# OCORA

**Open CCS On-board Reference Architecture**

## Configuration Management

## Concept

Discussion Paper



Document ID: OCORA-TWS07-060

Version: 1.3

Date: 22.06.2023

# Revision history

| Version | Change Description | Initial | Date of change |
|---|---|---|---|
| 1.0 | Initial version | TM | 25.11.2022 |
| 1.1 | Official version for OCORA Release R3 | TM | 30.11.2022 |
| 1.2 | Added introduction paragraph and chapter regarding safety aspects. | JB | 08.06.2023 |
| 1.3 | Official version for OCORA Release R4 | TM | 22.06.2023 |
| | | | |
| | | | |

# Table of contents

# Table of figures



# Table of tables

# References

Reader's note: please be aware that the numbers in square brackets, e.g. [1], as per the list of referenced documents below, is used throughout this document to indicate the references to external documents. Wherever a reference to a TSI-CCS SUBSET is used, the SUBSET is referenced directly (e.g. SUBSET-026). OCORA always reference to the latest available official version of the SUBSET, unless indicated differently.


[1] OCORA-BWS01-010 – Release Notes

[2] OCORA-BWS01-020 – Glossary

[3] OCORA-BWS01-030 – Question and Answers

[4] OCORA-BWS01-040 – Feedback Form

[5] OCORA-BWS03-010 – Introduction to OCORA

[6] OCORA-BWS03-020 – Guiding Principles

[7] OCORA-BWS04-010 – Problem Statements

[8] OCORA-BWS05-010 – Road Map

# 1 Introduction

As per today's practice, changing the configuration of an ETCS-OB system (FW, SW, parameter, etc.) means that the vehicle is out of service for one or more days and requires trained specialists, using vendor and/or vehicle type specific tools. With OCORA's modular system approach and its independent building blocks, an unharmonised (manual) configuration management would become even more complex and difficult to handle.

Hence, the aim of for a harmonised configuration management, able to remotely update vehicles, using a single toolchain for all vehicle types of all fleets even if they are from different vendors. Updates shall be planned (for the whole vehicle or, as applicable, for a single building block) and deployed/installed autonomously, when determined appropriate and safe - i.e., without the need to always take the vehicle out of service (in a depot) for a longer time and without manual interaction on the train.

Foremost, the configuration management described in this document is intended to be applicable for all OCORA compliant systems.

To facilitate a common approach, OCORA plans to shift its current specification activities regarding the configuration management into the ERJU System Pillar. The aim is to further shape to the topic as mirror group contributor. This approach shall pave the way for a harmonized configuration management for all railway components i.e., not only OCORA and its Building Blocks. Nevertheless, a later extension to be used for CCS TRK, Interlocking, RST is desirable.

## 1.1 Purpose of the document

OCORA, with its modular *Building Block* based On-Board architecture, requires a rather complex CCS-OB *Configuration Management* process that involves several stakeholders. This document provides a brieve high-level *Configuration Management* concept and highlights the objectives behind a standardisation of the CCS-OB *Configuration Management* process.

This document is addressed to experts in the CCS domain and to any other person, interested in the OCORA concepts for on-board CCS. The reader is invited to provide feedback to the OCORA collaboration and can, therefore, engage in shaping OCORA. Feedback to this document and to any other OCORA documentation can be given by using the feedback form [4].

If you are a railway undertaking, you may find useful information to compile tenders for OCORA compliant CCS building blocks, complete on-board CCS system, or on-board CCS replacements for functional upgrades or life-cycle reasons.

If you are an organisation interested in developing on-board CCS building blocks according to the OCORA standard, information provided in this document can be used as input for your development.

## 1.2 Applicability of the document

The content of this document is primarily intended to trigger a more in-depth discussion on the topic. Subsequent releases of this document will be developed based on a modular and iterative approach, evolving within the progress of the OCORA collaboration and eventually leading to a more detailed and jointly agreed concept.

## 1.3 Context of the document

This document is published as part of an OCORA Release, together with the documents listed in the release notes [1]. Before reading this document, it is recommended to read them. If you are interested in the context and the motivation that drives OCORA, we recommend reading the Introduction to OCORA [5], the Guiding Principles [6], the Problem Statements [7], and the Road Map [8]. The reader should also be aware of the Glossary [2] and the Question and Answers [3].

Chapter two tries to establish a common vocabulary used for the discussion of the presented concept. It also depicts a simple entity relationship diagram, showing how the different terms relate to each other. Having a common understanding of the terms and their relationship is important to follow the explanations in the following chapters.

Chapter three discusses the overall CCS *Configuration Management* process, introducing the five high-level activities and the involved Stakeholders and their responsibilities.

Chapter four dives into the high-level components that are part of the CCS *Configuration Management* process and chapter five explains fundamental configuration principles.

Finally, the appendix lists the high-level objectives of the CCS *Configuration Management*.

# 2 Terms and Entities

The following table introduces a set of terms that are used throughout this document.

| Term | Description |
|---|---|
| Activation | *Activation* refers to the process of applying a configuration to a *Building Block*. Depending on the type of change, the *Activation* may include the installation of new Firmware, OS, and/or Software. |
| *Authentication* | *Authentication* is the ability to prove that a user or a system is genuinely who that person or what that system claims to be. In the context of *Configuration Management*, *Identification* and *Authentication* are key. |
| *Boot Mode* | A *Building Block* is starting up or shutting down. |
| *Building Block* | A *Building Block* is a sourceable unit of the *CCS-OB System* (hardware and/or *Software*), having standardised functionality, standardised PRAMSS requirements (including Tolerable Functional Failure Rate [TFFR], Safety Integrity Level [SIL] and Safety Related Application Conditions [SRAC]), standardised interfaces (on all OSI Layers) towards other *Building Blocks* and/or external systems. *Building Blocks* are exchangeable and migratable, without impacting other *Building Blocks*.<br>*Building Blocks* are separately sourceable from different suppliers and capable of being integrated by a third party. |
| *BB Configuration* | The *BB Configuration* is an exhaustive, unambiguous description of all *Configuration Items* required to operate a physical instance of a *Building Block*. It includes default values for *Parameters.* |
| *BB Configuration Management Systems* | The *BB Configuration Management System* is an off-board technical system at the *BB Supplier* that is responsible for managing the *BB Configurations*. |
| *BB Manifest* | An exhaustive, unambiguous human readable description of a *BB Supplier* approved *Building Block Configuration* using a standardised lightweight data-interchange format e.g., JSON, XML, etc. |
| *Building Block Package* | The *Building Block Package* is a file containing an exhaustive list of *Building Block* specific *Configuration Items*. The content and the format of the file shall be supplier specific and not standardised. |
| *(Building Block) Supplier* | The *Building Block Supplier* is the manufacturer of a separately sourceable component of the *CCS-OB System*. He is in charge of the implementation, verification, validation, and certification of one or multiple *Building Blocks*. |
| *CCS Building Block Type* | The *CCS Building Block Type* is a unique identifier of a specific OCORA *Building Block* |
| *CCS-OB Configuration* | The *CCS-OB Configuration* is an exhaustive, unambiguous description of all *Configuration Items* necessary to operate a physical instance of a *CCS-OB System*. |
| *CCS-OB Deployment* | Refers to the physical deployment of a *CCS-OB System*. A *CCS-OB Deployment* consists of the CCS-OB hardware running a specific *CCS-OB Configuration*. |
| *CCS Configuration Management System (CCMS)* | The *CCMS* is an off-board technical system that is responsible for managing the *CCS-OB Configurations* of a defined set of *CCS-OB Systems*. Each *CCS-OB System* is managed by exactly one *CCMS*. |

| Term | Description |
|---|---|
| *CCS-OB Manifest* | The *CCS-OB Manifest* is used to describe a *CCS-OB Configuration*.<br><br>*Note:* the actual *Configuration Items* are not part of the *CCS-OB Manifest*, they are only referenced in the *CCS-OB Manifest*. |
| *CCS-OB Parameter Package* | The *CCS-OB Parameter Package* is a file containing *Parameters* required to configure a specific *CCS-OB Deployment*. The format of the file and the content shall be standardised. |
| *CCS-OB System* | The *Command, Control and Signaling On-Board System.* |
| *CCS Vehicle Configuration* | The *CCS Vehicle Configuration* is an exhaustive description of a vehicle, in respect to its physical CCS-OB hardware configuration and all train-born systems that the *CCS-OB System* directly interacts with. |
| *CCS Vehicle Type* | The *CCS Vehicle Type* is a unique identifier for a specific *CCS Vehicle Configuration* |
| *Configuration Items* | *Configuration Items* include *Software* and *Parameters*. They remain unchanged during *Operational Mode* of the *CCS-OB System*. |
| *Configuration Management* | *Configuration Management* refers to the management of all *Configuration Items* of a *CCS-OB System*.<br>From a process point of view, it covers activities along the complete chain, from the *Building Block Supplier(s)* to the *Integrators*, the operator, and the actual *CCS-OB Deployment* on a train. |
| *Configuration Mode* | The only mode of a *Building Block*, in which configuration changes are allowed. |
| *Distribution Job* | The *Distribution Job* contains the metadata required for the distribution of a *CCS-OB Configuration*. In addition to linking a referenced *CCS-OB Manifest* to a *CCS Vehicle Identifier*, it also includes information like distribution date/time, activation location, etc. This metadata will be evaluated by Off-Board and On-Board systems involved in the distribution process. |
| *Identification* | *Identification* is the ability to identify uniquely a user or a system. |
| *Integrators* | The *Integrators* are the entities in charge of building the *CCS-OB System* on behave of the operator. This includes integration, verification, validation, parametrisation, and certification of the *CCS-OB System* and covers the integration into a vehicle as well as the authorization for *Networks.* |
| *Network* | A *Network* is a system of intersecting rail routes of in a defined area. |
| *Network Identifier* | The Network Identifier is a unique identifier for one specific Network. |
| *Operational Mode* | The *Building Block* is executing its designed business logic. This includes full (normal) operation and degraded operation. |
| *Operator* | The *Operator* is the entity responsible for operating and maintaining vehicle(s) with installed *CCS-OB Deployments*. |
| *Parameters* | *Parameters* are variables that have configuration specific values. In the context of this document, we distinguish between the following *Parameters*:<br>• *Default Parameters*: parameters that are part of a *BB Configuration* received from the *BB Supplier*.<br>• *Vehicle Parameters*: Vehicle specific parameters<br>• *Fleet Parameters:* Fleet specific parameters |

| Term | Description |
|---|---|
| | • *Operator Parameters:* Operator specific parameters<br>• *Infrastructure Parameters:* Infrastructure specific parameters<br>• *System Parameters: CCS-OB System* specific parameters |
| *Software* | *Software* includes - but is not limited to - Firmware, Operating System, Runtime Environment, Application Software |

Table 1 Terms

### 2.1.1 Relations

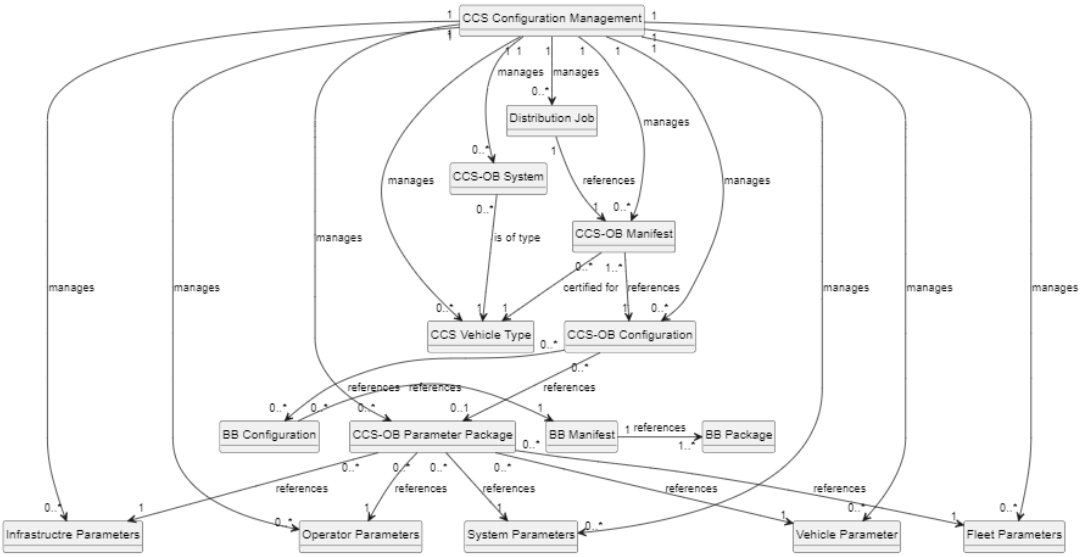


Figure 1 Entity Relationship

# 3 Configuration Management Process

The *CCS-OB Configuration Management Process* describes the conceptual mechanisms to produce, distribute, and activate *CCS Configurations.* It details all activities and stakeholders involved in the process.

As per OCORA's multi-supplier approach, a *CCS-OB Deployment* may consist of various separately sourced *Building Blocks*, integrated, configured, tested, and certified to be deployed on a well-defined *CCS Vehicle Configuration*.

In the context of this document, the *CCS-OB Configuration* is an exhaustive, unambiguous description of all *Configuration Items* required to operate a physical instance of a *CCS-OB System,* certified for deployment to a specific *CCS Vehicle Configuration* (identified via a unique *CCS Vehicle Type*) and authorised to operate on specific *Networks* (identified via unique *Network Identifiers*).

The overall *CCS-OB Configuration Management Process* can be divided into the following activities: BB Realisation, *CCS-OB System* Integration & Parameterization, Configuration Distribution, and Configuration Activation. The following diagram shows the Stakeholders involved in the *Configuration Management Process* along with their respective activities.

***Note:*** *For now, the concept deliberately excludes the initial deployment of a CCS-OB System. The process assumes a working CCS-OB Deployment that knows about its identity and as a minimum can connect to its Off-Board Configuration Server to check for configuration updates*.

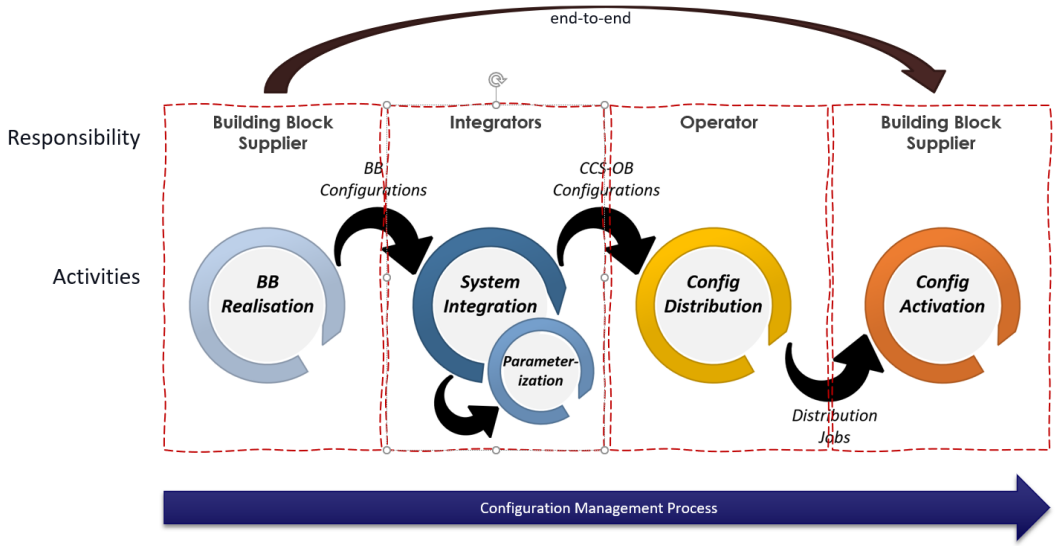


Figure 2 *Configuration Management* Stakeholders & Activities

## 3.1 Stakeholders

### 3.1.1 Building Block Supplier

The CCS On-Board system shall be composed of several separately sourced *Building Blocks* provided by various suppliers. Depending on the criticality of the function(s) provided by a particular Building Block, the correctness of its configuration is paramount for the safety of the overall system.

Each Supplier shall be responsible for having tools and processes in place to guarantee the correctness (in terms of consistency) of a released *BB Configuration*. Every released *BB Configuration* shall include metadata as part of the *BB Manifest* to allow the verification of the configuration's integrity and authenticity.

The *BB Supplier* has an end-to-end responsibility for *BB Configurations*: This means not only is he in charge of releasing *BB Configurations* to *Integrators*, but also for the activation of the *BB Configurations*.

Each *BB Configuration* shall include all procedures and required meta-data to ensure the activation of the respective *BB Configuration* on a corresponding *CCS Vehicle Configuration* may be executed in compliance with the required safety integrity level.

### 3.1.2 Integrators

The *Integrators* shall be responsible for the overall *CCS-OB Configuration* including verification, validation, and certification of the *CCS-OB System*. Besides integrating all *Building Blocks* that form the *CCS-OB System*, this also includes provisioning of all required *Parameters* including infrastructure, operator, fleet, vehicle, and system specific parameters. It also covers the integration of the CCS-OB System into a vehicle (CCS Vehicle Configuration) and the authorisation for use on a certain Network.

Even though there are multiple integration steps that typically are handled by different *Integrators*, for simplicity, this concept currently summarises all entities involved *Integrators*. In further evolutions of this document this shall be elaborated in more detail.

### 3.1.3 Operator

The Operator is the entity responsible for scheduling the distribution and activation of a *CCS-OB Configuration*. Using *Distribution Jobs*, *CCS-OB Configurations* released by the *Integrators* are amended with additional activation meta data. This meta data may include information like distribution date/time, activation location, activation trigger, etc. Jobs are associated with logical *CCS-OB Systems* and released for execution.

## 3.2 Activities

### 3.2.1 Building Block Realisation

The realisation of the CCS-OB *Building Blocks* is entirely in the responsibility of the respective Supplier(s). Even though *Building Block Configuration Management* is part of the overall end-to-end *CCS-OB Configuration Management Process*, it is fully managed within the responsibility of each individual *Building Block Supplier* as required by the TSI-CCS. Standardisation is only necessary in respect to the transition of released *BB Configurations* from Suppliers to *Integrators*.

Typical *BB Realisation* tasks are:

- Implement *Building Block* business logic and interfaces according to the OCORA specifications and compliant with the assigned safety integrity level.
- Implement *Building Block* activation (update) procedures compliant with the assigned safety integrity level.
- Create *BB Configuration* compliant with the assigned safety integrity level. The *BB Configuration* shall include all required *Software* (business logic & activation procedures) as well as possibly a default set of *Parameters*.
- Provide proprietary end-to-end safety/security layer to ensure a *BB Configuration* activation compliant with the required safety integrity level.
- Manage *BB Configurations* in proprietary *BB Configuration Management Systems* compliant with the assigned safety integrity level.
- Deliver (release and export) *BB Configurations* to *Integrators.*
- Proactively informs *Integrators* about available new *BB Configurations*.

### 3.2.2 System Integration

This activity includes the compilation, test, and certification/homologation of a *CCS-OB System* that is comprised of *Building Blocks* sourced from various suppliers. The tasks in this activity are in fact handled by several different stakeholders building on a modular certification approach. The details will be discussed in a later revision of this document, in particular the certification of the parametrization.

In the sub-activity Parametrization, specific *Parameters* required to configure a *CCS-OB System* are collected and applied to the *CCS-OB Configuration*.

In a first step, certification/homologation happens for a *CCS Vehicle Configuration.* Hence, each homologated CCS Configuration will be associated with one *CCS Vehicle Configuration* identified via a *CCS Vehicle Type.* Authorisation for a specific *Network* identified via *Network Identifier* happens in a second step.

Typical Integration tasks are:

- Import *BB Configurations* received from *BB Suppliers* into the *CCS Configuration Management System*
- Compile *CCS-OB Configurations* based on imported *BB Configurations*
- Associate *CCS-OB Configuration* with *CCS-OB Parameter Package* (see sub-activity System Parameterization)
- Test *CCS-OB Configuration* on selected *CCS Vehicle Configurations.*
- Certify *CCS-OB Configuration* for use on selected *CCS Vehicle Configurations* and authorisation for specific *Networks,* using a modular certification process
- Release certified *CCS-OB Configuration* to Operator for later distribution

#### 3.2.2.1 System Parameterization

*BB Configurations* received from suppliers are delivered with standard sets of *Parameters*. To compile a deployable *CCS-OB Configuration*, the default parameter values need to be replaced with project specific values provided by the different stakeholders.

Typical Parametrization tasks are:

- Collect *Parameters* from different stakeholders (operator, infrastructure managers, etc.)
- Compile *CCS-OB Parameter Package(s)*

### 3.2.3 Configuration Distribution

This activity includes all tasks required to distribute released *CCS-OB Configurations* to *CCS-OB Deployments* in the field. It includes associating a *CCS-OB Configuration* with a specific vehicle, defining distribution related meta data, like activation date/time, etc., and the monitoring of the distribution process.

Typical off-board tasks are:

- Create/edit *Distribution Jobs*
- Release *Distribution Jobs*
- Publish *Distribution Jobs* incl. Manifests, Packages, etc. to *CCS-OB Deployments*
- Monitor distribution status

Typical on-board tasks are:

- Periodic check of off-board system for new released *Distribution Jobs*
- Download files (Manifests, Packages, etc.) from off-board system
- Report distribution status to off-board system
- Validate integrity, perform identification check (is it for me?) and ensure authenticity of configuration (is it from a valid and trusted source?)
- Once all activation criteria are met (date/time, location, trigger, etc.), share new configuration with on-board *Building Blocks*

### 3.2.4 Configuration Activation

The activation procedure of a CSS-OB Configuration is handled by each *Building Block* independently and is *Building Block* specific. Certain standardised high-level modes and states ensure all *Building Blocks* have the same activation behaviour (see chapter 5.4).

Typical on-board tasks:

- At each start-up *Building Blocks* check if they are running a correct *CCS-OB Configuration*
- In case a new *CCS-OB Configuration* is available, each *Building Block* performs its required proprietary activation procedures that comply with the required safety integrity level.
- The *Building Blocks* report their activation status to the off-board system
- The *Building Blocks* handle activation failures in accordance with their automatic rollback capabilities and according to the defined failure handling configuration
- *Building Blocks* only get into operational service if they are running a valid *BB Configuration* as per the active *CCS-OB Configuration*.

Typical off-board tasks:

- Receive and handle configuration state updates from on-board systems
- Provide Distribution monitoring information to Operator

# 4 Involved high-level components

The following chapter describes in more detail the conceptual ideas regarding the *Configuration Management* process, focusing on the transitions between different areas of responsibility e.g., between stakeholders and between Off-board and On-Board. Figure 3 shows a detailed diagram of the end-to-end *Configuration Management* process, introducing certain key high-level components.

## 4.1 Off-Board

The Off-Board part of the *Configuration Management* process involves three different stakeholders with transitions of responsibilities and data handover in between.

### 4.1.1 Building Block Configuration Management

Each *Building Block Supplier* uses its own, non-standardised, *Configuration Management* system for managing the different *BB Configurations*. It is up to the discretion of the supplier how he manages the different configurations, providing it is handled compliant to the defined safety integrity level of the respective *Building Blocks* and in accordance with this *Configuration Management* concept.

The *BB Supplier* releases tested and approved *Building Blocks* to the *Integrators* for building a *CCS-OB System*. The format and mechanisms how *BB Configurations* are described, including the identification, authentication and integrity verification must be standardised: for example, using the described Manifest approach.

### 4.1.2 Building Block Configuration Repository

A *BB Supplier* specific repository for storing and managing *BB Configurations* compliant to the defined safety integrity level of the respective *Building Blocks*.

### 4.1.3 CCS Integration Management

Integration Management happens in the domain of the *Integrators*. On one hand the *Integrators* receive approved, released *BB Configurations* in a standardised form, on the other hand compiles tests certifies and approves *CCS-OB Configurations* consisting of several different *Building Blocks.* Integration Management also includes the system specific parametrisation of *Building Blocks* and the entire *CCS-OB System*.

Again, the CCS Integration Management shall be non-standardised and fully in the responsibility of the respective *Integrators*, providing everything is handled compliant to the defined safety integrity level of the respective *CCS-OB Configuration* and the *Integrators* follow the evolution management/optimised approval processes.

The *Integrators* release certified and approved *CCS-OB Configurations* to the Operator for distribution to physical *CCS-OB Deployments*. The format and mechanisms how *CCS-OB Configurations* are described, including the identification, authentication and integrity verification must be standardised: for example, using the described Manifest approach.

### 4.1.4 CCS Integration Repository

A repository for storing and managing *BB Configurations*, *Parameters* and *CCS-OB Configurations*, compliant to the defined safety integrity level of the respective items.

### 4.1.5 CCS Distribution Server

Distribution of *CCS-OB Configuration* is fully in the responsibility of the Operator. Certified and approved *CCS-OB Configurations* are received from the *Integrators* in a standardised format and imported into the CCS Distribution Repository.

The Operator manages and release *Distribution Jobs*. The format and mechanisms how *Distribution Jobs* are described is standardised. In addition, the full communication between the Off-Board CCS Distribution Server

and the On-Board CCS Configuration Agent is to be standardised and compliant with data management up to SIL4 and SL 4.

Considering all Off-Board relevant *Distribution Job* attributes (e.g., distribution date/time, etc.), the CCS Distribution Server offers new *CCS-OB Configurations* to physical *CCS-OB Deployments* i.e., it communicates with the On-Board CCS Configuration Agent.

### 4.1.6 CCS Distribution Repository

An Operator specific repository for storing and managing *CCS-OB Configurations* received from *Integrators*, along with *Distribution Job* created for distribution of *CCS-OB Configurations* to physical *CCS-OB Deployments*.

## 4.2 On-Board

The On-Board part of the CCS Configuration Management process is in the responsibility of two stakeholders: on one hand the Operator who oversees the distribution process, on the other hand the Building Block Supplier, responsible for implementing procedures (as part of the BB) to automatically activate the respective BB Configurations as per the active CCS-OB Configuration.

### 4.2.1 CCS Configuration Agent

The Configuration Agent is responsible for the communication between On-Board and Off-Board systems. It periodically checks with the Off-Board CCS Distribution Server if new *Distribution Jobs* are available. If so, it downloads the complete *Distribution Job* incl. all referenced files and stores it in the On-Board CCS Configuration Repository. Once the full *Distribution Job* (incl. all referenced files) has been downloaded, it informs the On-Board CCS Configuration Manager.

### 4.2.2 CCS Configuration Repository

A standardised repository that contains partially and fully downloaded *Distribution Jobs* including all referenced files.

### 4.2.3 CCS Configuration Manager

Considering all Off-Board relevant *Distribution Job* attributes (e.g., activation date/time, activation location, activation trigger, etc.) it always publishes the *CCS-OB Configuration* that must be active.

### 4.2.4 Building Block

At start-up *Building Blocks* check with the CCS Configuration Manager for the active *CCS-OB Configuration*. In case it is different from their currently active configuration, they retrieve the necessary *BB Configuration* including all referenced files from the CCS Configuration Manager and activate the new configuration.

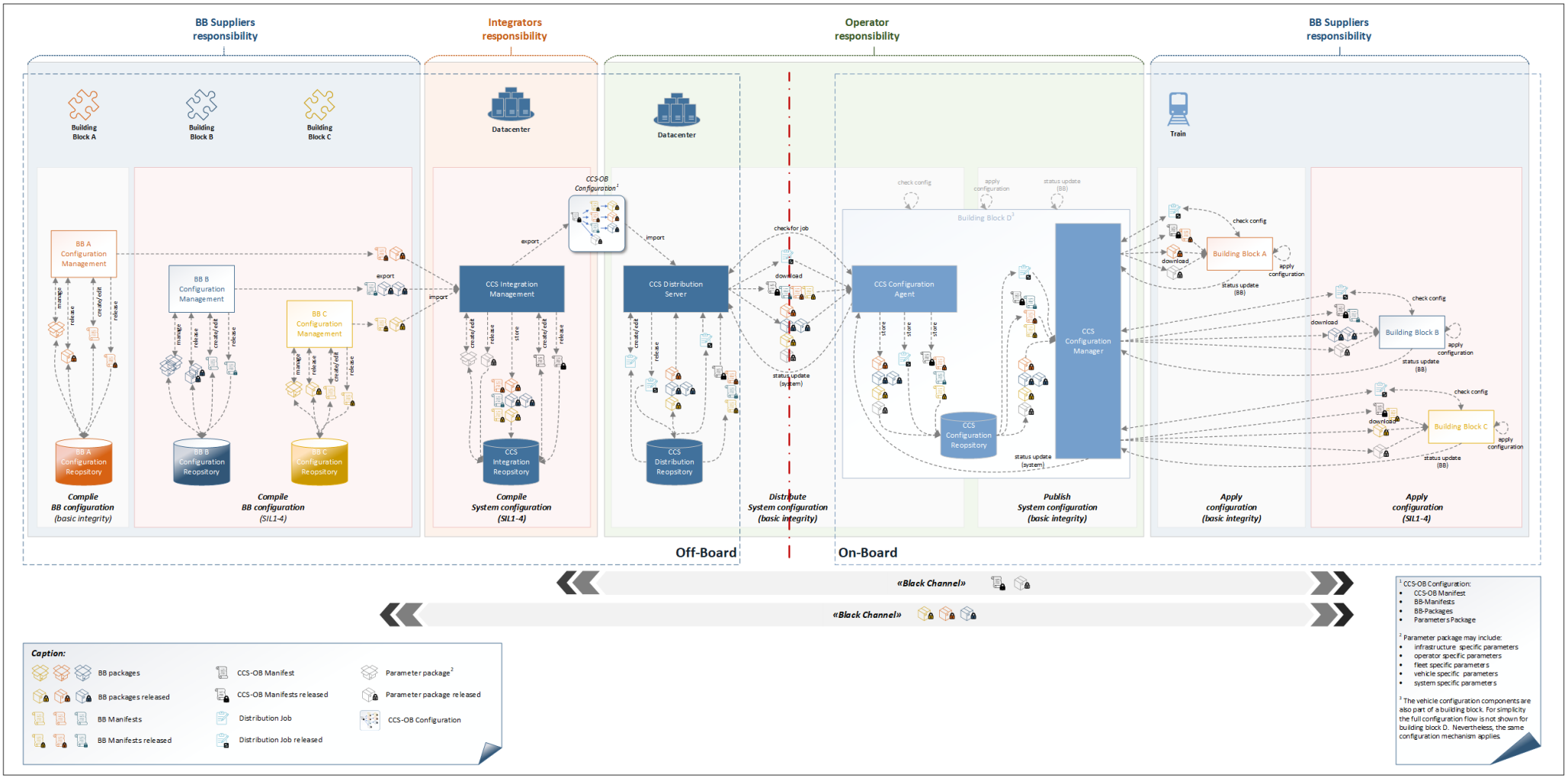


Figure 3 Detailed *Configuration Management* process

# 5 Configuration Principles

## 5.1 Identification & Authentication

Besides ensuring configuration integrity and confidentiality, the concept shall foresee mechanisms to ensure that only authenticated configurations from identified sources are applied.

Only known, authorised entities shall create, and release configurations. On one hand, a receiving entity must be able to verify the identity of the source entity to ensure it is from a trusted source. On the other hand, it must be able to check if a configuration has been specifically targeted to the receiving entity and was not supposed to be handled by some other entity.

To support the identification and authentication process, all entities involved in the *Configuration Management* need unique identifiers. Cryptographic techniques (for example certificate based) are needed for authenticating the involved systems. The mechanisms used for identification and authentication shall be standardised.

In conclusion, a configuration shall only be activated after a positive verification of the authenticity of its source, ascertainment that the source is a trusted *Configuration Management* entity, and that the configuration is targeted to the correct receiving entity.

The actual activation of a configuration as per the BB Manifest shall be handled by each *Building Block* independently. The procedure(s) required may strongly vary between *Building Block Suppliers*. Hence, the actual activation mechanisms are not to be standardised.

## 5.2 Manifest

A Manifest is an exhaustive, unambiguous human readable description of a configuration using a standardised lightweight data-interchange format e.g., JSON, XML, etc.

The information contained in a Manifest includes Meta Data and *Configuration Items* in form of referenced files: *Software* packages, *Parameter* packages or other Manifests (i.e. the CCS-OB Manifest references BB Manifests).

The Meta Data of a released Manifest shall contain information about the manifest type, the issuing entity as well as a unique identifier of the configuration it contains.

When releasing a Manifest, a cryptographically signed CRC is added to the Meta Data. This signature will help the receiver to identify and authenticate the originator and to verify the data contained in the Manifest has not been tampered or changed.

The Manifest concept shall allow for an exhaustive unambiguous representation of all *Building Block*- and *CCS-OB Configurations*.

### 5.2.1 BB Manifest

The *BB Manifest* describes the *BB Configuration* of one specific *Building Block*. Figure 4 depicts on a high-level the idea behind the *BB Manifest* – it is by no means intended to be a specification but shall provide a picture of the content. Table 2 provides an explanation of the elements shown in Table 2.

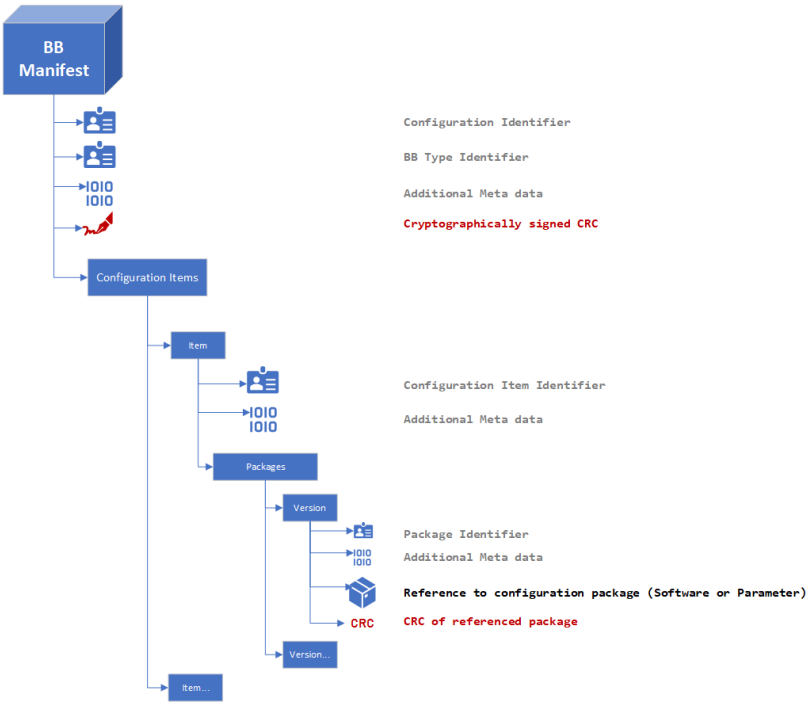


Figure 4 *Building Block Manifest*

| Element | Description |
|---|---|
| ***BB Manifest*** | An exhaustive, unambiguous human readable description of a *BB Supplier* approved *Building Block* configuration using a standardised lightweight data-interchange format e.g., JSON, XML, etc. |
| *Configuration Identifier* | UUID[1] identifying this specific *BB Configuration* |
| *BB Type Identifier* | UUID[1] identifying the OCORA *Building Block* Type e.g., LOC-OB, etc. |
| *Additional Meta data* | Supporting manifest specific data like descriptions, etc. |
| *Cryptographically signed CRC* | Cryptographically signed CRC calculated over the full content of the manifest and used for identification and authentication as well as the integrity check of the content of this manifest. It allows to detect any tampering and modifications. |
| ***Configuration Items*** | List of all *Configuration Items* that a part of this specific *BB Configuration*. As a minimum it contains two Items: one of type *Software* and one of type *Parameters.* The *Building Block Supplier* decides if its *Software* and/or *Parameters* are split into multiple Items. |
| ***Item*** | A configuration element of this *Building Block* with references to one or multiple packages containing the actual Configuration Data. |
| *Identifier* | UUID[1] identifying this specific configuration item |
| *Additional Meta data* | Supporting manifest specific data like required version to be active, alternative versions supported, rollback information, descriptions, etc. |
| ***Packages*** | List of all packages of a configuration item |
| ***Version*** | A specific version of a configuration package |
| *Configuration Identifier* | UUID[1] of this package version |
| *Additional Meta data* | Supporting manifest specific data like descriptions, etc. |
| *Reference to configuration package* | Reference to the linked configuration package file. The package content is supplier specific. |
| *CRC of referenced package* | Cyclic redundancy code of the referenced package file. Used to verify the integrity of the package file and to detect any tampering and modifications. |

Table 2 *Building Block Manifest* description

[1] The exact format of the identifiers will be defined at a later stage

### 5.2.2 CCS-OB Manifest

The *CCS-OB Manifest* describes a *CCS-OB Configuration* of one specific *CCS-OB System* that has been certified and approved for deployment to one specific *CCS Vehicle Type*. It contains references to *BB Manifests* as well as a set of deployment specific *Parameters*.

Figure 5 depicts on a high-level the idea behind the *CCS-OB Manifest* – again, it is by no means intended to be a specification but shall provide a picture of the content. Table 3 provides an explanation of the elements shown in Table 3.

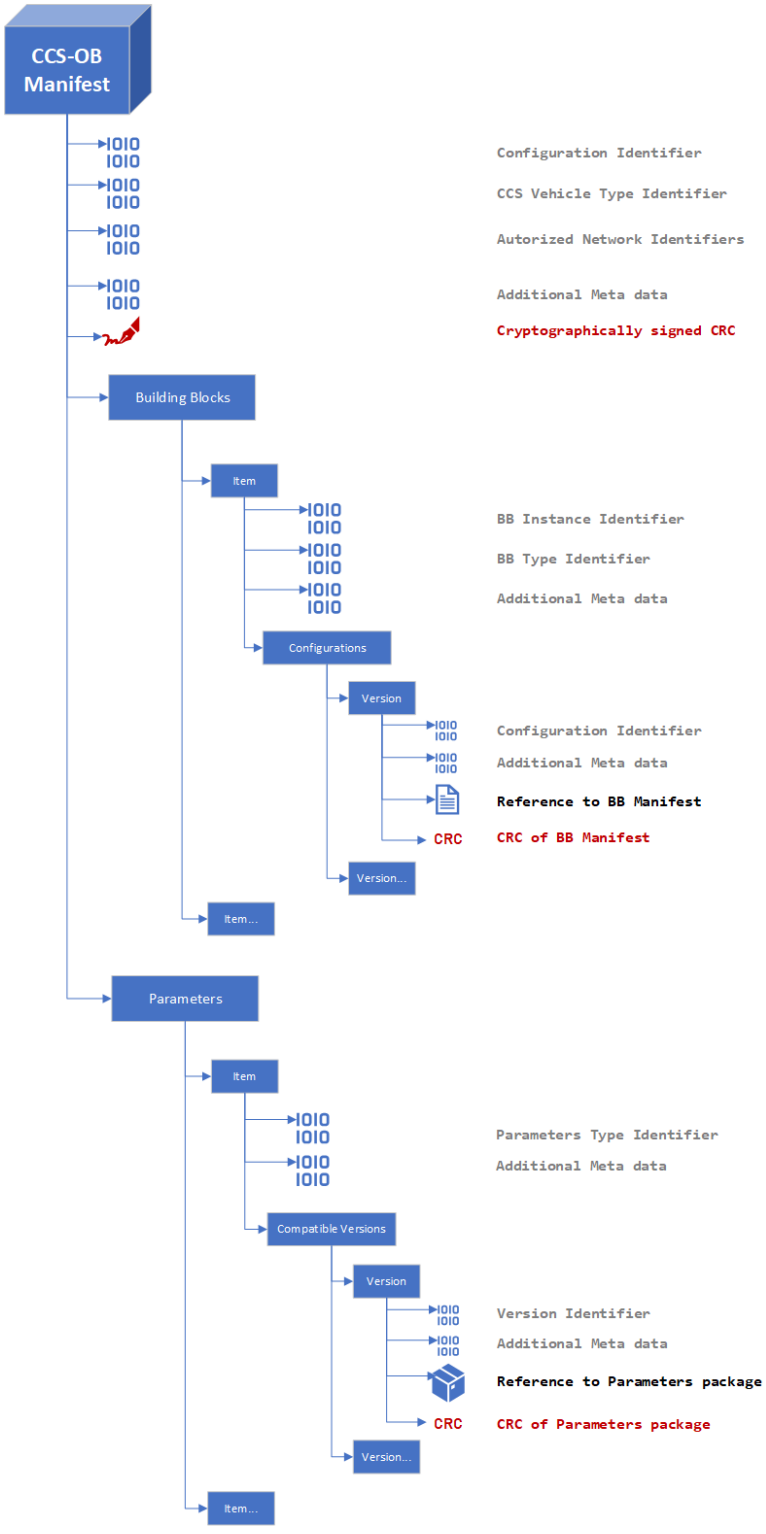


Figure 5 *CCS-OB Manifest*

| Element | Description |
|---|---|
| ***CCS-OB Manifest*** | An exhaustive, unambiguous human readable description of a *CCS-OB Configuration*, certified and approved for deployment on one *CCS Vehicle Type*, using a standardised lightweight data-interchange format e.g., JSON, XML, etc. |
| *Configuration Identifier* | UUID[1] identifying this specific *CCS-OB Configuration* |
| *CCS Vehicle Type Identifier* | UUID[1] identifying the *CCS Vehicle Configuration* that this configuration has been certified and approved for. |
| *Authorized Network Identifiers* | List of UUIDs[1] identifying the *Networks* on which the vehicle is authorized to operate. |
| *Additional Meta data* | Supporting manifest specific data like descriptions, etc. |
| *Cryptographically signed CRC* | Cryptographically signed CRC calculated over the full content of the manifest and used for identification and authentication as well as the integrity check of the content of this manifest. It allows to detect any tampering and modifications. |
| ***Building Blocks*** | List of all *Building Blocks* and their configurations that a part of this specific *CCS-OB Configuration*. |
| ***Item*** | A configuration for a specific *Building Block* Instance. |
| *BB Instance Identifier* | UUID[1] identifying this specific *Building Block* Instance. |
| *BB Type Identifier* | UUID[1] identifying the OCORA *Building Block* Type e.g., LOC-OB, etc. |
| *Additional Meta data* | Supporting manifest specific data like required version to be active, alternative versions supported, rollback information, descriptions, etc. |
| ***Configurations*** | List of configurations for this *Building Block* Instance |
| ***Version*** | A specific version of a *BB Configuration* |
| *Configuration Identifier* | UUID[1] identifying the *BB Configuration* as per the linked *BB Manifest* |
| *Additional Meta data* | Supporting manifest specific data like descriptions, etc. |
| *Reference to BB Manifest* | Reference to the linked *BB Manifest* file. |
| *CRC of BB Manifest* | Cyclic redundancy code of the referenced *BB Manifest* file. Used to verify the integrity of the *BB Manifest* file and to detect any tampering and modifications. |
| ***Parameters*** | |
| *Parameters Type Identifier* | UUID[1] identifying the *Parameters* Type |
| *Additional Meta data* | Supporting manifest specific data like required version to be active, alternative versions supported, rollback information, descriptions, etc. |
| ***Packages*** | |
| ***Version*** | |
| *Configuration Identifier* | UUID[1] of the package |
| *Additional Meta data* | Supporting manifest specific data like descriptions, etc. |
| *Reference to Parameters package* | Reference to the linked *Parameters* package file. The package format and content format are standardised. |
| *CRC of Parameters package* | Cyclic redundancy code of the referenced package file. Used to verify the integrity of the package file and to detect any tampering and modifications. |

Table 3 CCS-OnBoard Manifest description

## 5.3 Distribution Jobs

As per the proposed *Configuration Management* process, the preparation of the *CCS-OB Configuration* is handled by a different entity than the distribution. To reflect that split of responsibilities, the operator in charge of distributing the *CCS-OB Configurations* to *CCS-OB Deployments*, must not alter the *CCS-OB Configurations*.

Hence. the *Distribution Job* contains all additionally needed information for the distribution process and references a *CCS-OB Manifest* describing a specific *CCS-OB Configuration*. A similar standardised lightweight

[1] The exact format of the identifiers will be defined at a later stage

data-interchange format e.g., JSON, XML, etc. as for the Manifests could be used to describe the *Distribution Jobs*.

Typical attributes of a *Distribution Job* are, distribution date/time, activation date/time, activation location (e.g., at the depot, at a stopping point, etc.), activation trigger (e.g., automatic, driver/technician initiated, etc.), etc.

Figure 6 depicts on a high-level the idea behind the *Distribution Job* – not intended to be a specification but to provide a picture of the content. Table 4 provides an explanation of the elements shown in Figure 6.

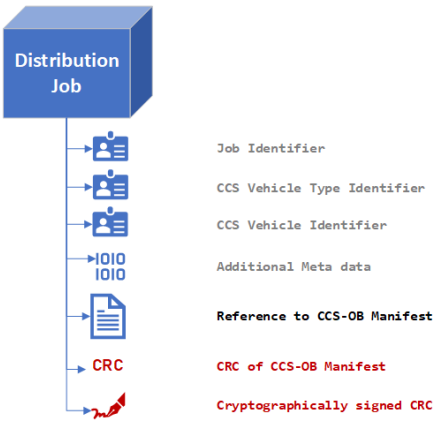


Figure 6 *Distribution Job*

| Element | Description |
|---|---|
| ***Distribution Job*** | An exhaustive, unambiguous human readable description of a distribution task to one CCS Vehicle (identified by its CCS Vehicle Identifier) using a standardised lightweight data-interchange format e.g., JSON, XML, etc. |
| *Job Identifier* | UUID identifying this specific *Distribution Job* |
| *CCS Vehicle Type Identifier* | UUID identifying the *CCS Vehicle Configuration* that this configuration has been certified and approved for. |
| *CCS Vehicle Identifier* | UUID identifying the physical CCS Deployment that this *Distribution Job* is targeted to. |
| *Additional Meta data* | Supporting manifest specific data like distribution time/date, activation time/date, activation location, activation trigger, descriptions, etc. |
| *Reference to CCS-OB Manifest* | Reference to the linked *CCS-OB Manifest* file. |
| *CRC of CCS-OB Manifest* | Cyclic redundancy code of the referenced *CCS-OB Manifest* file. Used to verify the integrity of the *CCS-OB Manifest* file and to detect any tampering and modifications. |
| *Cryptographically signed CRC* | Cryptographically signed CRC calculated over the full content of the manifest and used for identification and authentication as well as the integrity check of the content of this manifest. It allows to detect any tampering and modifications. |

Table 4 *Distribution Job* description

## 5.4 Building Block Modes & States

Certain high-level aspects of how *Building Blocks* manage configuration changes need to be standardised. Not in respect to how they activate configuration updates but rather in respect to their behaviour when processing configuration changes: it is key that all *Building Blocks* behave in a standardised, known, and predictable way.

Figure 7 shows a proposal of possible high-level Modes and States, assumed to be implemented in each *Building Block*. Modes and states handling the business logic of a *Building Block* heavily depend on the

*Building Blocks* functionality, are not to be standardised and are not in scope of this discussion. However, for completeness, the diagram shows some possible states of the *Operational Mode*.

This concept foresees that configuration changes are only applied when the *Building Block* is in a dedicated *Configuration Mode*. Switching to the *Configuration Mode* is only allowed from *Boot Mode* at start-up of the system.

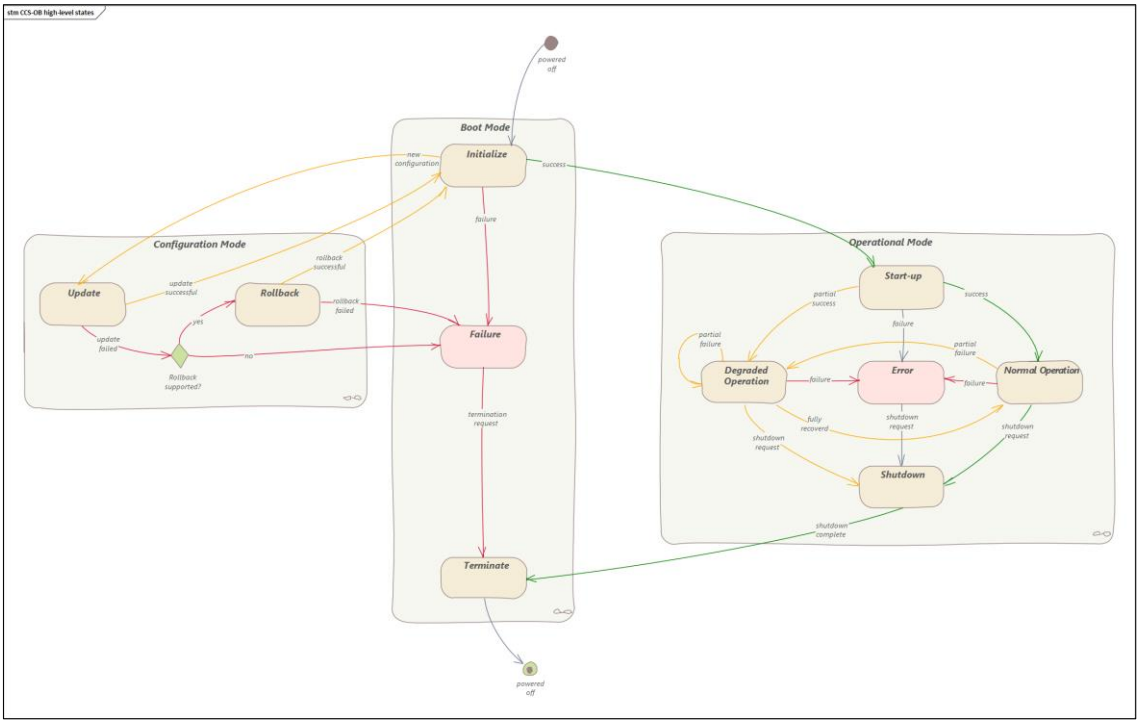


Figure 7 *Building Block* Modes & States

### 5.4.1 Boot Mode

The *Boot Mode* is the very first mode every *Building Block* gets into when it starts up and is the last mode it is in while terminating, before being powered off. It contains four distinct states: *Initialize*, *Failure* and *Terminate*.

#### 5.4.1.1 Initialize state

At power on each *Building Block* enters the *Initialize* state and performs some basic initialisation tasks (e.g., initial built-in tests). If those fail, it immediately enters the *Failure* state and remains there until it gets terminated.

Upon successful basic initialisation it checks with the Ob-Board Configuration Manager for configuration updates. If a new *CCS-OB Configuration* is available, it switches to *Configuration Mode* and transitions to the *Update* state.

In case no new *CCS-OB Configuration* is available and the active *BB Configuration* can be successfully validated, the *Building Block* switches to *Operational Mode* and enters the *Start-Up* state.

If the active *BB Configuration* is invalid, the *Building Block* transitions to *Failure* state and remains there until it gets terminated.

#### 5.4.1.2 Failure state

The *Building Block* has encountered an unrecoverable failure and needs to remain disabled. It does not perform any functional business logic.

It transitions to *Terminate* state once it receives a termination request.

#### 5.4.1.3 Terminate state

The *Building Block* performs any last remaining tasks before being turned off.

### 5.4.2 Configuration Mode

The *Configuration Mode* is the only mode where configuration changes may happen. The *Configuration Mode* can only be entered form *Boot Mode* and when completed always switches back to *Boot Mode.* It can never be entered from *Operational Mode*.

#### 5.4.2.1 Update state

The *Building Block* retrieves the *CCS-OB Manifest*, verifies the signature in the Manifest (identification and authentication) and performs an integrity check of the content.

In case the verification fails, the *Building Block* reports *configuration failure*, switches to *Boot Mode* and transitions to *Failure* state where it remains until terminated.

If the verification is successful, the *Building Block* searches in the *CCS-OB Manifest* for its configuration section (using its Instance Identifier and the BB Type Identifier).

In case the *BB Configuration* Identifier has not changed i.e., the active configuration is correct, the *Building Block* just updates the active *CCS-OB Configuration* Identifier. It reports *configuration success* and switches to *Boot Mode* and transitions to *Initialize* state.

In case the *BB Configuration* Identifier has changed, the *Building Block* retrieves the *BB Manifest* matching the required *BB Configuration* Identifier as per the *Building Blocks* configuration section in the *CCS-OB Manifest*.

It verifies the signature in the *BB Manifest* (identification and authentication) and performs an integrity check of the *BB Manifests* content.

In case the verification fails, the *Building Block* reports *configuration failure*, switches to *Boot Mode* and transitions to *Failure* state where it remains until terminated.

If the verification is successful, the *Building Block* searches in the *BB Manifest* for the referenced packages and retrieves and activates them according to the *Building Block* proprietary activation procedure.

If the activation is successful, it reports *configuration success*, updates the active *CCS-OB Configuration* Identifier, updates the active *BB Configuration* Identifier, switches to *Boot Mode* and transitions to *Initialize* state.

In case the activation fails, and rollback is allowed the *Building Block* transitions to *Rollback* state. Else, it reports *configuration failure*, switches to *Boot Mode* and transitions to *Failure* state where it remains until terminated.

#### 5.4.2.2 Rollback state

The *Building Block* checks if the previously active *BB Configuration* is listed as an alternative configuration in the *BB Manifest*.

If this is not the case, it reports *configuration failure*, switches to *Boot Mode* and transitions to *Failure* state where it remains until terminated.

If the previously active *BB Configuration* is listed as an alternative configuration, it performs a rollback as per the *Building Block* proprietary rollback process.

If the rollback fails, it reports *configuration failure*, switches to *Boot Mode* and transitions to *Failure* state where it remains until terminated.

If rollback is successful, the *Building Block* reports *configuration rolled back,* switches to *Boot Mode* and transitions to the *Initialize* state.

### 5.4.3 Operational Mode

The *Operational Mode* with its states is shown for illustration only, it is *Building Block* specific and not in scope of this concept.

# 6 Safety aspects

## 6.1 Management of configurations

Safety related systems in railway shall be designed in conformity with CENELEC standards EN 50126, EN 50128/50657 and EN 50129. All evidence shall be recorded into the SuC Safety Case where its structure is presented in EN 50129.

> *3.1.5*
>
> *configuration*
>
> *structuring and interconnection of the hardware and software of a system for its intended application*
>
> *[SOURCE: IEC 60050-821:2017, 821-12-12]*
>
> *5.3.10 Safety verification*
>
> *[…]*
>
> *Results of the planned safety verification activities shall be documented, including:*
>
> *— identity and configuration of the items verified;*
>
> *Part 1* [of the Safety Case] *— Definition of system*
>
> *This shall precisely define or reference the system, subsystem or equipment to which the Safety Case refers, including version numbers and modification status of all requirements, design and application documentation.*
>
> *When the Safety Case is issued or re-issued due to a change or reconfiguration, a delivery sheet or a release note reporting the complete configuration shall be referenced here. The delivery sheet or release note shall also list the current and previous versions of all the modified products and applications.*

Regarding the above statement, it basically means that anytime a change occurs in the SuC, its safety case shall be updated and therefore, request for a new ISA certificate. In the OCORA context, this approach is not compatible with smart modularity and fast evolvability from the Building Blocks to the overall System composed of train(s) and a network.

A methodology has been proposed by the RAMS team to help at solving this issue without degrading the overall safety level of the SuC [Evol Mngt]. Therefore, it will be possible to deploy updates of building blocks with a limited set of certification activities. Obliviously it does not concern all building block updates, only the ones considered as having no or low safety impact on the building block as defined in [Evol Mngt].

## 6.2 Transmission of data

To void RAMSS issues when deploying the present remote configuration process, it shall be ensured that the data (e.g. packages, manifests) are transferred with RAMSS mechanisms in line with the SIL, SL or RAM target SuC. Knowing that some building blocks will be SIL4 (e.g. ETP-ON, LOC-OB), the overall configuration management process shall be analysed during “Risk analysis” step (I.e. Phase 3 according to EN 50126), to determine its safety level. In addition, the communication layer from the “CCS Integration repository” to the on-board building block shall be compliant with EN 50159 and TS 50701.

A proper safety analysis will be performed in a future release of OCORA when the configuration process will be mature.

# Appendix A High-level Objectives

High-level objectives listed in this section are associated with one of the five *Configuration Management* activities.

## A1 Overall Configuration Process

| No. | The objective is … |
|---|---|
| # CP001 | • that the concept allows for new *CCS-OB Configurations* to be distributed over the air to *CCS-OB Deployments*. |
| # CP002 | • that the concept allows for new *CCS-OB Configurations* to be distributed off-line (e.g., via Maintenance Terminal) to *CCS-OB Deployments*. |
| # CP003 | • that the concept allows for a new *CCS-OB Configuration* to be activated without impacting the safety certification of the respective *CCS-OB Deployment*. |
| # CP004 | • that the concept allows for manual and automatic activation of new *CCS-OB Configurations* without impacting the safety certification of the respective *CCS-OB Deployment*. |
| # CP005 | • that the activation procedures for new *CCS-OB Configurations* are in the responsibility of the *Building Block Supplier* and are not to be standardised. |
| # CP006 | • that *Building Block Suppliers* shall support rollback in case of a failed activation of a *BB Configuration* (either automatic and/or manual on-board). |
| # CP007 | • that *Building Blocks* only enter *Operational Mode* if they comply with the required *BB Configuration* as per the active *CCS-OB Manifest*. |
| # CP008 | • that *a CCS-OB Configuration* may define 1..n permitted *BB Configurations* for the same *Building Block* in order to allow rollback in case of a failed activation of a *BB Configuration*. |
| # CP009 | • that *Building Blocks* activate *BB Configurations* independently and autonomously. |

Table 5 Overall Configuration Process Objectives

## A2 BB Realisation

| No. | The objective is … |
|---|---|
| # BR001 | • that each *Building Block Supplier* implements its *Building Block* according to the required safety integrity level |
| # BR002 | • that each *Building Block Supplier* implements its own proprietary configuration activation procedures compliant with the required safety integrity level of the respective *Building Block*. |
| # BR003 | • that each *Building Block Supplier* can use its own proprietary *Configuration Management* system. |
| # BR004 | • that each *Building Block Supplier* exports its *BB Configuration* including safety and cybersecurity means related to the SIL (Safety Integrity Level) and SL (Security Level). |
| # BR005 | • that each *Building Block Supplier* uses the same standardised lightweight data-interchange format to describe the *BB Configuration* in form of a *BB Manifest*. |
| # BR006 | • that the actual *Configuration Items* of one *Building Block* are provided in form of 1..* *Building Block Packages*. |
| # BR007 | • that each *BB Configuration* consists of a *BB Manifest* and 1..* referenced *BB Packages*. |

| No. | The objective is … |
|---|---|
| # BR008 | • that each *BB Package content* may have a supplier specific format. |

Table 6 BB Realisation Objectives

## A3 System Integration

| No. | The objective is … |
|---|---|
| # SI001 | • that the *Integrators* compile a *CCS-OB Configuration* based on the available *BB Configurations* and the necessary *Parameters*. |
| # SI002 | • that the *Integrators* ensure compatibility of *BB Configurations* which are part of a *CCS-OB Configuration*. |
| # SI003 | • that the *Integrators* associate the *CCS-OB Configuration* with a CCS-OB Vehicle Type |
| # SI004 | • that the *Integrators* are responsible to get the *CCS-OB Configuration* certified for use on the associated *CCS-OB System*. |
| # SI005 | • that the *Integrators* use a standardised lightweight data-interchange format to describe the *CCS-OB Configuration* in form of a *CCS-OB Manifest*. |
| # SI006 | • that the actual *Configuration Items* of a *CCS-OB System* are provided in form of:<br>  • 1..* *Building Block Packages*.<br>  • 1 *CCS-OB Parameter Package*<br>  • all necessary metadata to comply with the required safety integrity level |
| # SI007 | • that each CCS OB Configuration is described in form of a *CCS-OB Manifest* that references 1..* *BB Configurations*, exactly one *CCS-OB Parameter Package* including safety and cybersecurity means related to the SIL (Safety Integrity Level) and SL (Security Level). |

Table 7 System Integration Objectives

### A3.1 System Parameterization

| No. | The objective is … |
|---|---|
| # SP001 | • that the RU compiles and releases the static operator specific *Parameters* for its *CCS-OB Systems* as *Operator Parameter Packages* |
| # SP002 | • that the IMs compile and release the static infrastructure specific *Parameters* for all *CCS-OB* Systems running on their infrastructure as *Infrastructure Parameter Packages* |
| # SP003 | • that the RUs compile and release the static fleet specific *Parameters* for all *CCS-OB Systems* of a fleet as *Fleet Parameter Packages* |
| # SP004 | • that the RUs compile and release the static vehicle specific *Parameters* for a specific *CCS-OB* System as *Vehicle Parameter Packages* |
| # SP005 | • that the *Integrators* compile and release the static system specific *Parameters* for a specific *CCS-OB System* as *System Parameter Packages* |

Table 8 System Parameterization Objectives

## A4 Configuration Distribution

| No. | The objective is … |
|---|---|
| # CD001 | • that the Operator is responsible for defining the distribution and activation schedule of *CCS-OB Configurations* |
| # CD002 | • that the Operator is responsible to monitor the configuration distribution |
| # CD003 | • that *CCS-OB Deployments* check periodically if a new *CCS-OB Configuration* is available for download |
| # CD004 | • that *CCS-OB Deployments* download new *CCS-OB Configurations* when they become available (*CCS-OB Manifest* and all required *Packages*) |
| # CD005 | • that a new *CCS-OB Configuration* is published on-board only once it has been completely downloaded and all activation criteria are met (date/time, location, trigger, etc.) |
| # CD006 | • that *CCS-OB Deployments* report distribution status updates to *the CCS Configuration Management System.* |

Table 9 Configuration Distribution Objectives

## A5 Configuration Activation

| No. | The objective is … |
|---|---|
| # CA001 | • that each *Building Block* checks on its own if a new *BB Configuration* is available |
| # CA002 | • that each *Building Block* is responsible for activating new *BB Configurations* when triggered to do so. |
| # CA003 | • that each *Building Block* ensures compliance with the required safety integrity level when activating new *BB Configuration*. |
| # CA004 | • that each *Building Block* uses its own safety mechanism(s) when activating new *BB Configurations*. |
| # CA005 | • that each *Building Block* uses its own process/mechanism(s) for activating new *BB Configurations*. |
| # CA006 | • that each *Building Block* reports the currently active *BB Configuration*. |
| # CA007 | • that each *Building Block* reports update of its activation progress to the *CCS Configuration Management System* |
| # CA008 | • that each *Building Block* handles activation failures in accordance with its own rollback capability whilst considering the failure configuration as per the respective *CCS-OB Configuration*. |

Table 10 Configuration Activation Objectives