutions that are not only strong but also flexible, decentralized, and capable of autonomous operation. The traditional security architecture, which is centralized and strong, is not sufficient to protect IoT networks that are dynamic and heterogeneous in nature and where failure or attack can spread rapidly throughout the network. The approach that this work undertakes is to break free from traditional security models that offer only static security and to develop tools that facilitate security based on collective intelligence, where security functions are distributed throughout the network and dynamically coordinated depending on context, threats, and resources.

This report outlines the design and development of tools that facilitate the strategic distribution and orchestration of security functions throughout IoT devices, enabling these devices to function in collaboration and provide security to IoT networks. This approach enables the distribution of security sub-functions like monitoring, detection, cryptographic services, and response to be strategically positioned throughout the network where they are most effective at any given time. This approach to security makes all IoT devices contribute to the collective intelligence of the network, which is then able to respond to threats early and respond coherently to threats, thus providing enhanced security and resilience to IoT networks.

The significance of this work is that it has the potential to enhance IoT network resilience, scalability, and trustworthiness, especially when these networks are used to provide critical services and industrial processes. This approach to IoT security, which is based on collective intelligence and the dynamic distribution of security functions, provides a strong and forward-thinking approach to IoT security that is essential to European digital sovereignty and to safely deploying next-generation IoT systems.

# Executive Summary

The rapid growth of IoT systems and cyber-physical systems in critical infrastructure has significantly increased the complexity, scale, and impact of cybersecurity incidents, while traditional security methods have remained static, centralized, and inflexible to changing operating environments. The challenges facing organizations with large-scale IoT systems are increasing in sustaining continuous security, managing heterogeneous IoT systems, and achieving rapid recovery from cyber-attacks or system failures. The target audience of this report is IoT system operators, system integrators, security architects, and decision-makers seeking effective solutions to secure their distributed IoT systems and achieve reliable automated disaster recovery.

This report offers tools and techniques to optimally allocate security sub-functions to IoT systems through an intelligent orchestration framework. The CISSAN Orchestrator is responsible for continuous IoT device discovery, device management, and automated disaster recovery and works together with the CISSAN Management Server and Optimization Solver. The CISSAN Management Server maintains a global view of network resources and their associated trust levels, communicates with the Optimization Solver and Councilbox Eventchain System for obtaining optimal task allocations and performing blockchain transactions. It also triggers the execution of security tasks, performs access control and coordinates threat responses. The CISSAN Orchestrator works with the Arctos Labs Optimization Solver to determine the optimal allocation of security functions to available IoT devices. The security functions, which include monitoring, detection, protection, trust, and response, are dynamically allocated, reassigned, and coordinated in real time depending on the operating environment. The CISSAN Orchestrator plays a crucial role in achieving reliable automated disaster recovery, which enables the secure reconfiguration of the network to rapidly recover critical services.

For organizations, these capabilities translate to decreased downtime, lower security costs, stronger resiliency of critical services, and better utilization of existing infrastructure and security investments. The ability to dynamically adapt security placement and automate recovery processes helps organizations scale their IoT deployments safely, meet regulatory requirements efficiently, and minimize business risk in highly connected operational environments.

In conclusion, the tools outlined in this report have the potential to provide a solid base for adaptive, distributed, and autonomous security management in large-scale IoT deployments. By supporting interoperable, standard-based, and EU-led security solutions, the project contributes to enhanced European digital sovereignty, independence from external security providers, and sustainability and competitiveness of European digital and industrial infrastructure.

# List of Authors


- Lars-Göran Magnusson, Arctos Labs
- Ilgin Safak, University of Jyväskylä
- Pyry Kotilainen, University of Jyväskylä
- Stella Palenius, University of Jyväskylä
- Mikko Lehkonen, University of Jyväskylä
- Oliver Bölin, Techinova
- Kristian Kratschmer, Techinova

# Table of Contents

# List of Figures

# Abbreviations

| | |
|---|---|
| AI | Artificial Intelligence |
| API | Application Programming Interface |
| GPS | Global Positioning System |
| GUI | Graphical User Interface |
| HTTP | Hypertext Transfer Protocol |
| IoT | Internet of Things |
| IP | Internet Protocol |
| JSON | JavaScript Object Notation |
| mDNS | Multicast Domain Name System |
| MQTT | Message Queueing Telemetry Transport |
| OT | Operational Technology |
| PQC | Post-Quantum Cryptography |
| QoS | Quality of Service |
| REST | Representational State Transfer |
| RTU | Remote Terminal Unit |
| SCADA | Supervisory Control and Data Acquisition |
| SIEM | Security Information and Event Management |
| URL | Uniform Resource Locator |
| VLAN | Virtual Local Area Network |
| WASM | WebAssembly Module |

# Definitions

| | |
|---|---|
| Supervisor | Supervisor software runs on each device, providing a local WebAssembly runtime and interfaces for device monitoring and management. |
| MQTT | A lightweight, publish–subscribe, machine-to-machine network protocol for message queue/message queuing service. |
| mDNS | A computer networking protocol that resolves hostnames to IP addresses within small networks that do not include a local name server. It is a zero-configuration service, using essentially the same programming interfaces, packet formats, and operating semantics as unicast Domain Name System (DNS). |
| Orchestrator | The CISSAN system component that automatically coordinates and manages the optimal security task distribution and response to security incidents and disaster recovery processes in the IoT network to enable collective intelligence and provide continuous security and quick response to service disruptions. Synonymous with orchestration server. |
| WebAssembly | A portable binary-code format and a corresponding text format for executable programs as well as software interfaces for facilitating communication between such programs and their host environment. |
| Solver | A piece of mathematical software which takes problem descriptions in some sort of generic form and calculates their solution. |

# 1 Introduction

The security of IoT networks has become a critical concern with the increasing use of these interconnected systems to support critical services in industrial, infrastructure, and societal sectors. The IoT networks are characterized by their large-scale, highly distributed, and heterogeneous nature. The security challenges in these environments have also increased in terms of their sophistication, speed, and coordination, with these attacks becoming common due to the static nature of traditional security architectures. The ability to provide security in these environments requires novel approaches that can adapt security mechanisms dynamically to meet changing conditions.

This report provides an innovative solution for enabling collective intelligence by introducing an orchestrator and tools that can be used to achieve optimal distribution of security sub-functions in IoT networks through intelligent orchestration. The proposed solution does not rely on centralized security mechanisms but rather on the distributed capabilities of the network that can be coordinated in real-time. The proposed solution enables the IoT network to function as a cooperative system that can identify, respond to, and recover from cyber-attacks using shared intelligence and automated disaster recovery.

## 1.1 Objective of this Document

The objective of this document is to outline and document the tools, mechanisms, and architectural approach that have been created to optimally distribute security sub-functions across an IoT network and enable disaster recovery through collective intelligence. This document outlines, in a detailed and structured fashion, the security and automated disaster recovery orchestrator and all its associated device management, optimization mechanisms, deployment of distributed security functions and automated disaster recovery mechanisms. This document will function as a technical document for all project partners and stakeholders, detailing the implementation of adaptive and distributed security, while also acting as a basis for validation, reuse, and extension of the proposed approach.

## 1.2 Scope and Structure of the Document

The scope of this deliverable is to present the design, implementation, and operation of the orchestration tools designed for the distribution of security sub-functions across the IoT network and for the support of disaster recovery. This document focuses primarily on the architectural aspects, functionality, and operation of the orchestrator, including the interfaces and interactions with device management, optimization, and security enforcement. This document is intended for project partners, system integrators, and interested parties who wish to use this document as a technical reference for the implementation of adaptive and distributed security functions within the IoT network.

The rest of this document is organized as follows. Section 2 describes the system architecture and the role of the orchestrator within the platform. Section 3 explains how the orchestrator performs device discovery and management. Section 4 describes the process for determining the optimal security task allocation. Section 5 describes the normal runtime operation of the orchestrator. Section 6 focuses on the runtime operation during disaster recovery of the orchestrator. Section 7 describes the implementation of the orchestrator including its integration with other CISSAN platform components and the validation of the orchestrator in the CISSAN use cases, limitations, and lessons learned. Section 8 concludes this document and describes the possible extensions.

# 2 CISSAN Orchestrator Architecture

## 2.1 CISSAN Platform Architecture Overview

The CISSAN Platform (see Figure 1) comprise of contributions from the CISSAN partners and acts as a common integration and experimentation platform to implement, integrate, and validate the mechanisms developed within the project. An important target guiding the platform development is the project's strategic objective to facilitate a transition towards more distributed and collaborative system architectures while at the same time support the various use case system representations interconnected within a shared architecture.

The CISSAN deliverable D2.3 report defines the detailed implementation definition of the CISSAN Platform Architecture.

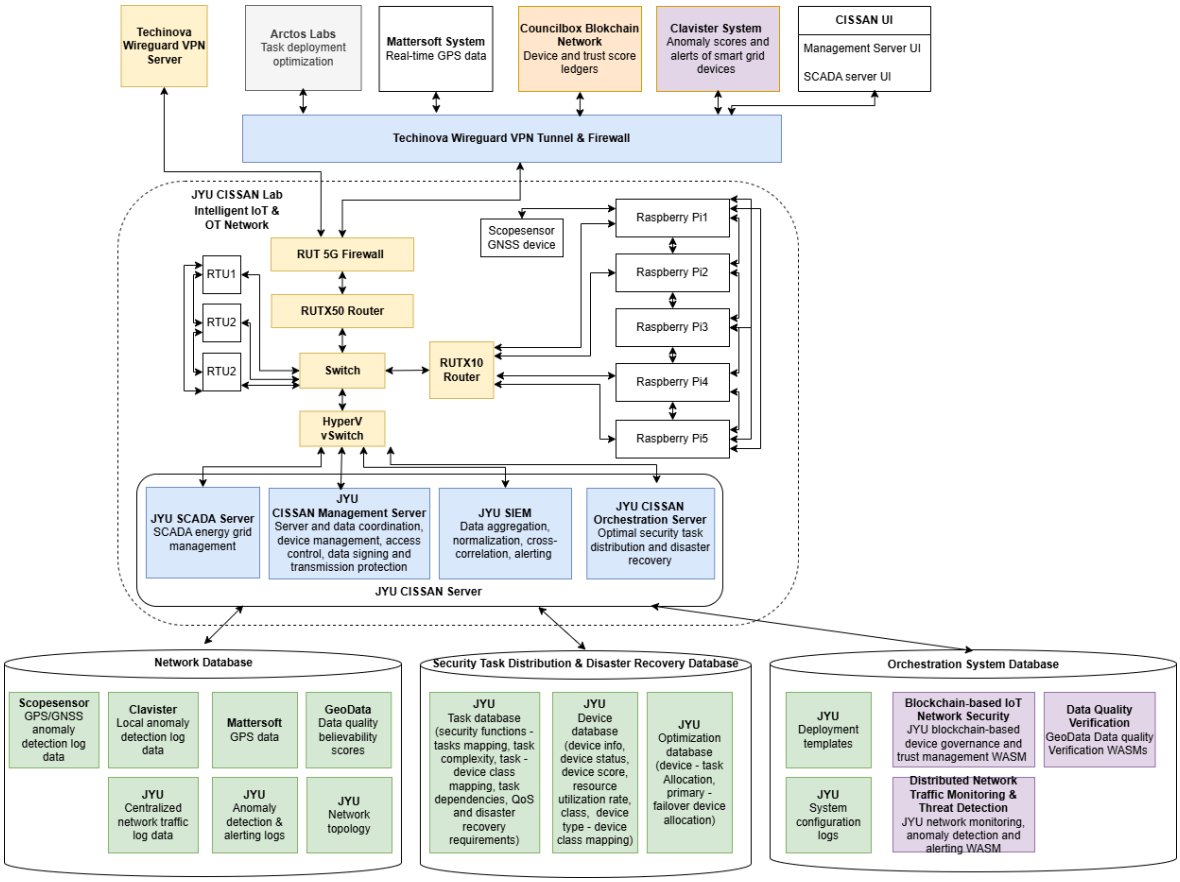


**Figure 1**. CISSAN Platform architecture

Figure 1 depicts the CISSAN Platform, including the CISSAN Orchestrator. The CISSAN Orchestrator belongs to the collective intelligence driven orchestration services and resides on the orchestration server overseeing the distribution of the WebAssembly modules (The target audience of this report is IoT system operators, system integrators, security architects, and decision-makers seeking effective solutions to secure their distributed IoT systems and achieve reliable automated disaster recovery.) containing security functions. The optimal distribution of the WASMs is received from the Arctos Labs Optimization Solver. The CISSAN Orchestrator communicates with the supervisors residing on the Raspberry Pi devices housing and executing the WASMs. The outputs of the modules are fetched by the CISSAN Management Server where they are used for blacklisting purposes and calculating global trust scores of the network. The CISSAN Management Server communicates via Application Programming Interface (API) with the blockchain network where the trust scores and device blacklist status are stored long term.

## 2.2 CISSAN Orchestrator Architecture

The CISSAN Orchestrator developed within CISSAN is a bespoke version of the orchestrator functionality originally developed within the Liquid AI project (project number: 21000057641, project title: 6G Bridge - Liquid AI for 6G software, supported by Business Finland) that supports disaster recovery and optimal task orchestration. The adaptations and extensions developed for CISSAN is the result of joint efforts by CISSAN and Liquid AI project teams.

The CISSAN Orchestrator works in conjunction with the CISSAN Management Server, which are coordination components of the distributed system as opposed to the more traditional concept of a centralized system. This means that the logical control plane is achieved by the coordination components, but the actual execution of the system is distributed across the devices, edge nodes, and domain systems. This avoids the possibility of having single point of failures or reverting to the concept of a centralized system. It is also possible to have multiple instances of such coordination components across different network segments or domains within the system, which can be considered as a federated structure. It is to be noted that, due to practical reasons, only one orchestrator and management server was deployed in the CISSAN platform. However, the deployment of multiple servers as backup and redundancy mechanisms stands out as key areas for future development.

For details on the CISSAN Orchestrator architecture, please refer to the CISSAN deliverable D2.3 report.

## 2.3 Optimization Solver Architecture

The Arctos Labs Optimization Solver assists the CISSAN Management Server by identifying the optimal distribution of tasks across the available devices in a network. The concrete optimization is performed by a 3rd party linear optimization solver component.

Optimal distribution is defined as a balanced decision between risk, performance impact, and network penetration. The decision is made with consideration to hard constraints in the execution environments such as capacity and latency demands, availability of capacity, network topology and network characteristics, combined with task and device characteristics such as risk exposure for a task on a device and performance impact of a task on a device.

The other elements in the Optimization Solver architecture contain functionality to construct, execute, parse, and transform data to and from the model which expresses the optimization problem in a representation supported by the solver component.

The detailed API specification is found in the CISSAN deliverable D6.2 - Annex 3. The Optimization Solver architecture is illustrated in Figure 2.

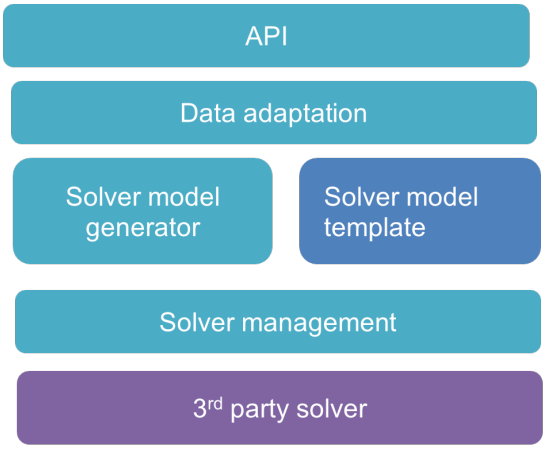


**Figure 2.** Optimization Solver Architecture

# 3 Device Discovery and Management

## 3.1 Device Discovery

The CISSAN Orchestrator maintains dynamic awareness of available network resources through a continuous device discovery process. Device discovery enables the system to detect new devices, track their capabilities, and keep the device registry up to date as devices join, leave, or change their network configuration.

The discovery mechanism is primarily based on multicast Domain Name Service (mDNS). Each supervisor advertises its presence on the local network for the duration of its uptime. The orchestrator continuously listens for these mDNS announcements and reacts to newly discovered or updated devices without requiring manual configuration.

When a previously unknown device is detected, the CISSAN Orchestrator initiates a discovery workflow by querying the supervisor over Hyper Text Transfer Protocol (HTTP). The supervisor provides a standardized, machine-readable device description that provides detailed information about the device's supported WebAssembly host functions, communication interfaces, and execution capabilities.

As part of the initial discovery, the CISSAN Orchestrator also queries the device's health endpoint to verify basic system availability. All collected information, including network addresses, ports, capabilities, and health status, is stored in the CISSAN Orchestrator's device database and becomes immediately available for deployment planning and scheduling decisions.

The device discovery workflow is illustrated in Figure 3.

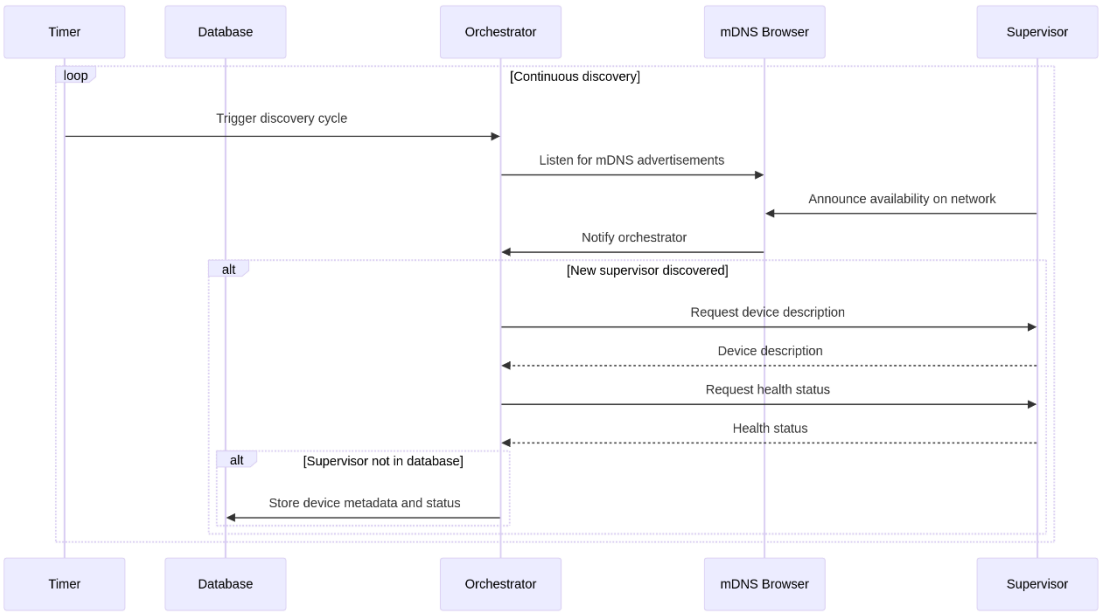


**Figure 3.** Device discovery workflow

## 3.2 Device Management

The CISSAN Orchestrator maintains a database of all discovered devices and continuously monitors their status through periodic health checks. Based on these checks, devices are updated as either active or inactive. Additionally, if the CISSAN Management Server flags a device as blacklisted, the CISSAN Orchestrator accordingly updates the status for the device, preventing it from being used in further deployments. To maintain security integrity, blacklisted devices are retained in the database rather than deleted. This ensures that a blacklisted device cannot be rediscovered as a new node and inadvertently reused in deployments.

Blacklisting on the management server is based on the results of the anomaly detection and trust scoring security functions distributed to the IoT / Operational Technology (OT) devices. The results of these functions and the current device status can be viewed on the management server Graphical User Interface (GUI). Blacklisting is triggered when a device's trust score drops below an acceptable threshold. The GUI also allows the manual blacklisting of devices.

The management server maintains its own device registry, which provides dynamic awareness of network resources. The registry is populated from several sources: devices discovered from the CISSAN Orchestrator API, devices announced in Message Queueing Telemetry Transfer (MQTT) trust messages (e.g. RTUs), and entries in the port isolation allowlist. Only devices in the allowlist are treated as allowed and shown in the CISSAN Management Server GUI. The management server publishes per-device data to MQTT (e.g. allowlist status, registration status, blacklist, and network info) so the GUI can show which devices are present and their status. When a device must be blacklisted, the management server calls the CISSAN Orchestrator's blacklist API so the CISSAN Orchestrator can exclude it from further deployments. Blacklisting on the management server is driven by the results of anomaly detection and trust-scoring functions run on the OT/IoT devices. The trust analyser compares each device's trust score to a configurable threshold; when the score falls below it, the management server triggers blacklisting: it instructs the port blacklister service (e.g. switch control) to block the device's physical port (quarantine Virtual Local Area Network (VLAN)), updates the device registry with blacklisted status and reason, and notifies the CISSAN Orchestrator via its blacklist API. The same flow is used when an operator manually blacklists a device from the GUI. The GUI sends an MQTT command to block a device; the management server subscribes to that topic, calls the port blacklister, updates the registry, and calls the CISSAN Orchestrator blacklist API. Device status, trust scores, anomaly counts, believability, and blacklist state is visible on the GUI because the management server publishes these to MQTT and the GUI receives them via the socket bridge. The GUI also supports manual add device, remove device, and unblock if its blacklisted. See Figure 4 for a snapshot of CISSAN-AI GUI.


**Figure 4.** CISSAN-AI GUI

# 4 Optimal Task Distribution

## 4.1 Optimal task allocation

The CISSAN Orchestrator works together with the CISSAN Management Server and the Arctos Labs Optimization Solver in obtaining the optimal security task allocation to primary and failover devices. The following supported optimization techniques are utilized for optimization solver.

**CISSAN Orchestrator, CISSAN Management Server, and Arctos Labs Optimization Solver Interwork**

The purpose with optimal task allocation is to identify which network devices that should host which tasks to 1) minimize overall risk for the tasks, 2) to minimize performance impact from the tasks whilst 3) adhering to constraints regarding aspects such as task dependencies, disaster recovery requirements, quality of service (QoS) and device capacities.

The CISSAN Management Server acts as the coordinating entity for the optimal task allocation process. In a preparation phase, the CISSAN Management Server first collects the necessary data on tasks and network from the CISSAN Orchestrator and injects all relevant supplementing data describing characteristics on task- and network entities to the Arctos Labs Optimization Solver.

The CISSAN Management Server then queries the Arctos Labs Optimization Solver for the optimal distribution for the set of tasks to be deployed. The query is provided in the form of a list of tasks subject to network deployment.

Upon reception of the query, the Arctos Labs Optimization Solver uses the supplementing data in combination with the tasks data and constructs a model tailored for the internal solver component. The Arctos Labs Optimization Solver initiates background execution of the model in the internal solver and returns a designated API resource for the CISSAN Management Server where the result in time will be made available. When the internal solver component has finished and produced an optimal task allocation result, the Arctos Labs Optimization Solver will transform the outcome into a data structure which for each task maps the tasks to a primary device and, depending on task criticality, a selected number of failover devices. Said data structure will also contain data on the scoring for risk, performance impact, and distribution as computed by the internal solver component. Once properly assembled, the data is associated with the designated API resource.

The CISSAN Management Server busy waits on the designated API resource until the Arctos Labs Optimization Solver provides a solution. A status indicator on the API resource informs the CISSAN Management Server of the present state of the task allocation query.

When the optimal task deployment allocation is available, the CISSAN Management Server will order deployment of the task from the CISSAN Orchestrator as described in the accompanying *Deployment Template*. The CISSAN Orchestrator will, upon reception of the order, conduct distribution of the corresponding WASMs to the designated devices in the network.

Figure 5 depicts the interwork between the CISSAN Management Server, the CISSAN Orchestrator, and the Arctos Labs Optimization Solver.

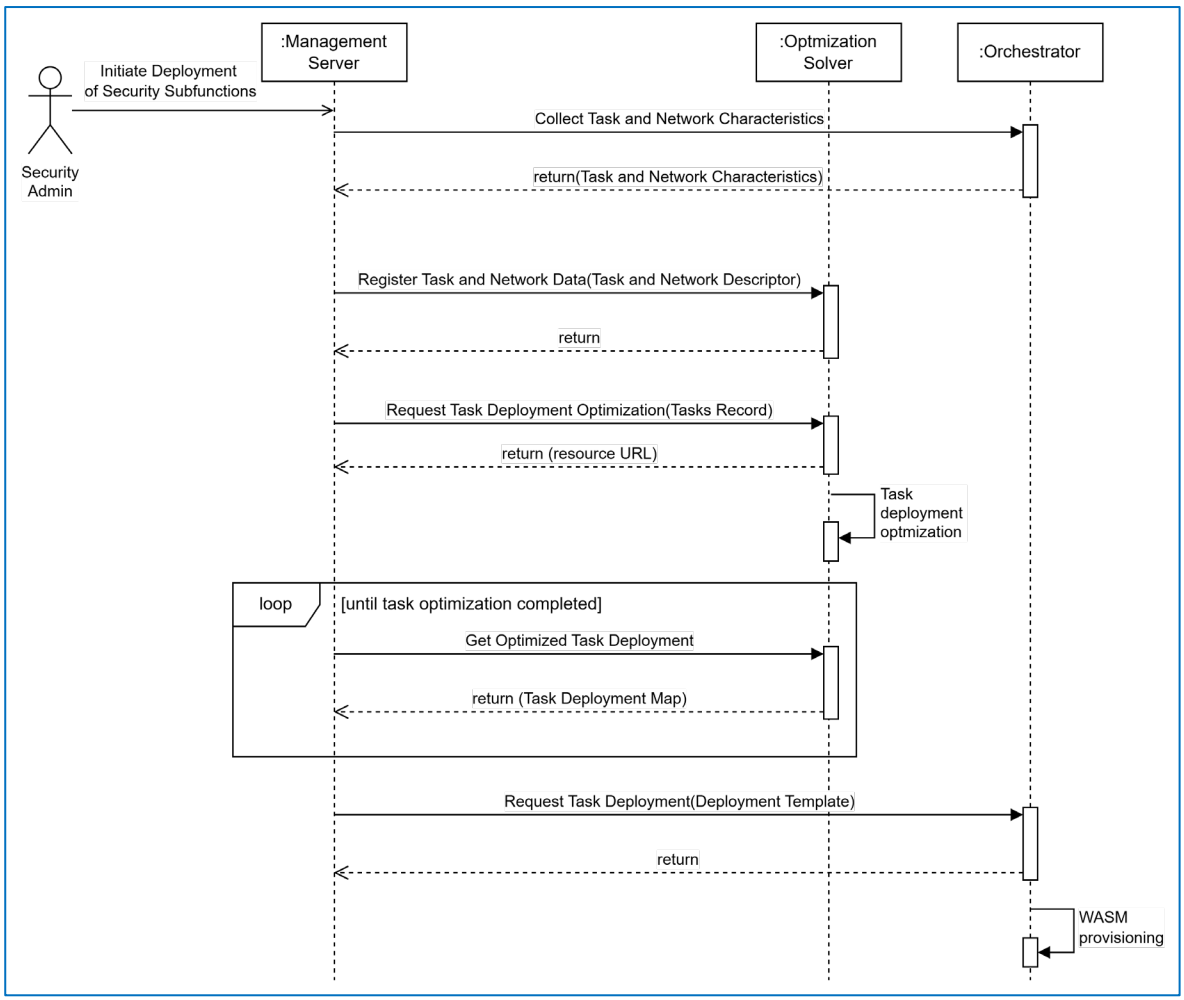


**Figure 5.** CISSAN Management Server, CISSAN Orchestrator and Arctos Labs Optimization Solver interwork

### Optimization techniques in the Arctos Labs Optimization Solver

The Arctos Labs Optimization Solver uses linear optimization (also referred to as linear programming) as the mechanism through which the optimal task allocation is determined. In linear optimization, a problem is modelled as a set of linear relationships. The purpose is often to minimize what is known as the objective function given by a set of constraints. The constraints are also linear and consist of both linear equalities and linear inequalities.

These linear relationships are known within the Arctos Labs Optimization Solver only. The data provided by the CISSAN Management Server through a) the Supplementing Task & Network Data and b) the Task Deployment Template will in some cases contain readily available parameter values for the linear relationships but in other cases there is a need to process provided data in various steps to determine parameter values. As a general principle, the system is designed such that the CISSAN Management Server as far as possible is unaware of and agnostic about the way the Arctos Labs Optimization Solver represents the optimization problem. The objective function for the task allocation problem expresses a trade-off between risk, performance impact, and network penetration. The constraints describe requirements on properties like task and device capacity, latencies for processing as well as inter-task communication and task criticality.

Another significant property of the Arctos Labs Optimization Solver implementation is the code generator for the representation of the linear optimization used by the solver component and the balance this has between which parts of the problem that is handled by the imperative domain (i.e. in Python) and which parts of the problem that are handled by the linear optimization domain of the Arctos Labs Optimization Solver. The linear optimization is expressed in the Minizinc language which is a high-level solver independent constraint modelling language. As for all programming languages,

the language constructs available are efficient both from a design as well as a run-time perspective. In this case, the solver characteristics are also important. The design of the Optimization Solver is therefore made such that some aspects, even though possible to express and process within the linear optimization solver, are in fact done outside the solver in the imperative domain to improve overall efficiency.

The CISSAN Orchestrator supports collective intelligence by distributing four different WASMs across the network: *TrustScoring*, *AnomalyDetection*, and Data Quality Verification, which is separated to two different modules: *DTW* (dynamic time warping) and *VB* (variance-based) modules.

The *AnomalyDetection* module analyses data retrieved from the Security Information and Event Management (SIEM) server using anomaly detection methods and is explained in CISSAN deliverable report D5.4. Data Quality Verification modules generate believability scores of data, which are also explained in the CISSAN deliverable report D5.4.

The *TrustScoring* module calculates trust scores for peer devices based on device responsiveness and the results of the *AnomalyDetection* and Data Quality Verification modules. The derived trust scores are shared with the CISSAN Management Server which performs access control, device blacklisting and global trust scoring of the network. The details for the *TrustScoring* module can be found in the CISSAN deliverable reports D5.3 and D5.4.

## 4.2 Provisioning of tasks

During the provisioning phase, the CISSAN Orchestrator is responsible for preparing and distributing all WASMs required by a deployment. Before modules can be provisioned to devices, they must first be uploaded to the CISSAN Orchestrator, where they are stored in the package manager. Provisioning the tasks is part of the deployment process, which is triggered when the actor requests a specific functionality from the CISSAN Orchestrator. At this point, the actor submits a deployment manifest, a high-level specification of the desired application workflow. The manifest defines the sequence of functions that the actor wants to execute. Optionally, the actor can also specify devices on which some or all functions should run. If specific devices are not specified, the CISSAN Orchestrator selects suitable devices based on their capabilities.

The CISSAN Orchestrator then produces a deployment solution, which maps the functions defined in the manifest to the selected or chosen devices. The solution also determines which software modules and assets each device requires, including WASMs, configuration files, or machine learning models. From the deployment solution, the CISSAN Orchestrator generates device-specific deployment instructions, detailing how each device should stage modules, set up its execution environment, and expose endpoints for function execution and chaining.

Once the devices receive their deployment instructions, they retrieve the required software modules and assets from the CISSAN Orchestrator’s package manager, which maintains a registry of all available modules. The devices then stage the modules in their WebAssembly runtimes and install the endpoints necessary for executing the assigned functions.

## 4.3 Execution of tasks

During the execution phase, the CISSAN Orchestrator initiates the application by invoking the starting function defined in the deployment solution**,** by sending an HTTP request to the device. This function is the entry point of the function sequence and is executed on the device responsible for the first stage of the workflow.

To support asynchronous execution, the device immediately responds with a resultUrl, allowing the CISSAN Orchestrator to continue monitoring the execution without blocking. The device then executes the assigned function, and if the function chain is not complete, it invokes the next function on another device, following the sequence defined in the solution. Once the final function in the chain completes, the execution result is available and returned to the actor via the CISSAN Orchestrator. The execution is represented as part of the diagram in Section 5.

The CISSAN Management Server does not use the CISSAN Orchestrator’s execute endpoint. It drives execution using an internal task ledger (current deployment and task-to-device assignments) and a periodic task executor that runs in a background loop (e.g. every 60 seconds). If automatic execution is enabled, it iterates over the ledger and, for each device, starts execution in a separate

thread. It resolves each device's Internet Protocol (IP) and port from the device registry and then performs direct execution on the device via HTTP. The CISSAN Management Server is therefore the actor that triggers execution; the CISSAN Orchestrator is used for deployment metadata and for deploying WASMs to devices, not for invoking runs.

The CISSAN Management Server sends an HTTP request directly to the Supervisor on each device to execute WASM functions. For functions that need input files it uses POST with multipart form data; otherwise GET with query parameters. Tasks on a single device run sequentially; different devices are run in parallel in separate threads. The device responds with JavaScript Object Notation (JSON) containing a Uniform Resource Locator (URL). The management server then polls that URL every 30 seconds up to 300 seconds until the response is 200 and includes an outputs array, then downloads those output URLs and parses them. If needed, it can also fetch known output paths directly. The CISSAN Management Server orchestrates the workflow: it uses the ledger to know which device runs which function(s), prepares input files for dependent tasks, and issues the next HTTP request to the right device. It does not rely on the device to call the next function on another device. After retrieving and parsing outputs, it computes per-device scores and publishes them to MQTT.

## 4.4 Data coordination and routing

The CISSAN Orchestrator plays the leading role in distributing the deployment templates to the participating devices. These templates contain the full configuration needed for each module, including the predefined endpoint addresses through which the devices communicate with one another. When the CISSAN Orchestrator pushes the template to a device, the module deployed on that device automatically receives the routing and endpoint information it needs to operate.

These endpoint definitions form the basis for execution: each module knows exactly where to send its outputs, and the devices can exchange results directly with one another without requiring additional discovery mechanisms. The deployment template therefore not only describes the module placement but also embeds the communication structure of the system. This ensures that, once the deployment becomes activated, all modules follow a consistent routing logic and can immediately begin coordinated execution.

From the CISSAN Management Server's perspective, routing is determined by the deployment template: at any time, each device has a fixed set of tasks and a known role in the workflow. The management server does not perform runtime discovery of endpoints except asking the CISSAN Orchestrator for new devices or checking latest information from Rotor. It builds a global view (devices, modules, constraints), calls the Arctos Labs Optimization Solver to get an assignment, and turns that into a deployment manifest (sequence and per-device assignments). The CISSAN Orchestrator then distributes that deployment template to the devices, so the routing is static for the lifetime of that deployment, each module knows where it fits and, from the deployment template, how to reach what it needs. Routing changes when the management server produces and deploys a new deployment template. That happens when the CISSAN Management Server's view of the network or policy changes: when new devices are discovered, when devices go offline or are blacklisted, or when the operator triggers a re-run of the deployed tasks. The CISSAN Management Server then runs the optimization pipeline again, gets a new solution from the optimization solver, creates a new deployment, possibly with a different task placement, and pushes it through the CISSAN Orchestrator. After that, routing is again determined until the next such update. So routing is deployment template-driven and centralized on the management server: the management server decides who does what and with whom and the devices execute according to the deployment template they receive and do not negotiate routes at runtime.

# 5 Normal Runtime Operation of the CISSAN Orchestrator

Under normal operating conditions, an actor can request functionalities from the CISSAN Orchestrator, which initiates the deployment phase as described in Section 4.2. The actor may also request the execution of existing deployments, as detailed in Section 4.3. The process of creating and executing a deployment under normal runtime conditions is illustrated in Figure 6.

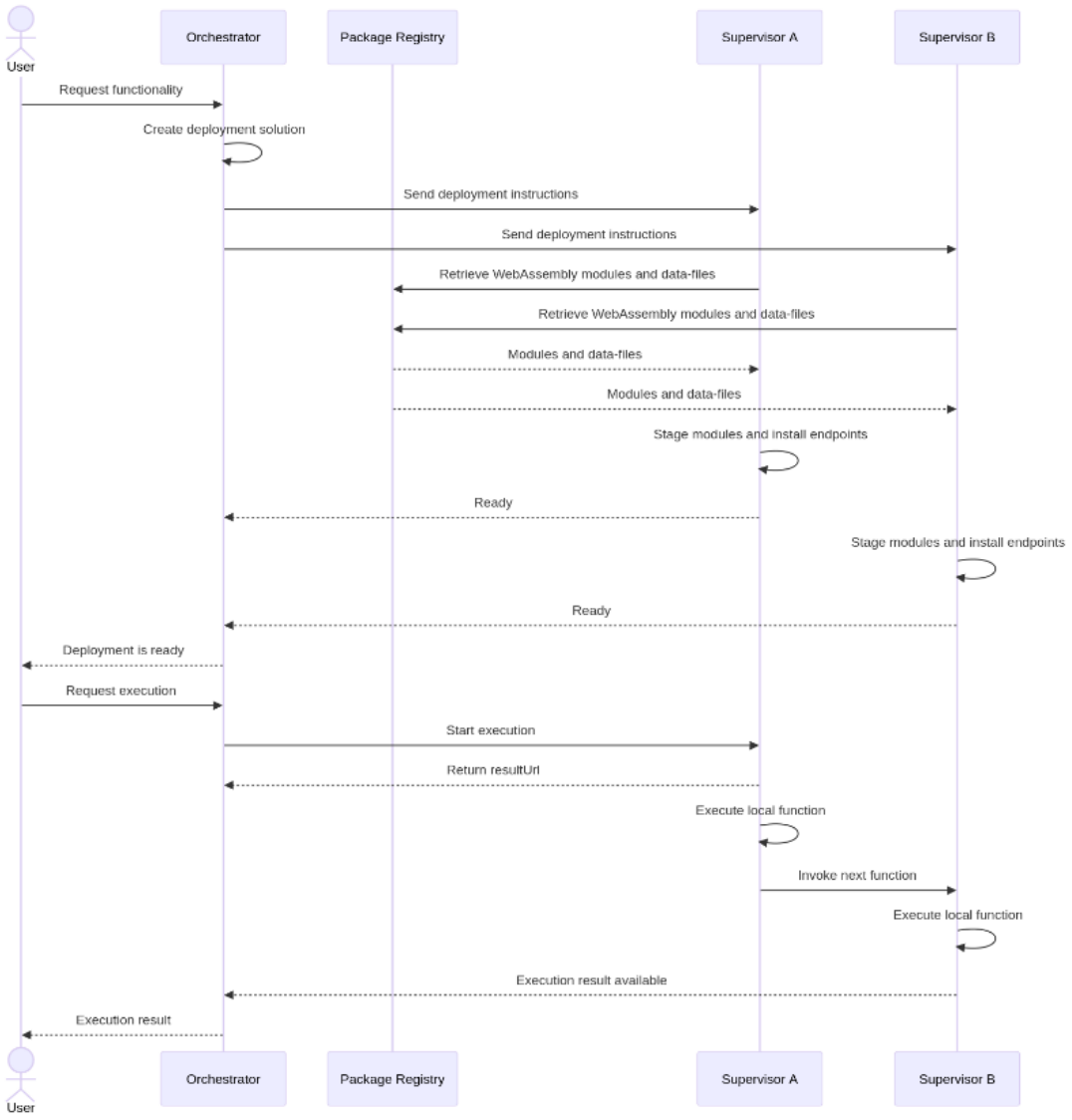


**Figure 6**. Process of creating and executing a deployment under normal runtime conditions

In addition to the deployment and execution logic, the CISSAN Orchestrator performs continuous health monitoring by periodically issuing HTTP-based health check requests to all registered supervisors. Each health check is implemented as an HTTP GET request targeting a device-specific health endpoint. Devices may expose multiple network addresses, and the CISSAN Orchestrator iterates through these addresses until a successful response is received.

A health check is considered successful if the device responds with an HTTP 200 (OK) status code. If the request fails or returns a non-200 status, the attempt is recorded as a failed health check. When all available addresses associated with a device fail to respond, the health check is considered unsuccessful. The CISSAN Orchestrator maintains a failure counter for each device, resetting the counter upon a successful response and incrementing it on failure. If a device fails to respond to five consecutive health checks, it is marked as inactive, and the disaster recovery process (described in Section 6) is triggered. This state transition is logged with a timestamp to ensure to provide an

auditable history of device availability. Figure 7 illustrates the decision logic for a periodic health check process.

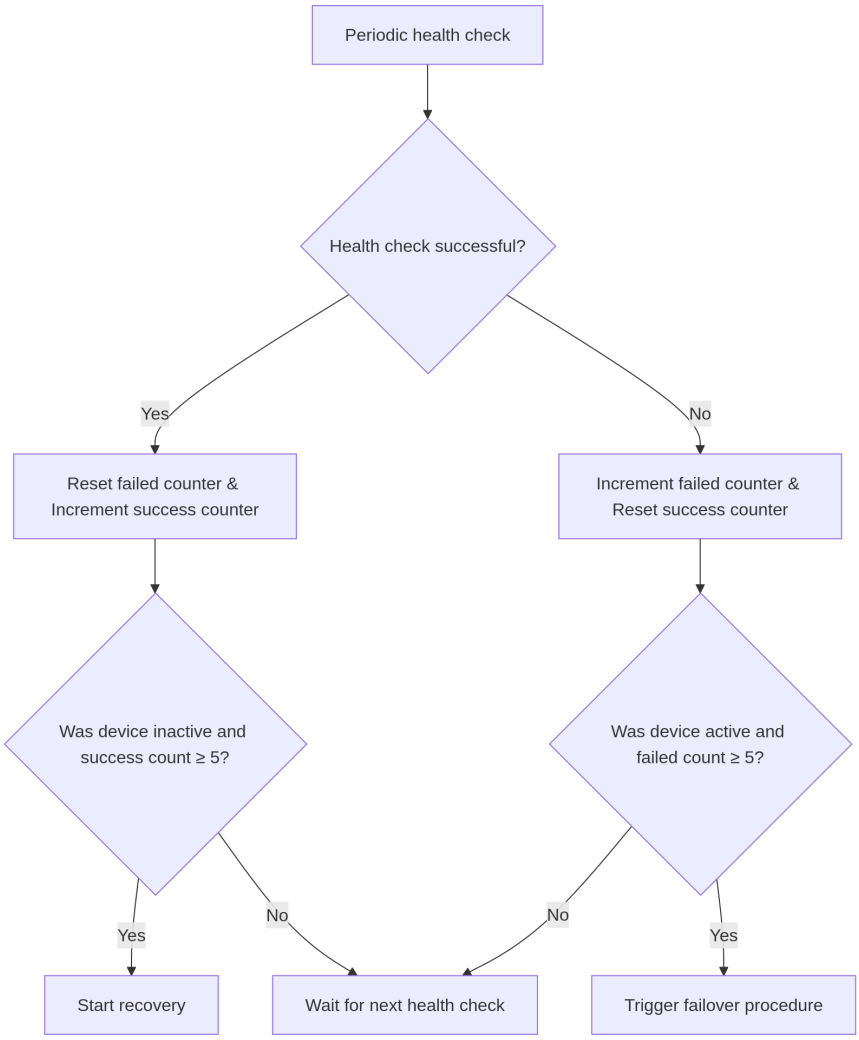


**Figure 7.** Decision log for periodic health check process

# 6 Disaster Recovery Runtime Operation of the CISSAN Orchestrator

Once a device is declared inactive, the CISSAN Orchestrator evaluates the impact of the failure by identifying all deployments that depend on the affected device. For each impacted deployment, the CISSAN Orchestrator initiates a failover procedure. Failover candidates are pre-defined as part of the deployment configuration provided by the Arctos Labs Optimization Solver, allowing each step in the execution sequence to be associated with one or more alternative devices capable of performing the same function.

To mitigate the impact of a device failure, the CISSAN Orchestrator updates the deployment by replacing the inactive device with the first available active failover device for the affected execution steps. The deployment is then marked as operating in failover mode, and the CISSAN Orchestrator redeploys the updated deployment instructions to the relevant devices, see Figure 8.

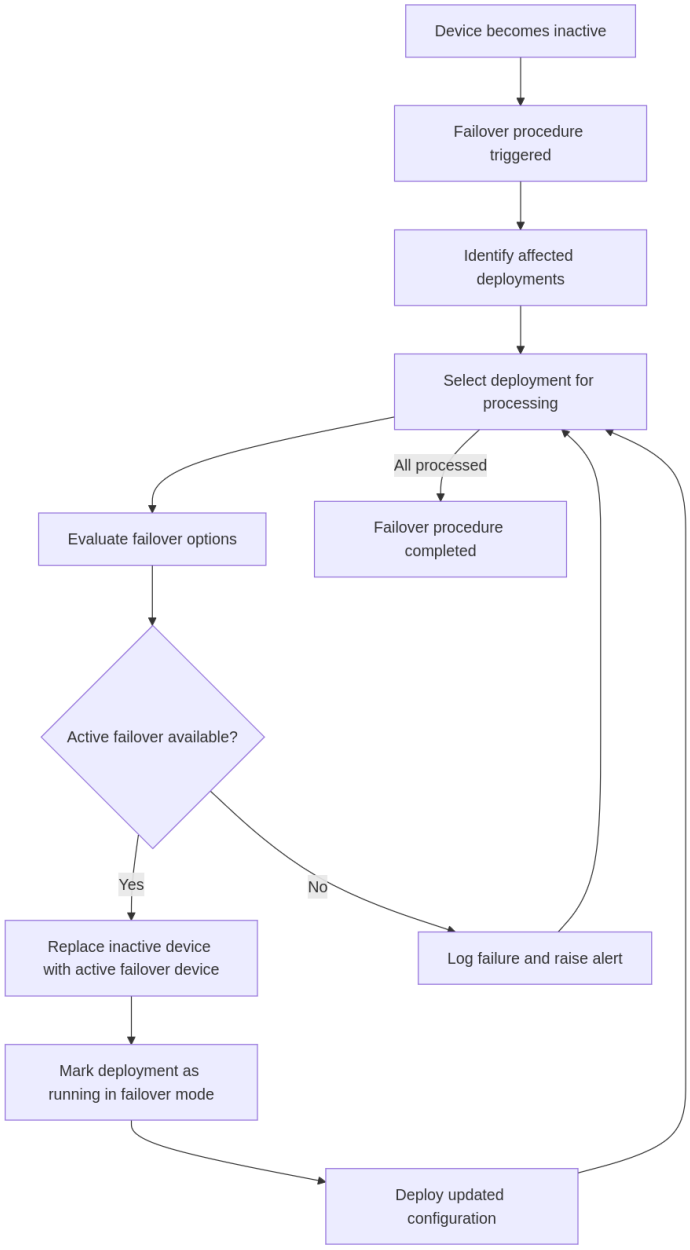


**Figure 8**. CISSAN Orchestrator failover handling process

In addition to failure handling, the CISSAN Orchestrator supports automatic recovery when a previously inactive device becomes healthy again, see Figure 9. When a previously inactive device

responds successfully to five consecutive health checks and reaches the configured success threshold, its status is transitioned back to active, making it available again for deployment and execution.

Upon recovery, the CISSAN Orchestrator identifies deployments that are currently operating in failover mode and for which the recovering device was part of the original deployment plan. For each such deployment, the CISSAN Orchestrator restores the recovered device to its original position in the execution sequence, replacing the temporary failover device. The deployment is then updated in the database, its failover status is cleared, and the restored deployment instructions are redeployed to the devices.

Finally, the device's status is transitioned back to active, and the state change is recorded. This recovery mechanism allows the system to automatically return to its intended execution topology once the failure condition has been resolved, without requiring manual intervention.

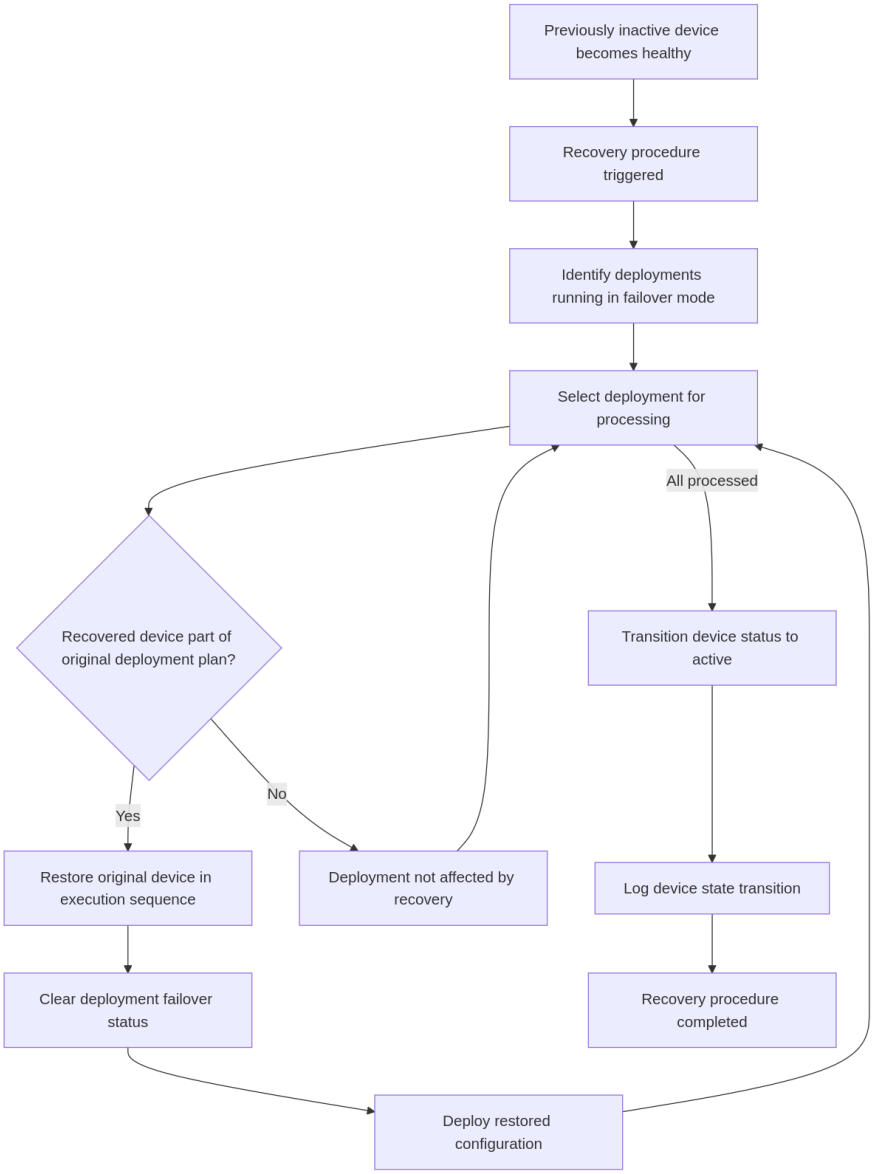


**Figure 9.** CISSAN Orchestrator failover recovery process

# 7 Implementation of the CISSAN Orchestrator

## 7.1 Implementation of the CISSAN Orchestrator

The CISSAN Orchestrator was implemented as a standalone software component in JavaScript and deployed in the CISSAN lab as a Docker container on an Ubuntu Server. Persistent data is stored in a MongoDB instance, with separate collections for devices, deployments, and modules, corresponding to the device database, deployment registry, and package manager. The CISSAN Orchestrator exposes a Representational State Transfer (REST)-based API through which an actor can submit deployment requests and trigger application execution.

Edge devices consisted of five Raspberry Pi units from two hardware generations, each running a supervisor component implemented in Rust. The supervisors host lightweight WebAssembly runtimes and communicate with the CISSAN Orchestrator over HTTP. WebAssembly was chosen as the runtime environment for application modules because it enables execution of code compiled from multiple programming languages in a safe, sandboxed environment. This allows the same modules to run on heterogeneous hardware without modification.

## 7.2 Validation of the CISSAN Orchestrator in the CISSAN Use Cases

Orchestration and task distribution form a common foundation for multiple use cases, as the correct placement and execution of tasks is a prerequisite for their operation. The CISSAN Orchestrator plays a leading role in ensuring that the CISSAN platform remains operational when failures occur through its disaster recovery capabilities, making the disaster recovery relevant to all the use cases.

Data Quality Verification modules are relevant for believability score calculation of Use Case 3, tunnelling, and construction. The *AnomalyDetection* module is used to detect anomalies from the Global Positioning System (GPS) data in Use Case 1, transportation. The *TrustScoring* module is relevant to all use cases for calculating the local trust scores. Furthermore, since Use Case 5 demonstrates the elements of the other use cases working in tandem with each other, the CISSAN Orchestrator plays an important role. Without the CISSAN Orchestrator distributing the different modules to the appropriate devices, the modules could not be executed, and the system would not be able to produce the results required to secure the network and enable coordinated operation.

## 7.3 Limitations and Lessons Learned

Even though the CISSAN Orchestrator was successful in demonstrating its advantages in terms of robustness, scalability, ensuring business continuity and enabling collective intelligence, there are some limitations identified during its implementation. First, the complexity involved in the integration of heterogeneous systems with different capabilities, protocols, and maturity levels was identified as one of the limitations with the proposed system. The seamless integration of the distributed security sub-functions was also challenging in terms of the execution of the distributed system. Even though the CISSAN Orchestrator and Arctos Labs Optimization Solver are effective in the distribution of security tasks in the system, the performance of the system depends on the availability of system-level information.

Another limitation identified was the deployment architecture and the level of decentralization achieved with the proposed system. Even though the CISSAN Management Server and the CISSAN Orchestrator components of the system are effective in the management of the distributed system, the deployment of the system was done with only one instance of the CISSAN Management Server and the CISSAN Orchestrator components of the system during the project implementation due to practical reasons. However, the level of decentralization achieved with the proposed system can be improved by having multiple instances of the Management Server and the CISSAN Orchestrator components deployed across different network segments or domains to avoid the possibility of bottlenecks and single points of failure in the system.

One key limitation is that failover devices alone are insufficient to guarantee uninterrupted execution: if a device fails during computation, its in-flight data and intermediate outputs are lost, and the downstream execution chain cannot continue even though a failover device exists. Since activating

a failover node requires detection, reallocation, and startup time, the ongoing execution will typically fail before the failover node becomes operational. This highlights the need for additional mechanisms such as state replication, checkpointing, or warm-standby devices.

In addition, some of the advanced and future-proof technologies were not fully addressed. In particular, the integration of post-quantum cryptography (PQC) was recognized as an important feature of future-proofing the CISSAN platform (see CISSAN D1.2 and D7.1 Standardization Action Plan reports for details), even though only a preliminary analysis was performed as part of this project, with the actual integration left as future work post-project.

Based on the experiences gathered during the implementation of the project, several key lessons have been learned. The phased approach to decentralization is essential when developing a system that is likely to find its way into real-world applications, especially when the application is as critical as the security of the system itself. The standardized data model and communication protocol are crucial when developing a system that is likely to find its way into real-world applications, especially when the application is as critical as the security of the system itself. The importance of cooperation cannot be overstated, especially when developing a system that is likely to find its way into real-world applications, especially when the application is as critical as the security of the system itself. Security by design, privacy by design, and zero trust principles are crucial when developing a system that is likely to find its way into real-world applications, especially when the application is as critical as the security of the system itself. In particular, the coordination of device workloads must be designed with security in mind: access control, integrity of deployment artifacts, and secure channels. A single, well-defined execution contract reduces complexity and avoids ad hoc access to devices. Publishing clear, per-device metrics over a standard interface helps operators and other systems reason about “who is trusted” and “who is anomalous.” Blacklisting that retains but excludes devices avoids bad or untrusted devices being rediscovered and reused. Threshold-based automation (e.g., using rules-based mechanisms such as “if trust score is below threshold, then blacklist”, or dynamic baselining methods) is straightforward to implement but needs tuning and recovery procedures so that operators can correct mistakes without requiring deep access to the system. Merging the CISSAN Orchestrator and the CISSAN Management Server into a single server may help reduce the possibility of failures.

# 8 Conclusions

This report describes the approach for the distribution of security sub-functions across the network for enabling collective intelligence in IoT and OT systems for the purpose of enhancing the overall network security. Through the delegation and distribution of detection, analysis, and response capabilities from centralized systems to the edge, CISSAN demonstrates a more robust and adaptable security model.

As an important aspect of the approach, the project has also created a security task and disaster recovery orchestrator, in collaboration with the Liquid AI project, that allows the coordinated execution of security and disaster recovery functions across the network. The CISSAN Orchestrator, working in conjunction with the CISSAN Management Server and supported by the Arctos Labs Optimization Solver, determines the optimal distribution of tasks across the network, taking into consideration the capabilities of the devices, the dependency of the tasks, quality of service, and disaster recovery, among others, in order to enable the distribution of security sub-functions across the network, thereby enabling collective intelligence in cybersecurity, as well as the execution of automated and distributed disaster recovery functions across the network.

The proposed approach has been successfully validated using the CISSAN platform through the joint use case demonstration, where several systems were used for detecting, sharing, and responding to security-related events. Although the validation was performed using one instance of the CISSAN Management Server and CISSAN Orchestrator, the results obtained demonstrate the viability of the proposed approach and indicate that it has the potential for further scaling towards a completely decentralized implementation. Future deployments of this approach are expected to use several instances of these coordination-related services implemented across different network segments, thus allowing for greater robustness, removing possible bottlenecks, and further enhancing the decentralized concept of this approach.

The CISSAN Orchestrator enables a practical approach for migrating traditional centralized systems towards a more distributed and collaborative paradigm without requiring disruptive changes. By facilitating collective intelligence and automated task and disaster recovery coordination between systems, the proposed approach enables better cybersecurity, business continuity, and resilience for critical infrastructures. Thus, it contributes to the strengthening of digital sovereignty in Europe and enables the development of new-generation cybersecurity solutions for interconnected and mission-critical environments.