

|  |  |
| --- | --- |
|  | **Horizon Europe Framework Programme**  HORIZON JU Research and Innovation Action  Reliable Services and Smart Security |



**Efficient, portabLe And Secure orchesTration for reliable servICes**

**Internal deliverable: Hardware Abstraction Layer†**

**Abstract**: Oratio n

|  |  |
| --- | --- |
| Date | 10/01/2024 |
| Contributors | Christian Gehrmann (LU)  Anthony Ferrari (THD)  Gunn Lachlan (AAL)  Dušan Borovčanin (UVC) |
| Internal Reviewers |  |

**The *ELASTIC* Consortium**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Part. No.** | **Participant organisation name** | **Participant Short Name** | **Role** | **Country** |
| 1 | POLYTECHNEIO KRITIS | TUC | Coordinator | EL |
| 2 | ERICSSON AB | ERS | Principal Contractor | SE |
| 3 | OY L M ERICSSON AB | ERF | Principal Contractor | FI |
| 4 | TELEFONICA INNOVACION DIGITAL SL | TID | Principal Contractor | ES |
| 5 | THALES SIX GTS FRANCE SAS | THS | Principal Contractor | FR |
| 6 | THALES DIS FRANCE SAS | THD | Principal Contractor | FR |
| 7 | INTERUNIVERSITAIR MICRO-ELECTRONICA CENTRUM | IME | Principal Contractor | BE |
| 8 | ULTRAVIOLET CONSULT DOO | UVC | Principal Contractor | RS |
| 9 | AALTO KORKEAKOULUSAATIO SR | AAL | Principal Contractor | FI |
| 10 | LUNDS UNIVERSITET | LUN | Principal Contractor | SE |
| 11 | ABSTRACT MACHINES SAS | AMA | Principal Contractor | FR |
| 12 | PRIVREDNO DRUSTVO ZENTRIX LAB DRUSTVO SA OGRANICENOM ODGOVORNOSCU PANCEVO | ZEN | Principal Contractor | RS |
| 13 | POLITECNICO DI TORINO | POLITO | Principal Contractor | IT |

**Document Revisions & Quality Assurance**

**Internal Reviewers**

1. *Reviewer 1, Beneficiary short name*

**Revisions**

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Date** | **By** | **Overview** |
| 0.1 | 20/05/2023 | Christian Gehrmann | TOC |
| 0.2 | 18/06/2024 | Christian  Gehrmann | First draft |
| 0.3 | 30/08/2024 | Christian Gehrmann | Second draft |
| 0.35 | 18/92024 | Christian  Gehrmann | Third draft |
| 0.36 | 25/9/2024 | Lachlan Gunn and Christian Gehrmann | Fourth draft |
| 0.4 | 25/11/2024 | Christian Gehrmann | Fifth draft with requirements included |
| 0.45 | 3/12/2024 | C. Gehrmann, Brent Mccredie, Anthony Ferrari, Merjlin Sebrecht, Gunn Lachland | Sixth draft with comments input on TEE HAL 0.4 incorporated as well as a first complete version of the architecture included. |
| 0.46 | 5/12/2024 | C. Gehrmann | Update with first interface function which will be used as template for all functions. |
| 0.48 | 10/1/2025 | C.Gehrmann, Ducan Borovcanin, Anthony Ferrari | First complete draft with a full interface function specification. |
| 0.50 | 24/1/2025 | C. Gehrmann | Final draft with review input from all contributors taken into account. |

**Disclaimer**

*The work described in this document has been conducted within the ELASTIC project. This project has received funding from the European Union’s Horizon Europe research and innovation programme under grant agreement No 101139067. This document does not reflect the opinion of the European Union, and the European Union is not responsible for any use that might be made of the information contained therein.*

*This document contains information that is proprietary to the ELASTIC Consortium partners. Neither this document nor the information contained herein shall be used, duplicated, or communicated by any means to any third party, in whole or in parts, except with prior written consent of the ELASTIC Consortium.*

Table of Contents

[List of Tables 6](#_Toc188622261)

[List of Figures 7](#_Toc188622262)

[List of Abbreviations 8](#_Toc188622263)

[Executive Summary 9](#_Toc188622264)

[1 Introduction 10](#_Toc188622265)

[1.1 Purpose and Scope of the Document 10](#_Toc188622266)

[1.2 Relation to Work Packages, Deliverables and Activities 10](#_Toc188622267)

[1.3 Contribution to WP3 and Project Objectives 10](#_Toc188622268)

[1.4 Structure of the Document 10](#_Toc188622269)

[2 Deployment Scenarios 11](#_Toc188622270)

[2.1 Scenario selection principle 11](#_Toc188622271)

[2.2 The ELASTIC HAL scenarios 11](#_Toc188622272)

[3 The HAL requirements 23](#_Toc188622273)

[3.1 Requirements Derivation Principle 23](#_Toc188622274)

[3.2 Requirements 23](#_Toc188622275)

[4 The HAL architecture 32](#_Toc188622276)

[4.1 Architecture Overview 32](#_Toc188622277)

[4.2 Architecture Components 33](#_Toc188622278)

[4.3 Implementation Considerations 33](#_Toc188622279)

[5 Interface Specification 34](#_Toc188622280)

[5.1 Interface Operation Principles 34](#_Toc188622281)

[5.2 Interfaces 34](#_Toc188622282)

[6 References 41](#_Toc188622283)

# List of Tables

[Table 1:Entities in the smart manufacturing deployments scenario 12](#_Toc187651085)

[Table 2:Entities in the IT services cloud migration deployment scenario. 17](#_Toc187651086)

[Table 3: Entities in the proactive maintenance deployment scenario. 19](#_Toc187651087)

[Table 4: Required functions identified from the smart manufacturing deployment case. 24](#_Toc187651088)

[Table 5: IT Service Cloud Migration functions. 26](#_Toc187651089)

[Table 6: Proactive Maintenance functions. 27](#_Toc187651090)

[Table 7: HAL Capabilities function. 30](#_Toc187651091)

[Table 8: The Clock Interface. 34](#_Toc187651092)

[Table 9:The Random Interface. 35](#_Toc187651093)

[Table 10: The Object Storage Interface. 35](#_Toc187651094)

[Table 11: The Sockets Interface. 36](#_Toc187651095)

[Table 12: The Cryptography Interface. 37](#_Toc187651096)

[Table 13: GPU Interface. 38](#_Toc187651097)

[Table 14: The Resource Allocation Interface. 38](#_Toc187651098)

[Table 15: The Event Handling Interface. 39](#_Toc187651099)

[Table 16: The Protected Internal Communication Interface. 39](#_Toc187651100)

[Table 17: HAL Platform Capabilities Interface. 40](#_Toc187651101)

# List of Figures

[Figure 1:Analytics smart manufacturing deployment scenario. 13](#_Toc187424011)

[Figure 2: Edge control smart manufacturing deployment scenario. 15](#_Toc187424012)

[Figure 3: IT migration deployment scenario. 18](#_Toc187424013)

[Figure 4: The proactive maintenance deployment scenario. 21](#_Toc187424014)

[Figure 5: The TEE HAL architecture. 32](#_Toc187424015)

# List of Abbreviations

**CCA** Confidential Compute Architecture

**DTLS** Datagram TLS

**ECU** Electronic Control Unit

**HAL** Hardware Abstraction Layer

**MAC** Message Authentication Code

**TCP** Transmission Control Protocol

**TLS** Transport Layer Security

**TEE** Trusted Execution Environment

**GPU** Graphical Processing Unit

**SEV** Secure Encrypted Virtualization

**UDP** User Datagram Protocol

**TDX** Trust Domain Extension

**WASI**  WebAssembly System Interface

# Executive Summary

This document contains the specifications for the Hardware Abstraction Layer (HAL) for confidential computing as defined by the ELASTIC project. The document specifies the TEE HAL and introduces the deployment scenarios for the HAL as defined by ELASTIC. This specification serves as the basis for the complete confidential computing framework within the project.

Requirements for the TEE HAL are derived based on the provided deployment scenarios. Next, interfaces on an abstract level fulfilling the requirements are listed. Detailed implementation recommendations will be made in the next phase of the project.

# Introduction

## Purpose and Scope of the Document

This document specifies the ELASTIC Hardware Abstraction Layer (HAL) for running sensitive applications on confidential containers. The document describes the ELASTIC HAL deployment scenarios, the HAL architecture and contains a specification of the HAL interface. The actual HAL implementation on different confidential container platforms is outside the scope of this specification except that in the last section of this document, we give recommendations for HAL implementation in AMD SEV and Intel TDX, which will be used as proof-of-concept platforms when verifying the ELASTIC HAL.

## Relation to Work Packages, Deliverables and Activities

ELASTIC WP3 addresses portable computing using confidential containers. The solution’s core is the ability to run Wasm application on different confidential container platforms. The HAL specification is essential to enabling this and is a core building stone to achieve the WP3 objectives. This specification is a contribution to D3.1 - Lightweight Confidential Computing Platform and will be integrated into this deliverable.

This specification is a prerequisite to fulfil the ELASTIC architecture concerning executing portable functions with confidential computing requirements. It supports the on-demand execution and orchestration needed to realize the WP1, WP2, and WP4 frameworks when the functions have strong security/privacy expectations. This HAL specification will also support the realization of the WP5 pilot use cases. The HAL as such will not be the only needed component to achieve these goals, but is important to ensure agnostic solutions that can be realized on important current and future confidential computing platforms.

## Contribution to WP3 and Project Objectives

The objective to create a privacy-preserving architecture-agnostic efficient and secure execution environment requires solutions that allow protected deployment of workloads on many different types of confidential containers. The adoption of such a solution is heavily dependent on the ability to run workloads on many different edge resources and platforms. The ELASTIC HAL design will enable exactly this. We put special efforts into identifying core HAL functionality and an architecture that can support a wide range of confidential computing platforms. Even if this will not allow agnostic secure deployment in all aspects, i.e. attestation handling, etc., it is an important step in achieving platform agnostic confidential platform resource accesses.

## Structure of the Document

The document starts by describing confidential container workload deployment scenarios. These scenarios are used to identify core HAL requirements. The next section contains the identified ELASTIC HAL requirements. Section 4 describes the ELASTIC HAL architecture. In Section 5, we specify the HAL interface. The final section gives recommendations for HAL implementation on AMD SEV and Intel TDX platforms. Even if the HAL in this specification is not limited to these two platforms, these two will be used as HAL benchmarks.

# Deployment Scenarios

The ELASTIC HAL will support a wide range of use cases with different security demands. To accomplish this, we have worked with a broad set of different deployment scenarios that represent typical usage for confidential computing. That does not mean that the HAL is *limited* to these scenarios, but these scenarios are carefully chosen to show the most important HAL deployments. Consequently, they will ensure that the specified HAL will be able to support the ELASTIC use cases and several other similar future use cases.

## Scenario selection principle

It is not feasible to work with too many deployment scenarios in parallel. Hence, we decided to limit the specification work to three different scenarios. An absolute requirement from ELASTIC point of view, is full support for the two pilots, which are defined in the ELASTIC project. It is natural to use these two pilots as deployment scenarios for the HAL work. The following two deployment scenarios were taken from the pilot use cases:

* Smart manufacturing
* IT Services Cloud Migration

In order to complement these deployments, we have searched for a complementing scenario. The scenario was chosen based on known needs among the contributing research teams. In the end, the following scenario was added:

* Automotive proactive maintenance

Below, we describe the three selected deployment scenarios in more detail. All scenario descriptions follow the same logical structure, i.e. scenario overview, description of involved entities, graphical description of the scenario and main actions, and finally a summary of the security and performance expectations for the particular scenario.

## The ELASTIC HAL scenarios

This section gives descriptions of the three ELASTIC HAL deployment scenarios. The first two scenarios used in this analysis correspond to the ELASTIC demonstrator scenarios. However, the work with the TEE HAL was planned to be done and executed already in the summer 2024. Hence, the deployment scenarios used for this study do not fully agree with the ELASTIC pilot scenarios. Consequently, additional requirements might come up to the ELASTIC TEE HAL in the next phase of the project.

* + 1. Smart manufacturing
       1. Scenario overview

Smart manufacturing allows advanced production analytics as well as real-time production control. Traditionally production plants have been controlled locally with only overall management at the central level. The Industry 4.0 paradigm [1] allows both more advanced analytics of production data with more advanced and coordinated production control, which can be enabled with the extension and complement of existing production infrastructures. However, to save cost and simplify management, production analytics, and control can also be outsourced to third-party edge cloud resources. Especially, this is an attractive solution for new, and small-scale production units. This TEE deployment scenario supports these two ELASTIC use cases. Smart manufacturing is also the first ELASTIC pilot use case. The analytics and control use cases have rather different demands and characteristics. Hence, rather than treating them as a *single* deployment scenario, we have chosen to describe them *separately.* This means this deployment scenario covers the following *two* deployments:

* Advanced federated analytics of production data
* Edge-based production control
  + - 1. Involved entities

Smart manufacturing can include a lot of different entities and different Industry 4.0 reference architectures have different structures and thus entities. Here we will only introduce entities essential for the understanding of the data flow and HAL platform usage in the deployment scenario. Other entities are for simplicity reasons excluded. The entities and their roles are listed in Table 1 below.

Table :Entities in the smart manufacturing deployments scenario

|  |  |  |
| --- | --- | --- |
| Entity | Description | Role |
| Controller - C | The controller is responsible for controlling a production plant. It communicates with sensors and curators to complete its tasks. The controller can be local, running on edge or in the cloud | The *local* control entity in the scenario |
| Product - P | The product is the entity produced by the production plants. Some products have advanced communication interfaces and can communicate with external entities. | Product subject to manufacturing in the scenario. |
| Analytics Engine - A | The analytics engine performs analytics tasks on production data and may also communicate with other analytics engines connected to the local or remote production plant. | Responsible for production analytics processing. |
| Plant Production Manager - M | The plant production manager function is responsible for scheduling production tasks and gathering production information to optimize production flows. | Production configuration and supervision. |
| Edge Computing Resource - E | An edge computing resource allows accelerating computing tasks close to the fabric. We here only consider edge computing resources with TEE and Wasm support. | Edge resource to perform computing task requested by M. |
| Edge Storage Resource - ES | An edge storage resource allows non-volatile storage at the edge. | Edge resource to store data from the production system. |
| Cloud Computing Resource - C | A cloud resource allows the execution of arbitrary computing tasks remotely. | Cloud resource to perform computing task requested by M. |
| Cloud Storage Resource - CS | A cloud storage resource enables remote non-volatile storage. | Cloude resource to store data from the production system. |

* + - 1. First manufacturing deployment scenario

This specification considers *two* different manufacturing deployment scenarios. The first scenario is an analytics scenario where edge and central computing and storage resources are used to perform production flow analytics. The scenario is depicted in Figure 1 below.

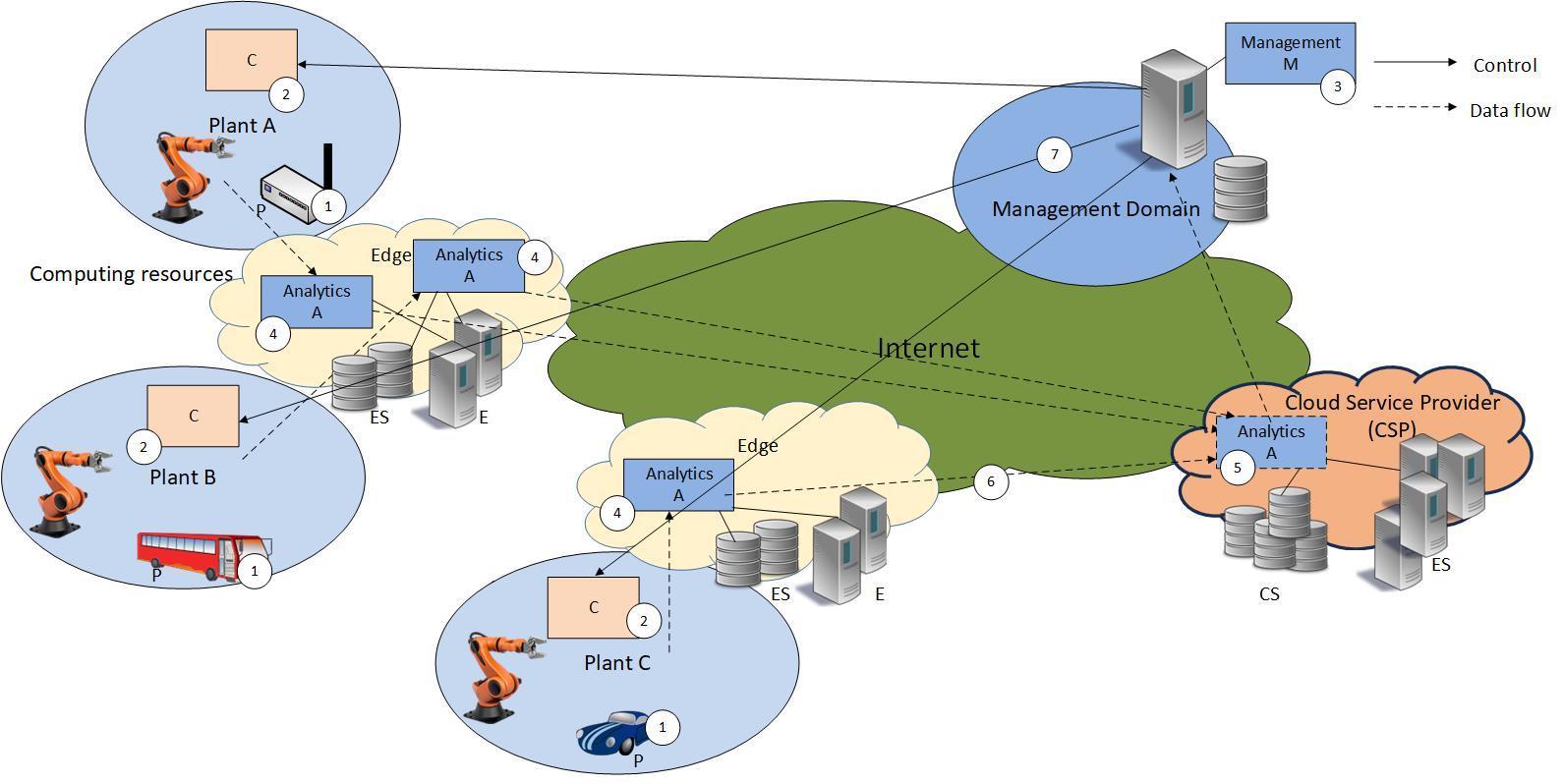


Figure :Analytics smart manufacturing deployment scenario.

To better understand the scenario, below we describe the typical information flows and actions in this deployment scenario. This flow is only one among several possible flows in the scenario but will give a view of the main interactions and data flows for one typical deployment.

1. The scenario shows three different plants producing three different products, i.e. cars, buses, and wireless routers. All these products have at the final production stage network connectivity possibilities.
2. The production is under control at each plant of one or several controllers. The controllers control actuators, robots, or similar in the plant. In this deployment scenario, we limit the description to local controllers.
3. A management system is in charge of the production plants. In the scenario description, we have assumed a remotely situated management function, but it can be placed anywhere in the system with similar functionality. The management system is responsible for configuring controllers and all other connected entities in the plants as well as analytics functions in the system independently of where they are launched.
4. According to this deployment scenario, the management functions launch analytics functions (A) on edge resources in the system. The analytics function is assumed to be run as Wasm processes on TEE enabled edge resources (E). The management functions first identify suitable edge computing and storage resources to associate with each analytics function in the system. Next, the management system selects a suitable Edge execution resource(s). The identified TEE enabled resource(s) is(are) verified and if the verification succeeds, the analytics Wasm(s) is(are) launched on the identified resource(s). Once, an analytics Wasm is launched, the management function configures protected analytics data passing from the production plant to the analytics Wasm. This includes setting up the needed connectivity channels and protection mechanisms of these channels, which feed data from the production plant(s) to the analytics Wasm. This can be data from controllers, actuators, or even data from products under production themselves. To perform its tasks, the analytics Wasm needs access to non-volatile Edge storage resources (ES) as well as a protected GPU hardware acceleration at E. The different analytics functions in the system are interconnected such that federated analysis between different plants can take place in the system.
5. Optionally as shown in the deployment scenario, analytics functions (A) can also be launched on external cloud resources. These analytics functions might collect analytics results from local analytics functions in order to for instance to build “global” analysis models for the system. When this configuration applies, it is the management function that is responsible for also identifying and verifying such analytics Wasm cloud resources and launching them in the system. It then is the management function that sets up the necessary data channel feeds from local (edgealso ) analytics functions to the central analytics function, as well as feeding analytics results back to the management system.
6. The dotted lines show the analytics data flows in the system. All data flows are assumed to be done using integrity and confidentiality-protected data channels.
7. The management function is assumed to control all production in the system based on the received analytics results. The non-dotted arrows show the control data flow used by the management system when re-configuring production controllers in the system based on analytics results. All these control flows are assumed to take place over integrity and confidentiality-protected channels.
   * + 1. **Second manufacturing deployment scenario**

The second manufacturing deployment situation we considered is a case where the production control is moved from local controllers to edge controllers. This will allow some more advanced control functionality. Figure 2 below shows this second deployment for smart manufacturing.

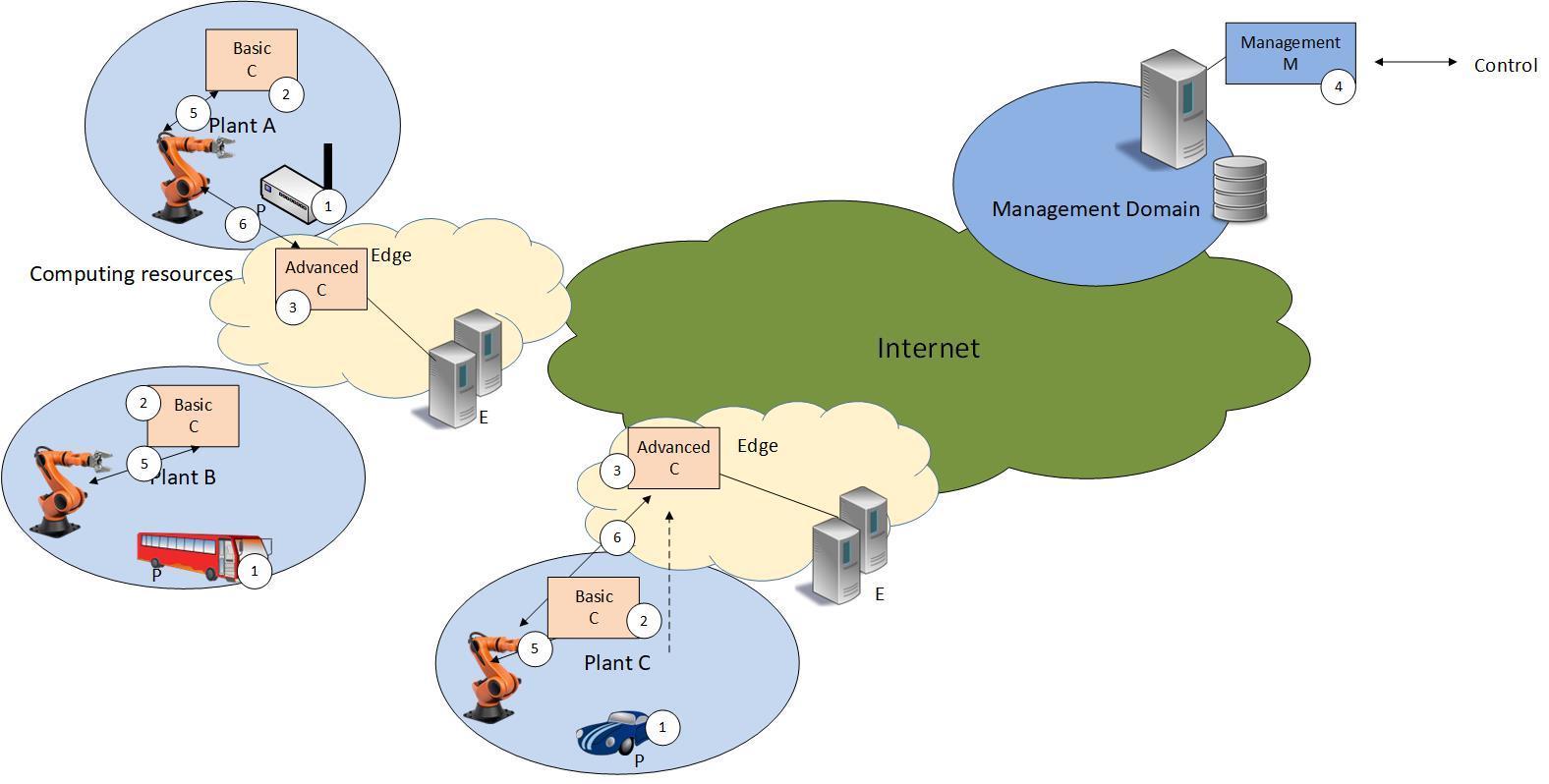


Figure : Edge control smart manufacturing deployment scenario.

The data processing and flow are a bit simpler for this case compared to the previous one. Below, we describe the main actions in this deployment:

1. The scenario shows three different plants producing three different products, i.e. cars, buses, and wireless routers. All these products have at the final production stage network connectivity possibilities.
2. The production is under control at each plant of one or several local controllers. The controllers control actuators, robots, or similar in the plants. These controllers handle basic and most time-critical tasks, while more advanced control functionality is handled by the advanced controllers (see next point).
3. The production plant is in addition controlled by one or several, more advanced, but less time-critical controllers. These controllers are executed as Wasm applications on edge TEE resources (E) in the network. This follows a cloud-based control model previously reported in the literature1). The integrity of the advanced controller and its control data must be preserved all the time in order not to jeopardize production at the plants.
4. A management system is in charge of the production plants. In the scenario description, we have assumed a remotely situated management function, but it can be placed anywhere in the system with similar functionality. The management system is responsible for configuring controllers and all other connected entities in the plants, independently of where they are launched. The management controller verifies edge cloud resources before launching any advanced controller on such a system. It is also responsible for configuring the secure control channels between launched advanced controllers, sensors and actuators etc. in the plant.
5. Within the plant the basic control loop takes place through direct integrity-protected communication between the local controller and the processes under control.
6. Advanced process control is done using Wasm advanced controllers executing in TEE-protected edge resources. All control communication with the plant processes takes place over an integrity-protected channel.
   * + 1. Data protection and performance

Here, we summarize the main data protection and performance implications for the smart manufacturing deployments described in Section 2.2.1.3 and Section 2.2.1.4.

Security

The first deployment scenario (analytics) has strict requirements on the data processed by the analytics functions executed on the edge resources or central cloud resources that it is never leaked to any external partner. This is also true for the analytics results produced by the analytics engine. The confidentiality must also hold for data processed by GPU units.

The first deployment scenario (analytics) also has strict requirements on the integrity of data processed by the analytics functions, as well as the analytics results produced by the engines. The integrity must also hold for data processed by GPU units.

The second deployment scenario (edge control) has strict requirements on the integrity of process data as well as control signals. The confidentiality and integrity of the advanced analytics engine must be guaranteed during the whole lifetime of the advanced analytics Wasm application.

Performance

The first deployment scenario requires that the analytics engine can complete its analytics task within the time limit determined by the analysis model used. We expect to be able to process a large amount of data, within a short time by utilizing GPU hardware resources. However, there are no strict requirements on response times. The bandwidth consumption requirement is determined by the amount of data collected from the plant.

The second deployment scenario has strict real-time requirements regarding the response time for the control task. The precise real-time requirements are determined by the specific plant process under control.

* + 1. IT Services Cloud Migration
       1. Scenario overview

Migration of IT Services to the cloud with edge and far edge network support could be of particular interest for IT organizations to provide lower latency and higher availability for the users of these services. Traditionally such migration is limited to less sensitive services while sensitive ones remain hosted in private data centres. This is notably due to the application of cybersecurity policies concerning organizations’ own rules or with applicable regulations. Privacy is one of the key aspects restricting the deployment of some IT services. The TEE deployment scenario considered here, which is also the second ELASTIC pilot use case, addresses the deployment of IT services while preserving such privacy aspects thanks to a confidential computing approach. For that objective, this deployment scenario particularly focuses on the deployment of such computing platforms supported by:

* Deployment and Handling of Remote Attestation framework
* Portability of such confidential computing platform while preserving and demonstrating attended security guarantees for the regulation needs
  + - 1. Involved entities

Cloud migration involves several different administrative entities, as well as the actual cloud resources and the service subject to migration. The different entities and their roles are listed in Table 2 below.

Table :Entities in the IT services cloud migration deployment scenario.

|  |  |  |
| --- | --- | --- |
| Entity | Description | Role |
| IT Organization - A | The IT Organization is responsible to make IT services available to their users. The IT Organization relies on the orchestrator platform to perform on-demand deployment of applications. | A has no direct role in the deployment but is the main stakeholder for the scenario. |
| Sensitive Service - B | The service can be deployed in the cloud to improve latency and availability of services to users. | The service subject to migration. |
| IT User - C | User of the IT Service. | The user of the migrating service. |
| Assessor/Auditor - D | The auditor will review and collect evidence of the proper enforcement of privacy rules (custom or regulatory ones) from the TEE platforms and associated services (orchestrator, data protection …). | The auditor of the security properties in the system. |
| Data Protection Service - E | A cloud resource providing encryption/decryption support | The service protecting the migrating service. |
| ELASTIC Orchestrator - F | Conduct orchestration as studied in this project and enforce security policies and related computation for attestation and authorization management. | The functions that ensure migration of services according to the defined security policies in the system. |
| IT Near Edge - G | An on-premise computing resource of the IT organization enforces security policies and ensures remote attestation and data protection of sensitive services. | Edge computing resource with shielding capabilities. |
| Public-Cloud - H | A cloud computing resource for the execution of IT organization services. | Cloud computing resources in the system. |

* + - 1. IT service cloud migration deployment scenario

The IT migration involves the IT user and the computing resources as well as orchestration services in the system. The deployment scenario is depicted in Figure 3 below.

A diagram of a cloud computing

Description automatically generated

Figure : IT migration deployment scenario.

To better understand the scenario, below we describe the typical information flows and actions in this deployment scenario (Figure 3):

1. IT Users (C) are requesting access to sensitive services (B), currently running on edge resources (G).
2. ELASTIC Orchestrator (F) performs remote attestation of IT Near Edge (G) runtime environment.
3. ELASTIC Orchestrator (F) performs an attestation of the *target* environment running in a public cloud (H) and enforces the additional security policies for accessing the services.
4. IT Near Edge (G) accesses to Data Protection Service (E) to request keys for the secure migration of service B from the source resource in G to the target resource in H.
5. The orchestration or attestation agent in G, connects to the data protection service to obtain the needed encryption and integrity protection keys required for the migration of service B. The agent encrypts the service.
6. The orchestration or attestation agent in H connects to the protection service to obtain the needed keys to decrypt and verify the integrity of the protected service.
7. The Orchestrator (F) downloads the protected service (B) from G.
8. The Orchestrator (F) securely launches the service on the target environment on H. The orchestration or attestation agent decrypts and verifies the service before launch.
9. The Orchestrator (F) sends successful migration evidence to the audit service.
10. The Orchestrator (F) redirects the end user to the service resource identifiers. The user (C) starts to use the new service now running in H.

A similar procedure can be used when migrating service from the public cloud to an Edge resource.

* + - 1. Data protection and performance

Here, we summarise the main data protection and performance implications for the IT Services Cloud migration deployment described in Section 2.2.2.3.

Security

The integrity of the source and target TEE (i.e., attestations) are critical prerequisites to authorize the migration of a running resource (e.g., workload) of a source TEE to another TEE instance. Hence, the protection service must be able to verify (i.e., attest) the source and target TEEs before releasing the keys needed for the migration of the resource.

The policy defined to access the source resource must be enforced and this must be verified before allowing the migration to happen.

Existing security associations between end users and the service must be preserved during the migration without leakage of any encryption keys or other sensitive information shared between the user and source resource.

Performance

The verification of source and target TEEs needs to be fast to allow quick migration from domain G to domain H.

The needed key management functions and corresponding authentication functions must be quick and the round-trip times to the protection service short, so as not to slow down the migration too much.

Service encryption must be very fast and the decryption and verification at the resource on the target platform needs to also be fast.

* + 1. Proactive Maintenance
       1. Scenario overview

Modern vehicles host increasingly capable computing platforms, a trend that will only increase as self-driving cars enter the market. These computing platforms have access to a wide variety of data that can be used to detect and diagnose faults before component failure. In the air transport industry, Flight Data Management systems are routinely used to obtain diagnostic information that can then be used to investigate the prevalence of newly-discovered issues and identify affected aircraft. Collecting this volume of data in consumer vehicles brings with it technical and regulatory challenges. However, many of these issues may be sidestepped by moving data processing to the vehicle itself, where raw data is available in great volumes and can be processed without impacting privacy.

For example, a manufacturer identifies a fault before complete failure by a temporary increase in engine temperature at a certain output level, can deploy a workload to the affected vehicles, identifying which and how many vehicles suffer from this fault. And issue recall notices or maintenance warnings without needing to collect huge volumes of data themselves, or issue a recall to all potentially-affected vehicles.

* + - 1. Involved entities

Proactive maintenance will depend on the type of vehicle and system. Here we assume a simplified scenario where cars are subject to maintenance operations. The different entities and their roles are listed in Table 3 below.

Table : Entities in the proactive maintenance deployment scenario.

|  |  |  |
| --- | --- | --- |
| Entity | Description | Role |
| Manufacturer - MFG | A server hosted by the manufacturer of a vehicle. | Receives data from maintenance providers and vehicles, and uses ELASTIC orchestration technology to distribute Wasm modules to vehicles. |
| Vehicle Application CPU - VAPP | A computer system embedded in the vehicle itself, contains a TEE. It receives some sensor data directly (e.g. the infotainment system's microphone and location data) and can communicate with the vehicle's ECU to obtain engine parameters. | Used to perform compute-intensive workloads within the vehicle itself and hosts the TEE containing the ELASTIC Wasm runtime. |
| Engine Control Unit - ECU | A computer system inside the vehicle is responsible for receiving engine data. | Manages data from the engine and determines what can be released to the outside world. |
| Maintenance provider - MNT | A manufacturer-specific tool held by a maintenance provider, such as a mechanic. | Identifies faults using physical means, and downloads relevant data from the ECU to be sent to MFG. |

* + - 1. Proactive maintenance deployment scenario

The vehicle maintenance scenario starts with a function in the vehicle, *or* through central identification of maintenance operations on one or several of the ECUs of the particular kind of vehicle. The deployment scenario is depicted in Figure 4 below.

A drawing of a bicycle

Description automatically generated

Figure : The proactive maintenance deployment scenario.

To better understand the scenario, we below describe the typical information flows and actions in this deployment scenario (number from Figure 4):

1. A car is taken to a maintenance facility, where the maintenance staff identifies a fault with the device. They use a device MNT that attests itself to the ECU to extract stored data, possibly performing some kind of data minimization before sending data to the manufacturer's server MFG.
2. MFG receives reports from several MNT and processes the ECU data to identify common features across vehicles that eventually failed. The manufacturer develops a Wasm module that can identify these features in real time. This is deployed using ELASTIC orchestration technology and runs on the vehicle's application processor VAPP using an ELASTIC Wasm runtime.  
     
   Example modules:
   1. command the ECU to perform an additional self-test of the problem component on start, resulting in a warning if the test fails.
   2. add new constraint on a combination of sensor values, issuing a warning if the vehicle parameters appear suspicious (e.g. additional strain on the engine that is not reflected in the vehicle's GPS trajectory, or excess mechanical noise during certain gear transitions as measured by the entertainment system's built-in microphone)
3. VAPP's Wasm runtime uses the TEE to attest itself to the ECU, which then provides data that would otherwise be unavailable to normal applications.
4. VAPP processes the data from the ECU inside the module, in combination with other data provided by the infotainment system, to determine whether a fault exists.   
     
   If so, and VAPP detects the fault before failure occurs, it signals the MFG across the internet, possibly using attestation, and indicates an appropriate action to the vehicle owner, e.g.
   1. check engine light
   2. recall notice delivered via a mobile app
   3. set code for mechanic to act upon during next service
      * 1. Data protection and performance

Here, we summarize the main data protection and performance implications for the proactive maintenance deployments described in Section 2.2.3.3.

Security

Wasm modules provided by the manufacturer to detect latent faults should not be accessible to vehicle owners or network operators, since they may contain commercially sensitive information such as proprietary fault detection algorithms. The use of ELASTIC's TEE-hosted Wasm runtime and orchestration technology will provide for this.

Data reported from the vehicle must not be modifiable by the end-user, to ensure owners for example are not able to forge fault reports to get a warranty replacement of a non-defective engine that has failed due to abuse, nor should they hide an already-detected fault in an attempt to hide it from a buyer. This can be ensured by VAPP's ELASTIC Wasm runtime using its containing TEE to attest itself to the manufacturer, and using secure storage (perhaps within the TEE, or perhaps by outsourcing it to the ECU) to record detected faults.

Commercially sensitive data collected by the ECU must not be easily available except to the manufacturer and manufacturer-approved software, nor must unapproved software be allowed to issue commands to the ECU that might affect sensitive functionality, such as that related to emissions. We do so in this case by requiring that the fault-diagnosing Wasm module attest itself to the ECU before it can begin communication.

Performance

The ELASTIC TEE-enabled Wasm runtime must be able to perform real-time processing of data from the highest-volume combination of sensors that the manufacturer wishes to monitor simultaneously. Not all sensors need to be monitored simultaneously, as some potential faults may manifest themselves only in certain regimes (e.g. faults that are only visible when the engine is outputting significant power need not be checked for when the vehicle is parked, and inconsistent outputs from a parking sensor cannot be diagnosed when a vehicle is on the highway).

The attested link between VAPP and ECU must have sufficient capacity to transmit all relevant data to the VAPP. Where this is not the case, it may be necessary to have an ELASTIC-enabled ECU that can perform data minimization using a sandboxed Wasm module.

VAPP must have the capability to run enough Wasm modules so that it can handle all active “soft recalls” (i.e. one module for each fault neither confirmed nor eliminated).

MFG's ELASTIC orchestration capability must be capable of scaling to the manufacturer's entire fleet of vehicles; that is, it must be possible to realistically deploy a module to millions or tens of millions of vehicles promptly.

# The HAL requirements

This section defines the ELASTIC HAL requirements. We start by giving a short overview of the requirement derivation principles used and then follow by the actual documentation of the requirements.

* 1. Requirements Derivation Principle

The deployment scenarios documented in Section 2 are the starting point for the HAL requirements. This is natural as no stakeholders are directly involved in the specification work. Considering this limitation, we follow the ISO/IES/IEEE 29148 standard for requirements engineering [2]. In particular, the HAL requirements are defined using the 29148 software requirement’s structure.

We have also identified stakeholder requirements following the standard process:

1. Define the constraints at the system level (The defined scope of this document)
2. Define a representative set of activity sequences to identify all required services (Obtained in the deployment scenarios documented in Section 2)
3. Identify the interaction between the users and the system (Obtained in the deployment scenarios documented in Section 2)
4. Specify security requirements (Done per deployment scenario in Section 2)

In addition, the requirements definitions were done according to the 29148 standard:

1. Definition of function boundary,
2. Define each function that the system is required to perform
3. Define necessary implementation constraints,
4. Define technical and quality in-use measurements,
5. Specify system requirements and functions
   1. Requirements
      1. Dependencies

The ELASTIC TEE HAL is needed to allow Wasm workloads with strong confidentiality and integrity demands to be executed on different platforms supporting secure execution. In particular, the ELASTIC HAL must allow workloads to be migrated and deployed on different TEE architectures without losing any functionality. Hence, the HAL functions in this specification will enable Wasm interoperability without compromising security or performance. This in turn implies that the HAL is fully dependent on the implementation and support given in the particular TEE. To achieve this, we have chosen to keep the HAL functionality on a generic level as much as possible.

Running Wasm inside a TEE requires a suitable Wasm runtime supporting the HAL. Hence, the HAL realization is strongly dependent on the runtime. However, the HAL specification as such does not assume a particular runtime but is defined on a generic level. The runtime as such is not enough to support the HAL but a complete *framework* is needed that can handle the secure execution and HAL support on a TEE.

* + 1. HAL functions

The HAL function supports the ELASTIC deployment case. To make it easy to follow the functions, we structure them according to their origin, i.e. the three different deployment scenarios presented in Section 2. Some functions are common for several different scenarios. In those cases, the functions are only specified in the first scenario in which it appears.

**Smart Manufacturing**

The manufacturing deployment case was presented in Section 2.2.1. We analyzed two different sub-cases; one case on distributed analytics and a second case for Edge and cloud-based manufacturing control. The analysis resulted in a set of identified functions. Those functions are listed, explained, and motivated in Table 4.

Table : Required functions identified from the smart manufacturing deployment case.

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Interface type | Description | Motivation |
| 1 | Secure transport, TCP and UDP | This interface allows Wasm application setting up external TLS and DTLS connections in client or server mode. | Wasm analytics engines must be able to collect data from external sources and set-up secure data channels with a central analytics engine without risk of data confidentiality and integrity. |
| 2 | Encryption/Decryption | An interface for the acceleration of common symmetric and asymmetric encryption and decryption processes. | Secure Wasm workloads for analytics will need an object security mechanism to process incoming data. Platform support for the main primitives accelerates analytics performance, simplifies Wasm deployment on different platforms, and reduces Wasm code size. Secure control of production processes will also need specific real-time adapted cryptographic protocols. |
| 3 | Integrity check | See ID 2 | See ID 2 |
| 4 | Accelerated computation, GPU | An interface for access to graphic processing for analytics. | Analytics for factory production performance will need access to hardware acceleration. Real-time control of production flow might require hardware acceleration for the control process. |
| 5 | Access to random | An interface to cryptographically sound random number generator. | Secure exchange of analytics data using for instance object security mechanisms will require access to a cryptographically sound random source. Protected real-time control will also need access to a cryptographically sound random source. |
| 6 | Clock | An interface to the system clock. | Wasm secure control will require access to a reliable system clock. Wasm analytics task will need access to a reliable system clock to perform their tasks. |
| 7 | Non-volatile storage | An interface for long-time storage of data. | A TEE for Edge control should be available in periods. It improves management and performance if not all control configuration data must be transferred when a workload is terminated. Furthermore, new control modules should be possible to deploy in real-time that can use configurations from an old Wasm control workload. |
| 8 | Secure non-volatile storage | An interface to secure long-time storage of data. | Data for non-volatile storage as described for ID7 might be confidential and must then be stored protected. |
| 9 | Resource allocation | An interface for the Wasm to request more or less computing and memory resources from the platform. | Analytics engines should have the possibility to request more resources and/or inform the platform of the resource needs. This allows better resource utilization of the TEE compute platform. |
| 10 | Secure external event handler | An interface to an external event handler can be used to receive a trig event for other Wasm workloads on the same or a different platform. | Secure control of remote processes will benefit a lot from the possibility of handling external events. Event handling will also enable coordinated control between different Wasm controlling workloads. |
| 11 | Wasm to Wasm protected internal communication | An interface for the communication between different Wasm workloads executing on the same TEE VM. | Some control tasks will be done with multiple workloads collaborating to perform the control tasks. A protected internal interface will enable such collaborative control without sacrificing security. |

**IT Service Cloud Migration**

The IT service cloud migration case was presented in Section 2.2.2. The analysis resulted in a set of identified functions. The most important platform function to support the IT migration case is the TEE agent. One can implement such agents in different ways. As the migration agent would be an essential *infrastructure* component, it is not natural to implement the agent as Wasm workload but instead, as a native component running on the TEE. We have done the function analysis based on this assumption which implies that functional requirements on the HAL for this case are rather limited. The identified functions are listed, explained, and motivated in Table 5.

Table : IT Service Cloud Migration functions.

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Interface type | Description | Motivation |
| 1 | Secure transport, TCP and UDP | This interface allows Wasm application to set up external TLS and DTLS connections in client or server mode. | The migrated service must be able to establish a connection with the user over a protected channel. It is also a requirement that migrated services shall be able to keep existing security association and transports channels after migration. |
| 7 | Non-volatile storage | An interface for long-time storage of data. | By supporting non-volatile storage for different Wasm workloads, it is possible to share data with new, migrated Wasm workloads, which speeds up and simplifies the migration for some workloads. |
| 8 | Secure non-volatile storage | An interface to secure long-time storage of data. | Data for non-volatile storage as described for ID7 might be confidential and must then be stored protected. |
| 9 | Resource allocation | An interface for the Wasm to request more or less computing and memory resources from the platform. | Migrated workload should be able to request the needed computing and memory resources. This allows better resource utilization of the TEE compute platform. |
| [12] | Runtime attestation | An interface for the Wasm to obtain the attestation (measurement) of the Wasm runtime | Attestation is an expected property of the platform and the Wasm framework to allow deployment of workloads. However, a migrated workload might need to obtain the result of the attestation operation. |
| [13] | Platform attestation | An interface for the Wasm to obtain the attestation (measurement) of the hosting platform | Attestation is an expected property of the platform and the Wasm framework to allow deployment of workloads. However, a migrated workload might need to obtain the result of the attestation operation. |

**Proactive maintenance**

The proactive maintenance case was presented in Section 2.2.3. The analysis resulted in a set of identified functions. According to this deployment case, the Wasm workload performed advanced analytics on a TEE-enabled vehicle platform. Such analytics tasks require the possibility to “poll” the available hardware/software on the vehicle, as well as the possibility to access such resources on the vehicle. This implies lots of additional interface capabilities of the TEE of a not-so-generic type. Consequently, we have here marked such functions as *optional.* Optional functions are indicated with brackets ([]) around the ID in the table. The reason for making these interface functions optional is that this deployment scenario is not an ELASTIC use case and will not be used for any pilots in the ELASTIC project. This also means that these functions will not be part of the ELASTIC core HAL. The identified functions are listed, explained, and motivated in 3.

Table : Proactive Maintenance functions.

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Interface type | Description | Motivation |
| 1 | Secure transport, TCP and UDP | This interface allows Wasm application setting up external TLS and DTLS connections in client or server mode. | The MNT needs to report status back to the MFG over a protected interface. |
| 2 | Encryption/Decryption | An interface for the acceleration of common symmetric and asymmetric encryption and decryption processes. | The MNT needs to make diagnostic requests to the vehicle’s internal entities. Some of those might run proprietary authentication, confidentiality and integrity protection mechanisms. Platform support for the main primitives accelerates diagnostics performance, simplifies Wasm deployment on different platforms, and reduces Wasm code size. |
| 3 | Integrity check | See ID 2 | See ID 2 |
| 5 | Access to random | An interface to cryptographically sound random number generator. | Protected collection of diagnostic data will require access to a cryptographically sound random source. |
| 6 | Clock | An interface to the system clock. | Diagnostics operations will require access to a reliable system clock. |
| [12] | Runtime attestation | An interface for the Wasm to obtain the attestation (measurement) of the Wasm runtime | The MFG might need to be able to verify integrity of the TEE runtime without communicating interactively |
| [14] | Vehicle system discovery | An interface for the diagnostics Wasm to collect information of the locally supported functions and status of the vehicle. | The MFG needs to be able to collect information on the hardware and software configurations applicable to the particular vehicle, subject to the diagnostics operations. Even if such information typically is already available at the MFG, specific local configurations and hardware changes might have been performed on the vehicle and some functions might even not be available due to faults. |
| [15] | Key handler | An interface to request the needed key used to enable inside vehicle secure connection to vehicle entities. | The MFG needs to set up secure connections with the different vehicle entities to collect the diagnostics information. This interface allows the MFG to collect the key and credential information needed to set up such secure connections. |
| [16] | Secure CAN transport | An interface to set-up secure connections between the TEE and other to vehicle entities over the CAN-bus and other, non-Ethernet local buses. | The Vehicle might support non-Ethernet internal communication means. Then the TEE needs to be able to set up connections to vehicle entities without risk of sniffing or interference. |

* + 1. User characteristics

The Wasm workloads are communicating with the external world through the HAL. All user interactions take place through a network connection with the workload through such interfaces. Hence, the HAL must provide good performance for setting up and maintaining secure connections through the HAL. All other user aspects are handled on the application level not directly affecting the HAL design or functions.

* + 1. Limitations

We have in our analysis only taken into consideration the HAL functions needed to support the three deployment scenarios covered in this specification. There are certainly additional requirements to support a broader set of HAL functions. Even if this is the case, the requirements identified in our analysis constitute a “typical” set for running Wasm on TEE. Hence, we are comfortable that the identified requirements will provide a sound basis for the ELASTIC TEE HAL architecture. With a generic and flexible architecture, we can easily incorporate additional HAL requirements in the future. Even if this is the case, it is important to notice that the HAL as specified here is *not* a generic TEE HAL that can be used to support a broad set of use cases and applications, and not serve in its present form as a draft standard. However, the requirements work, and specifications can provide a *basis* for such, potential future standardization efforts.

The requirements are not dependent on the actual runtime or TEE platform. However, this specification only considers *two* TEE platforms, and detailed interface implementation recommendations will be provided for only these two platforms. The two supported TEE platforms are:

* AMD SEV-SNP[[1]](#footnote-1)
* Intel TDX[[2]](#footnote-2)
  + 1. Assumptions and Dependencies

The HAL requirements assume the platform will support a suitable Wasm runtime inside the subject TEE. The actual realization of the runtime is not within the scope of this specification. Within the ELASTIC project, we have the working assumption that the ELASTIC TEE runtime will be built upon the Enarx project architecture as a code base [3]. The HAL support and implementation is heavily dependent on the runtime technology *as well* as the TEE platform as such (See also the limitations above).

* + 1. Apportioning of requirements

The HAL requirements are generic for any ELASTIC TEE platform claiming ELASTIC support. The requirements apply to the platform providing the runtime for ELASTIC secure Wasm. It is not applicable to other runtimes or system components within the scope of ELASTIC

* + 1. Specific requirements

The TEE HAL has not specific requirements beyond what is needed to be able to run Wasm on the TEE runtime.

* + 1. External interfaces

The TEE HAL is an interface that defines the Wasm interface on the platform. In order to implement the HAL support on a specific TEE, the runtime must be integrated with several other platform functions. How to provide these integrations is subject to the TEE-specific HAL implementation. That means in turn, that all the HAL functions listed required must have support on the TEE platform. This is the case for the two supported ELASTI TEEs, i.e. AMD SEV-SNP and Intel TDX in most configurations. However, for a platform that lacks the support for the needed HAL functions, some interfaces might not be available. A Wasm seeking HAL support must be able to execute independent of this fact. To handle, this we have identified on additional mandatory ELASTIC HAL interface function specified in Table 4 below.

Table : HAL Capabilities function.

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Interface type | Description | Motivation |
| 15 | Platform capabilities | This interface allows Wasm workloads to request the supported HAL functions on the current platform. | Different TEE platforms might have different external interfaces as well as internal capabilities. To allow the Wasm application to execute on platforms with different capabilities, it must be able to retrieve the specific HAL support on the current platform. |

* + 1. Usability requirements

The Wasm runs on the TEE without any direct user interface. It shall be easy to deploy and manage Wasm on different TEE platforms. The principles for managing and updating Wasm on the TEE are outside the scope of the TEE HAL specification.

* + 1. Performance requirements

The TEE HAL needs an acceptable response time and speed for interfaces allowing direct external and internal communication. What is considered acceptable performance depends on the use case. However, we will set lower bounds on acceptable response time as well as data rates for internal communication interfaces. Those will not be available until the final version of this document.

* + 1. **Design constraints**

The ELASTIC TEE HAL has been defined based on the included deployment cases. The main goal is to support the ELASTIC two pilots. The HAL has been defined using a generic and extendable architecture. It is also derived with just *two* TEEs within scope. Future extensions shall be straightforward though with the selected design approach.

* + 1. Standard compliance

The TEE HAL interface will follow the principles and recommendations as used by the WebAssembly System Interface (WASI) subgroup of the W3C standard [4]. The functions identified in this specification are *independent* of the WASI standard interface. However, whenever an appropriate function is already adopted by WASI, we have chosen to incorporate it also for the TEE HAL. The WASI is a living standard, and the most developed functions are currently in the “implementation” phase[[3]](#footnote-3).

* + 1. Verification

The TEE HAL will be verified once the specification is complete through a reference implementation. The reference implementation will be tested on at least *two* different TEEs, AMD SEV-SNP and Intel TDX, and the verification will be done by at least *two* different ELASTIC partners. Test scripts will be shared between partners but not released as part of this specification.

# The HAL architecture

The HAL architecture follows the common principle of the WASI as defined by the Bytecode Alliance [4]. WASI can be implemented both using core Wasm modules or modules built according to the so-called “component model”. The latter model is a way to build Wasm binaries with *wrappers* that allow interoperable function passing between different Wasm applications. WASI as defined by the Bytcode Alliance is currently in stage WASI 0.2. This latest release uses the component model, which is considered the future WASI model. Hence, a component model should be used for implementations claiming ELASTIC TEE HAL compliance. WASI can be used on different TEEs. However, the ELASTIC TEE HAL only supports VM-based TEE such as AMD SEV, Intel TDX and ARM CCA.

* 1. Architecture Overview

The TEE HAL is designed according to the general ELASTIC architecture. It is the architecture to be used when running ELATIC Wasm workload on TEEs. Figure 5 below gives a schematic overview of the TEE HAL.

A diagram of a diagram

Description automatically generated

Figure : The TEE HAL architecture.

The TEE HAL must be supported with the compliant Wasm runtime in the TEE. This specification does not mandate any specific runtime, but the ELASTIC TEE HAL reference implementation will be based on Wasmtime[[4]](#footnote-4), which is also the runtime used by the Enarx project[[5]](#footnote-5). This means that ELASTIC TEE HAL compliant Wasm workload will be possible to develop using the *standard* Wasmtime build principles with the correct *bindings* to the unique ELASTIC TEE HAL. functions. However, when using the WASI 0.2 functions part of the ELASTIC HAL, the standard bindings will be used, and no ELASTIC particular adaptations need to be done.

* 1. Architecture Components

The main architectural components are the following:

* The TEE hardware platform
* The runtime with WASI support implementing the ELASTIC TEE HAL
  + 1. The TEE hardware platform

The ELASTIC HAL is TEE agnostic but each runtime implementing the HAL needs to be carefully adapted to the specific TEE platform supporting it. The ELASTIC HAL reference implementation supports only the following two TEEs:

* AMD SEV-SNP
* Intel TDX
  + 1. Runtime

Any Bytecode Alliance[[6]](#footnote-6)-compatible runtime can be used to implement the ELASTIC HAL. Some runtimes will require considerably more effort to be compliant with this HAL specification. The ELASTIC HAL reference implementation will be based on the Enarx open-source project with Wasmtime as the runtime.

* 1. Implementation Considerations

The ELASTIC HAL will be based on WASI and using the WASI 0.2 as a baseline. This implies that the Bytecode Alliance *component* model must be used for implementing and using the ELASTIC HAL. Consequently, the interface functions will be specified using the standard .wit file format. All implementations claiming ELASTIC HAL TEE compliance need to support the defined “.wit” interfaces. This specification does *not* define the “.wit”-interfaces but the ELASTIC HAL functions (see next Section) already part of WASI 0.2 will reference the standardized interfaces and the corresponding .wit definitions. The ELASTIC HAL-specific interface functions, including “.wit” definitions, will be released as part of the ELASTIC HAL reference implementation.

# Interface Specification

This section contains the EASTIC TEE HAL interface functions. The interfaces follow the Bytecode Alliance WASI 0.2 specification whenever applicable with suitable extensions to cover the requirements identified in Section 3. The interface specification is *not* a full specification but lists the included functions and their motivations. The actual interface will be specified with .wit files part of the ELASTIC HAL reference implementation. The requirements numbering of the different interface functions refer to the identities in Section 3.2. The last colum in the interface lists referes to the WASI 0.2 support. Some functions have a draft WASI version, those versions are indicated as “proposed” or “No”. Proposed or draft WASI interfaces will be used whenever we find them suitable but this is subject to further evaluation when the detailed interfaces in the ELASTIC TEE HAL are defined/implemented.

This specification contains all HAL functions identified as required to support the deployment scenarios considered fully. However, as it is a *major* effort to provide full support for all functions, it is *not* within the scope of the ELASTIC project to give the full implementation of all the interface functions listed in Section 5.2 below. The ELASTIC TEE HAL reference implementation will include a subset of the interface functions. Exactly which one to include will be subject to prioritization and project resources available during 2025.

* 1. Interface Operation Principles

The interfaces are here described on a *generic level.* The HAL is expected to be of traditional request-response type but will also contain functions for setting up buffers and streams. Consequently, the interface operation will depend on the type of function it provides. The detailed operation of the different interface functions will be specified in the .wit files in the reference implementation. However, *not all* HAL functions will be provided as references. Then the precise function and implementation recommendation will be subject to if the ELASTIC TEE HAL will be adopeted in industry.

* 1. Interfaces
     1. Clock

This interface family covers requirement 6.

The clock interface allows the reading of the current time and the elapsed time measurement.

Table : The Clock Interface.

|  |  |  |
| --- | --- | --- |
| **Function** | **Description** | **WASI standard** |
| Read current time | Allows the read of the current system clock | Yes |
| Read time zone | Allows the read of the configured time zone for the system. | Yes |
| Start, stop and read elapsed time from monotonic clock | Allows starting, stopping and reading of the elapsed time for single monotonic clock. | Yes |

* + 1. Random

This interface covers requirement 5.

The random interface allows generating cryptographically-secure pseudo-random bytes.

Table :The Random Interface.

|  |  |  |
| --- | --- | --- |
| **Function** | **Description** | **WASI standard** |
| Generate random bytes | Allows getting cryptographically secure pseudo-random bytes | Yes |

* + 1. Object storage

The function family covers requirements 7 and 8.

The ELASTIC storage function is to be used to allow WebAssembly workloads to read and store data without any underlying knowledge of the actual non-volatile storage on the platform. Hence, we have adopted the Bytecode Allicance blob storage concept for object storage for this purpose[[7]](#footnote-7). This is at the moment a *proposed* WASI standard. In addition, we also would like to support the storage of protected (encrypted) objects. Hence, the Elastic WASI interface also includes protected storage.

Table : The Object Storage Interface.

|  |  |  |
| --- | --- | --- |
| **Function** | **Description** | **WASI standard** |
| Open container | Allows opening an object container if privileges are correct. | No |
| Read object from container | Allows reading of the container content. Optional the container content is encrypted. If this is the case, the container must first be associated with the expected key context. | No |
| Write object to container | Allows writing to file. If the container is associated with a key context, the object will be encrypted (encryption and integrity protected) prior to storing the object. | No |
| Load object key to container | Allows to assign a symmetric key context to a container object. A container object with a key will encrypted the object prior to storing it. | No |

* + 1. Sockets

The function family covers requirement 1.

The WASI 0.2 interface covers TCP and UDP sockets[[8]](#footnote-8). However, TLS/DTLS-protected sockets are considered out of scope. For the ELASTIC HAL this will not work as secure external connections are essential to support the expected use-cases. Consequently, we have extended the interface to cover TLS and DTLS support.

The sockets interface implements TCP & UDP sockets and domain name lookup. It adds the basic BSD socket interface with the intent to enable server and client networking software running on WebAssembly.

Table : The Sockets Interface.

|  |  |  |
| --- | --- | --- |
| **Function** | **Description** | **WASI standard** |
| TCP socket | Allows async read and write from and to TCP sockets | Yes |
| UDP socket | Allows async read and write from and to UDP sockets | Yes |
| Client TLS TCP socket | Allows protected async read and write from and to TCP client sockets | No |
| Server TLS TCP socket | Allows protected async read and write from and to TCP server sockets | No |
| Client DTLS UDP socket | Allows protected async read and write from and to UDP client sockets | No |
| Server DTLS UDP socket | Allows protected async read and write from and to UDP server sockets | No |
| Write server key context | PKCS12 compliant write of TLS/DTLS server key context to a TCP/UDP context. | No |
| Write client key context | PKCS12 compliant write of TLS/DTLS client key context to a TCP/UDP context. | No |

* + 1. Cryptography

The function family covers requirements 2 and 3.

Cryptographic support is a proposed WASI standard[[9]](#footnote-9) but has not been finalized or worked out in detail yet.

The cryptography interface allows relying on hardware available cryptographic accelerators for common symmetric and asymmetric encryption and decryption operations

Table : The Cryptography Interface.

|  |  |  |
| --- | --- | --- |
| **Function** | **Description** | **WASI standard** |
| Load key and algorithm context | Loading of key and algorithm context parameters | No |
| Public key encryption | Public key encryption | Proposed |
| Public key decryption | Public key decryption | Proposed |
| Digital signing | Digital signing support | Proposed |
| Signature verification | Digital signature verification support | Proposed |
| Hashing | Data hashing support | No |
| Symmetric encryption | Symmetric key encryption | Proposed |
| Symmetric decryption | Symmetric key decryption | Proposed |
| MAC calculation | MAC calculation function | No |
| MAC verification | MAC verification function | No |
| Authenticated encryption | Authenticated encryption support | No |
| Authenticated decryption/verification | Authenticated decryption/verification support | No |
| Sealing | Sealing of data (like key) to a particular platform state | No |
| Unseal | Unseal of data (like key) to platform state | No |
| Attestation context load | Platform attestation target (including key inf.) load | No |
| Platform attestation | Attestation request/response (for given attestation context) | No |

* + 1. GPU

The function family covers requirement 4.

Some WebAssembly workload would need GPU acceleration of some functions. To support GPU acceleration in WebAssembly is not straightforward and will require major adaptation depending on which GPU is accessible. In addition, GPU handling from an external WebAssembly is complex. A suitable model already exists in the W3C-specified WebGPU interface [5]. WebGPU is defined for JavaScript but contains a generic GPU workload handling and interaction API. Here, we assume the adoption of the WebGPU principles for the ELASTIC HAL while putting considerable effort into implementing such a framework. Especially, lots of context information and data structures must be defined and supported

The security of GPU workload handling needs to be handled with care. This specification assumes that the GPU resources made available to the confidential WebAssembly will be executed in a protected VM not generally accessible to the rest of the system. Also, the underlying GPU HAL support in the platform must ensure that different confidential workloads executed in parallel cannot influence each execution and that all GPU data is flushed once a workload has finalized its GPU task.

Table : GPU Interface.

|  |  |  |
| --- | --- | --- |
| **Function** | **Description** | **WASI standard** |
| Get GPU adapter | Get a reference to a GPU impl. on the platform | No |
| Create GPU device | Create a new virtual GPU device from an adaptor on the platform | No |
| Read and set GPU adapter features | Function to read and set the supported features of a GPU adapter on the platform | No |
| Create GPU compute pipeline | Create GPU computer and render pipelines for a GPU device | No |
| Create GPU compute pass and rendering | Passing GPU compute and rending task to compute pipelines | No |

* + 1. Resource allocation

The function family covers requirement 9.

The resource allocation interface allows a WebAssemby workload to list its current memory and CPU allocations and to request more resources. Allocation might be accepted or rejected by the platform.

Table : The Resource Allocation Interface.

|  |  |  |
| --- | --- | --- |
| **Function** | **Description** | **WASI standard** |
| List current memory and CPU allocation | This gives a listing of the current memory and CPU allocations given to the workload. | No |
| Request additional RAM memory | This request allows the workload to request additional memory by the platform. | No |
| Request additional CPU allocations | This request allows the workload to request CPU resources by the platform. | No |

* + 1. Event handling

The function family covers requirement 10.

The event handling allows workloads to set up and retrieve events. Events can be both platform internal and external.

Table : The Event Handling Interface.

|  |  |  |
| --- | --- | --- |
| **Function** | **Description** | **WASI standard** |
| Create an event handler | The creation of a new event handler. An event handler URI that can be subscribed to by other workloads" (so that some other high-performance local-only handle might be specified instead) can be reached over https is returned. | No |
| Request event subscription | Request the event handler to listen for the event from the given URI, i.e will trigger the event handler to subscribe to the given URI.. | No |
| Send event to event handler | Send a new event to the event handler. | No |
| Request event from event handler | Retrieve events from the event handler. | No |

* + 1. Protected internal communication

The function family covers requirement 1.

The secure communication interface allows protected communication between workloads. The main purpose of the channel is not to protect the communication as such as the channel is provided within the confidential VM but to have a controlled way to transfer information within the VM to other workloads and make sure that it is only the intended workloads that will get the information.

Table : The Protected Internal Communication Interface.

|  |  |  |
| --- | --- | --- |
| **Function** | **Description** | **WASI standard** |
| Set up a communication buffer | The buffer contains the ID of the sending and receiving workload, and a communication channel buffer reference is returned to the application. | No |
| Push data to buffer | Push (byte) to the secure communication channel. | No |
| Read data buffer | Read (byte) data from the communication channel. | No |

* + 1. HAL functions

This function covers requirement 15.

One cannot assume that all platforms have equal TEE HAL support. This interface allows the workload to request the specific support in the current running platform.

Table : HAL Platform Capabilities Interface.

|  |  |  |
| --- | --- | --- |
| **Function** | **Description** | **WASI standard** |
| Get platform HAL support | This request returns the HAL capabilities in the platform. | No |

# References

1. L. Y. Nakagawa, P. O. Antonino, F. Schnicke, R. Capilla, T. Kuhn and Pe. Liggesmeyer, “Industry 4.0 reference architectures: State of the art and future trends”, Computers & Industrial Engineering, Vol. 256, 2021, <https://doi.org/10.1016/j.cie.2021.107241>.
2. “System and software engineering – Lif cycl processes – Requirements engineering”, ISO/OES/IEEE 29148, 2011.
3. Enarx, Confidential Computing with WebAssembly, <https://enarx.dev/>
4. WASI W3C subgroup, <https://wasi.dev>
5. WebGPU ver. December, 2024, W3C, https://www.w3.org/TR/webgpu/

1. https://www.amd.com/en/developer/sev.html [↑](#footnote-ref-1)
2. https://www.intel.com/content/www/us/en/developer/tools/trust-domain-extensions/overview.html [↑](#footnote-ref-2)
3. https://github.com/WebAssembly/WASI/blob/main/Proposals.md [↑](#footnote-ref-3)
4. https://github.com/bytecodealliance/wasmtime [↑](#footnote-ref-4)
5. https://enarx.dev/ [↑](#footnote-ref-5)
6. https://github.com/bytecodealliance [↑](#footnote-ref-6)
7. https://github.com/WebAssembly/wasi-blobstore [↑](#footnote-ref-7)
8. https://github.com/WebAssembly/wasi-sockets [↑](#footnote-ref-8)
9. https://github.com/WebAssembly/wasi-crypto [↑](#footnote-ref-9)