




ERJU SYSTEM PILLAR

System Interface Description_SDI-XX Base (SERA Version)



System Interface Description_SDI-XX Base (SERA Version)

Author(s)	Karl-Albrecht Klinge
Abstract	This document describes the as required per SPPRAMSS-349 - EN 50126-1:2017 phase 5 (Architecture and apportionment of system requirements) between the and the
Config Item	System Interface Description
Document ID	TCCS Service Function Diagnostics _SFD_ L5/TCCS System Interface SDI-XX Base_SERA_Version#723882  System Interface Description_SDI-XX Base (SERA Version)
Classification	Public
Status	In Review by System Pillar
Version	1.0
Revision	723882
Last Change Date	02.10.2025
Copyright	Brussels: Europe's Rail Joint Undertaking, 2025

© Europe's Rail Joint Undertaking, 2025

This document is drafted by and belongs to EU Rail.

EU Rail encourages the distribution and re-use of this document, the technical specifications and the information it contains. EU Rail holds several intellectual property rights, such as copyright and trade mark rights, which need to be considered when this document is used.

EU Rail authorises you to re-publish, re-use, copy and store this document without changing it, provided that you indicate its source and include the following: EU Rail trade mark, title of the document, year of publication, version of document.

EU Rail makes no representation or warranty as to the accuracy or completeness of the information contained within these documents. EU Rail shall have no liability to any party as a result of the use of the information contained herein. EU Rail will have no liability whatsoever for any indirect or consequential loss or damage, and any such liability is expressly excluded.

You may study, research, implement, adapt, improve and otherwise use the information, the content and the models in the this document for your own purposes. If you decide to publish or disclose any adapted, modified or improved version of this document, any amended implementation or derivative work, then you must indicate that you have modified this document, with a reference to the document name and the terms of use of this document. You may not use EU Rail's trade marks or name in any way that may state or suggest, directly or indirectly, that EU Rail is the author of your adaptations.

EU Rail cannot be held responsible for your product, even if you have used this document and its content. It is your responsibility to verify the quality, completeness and the accuracy of the information you use, for your own purposes.


This work is currently a work in progress. The content presented is subject to change as it undergoes further review, refinement, and development. Please do not consider this version as final or authoritative.

INFO: History table is not displayed, because this document is in status **doc_contentApproval**.


RULE: History table is not displayed, in statuses: { draft doc_open doc_inprogress doc_contentApproval doc_contentDecision }

CONTACT: For more information contact Administrator

Review description

Attachments	REMINDER_[ERJU SP] Request to review SC2.4 List of deliverables - Task 2_ Transversal Systems .pdf , Review and Approval Jens Kilian.pdf , Review and Approval Virgil Lostun.pdf
Comments	#1 Approval comment by Golebniak, Udo (SMO RI ML ADC I&C) on 2025-10-01 12:23 The current TCCS Design is focused on IP Needs. The usage in Traffic CS Environment is still to be discussed and must follow top-down design. Traffic CS has currently not reached the necessary level in the design. Target date for the Traffic CS Specification work is 2027.
Approvals	Kilian Jens : Waiting , SANGO Marc (SNCF / DIR TECHNOLOGIES INNOVATION ET PROJETS GROUPE / IR DIR RECHERCHE - PSF) : Waiting , DE NICOLA, Giuseppe : Waiting , KEFALAS, Georgios : Waiting , Julien Bois : Waiting , Oliver Knapp : Waiting , Wischy, Markus Alexander (SMO RI R&D F IL) : Waiting , HENON Frédéric : Waiting , TEKE, Emre : Waiting , Renato Rodrigues : Waiting , IOVINO, Salvatore : Waiting , Davinder Bhatia : Waiting , BITSCH Friedemann : Waiting , Roman R Treydel : Waiting , Golebniak, Udo (SMO RI ML ADC I&C) : Waiting , Mirko Blazic : Waiting , Benameur, Malik (SMO NEE RC-CH RI PLM SYS) : Waiting , MOTTOLA, Giuseppe Diodato : Waiting , Jack Schneider : Waiting , Zeeshan Z Ansar : Waiting , LOSTUN Virgil : Waiting , Patrick Konix : Waiting , NANNI Marco : Waiting , DE MARCO TELESE Giancarlo : Waiting , Tione, Roberto : Waiting , Andreeva-Moschen Emilia (HOLDING) : Waiting
Type of Approval	 Document Review

Approval description

Attachments	REMINDER_[ERJU SP] Request to review SC2.4 List of deliverables - Task 2_ Transversal Systems .pdf , Review and Approval Jens Kilian.pdf , Review and Approval Virgil Lostun.pdf
Approvals	LOSTUN Virgil : Waiting
Type of Approval	 Document Approval

1 Preamble	4
1.1 Scope and intended audience	4
1.2 Purpose	5
1.3 Glossary	6
2 Overview	6
2.1 Overall description	6
2.2 Non-functional characteristics / non-functional requirements	6
3 Product Oriented Structure	7
3.1 Product-Oriented Structure for SDI Definition	7
4 System Structure and Interface Modeling	8
4.1 OPC UA Communication Requirements	9
4.2 General OPC UA requirements	9
5 Data description	11
5.1 Model Metadata	11
5.1.1 Standardisation of semantics	11

5.1.2 Attribute Metadata	11
5.1.2.1 Modeling Rule	11
5.1.2.2 Short Name	12
5.1.2.3 Localised Name and Description	12
5.1.2.4 Type	12
5.1.2.5 Category	15
5.1.2.6 Temporal Measurement Backend Requirements	16
5.1.2.7 Client Update Subscriptions	17
5.1.2.8 Units	17
5.1.3 Class Meta Data	19
5.1.3.1 Short Name	19
5.1.3.2 Localised Name and Description	19
5.1.4 References ("functional and hierarchical References")	20
5.2 OPC UA model Versioning	21
5.3 SDI-GEN	23
5.4 SDI-XX Product Group Models	23
5.5 SDI-XX Modelling Process	23
5.6 Adapting SDI-XX models to an OPC UA information model	24
5.7 Behaviour description	25
5.8 Interdependencies to other interface layers	26
6 Appendix	26
6.1 Input documents	26
6.2 Standards and References	26
7 Workspace for discussions, actions and issues	26
8 Scope of interface constraints	27
8.1 Role of interface	27

1 Preamble

1.1 Scope and intended audience

This document defines the general requirements for Standard Data Interface (SDI) implementations based on the OPC UA protocol, as outlined in Part 1 of the SDI Companion Specification. It provides the foundational principles, communication expectations, data modeling conventions, and versioning rules for designing interoperable, semantically consistent information models used within railway systems and other industrial applications adopting the SDI concept.

This specification is derived from and aligned with the following key documents:

- The System Definition, which describes the overall architecture and objectives of the SDI-based integration approach.
- The System Requirements Specification, which outlines functional, operational, and non-functional expectations for SDI implementations.
- The System Architecture Definition, which provides the structural blueprint for how system components interact, communicate, and are organized.

This document serves as the foundation for both the SDI-GEN (generic) models and the domain-specific SDI-XX (product group) models. Together, these models enable standardized semantic data exchange across heterogeneous systems by defining common type hierarchies, reference structures, metadata conventions, and modeling rules using OPC UA Information Modeling principles.

Intended Audience:

- System Architects and Integrators, responsible for the design and alignment of interoperable interfaces and data models across systems.
- Product Manufacturers and Suppliers, especially those contributing equipment or software components that expose OPC UA-based interfaces according to SDI standards.
- OPC UA Modelers and Developers, implementing companion models in NodeSet2 XML, transforming UML diagrams into OPC UA Information Models, and maintaining semantic consistency.
- Operators and Maintainers, who rely on standardized and interpretable data interfaces for diagnostics, monitoring, and operational decision-making.
- Standards Bodies and Certification Authorities, reviewing and validating SDI-based models for compliance, compatibility, and long-term maintainability.

The scope of this document is limited to general modeling and communication requirements, including semantic standardization, reference modeling, versioning strategies, and structural adaptation rules for mapping domain-specific models into OPC UA. It does not cover the detailed structure of the NodeSet2 files or product-specific information models, which are provided in separate parts of the companion specification (e.g., SDI-GEN and SDI-XX documents).

1.2 Purpose

This document describes the Standard Data Interface (SDI) in accordance with the requirements of – EN 50126-1:2017, Phase 5: Architecture and Apportionment of System Requirements. The described interface defines the communication and data exchange between and , forming a fundamental part of the system architecture.

The purpose of this document is to provide:

- A general but sufficiently detailed description of the system interface;
- Justification for the architectural and design decisions taken;
- A clear explanation of how the interface contributes to meeting the overall system requirements;
- A basis for consistent implementation, integration, and verification activities across different suppliers or subsystems.

This document aims to ensure a shared understanding of the architectural principles and interface structure, focusing on the rationale that led to the selected approach. It supports traceability from high-level system requirements to concrete technical solutions, particularly those realized using OPC UA.

Note: This Interface Definition may cover specific layer(s) of the overall system interface stack where appropriate, allowing reuse across multiple subsystems or interface definitions.

Depending on the context:

- For external interfaces, it serves as an extension of the System Definition, describing interactions from the perspective of the System Under Consideration (SuC).
- For internal interfaces, it complements the System Architecture Document, detailing the internal communication structure within the SuC.

This document is a key input for the development of OPC UA companion specifications and information models (e.g., SDI-GEN and SDI-XX), and provides the foundational guidance for implementing

semantically consistent OPC UA-based interfaces across the system.

1.3 Glossary

This section provides definitions for key terms and abbreviations used throughout this document. All definitions are based on the official System Pillar Glossary. To ensure consistency and traceability, terms referenced here are aligned with existing definition work items and maintained according to the Glossary Usage Guidelines.

SDI – Standard Diagnostic Interface	A standardized OPC UA-based interface framework for the integration of diagnostic, condition monitoring, and asset-related data across different subsystems and suppliers.
OPC UA – OPC Unified Architecture	A machine-to-machine communication protocol for industrial automation, designed for interoperability, scalability, and platform-independence.
NodeSet2	The XML-based format used to describe OPC UA Information Models for exchange between tools and implementations.
RedundancyGroup	A functional entity used to organize and monitor multiple redundant system components, ensuring availability through redundancy strategies.

2 Overview

2.1 Overall description

This section provides a high-level overview of the Standard Diagnostic Interface (SDI) architecture, illustrating the key elements and interconnections between components or subsystems that communicate via OPC UA. The intent is to give readers a quick understanding of the interface's structure, scope, and communication layers, without repeating the detailed content presented in later chapters.

The SDI enables standardized, semantically consistent data exchange between and , based on a layered interface approach aligned with the OSI/ISO communication model. The interface architecture distinguishes between logical, functional, and physical views, following the Viewpoint Guidelines provided in the Systems Engineering Management Plan – Annex M2.

2.2 Non-functional characteristics / non-functional requirements

This section outlines the non-functional properties of the SDI, derived from architectural decisions and system-level requirements. These characteristics are critical to ensuring that the interface is reliable, maintainable, safe, and performs as expected in its operational environment.

Key non-functional requirements include:

Reliability & Availability

The SDI must support robust data exchange and tolerate redundancy scenarios (see SPT2TS-130121). Redundant components must be monitored using RedundancyGroup and RedundancyStatus models.

Safety

The SDI contributes to the overall safety case by ensuring semantic clarity, preventing misinterpretation of data (e.g., through prohibition of homonyms and synonyms per SPT2TS-130091), and supporting traceable model versioning (SPT2TS-130663).

Cybersecurity

Secure communication shall follow the requirements defined in SP-SEC-CommSpec.

Performance

The interface must meet real-time or near real-time data update requirements depending on subsystem needs (e.g., consistent delivery of measurement timestamps, low-latency event propagation). Performance baselines are defined in the system requirements and validated during integration testing.

Environmental Conditions

Interface components must operate reliably under the physical conditions specified in the Physical Environment Requirements Specification (e.g., temperature, vibration, EMC compliance), especially relevant for edge-deployed OPC UA servers.

Each characteristic is supported by architectural decisions and referenced documents such as: System Requirements per EN 50126

3 Product Oriented Structure

3.1 Product-Oriented Structure for SDI Definition

In order to define a consistent and interoperable Standard Diagnostic Interface (SDI) applicable across all railway systems, the interface shall be structured based on a product-oriented structure.

Definition – Product-Oriented Structure:

A structure based on how a system is implemented, constructed, or delivered, using intermediate products and completed components.

- Note 1: A product-oriented structure represents the decomposition of a system into constituent objects from a product perspective, without considering the functional and/or locational aspects of these objects.
- Note 2: Documentation organized according to a product-oriented structure emphasizes the physical grouping of system component

[SPT2TS-130770]

This structure ensures that the SDI reflects the physical realization of the system, supports lifecycle traceability of equipment, and enables alignment with manufacturing, configuration management, and asset maintenance processes.

[SPT2TS-130771]

To ensure a harmonized and maintainable implementation of diagnostic interfaces across railway systems, the Standard Diagnostic Interface (SDI) shall be structured in accordance with the hierarchical decomposition of railway systems defined by Traffic CS domain, allowing clear differentiation between shared (generic) and domain-specific (product group) elements.

[SPT2TS-130774]

System Structure and Interface Modeling Approach

To ensure consistency, modularity, and interoperability in the development of diagnostic interfaces across railway systems, the Standard Diagnostic Interface (SDI) is structured according to system decomposition principles. This structuring distinguishes between domain-specific subsystems and cross-system elements, and forms the basis for the definition of SDI-XX and SDI-GEN models.

[SPT2TS-130775]

System

A system delivers the key functions and performance characteristics of a railway vehicle or infrastructure. It is composed of multiple main systems, each contributing to the overall operation and reliability.

[SPT2TS-130778]

Subsystems – SDI-XX

A subsystem is a major functional part of a vehicle or infrastructure system, responsible for a specific group of related functions. Each main system consists of one or more such subsystems.

Domain-specific diagnostic interfaces (SDI-XX) are defined for these subsystems by domain experts using standardized modeling conventions. These models capture the unique diagnostic, condition monitoring,

and operational characteristics of each product group.

[SPT2TS-130779]

Examples of SDI-XX models include:

- SDI-P: The diagnostic interface for railway switch systems (infrastructure)
- SDI-Door: The diagnostic interface for vehicle door systems

Each of these represents a distinct product group, with specific functional behavior, diagnostics, and performance indicators.

[SPT2TS-130780]

Cross-System Elements – SDI-GEN

Cross-system elements are components or structures that are used across multiple systems or subsystems. They provide shared or general-purpose functions and are not specific to a single product group.

The SDI-GEN model defines the generic diagnostic interface for such elements, ensuring semantic reuse and avoiding duplication across SDI-XX models. It includes foundational elements such as:

- The Equipment Model for physical representation of components
- The Network Model for interface communication structures
- The Redundancy Model for managing availability and failure

[SPT2TS-130776]

Integration Principle

This structuring approach supports:

- Modularization – by separating domain-specific and generic content
- Separation of concerns – enabling focused development by domain teams
- Semantic consistency – through shared definitions and reusable patterns
- System-wide integration – enabling SDI-XX and SDI-GEN to be composed into a complete diagnostic interface model for the vehicle or infrastructure system

Note: This architectural layering forms the basis for scalable, maintainable, and interoperable OPC UA-based diagnostics across the full range of railway applications.

[SPT2TS-130777]

4 System Structure and Interface Modeling

SDI connections must use the OPC UA protocol.

Why OPC UA for the Standard Diagnostic Interface

The OPC Unified Architecture (OPC UA) is selected as the foundation for the Standard Diagnostic Interface (SDI) because it provides a comprehensive, standardized, and future-proof framework for interoperable, machine-readable diagnostics in railway systems. OPC UA addresses both data exchange and information modeling, making it ideally suited to the requirements of SDI across vehicle and infrastructure applications.

[SPT2TS-130784]

Key Reasons for Using OPC UA

Integrated Information Modeling Language (NodeSet2 Format)

OPC UA is more than a communication protocol — it includes a robust, structured information modeling language, which is formally represented in the NodeSet2 XML format. This allows complex railway system behaviors, properties, and relationships to be modeled explicitly and in a tool-interoperable way.

Semantic Clarity and Standardization

OPC UA enforces structured typing and naming, making it impossible to hide meanings behind magic numbers or undocumented identifiers. All values and their meanings (e.g., enumeration values, unit metadata, and descriptions) must be declared, promoting transparency and semantic consistency.

Reusability and Extensibility

OPC UA allows base models (like SDI-GEN) to be extended in a modular fashion for domain-specific models (SDI-XX), encouraging reuse and minimizing duplication of common concepts (e.g., equipment structure, network connections, redundancy handling).

Predefined Reference Syntax and Semantics

OPC UA defines a standardized syntax for relationships (references) between objects, properties, and types. References like HasComponent, Organizes, or HasTypeDefinition provide consistent, machine-interpretable semantics across all models.

Platform and Vendor Independence

OPC UA is designed to be platform-neutral and supports communication over multiple transport protocols (e.g., TCP, HTTPS). It decouples the data model from the hardware or vendor-specific implementations, which is critical for multi-supplier railway systems.

Security and Lifecycle Management

OPC UA natively supports authentication, authorization, encryption, and certificate management, ensuring that SDI implementations can meet cybersecurity requirements without requiring a separate framework.

Wide Industry Adoption and Tooling Support

OPC UA is widely supported by open-source and commercial tools for model creation, validation, simulation, and deployment. This ecosystem simplifies implementation, validation, and long-term maintainability of SDI models.

[SPT2TS-130782]

Conclusion

By leveraging OPC UA, the SDI architecture benefits from a mature and proven foundation that integrates data access, information modeling, and security into a single, standardized framework. This ensures that SDI implementations are semantically rich, interoperable, auditable, and future-proof, supporting the growing complexity and integration needs of modern railway systems.

[SPT2TS-130783]

4.1 OPC UA Communication Requirements

The OPC UA communication requirements apply to all trackside applications. For trainside applications, the protocol is still under discussion.

The OPC UA protocol with binary binding via OPC UA Secure Conversation [OPC] via TCP must be used to transfer data between a Diagnosable BuildingBlock and a Service Function Diagnostics and a Configurable BuildingBlock and a Service Function Configuration or Service Function Loading.

The OPC UA Client-Server model must be used. The OPC UA Server is running on the Diagnosable or Configurable BuildingBlock, the OPC UA Client is part of the Service Function Diagnostics or Service Function Configuration and Service Function Loading.

The target address(es) and the corresponding communication ports of the OPC UA client in the Service Function Diagnostics for initiating the reverse connect shall be configurable in the Diagnosable Element. Note: If two network channels are used for the service function Diagnostics Collector, at least two target addresses need to be configurable.

4.2 General OPC UA requirements

Diagnostic data is represented through BaseDataTypes and PropertyType as component of object types and EventTypes. These types are defined in Nodeset2 XML information models, which can exist in multiple versions.

For SDI, standardized models exist for generic types used across different product groups, known as SDI-GEN. Additionally, there are specific models for each product group, named SDI-[short product group name]. The SDI-[short product group name] can use the object and event types defined in SDI-GEN.

The supplier of a Diagnosable BuildingBlock must always use and apply the generic model SDI-GEN. If standardized models exist for the product group(s) hosted by the Diagnosable BuildingBlock then these must be used and applied also.

The supplier of a Diagnosable Building Block can extend the standardized models by adding attributes. These additional attributes must be incorporated into a supplier-specific object or event type, which must be defined as a subtype of the standardized SDI object or event type using the OPC UA “HasSubtype” reference. The supplier-specific information model must be provided to the IM or RU in the standardized Nodeset2 XML format, along with a specification that adheres to the metadata specification.

The supplier specific extension must not create synonyms or homonyms to variables, objects and event types that already exist in the standardized models.

All versions of Nodeset2 XML information models must be made available for creating or updating the Service Function Diagnostics. Without them, the nodes cannot be correctly interpreted or constructed within the aggregated namespace due to missing type or attribute descriptions.

The version of both standardized and supplier-specific OPC UA information models must be included in the namespaceURI, enabling differentiation between different model versions.

The instantiated objects are represented as nodes within the object space. The designated product group-specific folders must be used accordingly: the product group model of a point must be placed in the “PointProductGroupSet”, while the associated equipment model must be stored in the “PointEquipmentSet”. These folders are part of the respective standardized product group models SDI-[short product group name].

The diagnostic data is provided as nodes representing instances of these types in the object space.

The variables, object and event types provided as instances in the object space of the Diagnosable BuildingBlock must respect the semantics defined in the meta data of the model, i.e. use the diagnostic variables according to their meaning and unit specified.

The object and event types in the object space must include at least all mandatory attributes. The backend of the Diagnosable BuildingBlock OPC UA server must receive attribute values with the accuracy and temporal behaviour specified in the meta data description belonging to each respective information model.

The OPC UA Node IDs of object space instances in the OPC UA server of the Diagnosable BuildingBlock must remain unchanged after a server reset or restart, unless an OPC UA Node ID has been explicitly modified, for example, due to a configuration update.

The Service Function Diagnostics monitors the nodes of the Diagnosable Building Block by subscribing to their value changes. This ensures that the system is continuously updated with the latest diagnostic data.

For analog values, a deadband mechanism can be applied as a threshold to reduce unnecessary updates. A deadband defines a range within which small fluctuations in value are ignored, preventing excessive data transmission and processing. This is particularly useful in scenarios where minor variations are not significant for diagnostics, improving efficiency and reducing network load.

There are typically two types of deadbands used in OPC UA:

1. Absolute Deadband – A fixed numerical threshold where value changes are only reported if the difference exceeds a predefined limit.
2. Percent Deadband – A dynamic threshold based on a percentage of the monitored value’s range, which is useful for scaling with larger values.

By using deadbands, the Service Function Diagnostics can optimize data handling, ensuring that only meaningful changes trigger updates while filtering out insignificant fluctuations.

Any changes to the OPC UA object space of the Diagnosable BuildingBlock structure at runtime must be communicated to the OPC UA client in the Service Function Diagnostics through Model Change Events.

These events notify clients about structural modifications such as:

- The addition or removal of nodes.
- Changes in node attributes or references.
- Updates to object hierarchies or relationships.

By emitting Model Change Events, the system ensures that Service Function Diagnostics remains synchronized with the latest structure, enabling to adapt dynamically without requiring manual intervention

or reconnection. This is essential for maintaining data consistency.

When an SDI-XX connection is established, the previous diagnostic data must be accessible using OPC UA Historical Data Access (HDA). HDA allows clients to retrieve historical values of monitored nodes, ensuring that no diagnostic information is lost due to temporary disconnections. The OPC UA server must store and manage historical data in compliance with HDA standards, allowing efficient querying and retrieval of time-stamped diagnostic records e.g. after the connection is re-established.

5 Data description

5.1 Model Metadata

5.1.1 Standardisation of semantics

For all standardised data it is forbidden to create the same data point using a different attribute name (synonym) and metadata (e.g. unit) or to use an already standardised attribute name with a different semantics any (homonym).

[SPT2TS-130091]

Non standardised manufacturer specific data points can exist. In case these data points are standardised later the manufacturer must adapt to the standardised data point in the next release. The meaning of datapoints must be disclosed to the operator (magic numbers are forbidden). [SPT2TS-130100]

Already existing standards might be taken into account to define data points, such as **IRS 50405** for rolling stock only. [SPT2TS-130099]

5.1.2 Attribute Metadata

For each attribute or class/type the following metadata must be provided: [SPT2TS-130098]

5.1.2.1 Modeling Rule

Mandatory or optional

Modeling Rule	
Mandatory	Data labeled as “Mandatory” must be provided by the (sub-)system. This data is essential for the system’s functionality and compliance with requirements.
Optional	Data labeled as “Optional” can be provided by the (sub-)system. While not strictly required, its inclusion may be subject to specific contractual agreements or use-case requirements.

[SPT2TS-130097]

5.1.2.2 Short Name

Short Name Requirements

Requirement	Description	Details/Comment
Mandatory Language	The short name must be provided in English (en) .	Ensures a consistent naming convention across systems and international projects.
Character Restrictions	The short name must not contain spaces, special characters, or accents (e.g., @, #, ä, é).	This ensures compatibility with database column names, programming variables, and system identifiers.
Allowed Characters	The short name can only contain alphanumeric characters (a-z, A-Z, 0-9) and underscores (_).	E.g., signalStatus or signal_status, temperatureReading etc.
Length Constraint	The short name must be concise, ideally limited to 32 characters or less.	Helps prevent issues in systems with field-length limitations (e.g., database schemas).
Uniqueness	The short name must be unique within the context of the system, database, or relevant namespace.	Avoids conflicts and ambiguity when mapping to data structures or schemas.

[SPT2TS-130107]

5.1.2.3 Localised Name and Description

Localisation

For each localisation the following attributes are needed:

Attribute	
locale	Language code (e.g., en, de, fr, es, it, etc.) indicating the language for the attribute's metadata.
name	The name of the attribute in the specified language of the locale.
description	A detailed explanation of the attribute's meaning in the language of the locale

[SPT2TS-130106]

Mandatory Locale

For the locale 'en' (English), the complete set of metadata attributes (name, description) must be provided. This ensures a consistent baseline for all systems and international usage. [SPT2TS-130105]

Preferred Locales

Providing metadata in additional locales supports localised systems, improves user experience for non-English users, and aligns with regional regulatory or contractual requirements. [SPT2TS-130104]

5.1.2.4 Type

Data Types

Data Type	Description
identifier	Definition: Data points that serve as unique, unchangeable identifiers for equipment, components, or entities within the system.

Data Type	Description
	<p>ToDo: differentiate physical id and logical id</p> <p>Characteristics:</p> <ul style="list-style-type: none"> • Immutable: Once assigned, they do not change over the lifespan of the equipment or component. • Uniqueness: Ensure the precise identification of an individual item among all others. • Essential for Tracking: Crucial for asset management, maintenance records, and traceability. <p>Examples:</p> <ul style="list-style-type: none"> • Serial numbers of equipment. • MAC addresses of network devices. • Universally Unique Identifiers (UUIDs) assigned to components. • Foreign keys of asset management systems
raw data	<p>Definition: Unprocessed measurements directly obtained from sensors or input devices, including appropriate units consistent with the International System of Units (SI).</p> <p>Characteristics:</p> <ul style="list-style-type: none"> • Precision: High level of detail without any abstraction or aggregation. • Usage: Fundamental input for analyses, monitoring, and prognostic modeling. <p>Examples:</p> <ul style="list-style-type: none"> • Temperature readings (e.g., 300.5°K). • Vibration frequencies measured in Hertz (Hz). • Electrical currents measured in Amperes (A). <p>Input for : diagnosis ; status ; prognosis</p>
diagnosis	<p>Definition: Interpreted data that identify the current state or condition of a system or component, often derived from raw data.</p> <p>Characteristics:</p> <ul style="list-style-type: none"> • Discrete Values: Typically represented as enumerated states. • Interpretation: Provides meaning to raw data through analysis. <p>Examples:</p> <ul style="list-style-type: none"> • Diagnostic states such as "Normal", "TooHot", ... <p>Input for : status</p>
status	<p>Definition: An aggregated representation of the system's health or operational condition, influenced by various diagnostic inputs.</p>

Data Type	Description
	<p>Characteristics:</p> <ul style="list-style-type: none"> • Hierarchical: Can represent statuses at different system levels, from components to entire systems. • Redundancy Consideration when aggregated: A failure on a lower system level must not be a failure on a higher system level if redundancies exist. E.g. if one subsystem of a 2 of 3 system fails the system is still working and has a warning on a higher system level. • The most critical diagnostic value will often determine the overall status of the system.. <p>Examples:</p> <ul style="list-style-type: none"> • status of a subsystem/system: "OK", "Warning", "Error", "Failure" <p>Input for : prognosis ; restriction</p>
prognosis	<p>Definition: Predictive information about future system states based on current data and trends. A prediction helps anticipate potential issues or failures, allowing for proactive maintenance and decision-making to ensure system reliability.</p> <p>Characteristics:</p> <p>Predictive Models: Utilises algorithms and models to forecast future diagnoses and status.</p> <p>Decision Support: Aids in planning maintenance and avoiding failures.</p> <p>Examples:</p> <p>Predicted time to failure (e.g., "Bearing will fail in 100 operating hours").</p> <p>Remaining useful life estimations for components.</p>
restriction	<p>Definition: Operational limitations imposed due to system conditions or failures.</p> <p>Characteristics:</p> <ul style="list-style-type: none"> • Impact on Operations: Directly affects how the system can be used. • Communication: Must be conveyed to systems like the Traffic Management System (TMS). <p>Examples:</p> <ul style="list-style-type: none"> • A track section closed for maintenance, requiring rerouting. • Signal limitations where only certain aspects can be displayed. • A point may be fixed in left position - no routes using it in right position can be set.
configuration parameter	<p>Definition: Data points that define the settings or parameters of a system or component.</p> <p>Characteristics:</p> <ul style="list-style-type: none"> • Static or Dynamic: May change infrequently (e.g., firmware version) or be adjustable (e.g., threshold settings). • System Behavior: Influences how the system operates. <p>Examples:</p> <ul style="list-style-type: none"> • Threshold values for sensor alerts.

Data Type	Description
	<ul style="list-style-type: none"> • Network configuration settings <p>Input for : diagnosis</p>

[SPT2TS-130103]

5.1.2.5 Category

Data Categories

Data Category	Description
Operation	<p>Definition: Data points related to the functioning and performance of the system during its operational phase.</p> <p>Characteristics:</p> <ul style="list-style-type: none"> • Performance Metrics: Indicators that help assess efficiency and effectiveness. • Usage Data: Information on how the system is being used. <p>Examples:</p> <ul style="list-style-type: none"> • Number of cycles completed by a switch. • Average speed of trains on a track segment. • Energy consumption rates.
Equipment	<p>Definition: Data associated with the physical components and hardware of the system. A model based on a modular principle that enables suppliers to represent their specific architecture.</p> <p>Characteristics:</p> <ul style="list-style-type: none"> • Asset Information: Details about individual pieces of equipment. • Maintenance Records: Historical data on servicing and repairs. • Online Inventory: This approach allows for the creation of an online inventory from the field, serving as a "single source of truth." <p>Examples:</p> <ul style="list-style-type: none"> • Serial numbers and model types. • Materials (type of equipment) • Installation dates. • Component specifications.
Network	<p>Definition: Information pertaining to the communication and data transmission aspects of the system.</p> <p>Characteristics:</p> <ul style="list-style-type: none"> • Connectivity: Details on how components communicate. • Redundancy: Information on backup channels and failover mechanisms.

Data Category	Description
	<p>Examples:</p> <ul style="list-style-type: none"> • Network topologies and configurations. • Protocols used (e.g., RASTA protocol). • Bandwidth and latency metrics.
Environment	<p>Definition: Data regarding external conditions that can affect system performance.</p> <p>Characteristics:</p> <ul style="list-style-type: none"> • External Factors: Elements not controlled by the system but impactful. • Aggregation: Data can be collected from multiple sources for comprehensive analysis. <p>Examples:</p> <ul style="list-style-type: none"> • Weather conditions like rain, snow, temperature, humidity. <p>Environmental data can be aggregated from different elements to provide a holistic overview. This approach reduces the need for multiple sensors while still offering comprehensive environmental insights. By correlating environmental conditions with other data, valuable insights can be gained, enhancing the overall analysis and understanding of the system.</p>

[SPT2TS-130102]

5.1.2.6 Temporal Measurement Backend Requirements

Defines when the measurement backend provides updated data for an attribute.

[SPT2TS-130101]

Measurement Timing Requirements

Requirement	Description	Unit	Comment
TemporalAccuracy	How accurate the time of the measurement is (depends on the clock).	s	Optional: specify 1-sigma accuracy in seconds if applicable.
SamplingFrequency	How often a measurement is taken per unit of time	Hz	<p>Defines the rate of data collection during events or under normal operation.</p> <p>Higher or event-triggered sampling rates may apply during critical events.</p>

[SPT2TS-130110]

Data Collection Requirements

Requirement	Description	Unit	Comment
DataCollectionInterval	Maximum duration of data collection before transferring to the upper system.	s	Equivalent to the PublishingInterval in OPC UA.
DataBufferingTime	Duration data is stored locally before being overwritten by new data.	s	Operational diagnosis/status data may require longer buffering compared to non-operational raw data.

[SPT2TS-130109]

Data Transmission Requirements

Requirement	Description	Unit	Comment
MaxDataTransmissionTime	Duration of the data transfer process (depends on the amount of data and bandwidth).	s	Depends on the backend architecture and network conditions.
MaxDataTransferLatency	How long it takes to transfer the measurements from the backend to the data server.	s	Includes delays caused by buffering, transmission, and processing.

[SPT2TS-130108]

Each measurement or event shall include both a SourceTimestamp, indicating the time the data was generated at the source, and a ServerTimestamp, representing the time the data was processed by the server. [SPT2TS-130769]

5.1.2.7 Client Update Subscriptions

Defines when a client gets an update from a backend.

Transmission Requirement	Description	Unit	Comment
OnChange	Transfer derived attributes (e.g., enumerations) immediately when their value changes.	-	For stateful data like enumerations, "onChange" ensures timely updates for downstream systems.
OnThreshold	Transfer derived or raw attributes when a specific threshold is crossed.	-	Applicable for metrics like temperature or pressure exceeding safe limits.
OnEvent	Event data is captured and transferred in batch. This makes comparison of one event with another event easy to see changing behavior.		Example: PointTurnEvent; MotorStartEvent

[SPT2TS-130119]

OPC UA

In OPC UA the data subscription is not static. It can be changed during runtime. [SPT2TS-130118]

5.1.2.8 Units

Units

Not all attributes have units. An enumeration for a diagnosis or status has no unit. [SPT2TS-130117]

SI-units

SI units must be used when describing an attribute of data type raw data.

SI Base Units

Quantity	Unit Name	Unit Symbol
Length	meter	m
Mass	kilogram	kg
Time	second	s
Electric Current	ampere	A
Thermodynamic Temperature	kelvin	K
Amount of Substance	mole	mol
Luminous Intensity	candela	cd

SI Derived Units

These units are derived from the seven base units.

Quantity	Unit Name	Unit Symbol	Expression in Base Units
Frequency	hertz	Hz	s^{-1}
Force	newton	N	$kg \cdot m \cdot s^{-2}$
Pressure	pascal	Pa	$N \cdot m^{-2} = kg \cdot m^{-1} \cdot s^{-2}$
Energy	joule	J	$kg \cdot m^2 \cdot s^{-2}$
Power	watt	W	$J \cdot s^{-1} = kg \cdot m^2 \cdot s^{-3}$
Electric Charge	coulomb	C	$A \cdot s$
Electric Potential	volt	V	$W \cdot A^{-1} = kg \cdot m^2 \cdot s^{-3} \cdot A^{-1}$
Capacitance	farad	F	$C \cdot V^{-1} = kg^{-1} \cdot m^{-2} \cdot s^4 \cdot A^2$
Electric Resistance	ohm	Ω	$V \cdot A^{-1} = kg \cdot m^2 \cdot s^{-3} \cdot A^{-2}$
Conductance	siemens	S	$A \cdot V^{-1} = kg^{-1} \cdot m^{-2} \cdot s^3 \cdot A^2$
Magnetic Flux	weber	Wb	$V \cdot s = kg \cdot m^2 \cdot s^{-2} \cdot A^{-1}$
Magnetic Flux Density	tesla	T	$Wb \cdot m^{-2} = kg \cdot s^{-2} \cdot A^{-1}$
Inductance	henry	H	$Wb \cdot A^{-1} = kg \cdot m^2 \cdot s^{-2} \cdot A^{-2}$
Luminous Flux	lumen	lm	$cd \cdot sr$
Illuminance	lux	lx	$lm \cdot m^{-2} = cd \cdot sr \cdot m^{-2}$
Radioactivity	becquerel	Bq	s^{-1}
Absorbed Dose	gray	Gy	$J \cdot kg^{-1} = m^2 \cdot s^{-2}$
Equivalent Dose	sievert	Sv	$J \cdot kg^{-1} = m^2 \cdot s^{-2}$
Catalytic Activity	katal	kat	$mol \cdot s^{-1}$

Additional Notes

Plane Angle: The unit radian (rad) is used for measuring plane angles and is dimensionless but accepted within SI.

Solid Angle: The steradian (sr) is used for measuring solid angles and is also dimensionless but accepted

within SI.

[SPT2TS-130116]

5.1.3 Class Meta Data

5.1.3.1 Short Name

Short Name Requirements

Requirement	Description	Details/Comment
Mandatory Language	The short name must be provided in English (en) .	Ensures a consistent naming convention across systems and international projects.
Character Restrictions	The short name must not contain spaces, special characters, or accents (e.g., @, #, ä, é).	This ensures compatibility with database column names, programming variables, and system identifiers.
Allowed Characters	The short name can only contain alphanumeric characters (a-z, A-Z, 0-9) and underscores (_).	E.g., signalStatus or signal_status, temperatureReading etc.
Length Constraint	The short name must be concise, ideally limited to 32 characters or less.	Helps prevent issues in systems with field-length limitations (e.g., database schemas).
Uniqueness	The short name must be unique within the context of the system, database, or relevant namespace.	Avoids conflicts and ambiguity when mapping to data structures or schemas.

[SPT2TS-130115]

5.1.3.2 Localised Name and Description

Localisation

For each localisation the following attributes are needed:

Attribute	
locale	Language code (e.g., en, de, fr, es, it, etc.) indicating the language for the attribute's metadata.
name	The name of the attribute in the specified language of the locale.
description	A detailed explanation of the attribute's meaning in the language of the locale

[SPT2TS-130114]

Mandatory Locale

For the locale 'en' (English), the complete set of metadata attributes (name, description) must be provided. This ensures a consistent baseline for all systems and international usage. [SPT2TS-130113]

Preferred Locales

Providing metadata in additional locales supports localised systems, improves user experience for non-English users, and aligns with regional regulatory or contractual requirements. [SPT2TS-130112]

5.1.4 References ("functional and hierarchical References")

Hierarchical References: Structure and Redundancy Tracing

Hierarchical references define parent-child relationships like "HasChild", "Organises" or "HasEventSource" within the system architecture, typically representing structural (ProductGroupSet model) or physical (EquipmentSet model) dependencies.

Characteristics:

- Tree-like structure: Components inherit relationships from their parent elements.
- Supports redundancies: If a higher-level component fails, all dependent subcomponents are at risk.
- Used for asset management: Helps in visualising system architecture and failure propagation.

Example in Digital Twin:

Hierarchical relationship: Railway Network → Station → Interlocking System → Switch

If the interlocking system fails, it can potentially impact all switches managed by that system.

[SPT2TS-130121]

Non-Hierarchical (Functional) References: Context and Root Cause Analysis

Non-hierarchical references (or functional references) link components based on their functional interactions, rather than structure.

Characteristics:

- Graph-based relationships: More flexible than hierarchical structures.
- Supports root cause analysis: When an issue arises, functional dependencies help trace its impact.
- Captures logical and operational dependencies: Shows how systems interact beyond direct physical connections.

Example in Digital Twin:

An Equipment may "consumePower" from a PowerSupply. If that PowerSupply fails (root cause) , all supplied Equipments will fail. The PowerSupply needs to be repaired , not the supplied Equipments, although these all fail as well.

The railway system is full of these functional references. If one route is being set all components like switches, level crossings, TVP-sections and Movement Authority transmitting systems like RBC and radio or signal and the interlocking have to work with a logical "AND".

If these relationships are explicit knowledge, a higher system could calculate what to repair first to get the most operational capacity.

[SPT2TS-130120]

Why Hierarchical and Non-Hierarchical References Are Needed?

In configuration management and digital twins, both types of references are essential:

1. Hierarchical references ensure that structural dependencies are clear, helping with redundancy planning and asset management.
2. Non-hierarchical (functional) references allow dynamic system analysis, which is crucial for root cause detection, impact analysis and predictive maintenance.

For automated configuration management in railway systems, a combination of both reference types helps ensure:

- Safe system behavior (by enforcing structural constraints),
- Efficient troubleshooting (by allowing functional impact assessments),
- Better decision-making (by integrating operational dependencies into the digital twin model).

[SPT2TS-130111]

Non hierarchical references:

Forward Name and Description	Inverse Name	Remarks
Implements Reference from a physical equipment to an element of a product group to show which physical equipment with a serial number is implementing which operational product groups. In case of a multi object controller there would be a reference to each of the (logical) product groups that are implemented.	IsImplementedBy	m:n relation between equipments and product groups
IsPartOfRedundancyGroup Each RedundancyStatus entity references a RedundancyGroup, which organises and manages multiple redundant entities. The RedundancyGroup monitors the availability of its associated RedundancyStatus entities and compares the count of available entities against a predefined minimum requirement. If the number of available entities falls below this minimum, the RedundancyGroup is marked as isAvailable = false.	HasRedundantItemStatus	
Drives The concept of Drives Reference establishes a relationship where a physical output (e.g., hardware) is responsible for driving or controlling a logical entity in a product group model. This structure enables the mapping of physical components, such as actuators or motors, to the logical representations of operational systems they influence, such as points, switches, or other functional entities in the system.	IsDrivenBy	
ConnectsTo A symmetric reference indicating a physical or logical connection between two entities. Useful for modeling communication paths, cabling, or logical interfaces between components.	ConnectsTo	
IsTlsCertificateOf Establishes the relationship between a TLS certificate and the secured entity (e.g., a communication interface, server, or component using TLS).	HasTlsCertificate	
Reads Indicates that a component (e.g., a sensor or software function) reads values from another entity. Useful for expressing data acquisition flows.	IsReadBy	
ServesInterface Defines which interface (logical or physical) is served by a component. Indicates functional service provision (e.g., APIs, communication interfaces).	IsServedBy	
ProvidesPowerTo Expresses the power supply relationship. A component providing power is linked to the one consuming it. Useful for modeling power dependency or hierarchy between electrical systems.	ConsumesPowerFrom	

[SPT2TS-130136]

5.2 OPC UA model Versioning

Information on Compatible vs. Incompatible Changes

Compatible changes refer to modifications that do not break existing opc ua clients or require a new namespace.

Examples:

- Addition of new types
- Extension of an enum (adding new values, retaining old ones)
- Addition of optional attributes or references
- Enhancement of descriptions or comments

Incompatible changes refer to structural modifications that may break compatibility with existing clients and require a new namespace URI.

Examples:

- Removal or renaming of a type, attribute, or reference element
- Change of NodeId (e.g., identifier or structure)
- Removal of enum values
- Change of data types or model structures impacting client implementation

[SPT2TS-130663]

Compatible vs. Incompatible Changes

Compatible changes must not necessarily result in a change of the namespace URI.

Incompatible changes must result in a new namespace URI to explicitly indicate a breaking change.

[SPT2TS-130661]

Information on Role of the Namespace URI in the Service Function Diagnosis (SFD)

The Service Function Diagnosis (SFD) aggregates diagnostic data from multiple sources (e.g., Object Controllers).

It must be able to process multiple incompatible model versions in parallel, since system components may be updated at different times.

To distinguish these versions, the model version is derived from the namespace URI.

Example:

- <http://rail-research.europa.eu/eu-rail.sdi-tds.1/>
- <http://rail-research.europa.eu/eu-rail.sdi-tds.2/>

These represent incompatible versions of a model, even if parts of the structure are similar.

[SPT2TS-130664]

The SFD must be able to handle and evaluate multiple incompatible model versions in parallel.

[SPT2TS-130666]

Information: Implications for Model Maintenance

For compatible changes: For changes that are backward-compatible:

- The namespace URI remains unchanged.
- The publishing version is incremented in the bugfix segment, e.g., 1.0.1, 1.0.2.
- If the version introduces new extensions that require notification but remains compatible, the minor version should be incremented, e.g., 1.1, 1.2, 1.3.

For incompatible changes:

- The namespace URI must be changed, typically by incrementing the major version.
- This clearly signals to the SFD that a new, incompatible model version is in use.
- The SFD must continue to support parallel operation of older and newer model version

[SPT2TS-130669]

Incompatible model versions must be assigned a new namespace URI by incrementing the version number (e.g., 4.3 → 4.4).

[SPT2TS-130667]

The SFD must always include the latest version within a series of compatible changes.
[SPT2TS-130670]

5.3 SDI-GEN

The models are partitioned in a generic part SDI-GEN and product group specific SDI-[short product group name] according to the General OPC UA requirements above. [SPT2TS-130142]

describe generic models, contains equipment, network , motor turn data, so be used by many ...
[SPT2TS-130210]

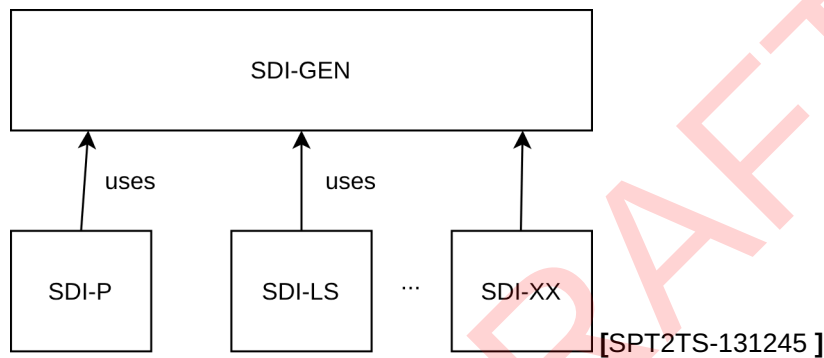
5.4 SDI-XX Product Group Models

Product Group Models are domain specific models that describe a logical component. [SPT2TS-130092]

The artifacts to describe a domain specific product group model are:

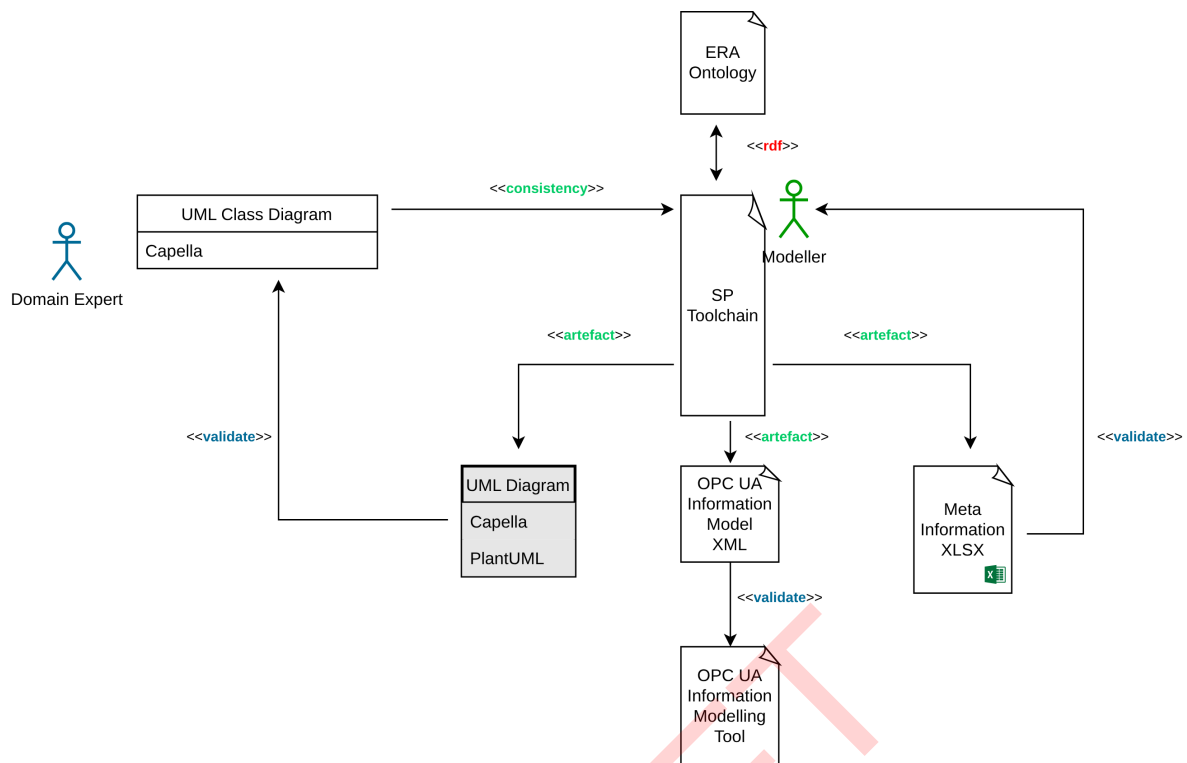
- A UML class diagram representing the static structure of the types used including the hierarchical and non-hierarchical (functional relationships (references)).
- A table according to the following meta data attributes
- A Nodeset2 XML OPC UA information model

[SPT2TS-130141]




5.5 SDI-XX Modelling Process

The modelling process from draft to artefacts is shown in the figure below: [SPT2TS-131400]



Modeling of Diagnostic Data

Process is defined in SEMP  SPPR-6083 - Define information model process

1. Partition the system into product groups: Domain experts should collectively identify and agree on the set of product groups to be modeled (e.g., Point, Doors, Checkpoints).
2. Define the static structure within each product group: For each product group, domain experts should determine the relevant subcomponents (e.g., Point → Point Machine, Door Control Unit → Door Actuator, Wheelset Profile → Wheel Profile). When defining these structures, domain experts should consider SDI-XX and SDI-GEN standards to reuse existing data points and avoid unnecessary duplication.
3. Draft the UML class diagram: A small subset of domain experts with knowledge of UML class diagrams should draft the respective diagrams based on the agreed product group partitions and static structures. These diagrams will serve as input for the modeller in the SP toolchain to generate the corresponding information model for the product group set.
4. Validate the generated information model. Domain experts should review and validate all artefacts produced by the modeller, including:

- Documentation
- Generated UML class diagram
- Metadata attributes
- Data model
- NodeSet2 XML OPC UA Information Model

5. Approval or feedback: Based on the validation, domain experts should either approve the information model or provide comments for further changes.

[SPT2TS-131401]

5.6 Adapting SDI-XX models to an OPC UA information model

When converting the UML class diagrams into a Nodeset 2 XML OPC UA information model, certain adaptations must be made due to naming conventions and structural differences. The key changes

include:

1. Naming Conventions:

- Variable and method names are capitalized with the first letter (e.g., speed in the class diagram becomes Speed in OPC UA).
- Class names receive the suffix "Type" to align with OPC UA object modeling standards (e.g., Sensor in the class diagram becomes SensorType in OPC UA).

2. Reference Adjustments:

- Reference names may differ in the OPC UA model when necessary, to conform with standardized OPC UA relationships.

3. Cardinality Constraints:

In OPC UA, enforcing an exact upper limit for cardinalities greater than 1 (e.g., 1..n) is not always possible in the Nodeset2 OPC UA information model. Unlike UML class diagrams, where cardinality constraints are explicitly defined, OPC UA primarily relies on references to establish relationships between objects. In UML, relationships between classes are constrained by specific cardinality limits (e.g., 1..3 means a maximum of 3 instances). In OPC UA, references (such as HasComponent, HasProperty, or Organizes) define the relationships but do not enforce strict upper limits. The references must still follow the requirements expressed in the UML class diagram:

- Although OPC UA cannot strictly enforce 1..n constraints, the structural relationships derived from UML class diagrams must still be reflected in the reference definitions.
- This means that every required relationship in the UML model must be mapped to an appropriate OPC UA reference type, ensuring semantic correctness.

4. Data Type Adjustments:

- Certain data types are modified to leverage OPC UA-specific structures:
- Enumeration-like variables can be converted to MultiStateDiscreteType, allowing for standardized state representation.
- Synonymous data types are recognized, e.g. real in the modeling tools is equivalent to float in OPC UA.

[SPT2TS-130156]

5.7 Behaviour description

The OPC UA server on the Diagnosable Element shall trigger the opening of the OPC UA connection by the client via reverse connect (reverse hello).

In case the Service Function Diagnostics does not start with the establishment of the OPC UA secure channel connection as a reaction to the reverse connect within 5 seconds, the Diagnosable Element system shall resend the reverse connect.

If two network channels are used for the Service Function Diagnostics shall be possible to establish the connection to the Service Function Diagnostics via both network channels.

If the OPC UA server in the diagnosable Element is configured with multiple addresses of the Service Function Diagnostics:

1. the establishment of the connection shall first be tried via the first address of the Service Function Diagnostics configured
2. when the Service Function Diagnostics does not start with the establishment of the OPC UA connection as a reaction to the reverse connect within 5 second, the Diagnosable Element shall try the next configured address of the Service Function Diagnostics.
3. If all configured addresses are tried the Diagnosable Element shall start again with the first address and repeat the process until a connection is being established.

If no connection is established but the Service Function Diagnostics expects to interact with the OPC UA server of a Diagnosable Element, the Service Function Diagnostics shall establish the OPC UA connection.

5.8 Interdependencies to other interface layers

Analysis of dependencies between levels like time-out values among OSI layers, disconnection detection and reconnection, ...

DK: Consideration of Service access points (vertical between OSI layers), APIs? This might be too detailed in some cases.

6 Appendix

6.1 Input documents

6.2 Standards and References

7 Workspace for discussions, actions and issues

Define and review template for the System Interface Definition deliverable [SPT2TS-129410]

Discussion topic: interface definition vs. interface specification

Visualisation:

<Image: diagram_20240710-1712.51421.mxg.svg>

Approach 1: interface definition (without requirements)

only static + dynamic definitions and descriptions for System A and B (example: protocol description, overview, protocol stack definition (example TACS SCI) --> statement of facts, no "shall" -> no requirements

Approach 2: interface specification (with requirements)

same as approach 1 but with shall statements for each System -> with requirements for System A and System B regarding the interface A <-> B)

General

- There should be no redundancy, means for example that behaviours (e.g. data exchanges, handshakes etc.) are not described here and in the requirements specification again.

Futher evaluation (in relation with Polarion work item content and outputs in Capella model explorer):

- recommendation use approach 1 (better because requirements are then placed at once and distributed across multiple documents which are referring)
- TACS / EULYNX uses already approach 1 in principle, easier to integrate them at later stage
- show exchange items + all the detailed of their exchange item elements (unit, data type, multiplicity, ranges, ...)
- one Polarion document per component exchange (or physical link?)?
- considering the use PVMT for attributing OSI layers
- focus on application payload, maybe not lower layers in all cases, just refer then to lower layer specs

Meeting 2024-08-01 (Dennis, Sayfeddine , Gilles)

Let's try first **approach 2**.

Rationale:

- one single point of truth (even if it is splitted into two documents, definition and specification to hold the requirements) At least all subsystems should refer to the same (set of) documents for a given interface.

- Still called it interface definition for now and see how many requirements to be included and where to stop. Important: avoid redundancy of information between interface def. and req. spec. document
Requirements for the interface def. would be: general performance values, master and slave roles, sequencing between the two interfacing systems (e.g. handshakes). The req. specs. would be then refer to this, but only for each corresponding side.

- Some requirements may be included in this documents to have a middleway to establish traceability of system requirements and (or) with the system design satisfies these requirements (to be elaborated in detail).

Glossary definitions to be considered:

 SPT2OD-6831 - FFFIS - Form Fit Functional Interface Specification

 SPT2ARC-1015 - FORM FIT FUNCTIONAL INTERFACE SPECIFICATION

 SPLI-1157 - FUNCTIONAL INTERFACES SPECIFICATION

8 Scope of interface constraints

8.1 Role of interface

DK: If there is the case that there will be a new protocol developed and defined by a domain for e.g. the application layer, should we give the possibility to move it to a separate document (e.g. protocol definition) which is referenced here like the standards? In case of new and detailed content, maybe one single interface definition gets to big.