OPC Unified Architecture

for

Safety over OPC UA

Companion Specification

Release Candidate 1.0

March 2019

Send comments to:
UAcomments@opcfoundation.org

Standard Type:	Industry Standard Specification for OPC Unified Architecture	Comments:	
Title:	Safety over OPC UA	Date:	21st of March 2019
Version:	Release Candidate 1.0	Software:	MS-Word
Editors:		Source:	OPC_UA_Companion_Specification_Safety_over_OPC_UA_20190322.docx
Owner:	OPC Foundation and PROFINET International	Status:	Draft

CONTENTS

Page

FIGURES

TABLES

Revision Log

Version	Originator	Date	Change Note / History / Reason

OPC FOUNDATION, PROFINET STANDARDIZATION GROUP

AGREEMENT OF USE

Should any provision of this Agreement be held to be void, invalid, unenforceable or illegal by a court, the validity and enforceability of the other provisions shall not be affected thereby.

This Agreement shall be governed by and construed under the laws of Germany.

This Agreement embodies the entire understanding between the parties with respect to, and supersedes any prior understanding or agreement (oral or written) relating to, this specification.

1 General

1.1 OPC Foundation

OPC is the interoperability standard for the secure and reliable exchange of data and information in the industrial automation space and in other industries. It is platform independent and ensures the seamless flow of information among devices from multiple vendors. The OPC Foundation is responsible for the development and maintenance of this standard.

OPC UA is a platform independent service-oriented architecture that integrates all the functionality of the individual OPC Classic specifications into one extensible framework. This multi-layered approach accomplishes the original design specification goals of:

- Platform independence: from an embedded microcontroller to cloud-based infrastructure
- Secure: encryption, authentication, authorization and auditing
- Extensible: ability to add new features including transports without affecting existing applications
- Comprehensive information modelling capabilities: for defining any model from simple to complex

1.2 PROFINET Standardization Group (PNO)

The PROFIBUS and PROFINET user organization (PNO: Profibus Nutzerorganisation e. V.) was founded in 1989 and is the largest automation community in the world and responsible for PROFIBUS and PROFINET, the two most important enabling technologies in automation today. The PNO is member of PROFIBUS and PROFINET International (PI).

The common interest of the PNO global network of vendors, developers, system integrators and end users covering all industries lies in promoting, supporting and using PROFINET. Regionally and globally about 1,400 member companies are working closely together to the best automation possible. No other fieldbus organization in the world has the same kind of global influence and reach.

The name of the Joint Working Group (JWG) is “Safety over OPC UA”.

1.3 Relation to safety-, security- and OPC UA-standards

This specification explains the relevant principles of functional safety for communication with reference to the IEC 61508 series as well as IEC 61784-3 and others (see Figure 1) and specifies a safety communication layer based on the OPC Unified Architecture.

Figure 1 shows the relationship between this specification and the relevant safety and OPC UA standards in an industrial environment. An arrow from Document A to Document B means “Document A is referenced in Document B”.

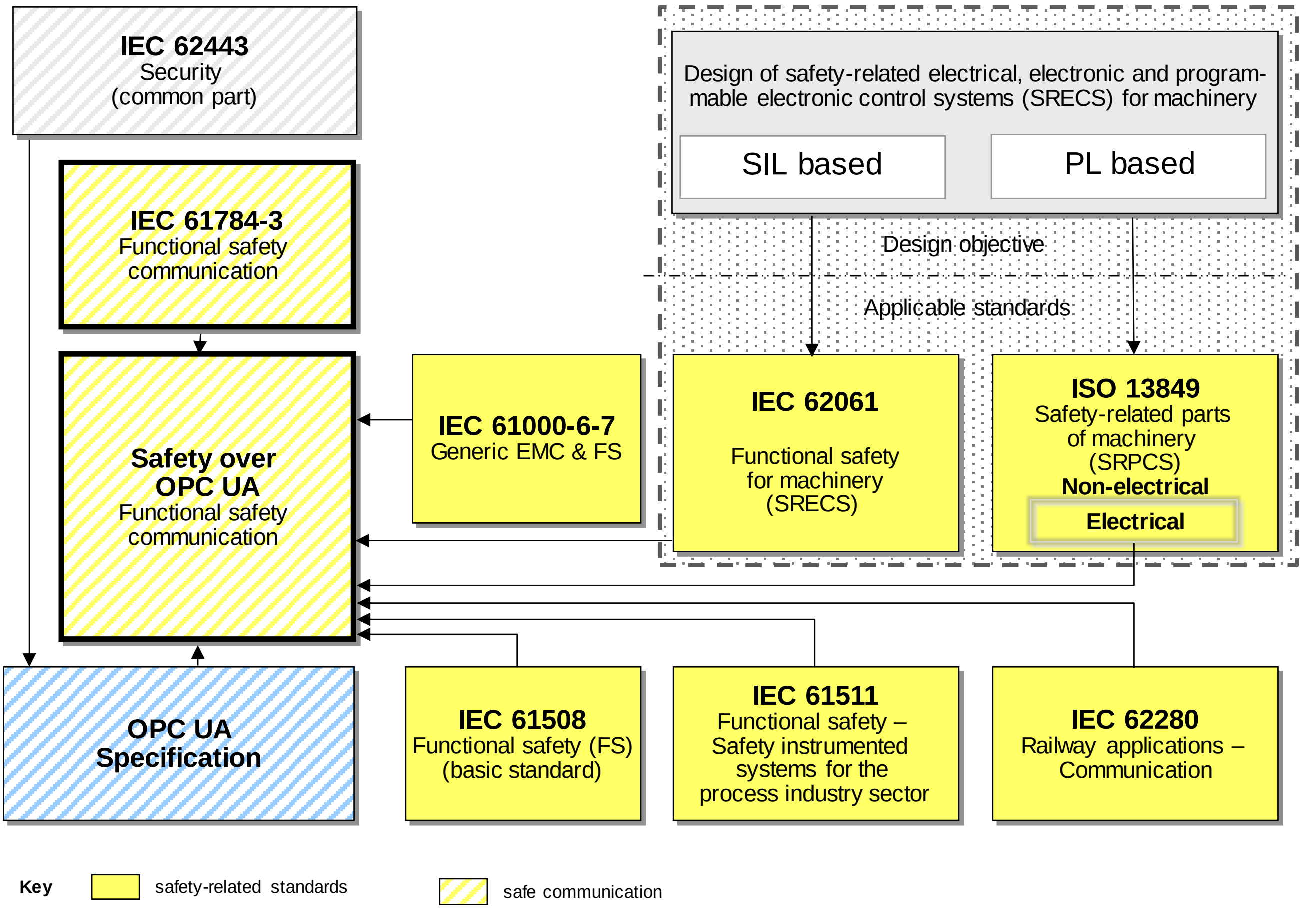


Figure 1 – Relationships of Safety over OPC UA with other standards

Safety over OPC UA do this in such a way that OPC UA can be used for applications requiring functional safety up to the Safety Integrity Level (SIL) 4.

The resulting SIL claim of a system depends on the way implementation of Safety over OPC UA is implemented within this system. That means that if a certain SIL is desired, this specification has to be implemented on a device which fulfils the requirements for this SIL as described in IEC 61508. In particular, measures against random hardware failures and systematic errors (e.g. software bugs) must be taken.

Table 1 – Intended for implementation of Safety over OPC UA

Safety over OPC UA is intended for implementation in safety devices exclusively. Simply implementing this specification in a standard device (i.e. a device not fulfilling the requirements of IEC61508) is not sufficient to qualify it as a safety device. A safety device with Safety over OPC UA shall fulfil the requirements of IEC 61508 (according the SIL-level as described) when used in live operation.

This specification does not cover electrical safety and intrinsic safety aspects. Electrical safety relates to hazards such as electrical shock. Intrinsic safety relates to hazards associated with potentially explosive atmospheres.

This specification defines mechanisms for the transmission of safety-relevant messages among participants within a network using OPC UA technology in accordance with the requirements of IEC 61508 series and IEC 61784-3 for functional safety. These mechanisms may be used in various industrial applications such as process control, manufacturing automation and machinery.

This specification provides guidelines for both developers and assessors of compliant devices and systems.

1.4 Patent declaration

The PNO draws attention to the fact that it is claimed that compliance with this document may involve the use of patents concerning the functional safety communication as follows, where the [xx] notation indicates the holder of the patent rights:

US 6907542	[SI]	System, device and method for determining the reliability of data carriers in a fail-safe system network

PNO takes no position concerning the evidence, validity and scope of these patent rights.

The holders of these patents rights have assured the PNO that they are willing to negotiate licenses either free of charge or under reasonable and non-discriminatory terms and conditions with applicants throughout the world. In this respect, the statements of the holders of these patent rights are registered with PNO.

Information may be obtained from:

[SI]	Siemens Aktiengesellschaft CT IP M&A Otto-Hahn-Ring 6 81739 München GERMANY

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights other than those identified above. PNO shall not be held responsible for identifying any or all such patent rights.

2 Normative references

The following referenced documents are relevant for the application of this specification. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

IEC 61784-3:2017, *Industrial communication networks – Profiles – Part 3: Functional safety fieldbuses – General rules and profile definitions*

IEC 61000-6-7, Electromagnetic compatibility (EMC) – Part 6-7: Generic standards – Immunity requirements for equipment intended to perform functions in a safety related system (functional safety) in industrial locations

IEC 61508 (all parts), Functional safety of electrical/electronic/programmable electronic safety-related systems

IEC 61511 (all parts), Functional safety – Safety instrumented systems for the process industry sector

IEC 62061, Safety of machinery – Functional safety of safety-related electrical, electronic and programmable electronic control systems

ISO 13849-1:2015, Safety of machinery – Safety-related parts of control systems – Part 1: General principles for design

ISO 13849 2:2012, Safety of machinery – Safety-related parts of control systems – Part 2: Validation

OPC UA Specification: Part 1: Overview and Concepts

OPC UA Specification: Part 2: Security Model

OPC UA Specification: Part 3: Address Space Model

OPC UA Specification: Part 4: Services

OPC UA Specification: Part 6: Mappings

OPC UA Specification: Part 8: Data Access

OPC UA Specification: Part 14: PubSub

3 Terms, definitions and conventions

3.1 Overview

This specification will use these concepts of OPC UA information modeling to describe Safety over OPC UA. For the purposes of this document, the terms and definitions given in OPC UA Part 1, OPC UA Part 3, OPC UA Part 6, and IEC 61784-3 as well as the following apply.

3.2 Terms

3.2.1
Cyclic Redundancy Check
CRC
<value> redundant data derived from, and stored or transmitted together with, a block of data in order to detect data corruption

<method> procedure used to calculate the redundant data

NOTE 1 to entry: Terms “CRC code” and "CRC signature", and labels such as CRC1, CRC2, may also be used in this specification to refer to the redundant data.

[SOURCE: IEC 61784-3:2017, 3.1]

3.2.2
error
discrepancy between a computed, observed or measured value or condition and the true, specified or theoretically correct value or condition

NOTE 1 to entry: Errors may be due to design mistakes within hardware/software and/or corrupted information due to electromagnetic interference and/or other effects.

NOTE 2 to entry: Errors do not necessarily result in a *failure* or a *fault*.

[SOURCE: IEC 61508-4:2010, 3.6.11]

3.2.3
failure
termination of the ability of a functional unit to perform a required function or operation of a functional unit in any way other than as required

NOTE 1 to entry: Failure may be due to an *error* (for example, problem with hardware/software design or message disruption).

[SOURCE: IEC 61508-4:2010, 3.6.4, modified – notes and figures deleted]

3.2.4
fail-safe
F
ability of a system that, by adequate technical or organizational measures, prevents from hazards either deterministically or by reducing the risk to a tolerable measure

NOTE 1 to entry: Equivalent to functional safety

3.2.5
fail-safe substitute values
FSV
values which are issued or delivered instead of process values when the safety function is set to a fail-safe state

NOTE 1 to entry: In this specification, the fail-safe substitute values (FSV) shall always be set to binary "0".

3.2.6
fault
abnormal condition that may cause a reduction in, or loss of, the capability of a functional unit to perform a required function

NOTE 1 to entry: IEV 191-05-01 defines “fault” as a state characterized by the inability to perform a required function, excluding the inability during preventive maintenance or other planned actions, or due to lack of external resources.

[SOURCE: IEC 61508-4:2010, 3.6.1, modified – figure reference deleted]

3.2.7
flag
a non-safe bit indication to perform test information.

3.2.8
Globally Unique Identifier
GUID

A universally unique identifier (GUID) is a 128-bit number used to identify information in computer systems. The term globally unique identifier (GUID) is also used

[SOURCE: Wikipedia]

3.2.9
MonitoringNumber
MNR
a means used to ensure the correct order of transmitted safety PDUs and to monitor the communication delay. The MNR starts at a random value and counts up with each request. It rolls over to a minimum threshold value that is not zero.

NOTE 1 to entry: Instance of *sequence number* as described in IEC 61784-3.

NOTE 2 to entry: The transmitted MNR is secured via the transmitted CRC signature of the SDataPDU

3.2.10
OPC UA Mapper
part of the Safety over OPC UA implementation which maps the SPDU to the actual OPC UA services. Depending on which services are used (e.g. client/server or pub/sub), different mappers can be specified

3.2.11
performance level
PL
discrete level used to specify the ability of safety-related parts of control systems to perform a safety function under foreseeable conditions

[SOURCE: ISO 13849-1:2015, 3.1.23]

3.2.12
process values
PV
input and output data (in a safety PDU) that are required to control an automated process

3.2.13
qualifier
Q
Qualifier is an attribute (bit or boolean), indicating whether the corresponding value is valid or not (e.g. being a fail-safe substitute value)

3.2.14
residual error probability
RP
probability of an error undetected by the SCL safety measures

[SOURCE: IEC 61784-3:2017, 3.1]

3.2.15
residual error rate
statistical rate at which the SCL safety measures fail to detect errors

[SOURCE: IEC 61784-3:2017, 3.1]

3.2.16
safety communication layer
SCL
communication layer above the OPC UA Communication Stack (OPC UA Server API or OPC UA Client API) that includes all necessary additional measures to ensure safe transmission of data in accordance with the requirements of IEC 61508.

The SafetyProvider together with the SafetyConsumer represent the SCL.

[SOURCE: IEC 61784-3:2017, 3.1 modified]

3.2.17
SafetyConsumer
The SafetyConsumer sends a request to the SafetyProvider and checks the response (SPDU) for timeliness, authenticity and data integrity in accordance with IEC 61784-3

3.2.18
safety data
SData
application data transmitted across a safety network using a safety protocol

NOTE 1 to entry: The Safety Communication Layer does not ensure the safety of the data itself, but only that the data is transmitted safely.

3.2.19
safety function response time
SFRT
worst case elapsed time following an actuation of a safety sensor connected to a fieldbus, until the corresponding safe state of its safety actuator(s) is achieved in the presence of errors or failures in the safety function

NOTE 1 to entry: This concept is introduced in IEC 61784-3:—, 5.2.4 and is addressed by the functional safety communication profiles defined in this specification.

[SOURCE: IEC 61784-3:2017, 3.1]

3.2.20
safety integrity level
SIL
discrete level (one out of a possible four), corresponding to a range of safety integrity values, where safety integrity level 4 has the highest level of safety integrity and safety integrity level 1 has the lowest level of safety integrity

NOTE 1 to entry: The target failure measures (see IEC 61508-4:2010, 3.5.17) for the four safety integrity levels are specified in Tables 2 and 3 of IEC 61508-1:2010.

NOTE 2 to entry: Safety integrity levels are used for specifying the safety integrity requirements of the safety functions to be allocated to the E/E/PE safety-related systems.

NOTE 3 to entry: A safety integrity level (SIL) is not a property of a system, subsystem, element or component. The correct interpretation of the phrase “SIL*n* safety-related system” (where *n* is 1, 2, 3 or 4) is that the system is potentially capable of supporting safety functions with a safety integrity level up to *n*.

[SOURCE: IEC 61508-4:2010, 3.5.8]

3.2.21
safety measure
measure to control possible communication *errors* that is designed and implemented in compliance with the requirements of IEC 61508

NOTE 1 to entry: In practice, several safety measures are combined to achieve the required safety integrity level.

NOTE 2 to entry: Communication *errors* and related safety measures are detailed in IEC 61784-3:2017, 5.3 and 5.4.

[SOURCE: IEC 61784-3:2017, 3.1]

3.2.22
safety PDU
SPDU
PDU transferred through the safety communication channel

NOTE 1 to entry: The SPDU may include more than one copy of the safety data using differing coding structures and hash functions together with explicit parts of additional protections such as a key, a sequence count, or a time stamp mechanism.

NOTE 2 to entry: Redundant SCLs may provide two different versions of the SPDU for insertion into separate fields of the OPC UA frame.

[SOURCE: IEC 61784-3:2017, 3.1]

3.2.23
SafetyProvider
The SafetyProvider answers the request from the SafetyConsumer with the current Safety Data together with the additional data to give the SafetyConsumer the ability to check these SPDU for timeliness, authenticity and data integrity in accordance with IEC 61784-3

3.2.24
SDataBaseID
Authenticity ID which is in a hierarchical view in the level above the SDataProvID. The typical use case is one SDataBaseID at machine-level for a lot of SafetyProviders at field-level. The pair of SDataBaseID together with the SDataProvID is used to check the authenticity of incoming data

NOTE 1 to entry: Together with the SDataProvID it is the instance of *connection authentication* as described in IEC 61784-3.

3.2.25
SDataProvID
Within an area of same SDataBaseID the SafetyProviders shall have unique SDataProvIDs. The pair of SDataProvID together with the SDataBaseID is used to check the authenticity of incoming data by the SafetyConsumer

NOTE 1 to entry: Together with the SDataBaseID it is the instance of *connection authentication* as described in IEC 61784-3.

3.2.26
SPDU sample rate

Number of SPDUs checked by the receiving SCL per hour

3.3 Abbreviations and symbols

BSC	Binary Symmetric Channel
CRC	Cyclic Redundancy Check
FIT	Failure In Time (equals 10E-9 failure per hour)
FS	Functional Safety
FSV	Fail-safe substitute Values
HMI	Human-machine interface

IACS	Industrial Automation and Control System	
ID	Identifier	
LSB	Least significant bit	
MNR	MonitoringNumber	
MSB	Most significant bit	
NSR	Non Safety Related	
OA	Operator Acknowledge	
PDU	Protocol Data Unit	[ISO/IEC 7498-1]
Pe	Bit error probability	
PL	Performance Level	[ISO 13849-1]
PLC	Programmable Logic Controller	
SoOPC	Safety over OPC UA	
RP	Residual Error Probability	
PV	Process Values	
WDTO_SA	maximum sample time period of a cyclic SafetyConsumer application	
SAPI	Safety Application Program Interface	
SCL	Safety Communication Layer	
SFRT	Safety function Response Time	
SIL	Safety Integrity Level	[IEC 61508-4:2010]
SIS	Safety Instrumented Systems	
SL	Security Level	[IEC 62443]
SMS	Security Management System	[IEC 62443]
SData	Safety Data	
STrailer	Safety Trailer	
SPDU	Safety PDU	
SPI	Safe Parameter Interface	
SR	Safety Related	
WDTO_SoOPC	Watchdog of Safety over OPC UA	

3.4 Suffixes at Variables

.._C	Control Input from application program
.._P	Parameter Input to SafetyProvider and SafetyConsumer driver
.._S	Status Output to application program
.._SF	Variable as part of the control safety Flags
.._NF	Variable as part of the nonsafety Flags

3.5 Conventions

3.5.1 Conventions in this specification

In this specification, the following conventions are used:

- The abbreviation "F" is an indication for safety related items, technologies, systems, and units (fail-safe, functional safe).
- The default data that shall be used in case of unit failures or errors, are called Fail-safe substitute Values (FSV) and are set to binary "0".

- Reserved bit ("res") shall be set "0" and ignored by the receiver for avoiding problems with future versions of Safety over OPC UA.
- Terms and names are often written in PascalCase (the practice of writing compound words or phrases in which the elements are joined without spaces, with each element's initial letter capitalized within the compound). Terms or names where two capital letters of abbreviations are in sequence or for separation to a suffix are written with underscores in between.
- Notation 0x… represents a hexadecimal value.

3.5.2 Conventions on CRC calculation

- Any CRC signature calculation is starting with a preset value of "1".
- Any CRC signature calculation resulting in a "0" value, will use the value "1" instead.
- SPDU with all values (incl. CRC signature) being==0 shall be ignored by the receiver (SafetyConsumer and SafetyProvider).

3.5.3 Conventions in state machines

Table 2 – Conventions used in state machines

Convention	Meaning
:=	Assignment: value of an item on the left is replaced by value of the item on the right.
<	Less than: a logical condition yielding TRUE if and only if an item on the left is less than the item on the right.
<=	Less or equal than: a logical condition yielding TRUE if and only if an item on the left is less or equal than the item on the right.
>	Greater than: a logical condition yielding TRUE if and only if the item on the left is greater than the item on the right.
>=	Greater or equal than: a logical condition yielding TRUE if and only if the item on the left is greater or equal than the item on the right.
==	Equality: a logical condition yielding TRUE if and only if the item on the left is equal to an item on the right.
<>	Inequality: a logical condition yielding TRUE if and only if the item on the left is not equal to an item on the right.
&&	Logical "AND" (Operation on binary values or results)
||	Logical "OR" (Operation on binary values or results)
⊕	Logical "XOR" (Operation on binary values or digital values)
[..]	UML Guard condition, if and only if the guard is TRUE the respective transition is enabled

4 General

4.1 Introduction for Safety over OPC UA

Safety over OPC UA specifies a safety communication layer based on the OPC Unified Architecture.

Safety over OPC UA is an application-independent, general solution. Application-dependent companion specifications will be specified by application-experts later on (for example: robot safety, AGVs, (Automated Guided Vehicles), etc.)

4.2 Safety functional requirements

The following requirements apply for the development of the Safety over OPC UA technology.

a) Safety communication shall be suitable for Safety Integrity Level up to SIL4 (see IEC 61508) and PL e (see ISO 13849-1).

b) Combination of SIL 1 – 4 Safety over OPC UA devices as well as non-safety devices on one communication network.

c) Implementation of the safety transmission protocol shall be restricted to the safety layer.

d) The transmission duration times shall be monitored by the safety layer.
e) Safety communication shall meet the requirements of IEC 61784-3:2017.
f) Safety over OPC UA stack is intended for implementation in safety devices exclusively.

4.3 Communication structure

Safety over OPC UA is based on:

- the standard transmission system OPC UA
- an additional safety transmission protocol on top of this standard transmission system

Safety applications and standard applications are sharing the same standard OPC UA communication systems at the same time. The safe transmission function comprises all measures to deterministically detect all possible faults / hazards that could be infiltrated by the standard transmission system or to keep the residual error rate under a certain limit. This includes

- Random malfunctions, for example due to electromagnetic interference on the transmission channel;
- Failures / faults of the standard hardware;
- Systematic malfunctions of components within the standard hardware and software.

This principle delimits the assessment effort to the "safe transmission functions". The "standard transmission system" ("Black Channel") does not need any additional functional safety assessment.

The basic communication layers of Safety over OPC UA are shown in Figure 2.

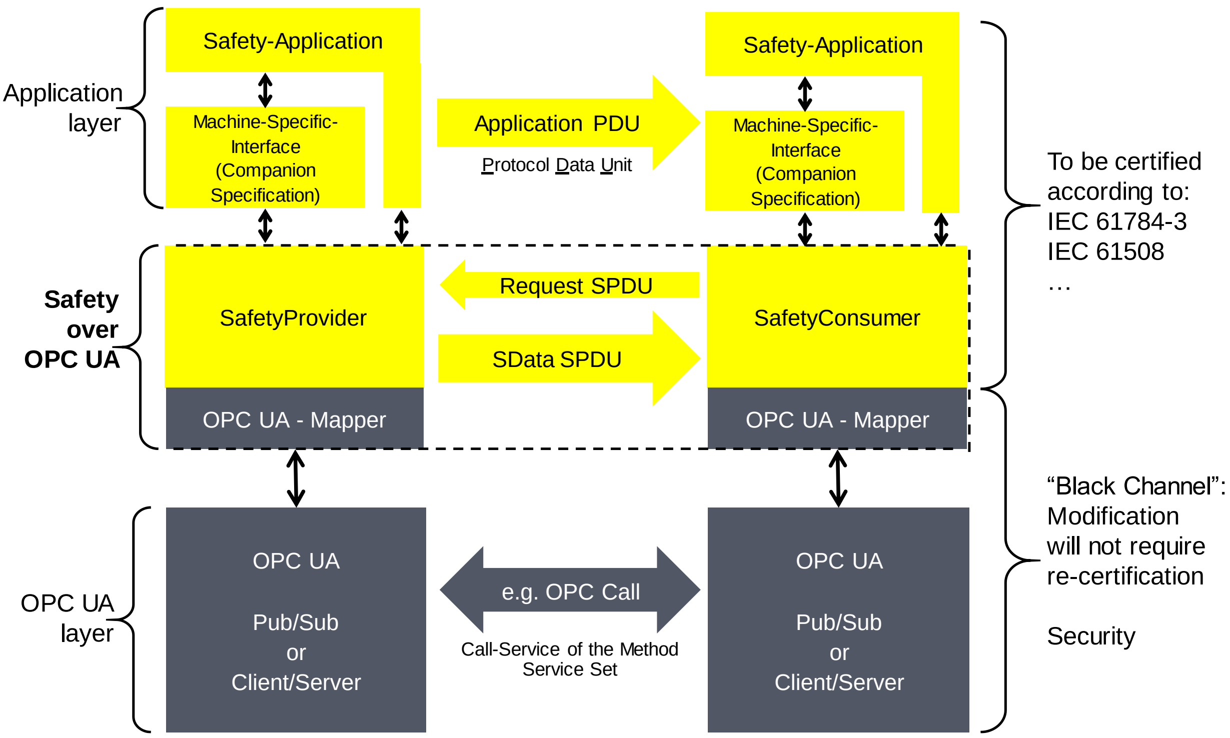


Figure 2 – Safety layer architecture

Summary of the Safety layer architecture:

Part: Application layer

The Safety application is either connected directly with the SafetyProvider / SafetyConsumer, or it is connected with a Machine-Specific-Interface, which is specified in a Companion Specification.

The Safety application layer shall be designed and implemented according IEC 61508.

The Safety application layer is not in the scope of this specification.

Part: Safety over OPC UA

This layer is in the scope of this specification.

The two basic building blocks are SafetyProvider and SafetyConsumer. These together are the SCL.

The transmission of safety data is a point-to-point communication (unidirectional). One unidirectional channel needs bidirectional communication, to guarantee SFRT.

For connection establishment: a SafetyConsumer connects to the SafetyProvider with a known ID, a SafetyProvider does not need to know the ID of the SafetyConsumer.

Error detection: a SafetyProvider is designed such it does not need error detection, the Safety-Consumer performs all required error detection.

If the safety application at the SafetyProvider needs information from the application at the SafetyConsumer, an additional reverse SafetyProvider – SafetyConsumer connection has to be established.

The OPC UA Mapper is used to be independent between the safety layer and OPC UA communication “Pub/Sub” or “Client/Server”.

Part: OPC UA layer

The SafetyProvider side uses either OPC UA Server, or in the future it might be an OPC UA Publisher.

The SafetyConsumer side uses either the OPC Call service, or in the future it might be an OPC UA Subscriber.

4.4 Implementation aspects

All technical measures for error detection of Safety over OPC UA shall be implemented within the SCL in devices designed in accordance with IEC 61508 and shall meet the target SIL.

4.5 Features of Safety over OPC UA

1) Runs on top of:
    a) OPC UA Client/Server (TCP/IP) with the Method Service Set
    b) The future OPC UA Pub/Sub
    c) goal: no modification on existing OPC UA framework
2) Modest requirements on safety network nodes:
    a) no clock synchronization needed (no requirements regarding the accuracy between clocks at different nodes)
    b) to support watchdog time monitoring a sufficient accuracy of the internal timer at the SafetyConsumer is required
3) “Black Channel” principle: No functional safety requirements for non-safety network nodes and OPC UA stack
4) “Dynamic” systems:
    a) Safety communication partners may change during runtime,
    b) and/or increased/decreased in number
5) Specified diagnosis texts are used
6) Security is part of OPC UA and is not covered by this companion specification, see 4.7
7) Safety communication and standard communication shall be independent. However, standard devices and safety devices shall be able to use the same communication channel at the same time
8) Safety communication may use a single-channel communication system. Redundancy may be used optionally for increased availability
9) For diagnostic purposes, the SPDU sent and received is also stored in OPC UA variables accessible by remote read accesses, both at SafetyProvider and SafetyConsumer side
10) The state machines of Safety over OPC UA shall be independent from the used part of the OPC UA Mapper

11) Length of user data: 1-1500 bytes, structures of basic data types, see 6.3

12) Ready for Wireless transmission channels.

4.6 Requirements on OPC UA

The (atomic) consistent Data exchange for the SPDU from SafetyProvider to SafetyConsumer is required.

4.7 Security policy

In the final application an appropriate security environment needs to be in place for protecting both the operational environment and the safety-related systems.

For this purpose, a threat and risk analysis (TRA) according to IEC 62443 needs to be carried out on a final application system level.

An adequate reduction of risk against malevolent attacks is a necessary prerequisite for a meaningful application of this specification. This specification does not describe any measures which will lower the risk of malevolent attacks, but addresses the topic “function safety”, only.

During compliance tests to this standard, security aspects are not part of the scope.

4.8 Safety measures

For the realization of Safety over OPC UA, the following measures were chosen:

- consecutive MonitoringNumber without any gaps
- watchdog time monitoring with acknowledgment;
- Identifier for each SafetyProvider
- Signature over the structure of SData
- Means to detect masquerade of SPDU from SafetyProvider with lower SIL
- Cyclic redundancy check (CRC) for data integrity.

The MonitoringNumber uses a range that is large enough to secure any malfunction caused by message storing network elements.

The watchdog time monitoring at the SafetyConsumer requires a cyclic call of the SafetyConsumer, see Table 28.

The ID for SafetyProvider is established for authentication reasons.

These safety measures, also mentioned in Table 3, to detect possible transmission errors and are a significant component of the Safety over OPC UA profile. The selection in Table 3 of the generic safety measures listed in IEC 61784-3:2017, 5.5 is required for Safety over OPC UA to satisfy functional safety requirements.

The safety measures shall be processed and monitored within the SCL.

Table 3 – Deployed measures to detect communication errors

Communication error	Safety measures			
	MonitoringNumber [a]	Timeout with receipt [b]	ID for SafetyProvider [c]	Data integrity check [d]
Corruption	–	–	–	X
Unintended repetition	X	X	–	–
Incorrect sequence	X	–	–	–
Loss	X	X	–	–
Unacceptable delay	–	X	–	–
Insertion	X	–	–	–
Masquerade	X	–	X	X
Addressing		–	X	–
Out-of-sequence	X	–	–	–

[a] Instance of "sequence number" of IEC 61784-3.

[b] Instance of "time expectation" (Timeout) and "feedback message" (Receipt) of IEC 61784-3.

[c] Instance of "connection authentication" of IEC 61784-3.

[d] Instance of "data integrity assurance" of IEC 61784-3, based on CRC signature.

The SafetyConsumer is specified that in case of communication errors according to Table 3, a defined fault reactions will occur.

If the SafetyConsumer detects a CRC error, or an MNR error, or an ID error and the error rate is lower than the limit, the state machine repeats the request, see 11.4.
In all other cases, the SafetyConsumer will deliver fail safe substitute values to the safety application instead of actual process values. In addition, an indication at the Safety Application Program Interface is set which can be queried by the safety application.

4.9 OPC UA profiles for safety components

This is a placeholder section for defining different OPC UA profiles. Each profile will specify a set of mandatory and optional services, taken from this specification.

Components should indicate the services/roles which are implemented.

Table 4 – OPC UA profiles for safety components

	SafetyProvider	SafetyConsumer	SafetyProsumer Classic (Host)	SafetyProsumer Classic (Device)
Service Name	Provider	Consumer	Host Prosumer	Device Prosumer
ProfileA	Mandatory	Mandatory	Recommended	Optional
ProfileB	Optional	Optional	not recommended	Optional

The SafetyProvider and the SafetyConsumer are useable for controller – controller safety communication and safety communication to field level.

The SafetyProsumer Classic is compatible with PROFIsafe V2.6. This is the classic version of safety communication and is optimized for safety communication to field level.

5 Use cases

5.1 Use cases for different types of communication links

5.1.1 Unidirectional communication

The most basic type of communication is unidirectional communication, where a safety application on one controller (A) sends data to a safety application on another controller (B).

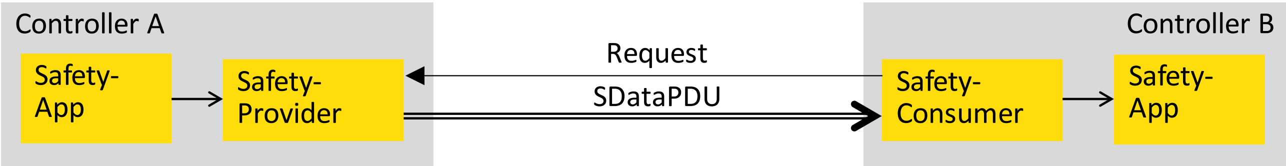


Figure 3 – Unidirectional Communication

This is accomplished by placing a provider on controller A, and a consumer on controller B. The connection between SafetyProvider and consumer can be established and terminated during runtime, allowing different consumers to connect to the same SafetyProvider at different times. Furthermore, the protocol is designed in such a way, that the consumer needs to know the parametrized ID of the SafetyProvider, for being able to safely check whether the received data is coming from the expected source. On the other hand, as safety data flows in one direction, only, there is no need for the SafetyProvider to check the ID of the consumers. Hence, controller A can – one after another- serve an arbitrarily large number of consumers, and new consumers can be introduced into the system without having to update controller A.

5.1.2 Bidirectional communication

Bidirectional communication means exchange of data in both directions, which is accomplished by placing a SafetyProvider and a consumer on each controller. Hence, bidirectional communication is realized using two Safety over OPC UA connections.

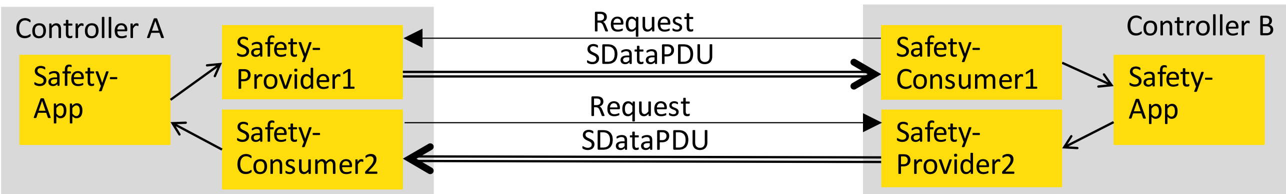


Figure 4 – Bidirectional Communication

Connections can be established and terminated during the runtime.

5.1.3 Safety Multicast

Multicast is defined as sending a set of data from one controller (A) to several other controllers (B1, B2, B3,…,BN) *simultaneously*.

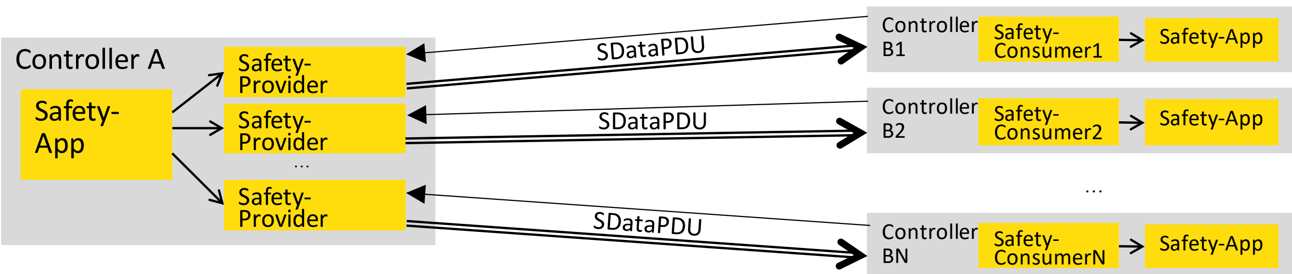


Figure 5 – Safety Multicast

Safety Multicast is accomplished by placing multiple SafetyProviders on controller A, and by connecting each of them to a consumer on one of the controllers B1, B2, … BN, each.

The protocol Safety over OPC UA is designed in such a way that:

- the state machine of the SafetyProvider has a very low number of states, and thus very low memory demands,
- all safety-related telegram-checks are executed on the consumer, and, thus the computational demand on the SafetyProvider is very low.

Therefore, even if a large number of SafetyProviders are instantiated on a controller, the performance requirements are still moderate.

The properties of simple unicast are also valid for multicast: different sets of consumers can connect to SafetyProviders at different times, and new consumers can be introduced into the system without having to reconfigure the SafetyProvider instances. As the SafetyProvider instances send data from the same Safety application (same data source), it is irrelevant from a safety point of view to which actual SafetyProvider instance each of the consumers is connected. Thus, all SafetyProvider instances can be parametrized with the same SDataProvID and same SDataBaseID.

5.2 Cyclic and acyclic safety communication

Safety over OPC UA supports cyclic and acyclic safety communication.

Usually safety applications with safety communication require to communicate a demand. In these applications a cyclic safety communication must be established. That means after evaluation of the SData at the safety application at SafetyConsumer side, the time until the next call of the SafetyConsumer shall be limited. This limitation time is part of the SFRT.

But in some safety applications a SafetyConsumer requires to read safety data but not a demand. This requirement can be fulfilled by acyclic safety communication. This acyclic safety communication uses a temporal time monitoring for the response (as SDataPDU) to the RequestSPDU.

The function for acyclic safety communication can be reused to implement the cyclic request.

5.3 Principle for “Application Variables with qualifier”

Every single safety variable shall add a safety qualifier to inform the SafetyConsumer Application Program whether the value is good or bad. The Qualifier shall be synchronous and consistent with the Value.

Table 5 – Example “Application Variables with qualifier”

Value	Qualifier
valid	0x1 (= good)
not valid	0x0 (= bad)

Motivation for “Application Variables with qualifier”:
This principle allows an individual safety reaction for that safety function, which is impacted by the failed safety variable.

The Qualifier shall be checked at safety application due to application specific reasons.

Exception: In safety applications where the value “0” at the variable leads to the safe state, the variable can be used without checking the Qualifier.

The presentation format for the Qualifier for single SData (Variable) will be specified in a separate companion standard. This is not in the scope of this standard.

6 Information Model

6.1 Example ObjectType Definition

The NamespaceUri of Safety over OPC UA is http://opcfoundation.org/UA/Safety.

Under this URL the Nodeset plus the list of nodes including the NodeIds can be found.

Each server has a singleton folder called SafetyDeviceSet with a fixed NodeId in the namespace of safety over OPC UA. Because all SafetyProviders on this server comprise a nonhierarchical reference to this variable, it can be used to directly access all SafetyProviders by following the references in backward direction.

In addition, the servers comprise one OPC UA object derived from type SafetyProviderType for each SafetyProvider they implement. The corresponding information model is shown in Figure 9.

A description of the graphical notation for the different types of Nodes and References (shown in Figure 6, Figure 7, Figure 8, and Figure 9) can be found in OPC UA Part 3 Annex C.

Figure 6 shows the Safety Parameters for SafetyProvider.

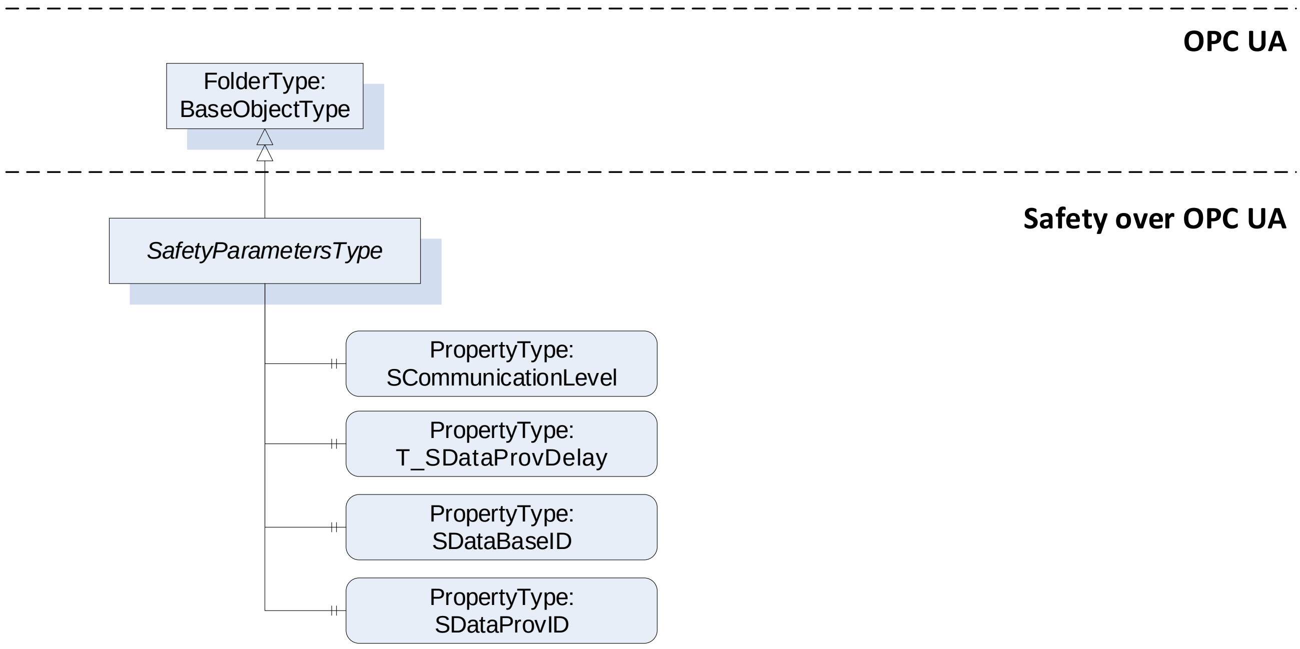


Figure 6 – Safety over OPC UA Parameters for SafetyProvider

Figure 7 describes SafetyProvider Type.

Safety over OPC UA requires (atomic) consistent data exchange.

For Safety over OPC UA, the Call-Service of the Method Service Set (see OPC Unified Architecture, Part 4; Chapter 5.11.2 Call) is used. The Call-Service supports the consistent Data exchange. The Method "ReadSafeDataV1" uses the OPC UA-Server with a set of InputArguments that make up the RequestSPDU and a set of OutputArguments that make up the SDataPDU. The "SafetyConsumer" uses the OPC UA-Client with the OPC UA Service Call.

For diagnostic purposes, the SPDU received and sent is accessible by calling the method ReadDiagnosisDataV1.

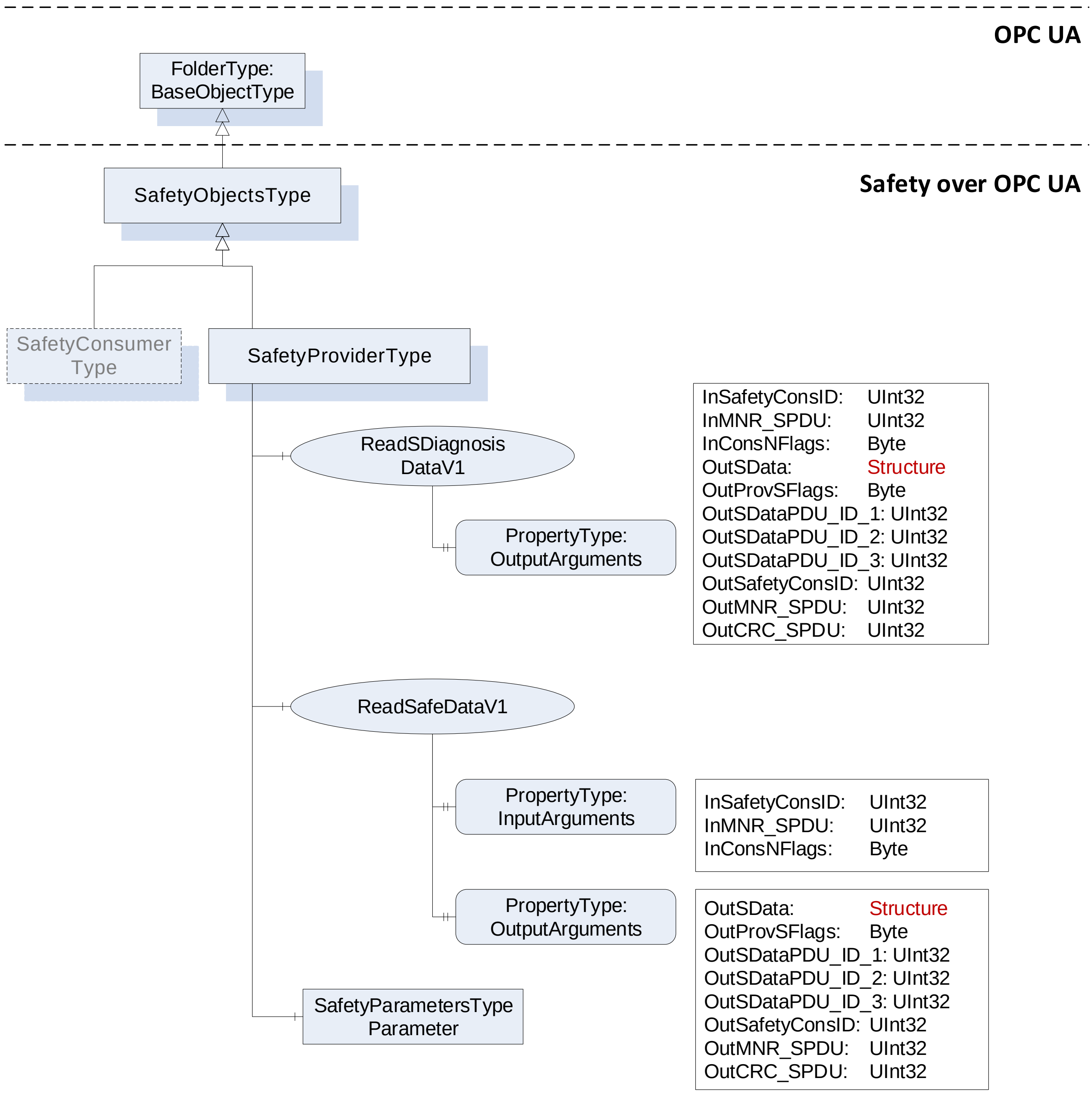


Figure 7 – Server Objects for Safety over OPC UA

NOTE The Model of the SafetyConsumer is not required at this stage of companion specification as a SafetyConsumer application has to know from the safety point of view all relevant information of the SafetyProvider. It can be added for Diagnosis.

Figure 8 shows the vendor specific definition of safety data.

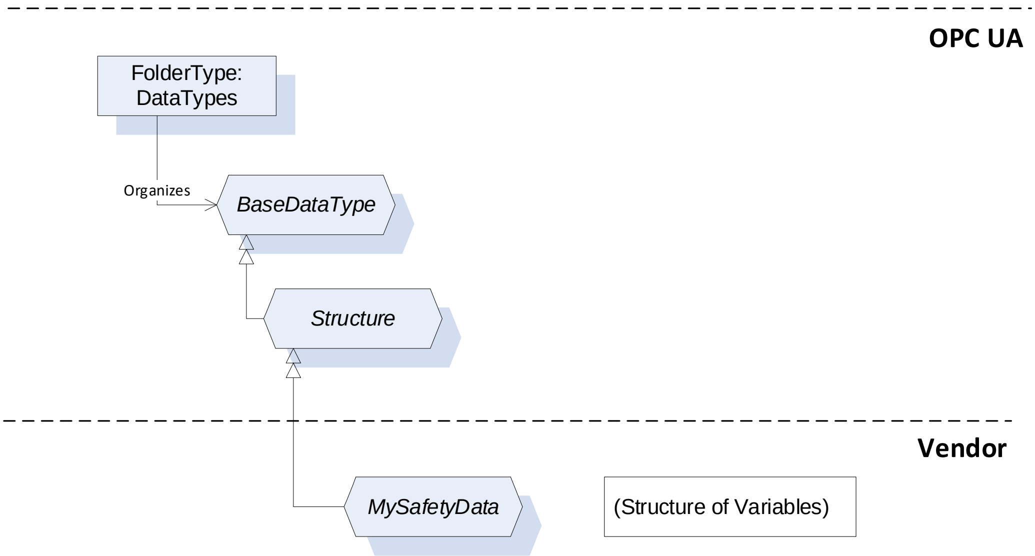
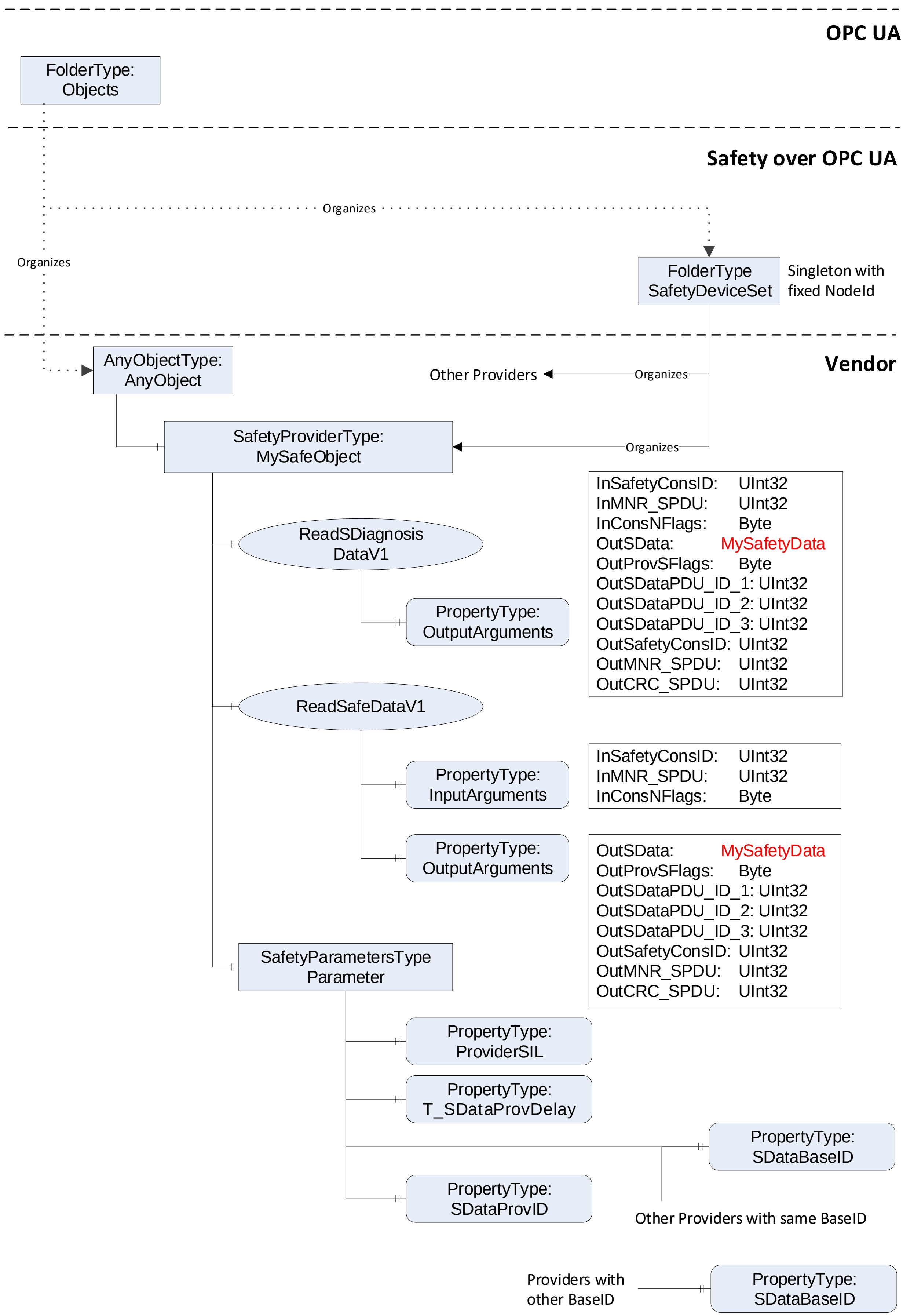


Figure 8 – DataTypes for Safety over OPC UA

Figure 9 shows the Instances of server objects for Safety over OPC UA. There are two things worth noting:

- The ObjectType for the SafetyProvider contains the methods with the abstract DataType BaseDataType. Each instance of a SafetyProvider needs its own copy of the methods which contains the concrete DataType of the SafetyData.
- The Property SDataBaseID is shared for all SafetyProviders with the same SDataBaseID value.

550


551 Figure 9 – Instances of server objects for Safety over OPC UA

Table 6 – Type Definition of Safety over OPC UA Parameters

Attribute	Value				
BrowseName	SafetyParametersType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Modelling Rule**
Subtype of BaseObjectsType					
HasProperty	Variable	SCommunicationLevel	Byte	PropertyType	Mandatory
HasProperty	Variable	T_SDataProvDelay	UInt32	PropertyType	Mandatory
HasProperty	Variable	SDStructSignVersion	UInt16	PropertyType	Mandatory
HasProperty	Variable	SDataBaseID	Guid	PropertyType	Mandatory
HasProperty	Variable	SDataProvID	UInt32	PropertyType	Mandatory

Table 7 – Type Definition of Safety over OPC UA SafetyProvider

Attribute	Value				
BrowseName	SafetyProviderType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Modelling Rule**
Subtype of SafetyObjectsType					
HasComponent	Method	ReadSafeDataV1			Mandatory
HasComponent	Method	ReadSDiagnosisDataV1			Mandatory
HasComponent	Object	Parameter		SafetyParametersType	Mandatory

6.1.1 Method ReadSafeDataV1

This method reads safe data from the SafetyProvider.

Signature

```
ReadSafeDataV1 (
    [in]   InMNR_SPDU            UInt32
    [in]   InSafetyConsID        UInt32
    [in]   InConsNFlags          Byte
    [out]  OutSData              Structure
    [out]  OutProvSFlags         Byte
    [out]  OutSDataPDU_ID_1      UInt32
    [out]  OutSDataPDU_ID_2      UInt32
    [out]  OutSDataPDU_ID_3      UInt32
    [out]  OutSafetyConsID       UInt32
    [out]  OutMNR_SPDU           UInt32
    [out]  OutCRC_SPDU           UInt32
);
```

Table 8 – Arguments of the Method ReadSafeDataV1

Argument	Description
InMNR_SPDU	“Monitoring NumbeR of the SPDU” details see MNR_S in Table 10
InSafetyConsID	“Safety Consumer Identifier” details see SafetyConsID_S in Table 10
InConsNFlags	“Byte with Non safety Flags from Consumer“ details see ConsNFlags in Table 15
OutSData	“Safety Data” details see 8.1.1.2
OutProvSFlags	“Byte with Safety Flags from Provider“ see ProvSFlags in Table 16
OutSDataPDU_ID_1	“Safety Data PDU Identifier Part1” see SDataPDU_ID_1 in Table 17
OutSDataPDU_ID_2	“Safety Data PDU Identifier Part2” see SDataPDU_ID_2 in Table 17
OutSDataPDU_ID_3	“Safety Data PDU Identifier Part3” see SDataPDU_ID_3 in Table 17
OutSafetyConsID	“Safety Consumer Identifier” details see SafetyConsID_S in Table 14 and Table 10
OutMNR_SPDU	“Monitoring NumbeR of the SPDU” details see NR_SPDU in Figure 16 and Figure 14
OutCRC_SPDU	“CRC of the SData PDU” details see CRC_SPDU in Figure 16

6.1.2 Method ReadSDiagnosisDataV1

This method is described in 9.2.

Signature

```
   ReadSDiagnosticDataV1 (
       [out]  InMNR_SPDU              UInt32
       [out]  InSafetyConsID          UInt32
       [out]  InConsNFlags            Byte
       [out]  OutSData                Structure
       [out]  OutProvSFlags           Byte
       [out]  OutSDataPDU_ID_1        UInt32
       [out]  OutSDataPDU_ID_2        UInt32
       [out]  OutSDataPDU_ID_3        UInt32
       [out]  OutSafetyConsID         UInt32
       [out]  OutMNR_SPDU             UInt32
       [out]  OutCRC_SPDU             UInt32
     );
```

For the description of the arguments see Method ReadSafeDataV1, see Table 8.

Instances of SafetyProviderType shall use non-abstract DataTypes for the OutSData argument.

6.2 Safety over OPC UA Version

The Version of the SafetyProvider is represented in the name of the two methods (“V1”).

6.3 DataTypes

Safety over OPC UA allows for sending the following basic data types listed in OPC UA within the SData (see OPC UA Part 3 - Address Space Model, Chapter BaseDataType and Part 6 - Mappings, Table – Built-in DataTypes).

Table 9 – DataTypes for Safety over OPC UA

ID	DataType name	value range	Number of octets	Description
1	Boolean (true or false)	0x0, 0x1	1	Using any other value than 0x1 for “true” may result in spurious errors in the cyclic redundancy check.
3	Byte	0 … 255	1	
4	Int16	−32 768 … 32 767	2	
6	Int32 / same for Enumeration	−2 147 483 648 … 2 147 483 647	4	
5	UInt16	0 … 65 535	2	
7	UInt32	0 … 4 294 967 295	4	
10	Float32 (ISO/IEC/IEEE 60559:2011)	single precision (32 bit) floating point value	4	

Currently, only scalar data types supported. No arrays are supported.

6.4 Connection establishment

Safety over OPC UA uses the OPC UA services for session establishment. For connection establishment Safety over OPC UA needs no additional requirement to these services.

This version of the specification includes configuration only at engineering time.

7 Safety communication layer services and management

7.1 Overview

Figure 10 gives an overview of the safety communication layer and its interfaces. It thereby also shows the scope of this specification. The main function of the Safety over OPC UA layer services is the state machine which handles the protocol. (As the Safety over OPC UA state machines are not influenced by OPC UA means it is described in UML). The state machines have several interfaces.

- The Safety Application Program Interface (SAPI) to control the safety data.

- The Safety Parameter Interface (SPI) supports adoption to the requirements of the application.

- The Diagnosis Interface (DI) supports the troubleshooting of the safety communication.

- the OPC UA platform interface (OPC UA PI) to the OPC UA stack.

These interfaces are vendor specific.

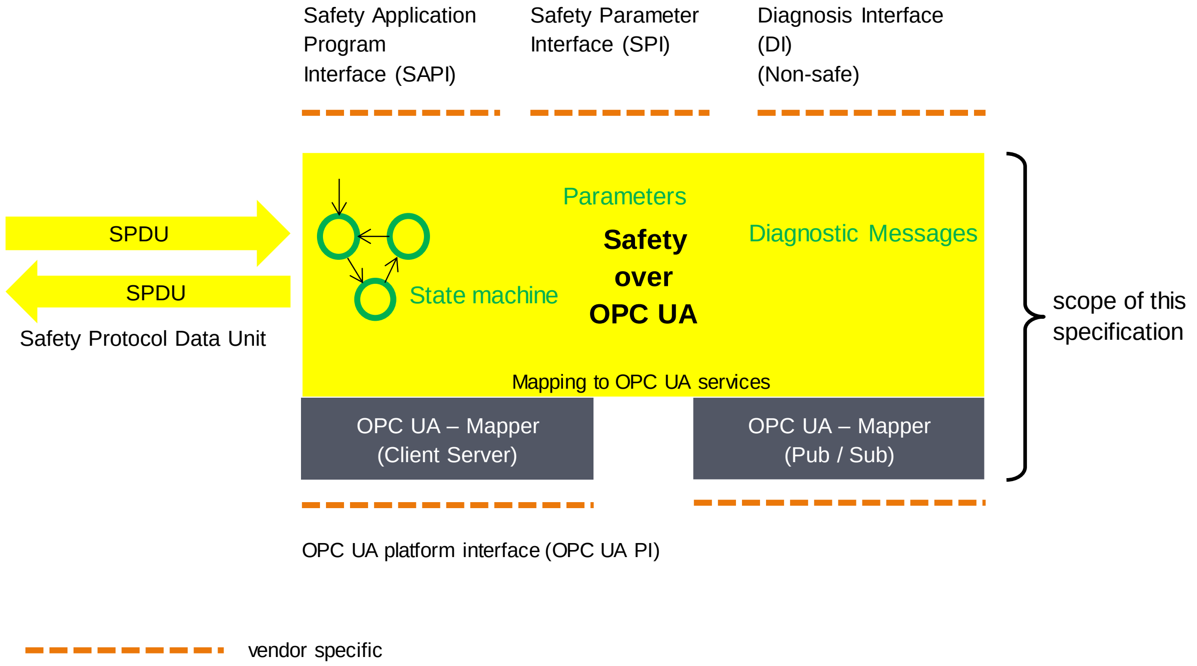


Figure 10 – Safety communication layer overview

7.2 Platform interface (OPC UA PI)

The state machines of Safety over OPC UA are independent from the used Part of the OPC UA Data Access. For this reason, the so called OPC UA Mapper is introduced.

One part of the OPC UA Mapper is the Method ReadSDiagnosticDataV1.

7.3 SafetyProvider interfaces

The Figure 11 shows an overview of the SafetyProvider interfaces. The SAPI is specified in 7.3.1, the SPI is specified in 7.3.2.

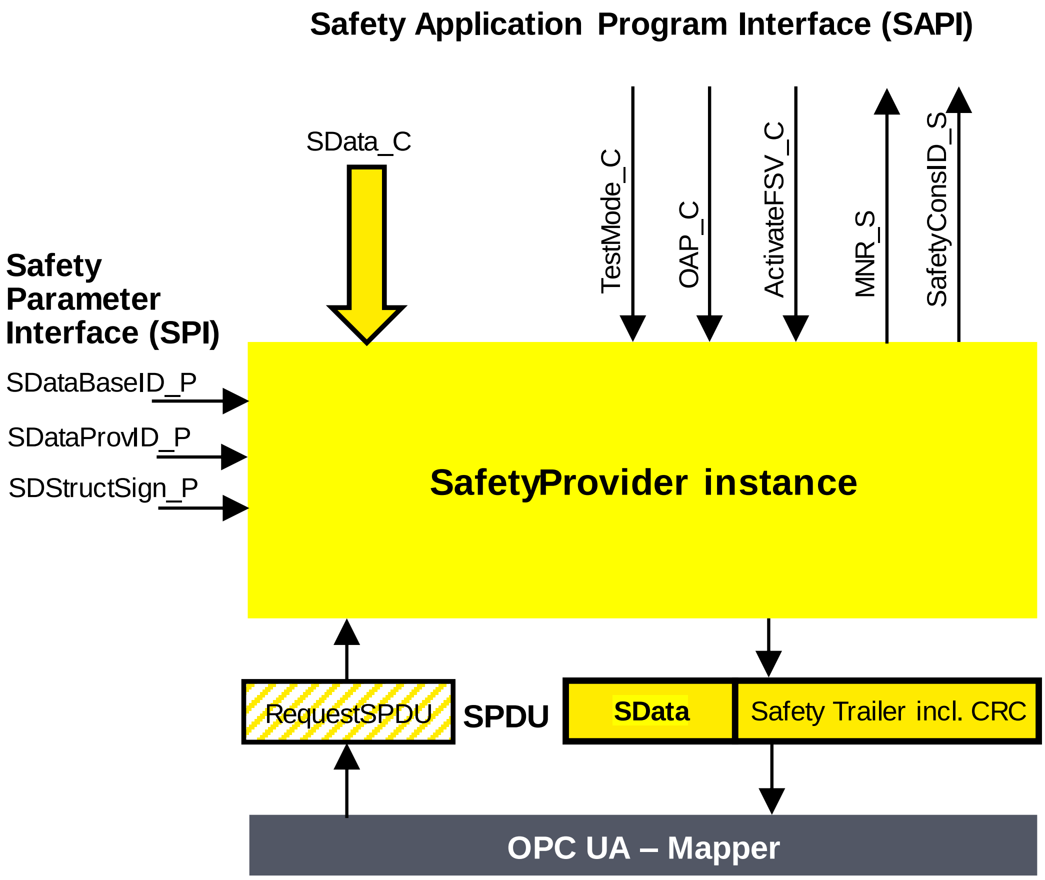


Figure 11 – SafetyProvider interfaces

7.3.1 SAPI of SafetyProvider

The SAPI of the SafetyProvider represents the Safety communication layer services of the SafetyProvider. Table 10 lists all inputs and outputs of the SAPI of the SafetyProvider.

Table 10 – SAPI of the SafetyProvider

SAPI Term	Type	Definition
SData_C	MySafeData	This input is used to accept the user data which is then transmitted as SData in the SPDU. NOTE: Whenever a new MNR is received from a SafetyConsumer, the state machine for the SafetyProvider will read a new value of the SData_C from its corresponding Safety Application and use it until the next MNR is received. NOTE: If no valid user data is available at the Safety Application, ActivateFSV_C shall be set to "1".
TestMode_C	Boolean	By setting this input to "1" the **remote** SafetyConsumer is informed that the SData are test Data, and should not be used for safety related decision.

SAPI Term	Type	Definition
		NOTE: The Safety over OPC UA stack is intended for implementation in safety devices exclusively, see 4.2.
OAP_C (Operator Acknowledge Provider)	Boolean	By changing this input to "1" (rising edge) after request for OA (OA_Req_S==1) at the **remote** SafetyConsumer, the SafetyConsumer is able to resume the SData from FSV to PV, see A.2.4.
ActivateFSV_C	Boolean	By setting this input to "1" the SafetyConsumer will deliver FSV instead of PV to the safety application program. NOTE: The “1” value is purposely used to guard against the case where an entire message is all 1's due to some faulty hardware. NOTE: if the replacement of process values by FSV should be controllable in a more fine-grained way, this can be realized by using qualifiers within the SData, see 5.3.
MNR_S	UInt32	This output yields the MNR. The MNR_S in the safety application is updated when a new request comes in from the SafetyConsumer.
SafetyConsID_S	UInt32	This output yields the ConsumerID used in the last access to this SafetyProvider by a SafetyConsumer.

7.3.2 SPI of SafetyProvider

The SPI of the SafetyProvider represents the parameter of the Safety communication layer management of the SafetyProvider.

Table 11 – SPI of the SafetyProvider

Name	Type	Range	Note
SDataBaseID_P	GUID	See GUID	See 8.1.1.6
SDataProvID_P	UInt32	1 - 0xFFFFFFFF	See 8.1.1.7
SDStructSign_P	UInt32	1 – 0xFFFFFFFF	Signature of the SData structure, see 8.1.1.2

7.3.3 Characteristics of SafetyProvider

The property value T_SDataProvDelay has no influence on the functional behavior of the SafetyProvider. However, it will be provided in its OPC UA information model of a SafetyProvider for engineering purposes.

The property value SCommunicationLevel gives coding for the SCommunicationLevel_ID to calculate the SDataPDU_ID.

Table 12 – Properties of SafetyProvider

Name	Type	Range	Note
T_SDataProvDelay	UInt32	1 – 0xFFFFFFFF	In microseconds (µs). It can be set in the engineering phase of the SafetyProvider or set during online configuration as well. T_SDataProvDelay is the maximum time at the SafetyProvider from receiving the RequestSPDU to start the transmission of SDataSPDU, see 10.2.
SCommunicationLevel	Byte	1 - 4	The maximal SIL the SafetyProvider implementation (hardware & software) is capable of, see Figure 12. It is used to inform the SafetyConsumer to parametrize the appropriate SCommunicationLevel_IP and then to generate the appropriate. SCommunicationLevel_ID NOTE: It is independent from the generation of the SData at SAPI.

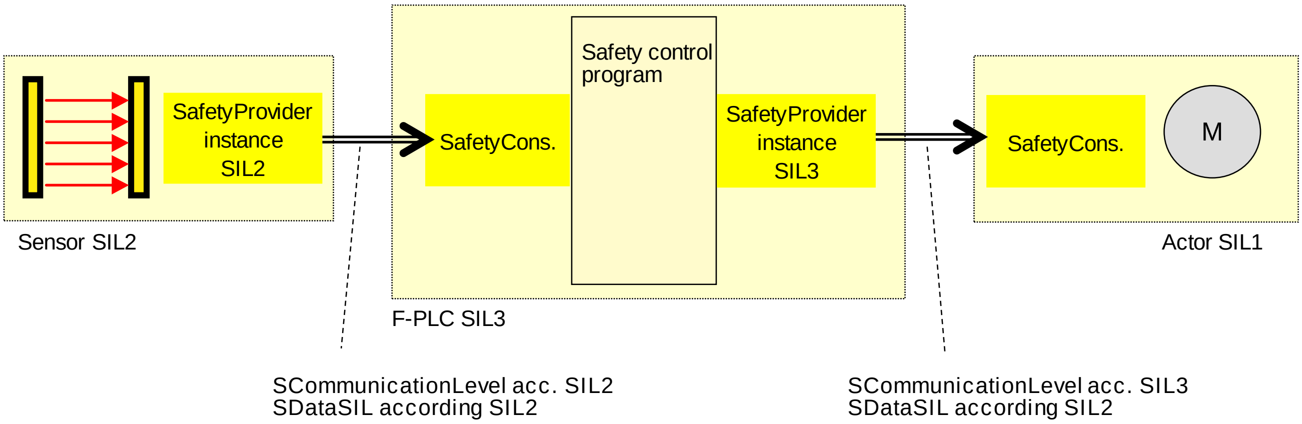


Figure 12 – Example combinations of SIL capabilities

The SIL capability of the SafetyConsumer implementation is independent on the SIL capability of the SafetyProvider implementation (SCommunicationLevel).

A SafetyConsumer implementation with a SIL 1, or 2, or 3, or 4 may accept SPDUs from SafetyProvider with a capability of SIL 1, or 2, or 3, or 4.

7.4 SafetyConsumer interfaces

The Figure 13 shows an overview of the SafetyProvider interfaces. The SAPI is specified in 7.4.1, the SPI is specified in 7.4.3.

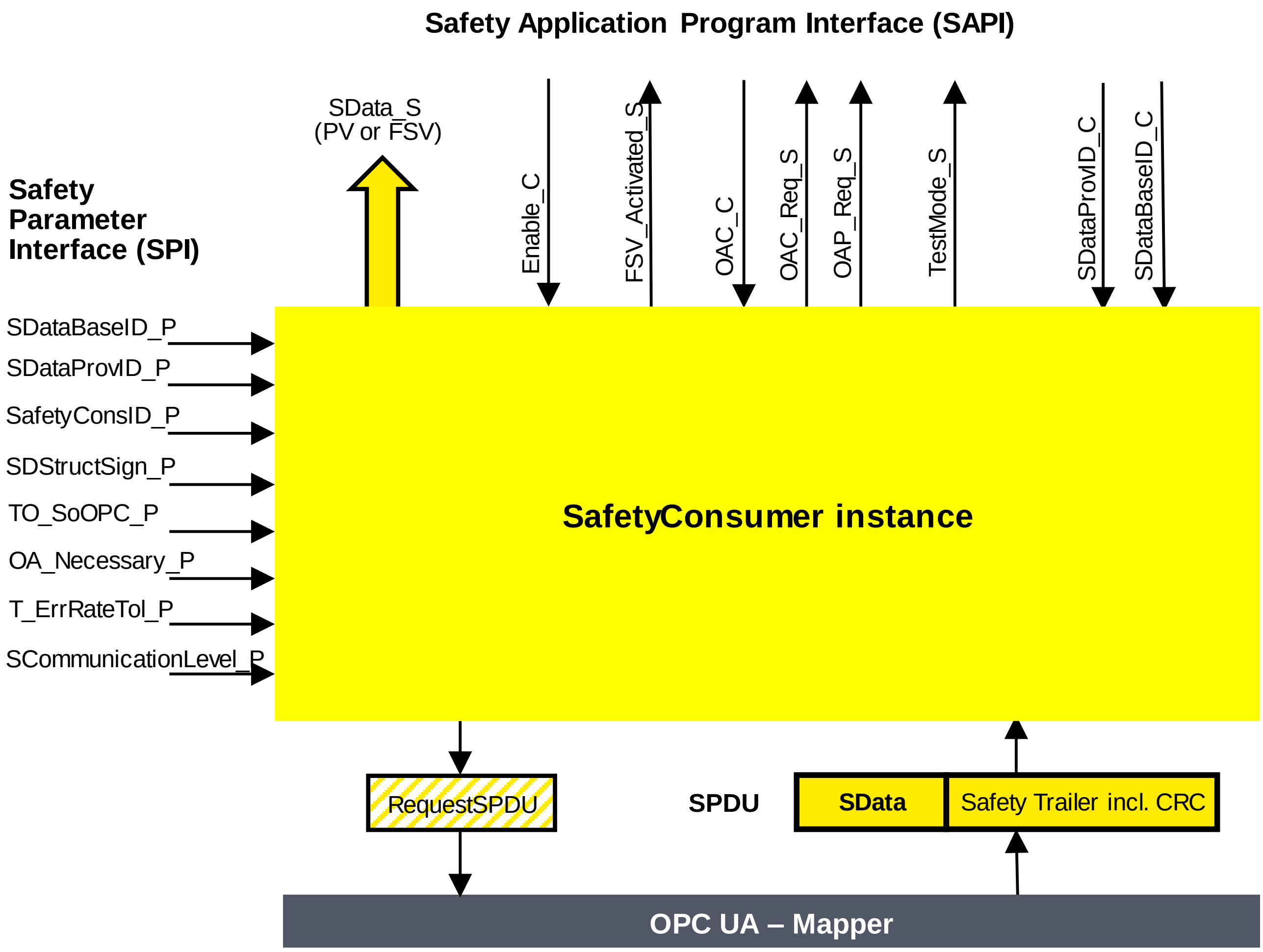


Figure 13 – SafetyConsumer interfaces

7.4.1 SAPI of SafetyConsumer

The SAPI of the SafetyConsumer represents the Safety communication layer services of the SafetyConsumer. Table 13 – SAPI of the SafetyConsumer lists all inputs and outputs of the SAPI of the SafetyProvider.

Table 13 – SAPI of the SafetyConsumer

SAPI Term	Type	Definition
SData_S	MySafeData	This output ether delivers the process values received from the SafetyProvider in the SPDU field SData, or FSV.
Enable_C	Boolean	By changing this input to "0" the SafetyConsumer will change each and every variable of the SData to "0" to stop sending requests. When changing Enable_C to "1" the SafetyConsumer will restart safe communication. The variable can be used to delay the start of the Safety over OPC UA communication, after power on until "OPC UA connection ready". The delay time <u>is not</u> monitored.
FSV_Activated_S	Boolean	This output indicates via "1", that on the output SData_S FSV (all binary "0") are provided. NOTE: if an application needs different FSV than “all binary 0”, it should use appropriate constants instead of the output of SData_S, in case FSV_Activated_S is set.

SAPI Term	Type	Definition
OAC_C (Operator Acknowledge Consumer)	Boolean	Motivation for Operator Acknowledge see 7.4.2. After an indication of OAC_Req_S this input shall be changed by means of the operator. By changing this input from “0” to "1" (rising edge) the SafetyConsumer is instructed to switch SData from FSV to PV. The OAC_C is processed only if this rising edge arrives after OAC_Req_S is set to “1”, see Figure 22. If a rising edge of OAC_C arrives before OAC_Req_S becomes 1, this rising edge is ignored. As soon as the OAC_Req_S is reset to “0” the OAC_C shall also be set to “0” by the safety application.
OAC_Req_S	Boolean	This output indicates the request for operator acknowledgment. The bit is set to “1” by the SafetyConsumer, after the rate of error has been too high and now the communication runs error free and hence operator acknowledgement is possible. The bit is reset to “0”, when a rising edge at OAC_C is detected
OAP_Req_S	Boolean	This output indicates an operator acknowledgement has taken place on the SafetyProvider. If operator acknowledgement at the SafetyProvider should be allowed this output must be connected to OAC_C, see A.2.4 and A.2.5.
TestMode_S	Boolean	Every SafetyConsumer application program shall evaluate this variable. This output indicates by “1” that the application at the SafetyProvider is in the state “TestMode”, e.g. during commissioning. A value of “0” represents the “normal safe state”. Motivation: The TestMode enables the programmer and commissioner to validate the application with test data.
SDataProvID_C	UInt32	By changing this input to a non-zero value, the SafetyConsumer uses this variable instead of the SPI-Parameter SDataProvID_P. This input is only read in the first cycle, or when a rising edge occurs at the input Enable_C. See also Table 14. If it is changed to “0”, the Parameter SDataProvID_P will become activated.
SDataBaseID_C	GUID	By changing this input to a non-zero-value the SafetyConsumer uses this variable instead of the SPI-Parameter SDataBaseID_P. This input is only read in the first cycle, or when a rising edge occurs at the input Enable_C. See also Table 14. If it is changed to “0”, the Parameter SDataBaseID_P will become activated.

7.4.2 Motivation for SAPI Operator Acknowledge (OAC_C)

NOTE The characteristic of Operator Acknowledge is used to limit directly the rate of detected faulty SDataPDUs and indirectly the rate of undetected faulty SDataPDUs entering the SafetyConsumer (see 11.4).

As long as communication errors are detected too frequent the SafetyConsumer continuously delivers FSV. When communication errors are no longer detected, the SafetyConsumer will return to deliver PV after an Operator Acknowledge.

Operator Acknowledge must be initiated by a human operator as he is responsible to check the installation, see “Table 28, row Operator Acknowledge”. For this reason the OAC_C is pushed up to the safety application program to deal with.

Timeout errors do not need OAC_C. As an option, if setting OA_Necessary_P==1 then the use of OAC_C is required.

7.4.3 SPI of SafetyConsumer

The SPI of the SafetyConsumer represents the parameter of the Safety communication layer management of the SafetyConsumer.

Table 14 – SPI of the SafetyConsumer

SPI Name	Type	Range / Default	Note
SDataBaseID_P	GUID	See GUID	See 8.1.1.6 If the Parameter SDataBaseID_P can be changed by the Safety Application Program, then it can be combined with the interface variable SDataBaseID_C
SDataProvID_P	UInt32	1 - 0xFFFFFFFF	See 8.1.1.7 If the Parameter SDataProvID_P can be changed by the Safety Application Program, then it can be combined with the interface variable SDataProvID_C
SafetyConsID_P	UInt32	1 - 0xFFFFFFFF	See 8.1.1.8
SDStructSign_P	UInt32	1 – 0xFFFFFFFF	32 bit signature of the SData structure See 8.1.1.2

TO_SoOPC_P	UInt32	1 – 0xFFFFFFFF Default value 1	In µs SPDU WatchDog time period at SafetyConsumer from request to response with error free SDataPDU or in case of an SPDU-error. See 10.2
OA_Necessary_P	Boolean	0 / 1 Default 1	This parameter controls whether an OA is necessary in case of timeout (TO_SoOPC_P) or when the SafetyProvider has activated FSV (ActivateFSV_C). 1: in case of timeout, or ActivateFSV_C the values remain in FSV until OA. 0: in case of timeout (TO_SoOPC_P), the values change from FSV to PV as soon as the communication is free of errors. In case of ActivateFSV_C the values change from FSV to PV as soon as ActivateFSV_C returns to “0”.
T_ErrRateTol_P	UInt16	6, 60, 600	Value in minutes. The parameter T_ErrRateTol effects the maximum tolerated rate of errors detected by Safety over OPC UA and therefore the PFH of this Safety over OPC UA link, see 11.4
SCommunicationLevel_P	Byte	1 - 4	This parameter represents the SIL of a SafetyProvider instance, from 1 to 4. This parameter gives Coding for the SCommunicationLevel_ID to calculate the SDataPDU_ID. See 8.1.1.11and Figure 12

7.4.4 Motivation for SPI OA_Necessary_P

This parameter shall be set to 1 at application where recovery only by human interaction is permitted.

This parameter shall be set to 0 at application where auto recovery (without human interaction) is permitted.

7.5 OPC UA platform interface for SafetyProsumer Classic

7.5.1 SafetyProsumer Classic (Host) services and parameter

The SafetyProsumer Classic (Host) will be conform to the behavior of IEC 61784-3-3 Ed3. Its integration in OPC UA will be described in the next version of this specification.

7.5.2 SafetyProsumer Classic (Device) services and parameter

The SafetyProsumer Classic (Host) will be conform to the behavior of IEC 61784-3-3 Ed3. Its integration in OPC UA will be described in the next version of this specification.

8 Safety communication layer protocol

8.1 SafetyProvider and SafetyConsumer

8.1.1 SPDU format

8.1.1.1 SPDU summary

Figure 14 shows the structure a RequestSPDU which originates at the SafetyConsumer and contains a SafetyConsID, a MonitoringNumber (MNR_SPDU), and one byte of (non-safety-related) flags (ConsNFlags).

NOTE: The ConsNFlags are named “NFlags” (non safety) in opposite to the “SFlags” in the SDataPDU, as the SFlags in this specification are safety-relevant.

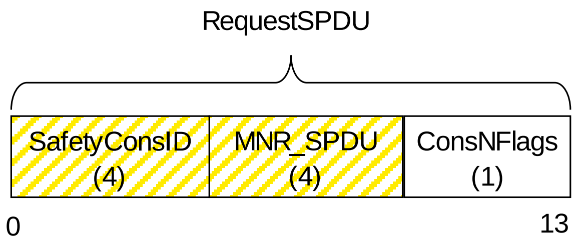


Figure 14 – RequestSPDU

NOTE The SafetyConsID and the MNR_SPDU are not safety relevant in the RequestSPDU, they become safety relevant as part of the SDataPDU. For this reason, they are crosshatched in the RequestSPDU.

NOTE The RequestSPDU does not need an own different BaseID as in Safety over OPC UA the SDataBaseID is enough from the safety point of view in the specified design.

Figure 15 shows the structure of a SDataPDU which originates at the SafetyProvider and contains the safety data (1 – 1500 Byte) and additional 25 Byte safety code (STrailer) as described in the subsequent sections.

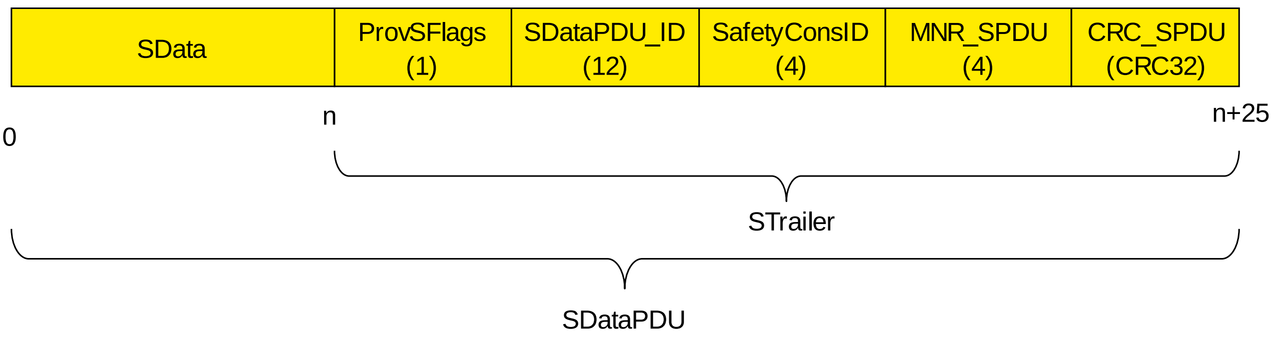


Figure 15 – SDataPDU

(x) lists the number of bytes in the SPDU part.

In order to avoid spurious trips, the SPDU must be transmitted in an atomic (consistent) way from the OPC UA platform interface of the SafetyProvider to the OPC UA platform interface of the SafetyConsumer. This is the task of the respective OPC UA mapper, see Figure 2.

8.1.1.2 SData

SData contains the safety-related application data transmitted from provider to consumer. It may comprise multiple basic OPC UA variables taken from Table 6. Note that SData should be collected in a Structure.

For the calculation of the CRC Signature, the order in which this data is processed by the calculation, is important. Provider and Consumer have to agree upon the number, type and order of application data transmitted in SData. The sequence of the SData is fixed.

NOTE SData may contain qualifier bits for a fine grained activation of fail-safe substitute values. For a valid process value, the respective qualifier should be set to 1 (good), whereas the value 0 (bad) should be used for invalid values. Invalid process values must be replaced by a fail-safe substitute value in the consumer's safety application. See Table 4

8.1.1.3 ConsNFlags: Flags of the Safety Consumer

The value of the ConsNFlags carries the flags according to Table 15.

Table 15 – Structure of ConsNFlags

NFlag Bit nr.	Name	Description
LSB = Bit 0	SComErrDiag_NF	0: No error 1: A communication error occurred in the previous SDataPDU.
Bit 1	OAC_Req_NF	Equivalent of OAC_Req_S to inform the SafetyProvider
Bit 2	FSV_Activated_NF	Is used for conformance test of FSV_Activated_S
Bit 3 ..7	Reserved for future use	

The SafetyConsumer sets SComErrDiag_NF to enable a communication analysis tool to trigger in case of an error.

The N of _NF stands for non safety. These flags are <u>not</u> evaluated at the SafetyProvider.

All other bits in ConsNFlags are reserved for future use. They must be set to zero by the SafetyConsumer and must not be evaluated by the SafetyProvider.

8.1.1.4 ProvSFlags: Flags of the SafetyProvider

The byte ProvSFlags carries the bits shown in Table 16

Table 16 – Structure of ProvSFlags

SFlag Bit nr.	Name	Description
LSB = Bit 0	OAP_SF	See OAP_C
Bit 1	ActivateFSV_SF	See ActivateFSV_C
Bit 2	TestMode_SF	See TestMode_C
Bit 3 .. 7	Reserved for future use	

The SafetyConsumer evaluates the ProvSFlags and transmits the info to the SAPI if the SDataPDU is checked without error.

The S of _SF stands for safety related.

If the SDataPDU is checked with error: *TestMode_C:=0, ActivateFSV_C:=1, OAP_SF* remains last value. If the SDataPDU is checked without error: *TestMode_C:= TestMode_SF*, and *OAP_C:= OAP_SF*.

8.1.1.5 MonitoringNumber (MNR)

The SafetyConsumer uses the MNR (MNR_SPDU together with the SafetyConsID) to check whether the SDataPDU is the response to the RequestSPDU sent earlier. The MNR_SPDU is used in the acknowledgment mechanism for monitoring the propagation delay from SafetyProvider to SafetyConsumer.

Safety over OPC UA is transmitting the MNR with each and every SPDU as MNR_SPDU.

The checking for correctness of the MNR_SPDU is performed by the SafetyConsumer.

With each request, the MNR is incremented. In case of an overflow (0xFFFFFFFF + 1) the MNR_SPDU is set to MNR_min. Values between 0 and MNR_min are reserved.

MNR_min is 0x100.

NOTE: these reserved values can be used in the future e.g. to transmit information from SafetyConsumer to SafetyProvider.

Example sequence for MNR:

- …
- 0xFFFF FFFE
- 0xFFFF FFFF
- 0x0000 0100
- 0x0000 0101
- …

Refer to 11.2 for relevant constraints.

8.1.1.6 SDataBaseID

The SDataBaseID is used as one global ID for a set of SafetyProviders with unique SDataProvID, see 3.2.24

Refer to 11.1.1 for relevant constraints.

8.1.1.7 SDataProvID

The SDataProvID is the ID for the individual SafetyProvider. See 3.2.25

In case of a machine-type every machine may have the same set of SDataProvIDs. In this case the SDataBaseID shall be set up for every machine instance individually.

Refer to 11.1.1 for relevant constraints.

8.1.1.8 SafetyConsID

Identifier of the SafetyConsumer instance.

Refer to 11.1.2 for relevant constraints.

8.1.1.9 Build SDataPDU

The task of the SafetyProvider is to take over the MNR_SPDU and the SafetyConsID of the received RequestSPDU into the STrailer. After this, it adds the SDataPDU_ID, ProvSFlags, and the SData to calculate the CRC_SPDU (see 8.1.1.9).

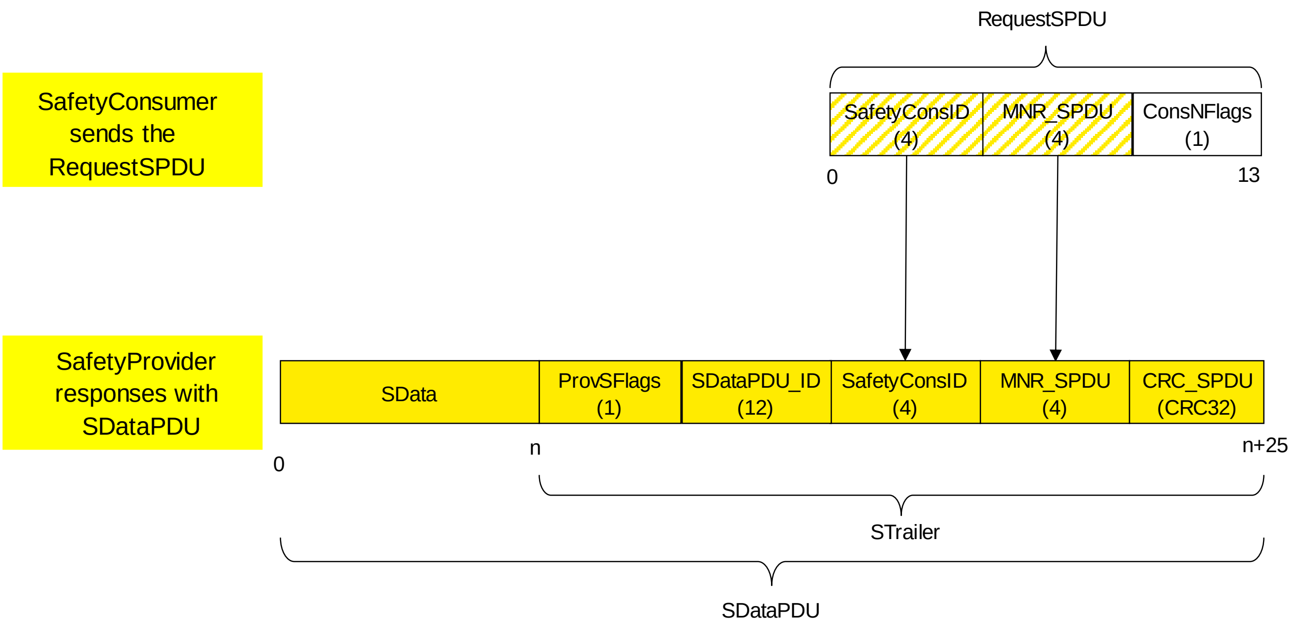


Figure 16 – Overview of task for SafetyProvider

Calculation of the SDStructSign see 8.1.1.12

Coding of the SDataBaseID, see 8.1.1.6

Coding of the SDataProvID, see 8.1.1.7

8.1.1.10 Calculation of the SDataPDU_ID

Figure 17 shows the calculation of the SDataPDU_ID.

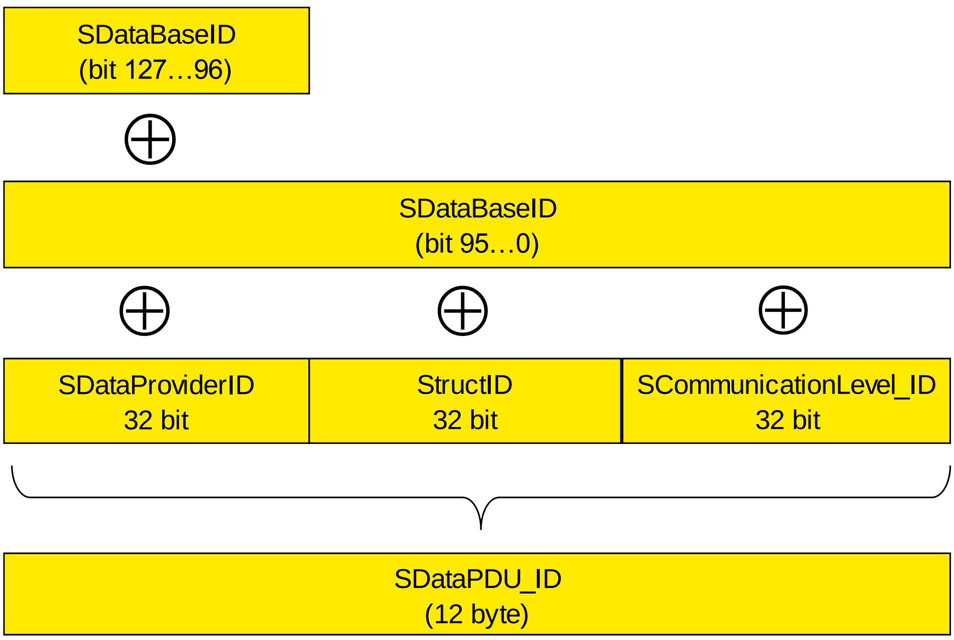


Figure 17 – Calculation of the SDataPDU_ID

The SDataPDU_ID is calculated according Table 17.

Table 17 – Presentation of the SDataPDU_ID

SDataPDU_ID (bit 31 .. 0) := SDataBase_ID (bit 31 .. 0) **XOR** SCommunicationLevel_ID
SDataPDU_ID (bit 63 .. 32) := SDataBase_ID (bit 63 .. 32) **XOR** SDStructSign
SDataPDU_ID (bit 95 .. 64) := SDataBase_ID (bit 127 .. 96) **XOR** SDataBase_ID (bit 95 .. 64) **XOR** SDataProvID

8.1.1.11 Coding of the SCommunicationLevel_ID

Table 18 – Coding for the SCommunicationLevel_ID

Property	Value of SCommunicationLevel_ID
Internal equivalent of the SCommunicationLevel	SIL 1 communication: 0x11912881 SIL 2 communication: 0x647C4654 SIL 3 communication: 0xDEAA9DEE SIL 4 communication: 0xAB47F33B

These values of SCommunicationLevel_ID differ by maximum number of bits (hamming distance of 21).

A SafetyProvider shall know only the one appropriate value (Internal equivalent value of the SCommunicationLevel) especially as SafetyProvider with SIL 1-3 to be not able to choose a wrong one (like equivalent for used for SIL 4 / SCommunicationLevel==4) by errors.

It shall be set equivalent to the SIL capability of the SafetyProvider instance.

The SCommunicationLevel is independent to the SIL capability of the provided SData.

8.1.1.12 SDataStructure Signature (SDStructSign)

The SDStructSign is used to check the number, type and order of application data transmitted in SData. If the SafetyConsumer is expecting, anything different than the SafetyProvider provides, the SDStructSign will differ, allowing the SafetyConsumer to enable fail-safe substitute values.

In addition, applications may also define a 32-bit SData-ID which will allow for the discrimination of data which is otherwise not differentiable. For instance, three integers could be interpreted as cartesian coordinates or three Euler angles. By defining a SData-ID for each of the two types, online detection of a mismatch becomes possible.

SData-IDs which are defined in application specific companion specifications published by the OPC Foundation should have their highest bit set to 0 (range 0x80000000 to 0xFFFFFFFF). Other SData-IDs should have their highest Bit set to 1 (range 0x00000001 to 0x7FFFFFFF). The value 0x00000000 shall be used if no SData-ID has been defined.

The SDStructSign is calculated as CRC32-signature over the SData-ID, the version of presentation and the sequence of the DataType IDs. After each datatype ID, a 16-bit zero-value (0x0000) is inserted.

Example of a SDStructSign:

ID of companion specification for Safety over OPC UA (0xyyyyy, 0xzzzz),
Version of list presentation for SDStructSign:= 0xvvvv,
1. DataType Int16: (Id = 4),
2. DataType Boolean: (Id = 1),
3. DataType Float32: (Id =10)

SDStructSign== CRC32(0xyyyyy, 0xzzzz, 0xvvvv, 0x0000, 0x0004, 0x0000, 0x0001, 0x0000, 0x000A)

NOTE: The insertion of 0x0000 values before the DataType ID, allows for introducing arrays in later version of this specification.

The DataType ID can be found at the DataType or at the derived DataType.

The OPC UA Information model supports not only built-in DataTypes, but also from these DataTypes derived from built-in DataTypes. In case of derived DataTypes, the Data Structure CRC uses the ID of a built-in DataType is used (which is found at the end of the tree).

The base type "enumeration" uses the DataType Int32 (ID=6); for therefore, this specification uses in case of "DataType enumeration" the ID for Int32.

In the first version of this specification arrays are not supported. Instead, multiple variables of the same type must be used.

8.1.1.13 CRC_SPDU

The SafetyProvider calculates the CRC signature (CRC_SPDU) and sends it to the SafetyConsumer as part of SPDU. This enables SafetyConsumer to check the correctness of the SPDU including the SData, ProvSFlags, MNR, and the SDataPDU_ID by recalculating the CRC signature.

The generator polynomial *0xF4ACFB13* shall be used for the 32 bit CRC signature.

If SData is longer than one byte (e.g. UInt16; Int16, Float32) , it shall be decoded and encoded using big-endian integers in which the least significant byte appears last in the incremental memory address stream.

The calculation sequence shall begin with the highest memory address (n) of the SData counting back to the lowest memory address (0) and then include also the STrailer beginning with the highest memory address.

Figure 18 shows the calculation sequence of the CRC_SPDU.

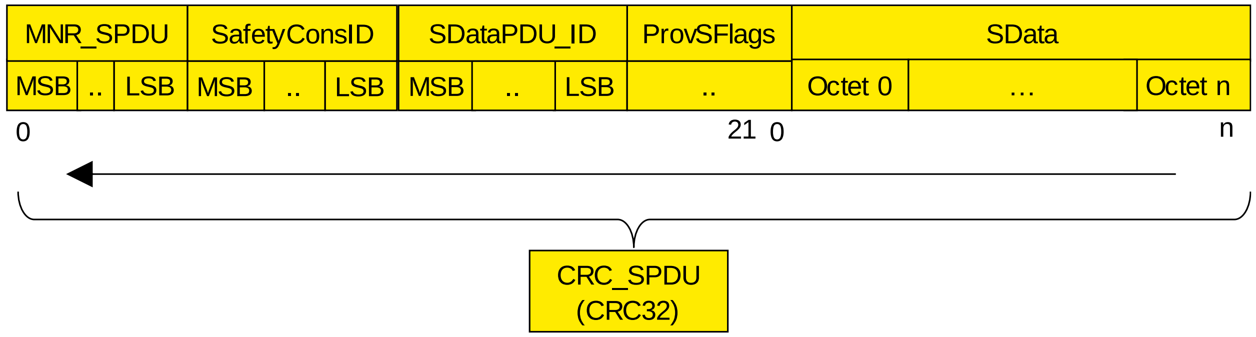


Figure 18 – Calculation of the CRC_SPDU

It is allowed to calculate the CRC of the SData and take this CRC as start value for the CRC calculation of the STrailer.

8.1.2 Safety over OPC UA behavior

8.1.2.1 General

The core of the safety layers is within the SafetyProvider and SafetyConsumer. The SafetyProvider and SafetyConsumer are specified in state diagrams.

Table 19 – Definition of terms used in the state diagrams

Term	Definition
Initial values	Any SPDU values =0
SPDU received	Any new SPDU received; ignore SPDU with all values = 0 NOTE this is executed in the macros <Get RequestSPDU> and <Get SDataPDU>
CRC_SPDU	CRC as part of the SPDU, Range 1 … 0xFFFFFFFF
MNR_SPDU	MNR as part of the SPDU, Range see 8.1.1.5
RequestSPDU	Structure: SafetyConsID, MNR

8.1.2.2 SafetyProvider/-Consumer Sequence diagram

Figure 19 shows the sequence of request and response with SData and the timeouts for Safety over OPC UA.

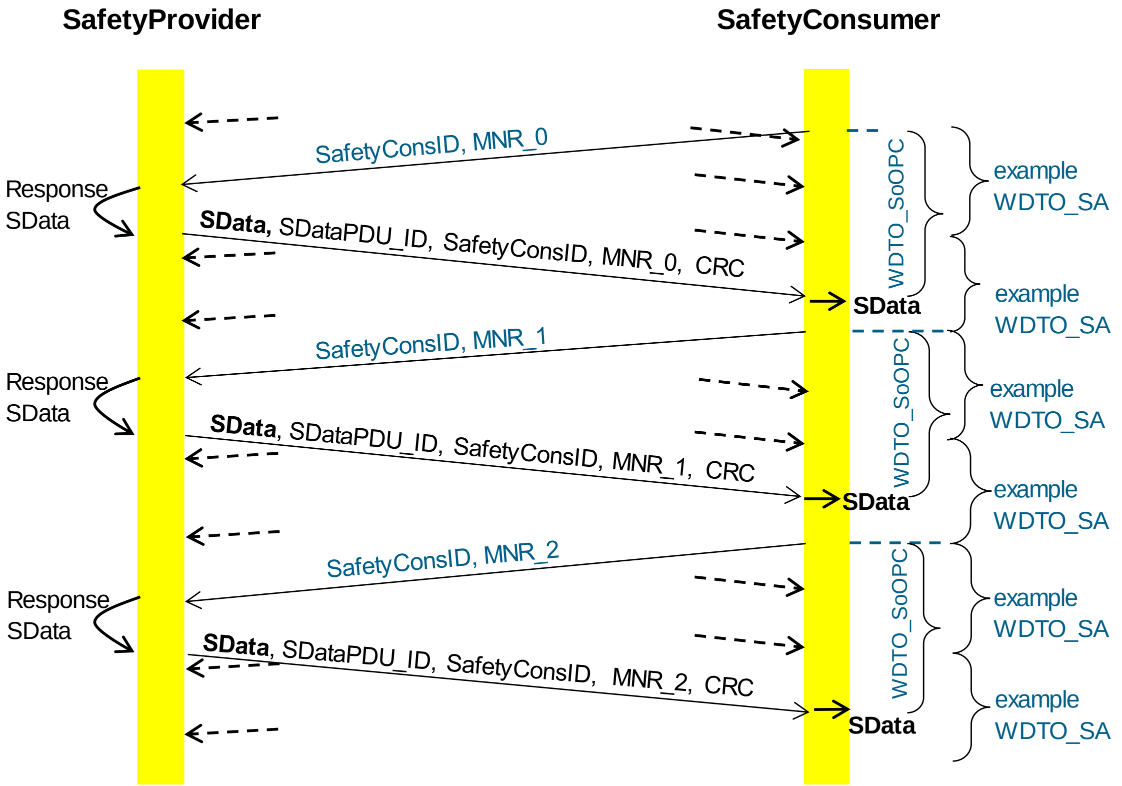


Figure 19 – Sequence diagram for Safety over OPC UA

The WDTO_SoOPC is the watchdog time for Safety over OPC UA.

The WDTO_SA is the maximum time for the cyclic update of the SafetyConsumer. It is the timeframe from one call of the SafetyConsumer to the next call of the SafetyConsumer. The implementation and error reaction of WDTO_SA is not part of this specification, it is vendor specific.

The short-dashed arrows show possible repetitions of SPDUs at the black channel, especially with Pub/Sub.

The long arrows show the new SPDU.

The WDTO_SoOPC monitors the timeframe from "<Set RequestSPDU>” to the OPC UA Communication Stack" to "accept the SDataPDU” (all checks for authenticity, timeliness and data integrity are accepted and positive)"

8.1.2.3 SafetyProvider state diagram

Figure 20 shows the principle state diagram of the SafetyProvider. Use Table 20, Table 21, and Table 22 for implementation and test.

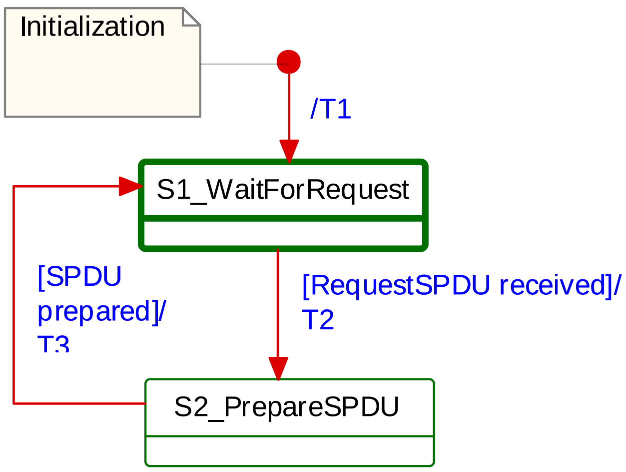


Figure 20 – Principle state diagram for SafetyProvider

The transitions are fired in case of an event for example receiving a SPDU. In case of several possible transitions, so-called guard conditions (refer to […] in UML diagrams) define which transition to fire.

The diagram consists of activity and action states. Activity states are surrounded by bold lines, action states are surrounded by thin lines. While activity states may be interruptible by new events, action states are not. External events occurring while the state machine is in an action state, are deferred until the next activity state is reached.

Table 20 – States of SafetyProvider instance

STATE NAME	STATE DESCRIPTION
Initialization	// Initial state SData_S:= 0 MNR_S:= 0 SafetyConsID_S:= 0 RequestSPDU_i:= 0
S1_WaitForRequest	// waiting on next RequestSPDU from SafetyConsumer <Get RequestSPDU>
S2_PrepareSPDU	**if** ActivateFSV_C== 1 **then** ActivateFSV_SF:= 1, **else** ActivateFSV_SF:= 0, **if** OAP_C== 0 **then** OAP_SF:= 0 **else** OAP_SF:= 1 **if** TestMode_C== 0 **then** TestMode_SF:= 0 **else** TestMode_SF:= 1 <build SDataPDU>

Table 21 – SafetyProvider driver transitions

TRAN-SITION	SOURCE STATE	TARGET STATE	GUARD CONDITION	ACTIVITY
T1	Init	1	-	
T2	1	2	**//** RequestSPDU received <Get RequestSPDU> **When:** [RequestSPDU_i<> RequestSPDU]	// **Operate Request** RequestSPDU_i:= RequestSPDU MNR_S:= MNR SafetyConsID_S:= SafetyConsID
T3	2	1	// **SPDU is prepared** **-**	<Set SDataPDU>

Table 22 – SafetyProvider instance internal items

INTERNAL ITEMS	TYPE	DEFINITION
RequestSPDU_i	Variable	Local Memory for RequestSPDU
state_x	Activity State	Within these interruptible "activity" states the SafetyProvider waits for new inputs.
state_y	Action State	Within these non-interruptible "action" states events like new Request is deferred until the next "activity" state **[**1] is reached.
<Get RequestSPDU>	Macro	Instruction to take the whole RequestSPDU from the OPC UA Mapper.
<Set SDataPDU>	Macro	Instruction to transfer the whole SDataPDU to the OPC UA Mapper
<build SDataPDU>	Macro	Take the MNR_SPDU and the SafetyConsID of the received RequestSPDU. Add the SDataPDU_ID, ProvSFlags, and the SData to calculate the CRC_SPDU. See 8.1.1.9

8.1.2.4 SafetyConsumer state diagram

Figure 21 shows the principle state diagram of the SafetyConsumer. Use Table 24, Table 25, and Table 23 for implementation and test.

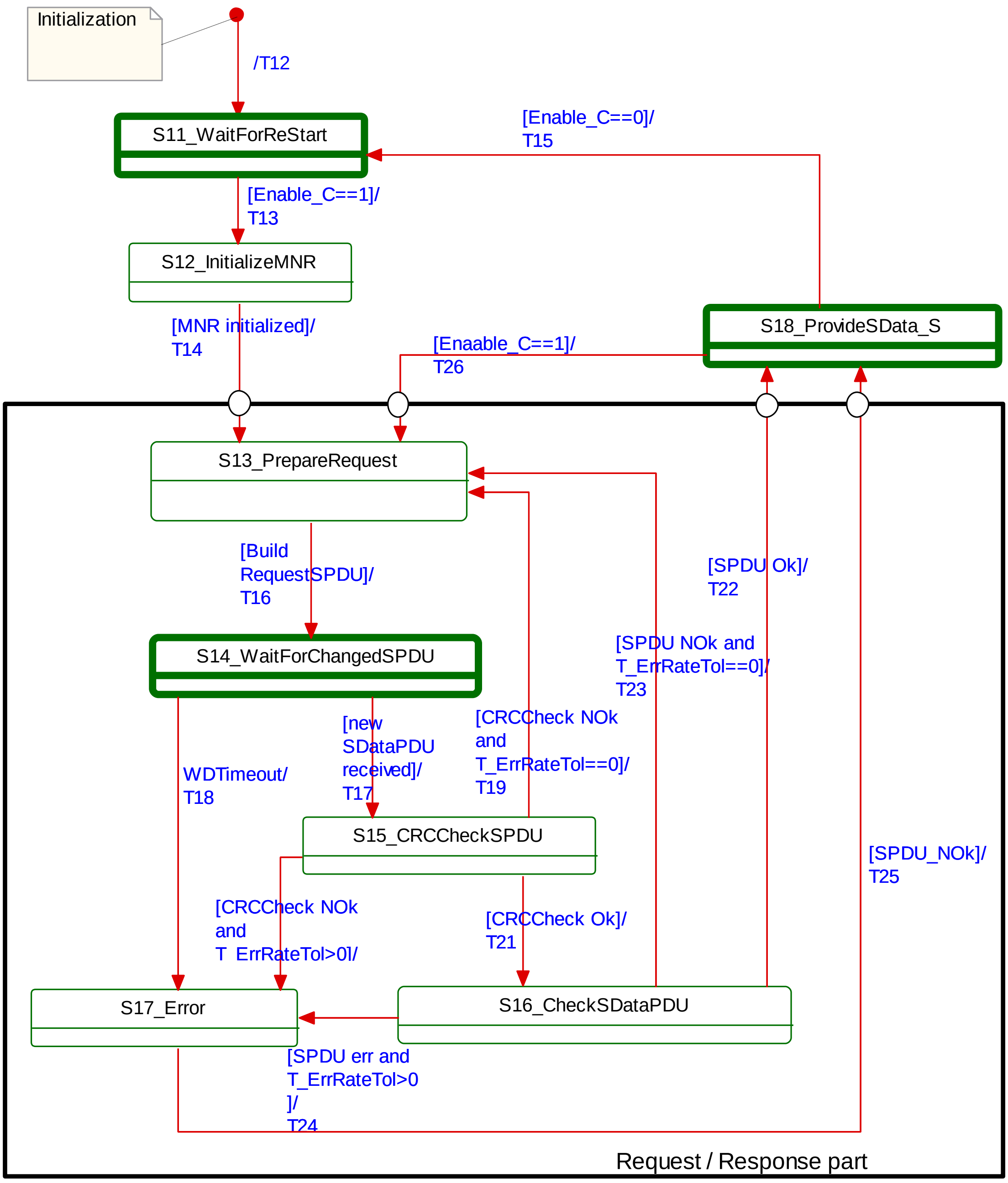


Figure 21 – principle state diagram for SafetyConsumer

Table 23 – SafetyConsumer driver internal items

INTERNAL ITEMS	TYPE	DEFINITION
Variable / Flag		
FaultReqOA_i	SFlag	Local memory for errors which request Operator Acknowledge.
OAC_i	SFlag	By means of this auxiliary variable (bit) a rising edge of OAC_C is memorized.
OAC_Req_S	Interface / SFlag	It is used at the Interface and at the same time for internal logic.

INTERNAL ITEMS	TYPE	DEFINITION
prevMNR_i	Variable	Local memory for previous MNR
SDataProvID_i	Variable	Local memory for SDataProvID in use
Timer		
WDTO_SoOPC	Timer	This timer is used to check whether the next valid SDataPDU arrived on time.
T_ErrRateTol	Timer	The CountdownTimer T_ErrRateTol is set to the value T_ErrRateTol_P. The first detected “SDataPDU error” starts the CountdownTimer T_ErrRateTol. It stops at 0. Only if a second “SDataPDU error” is detected before it stops the SData_S change to FSV. The CountdownTimer T_ErrRateTol is started after Start / Restart of the safety communication, too. The timer value is related to the residual error probability described later, as a different PFH is used to correlate between different timeout value settings for the enforcement of a BER at the safety layer. “SDataPDU error” consists of CRC error and an MNR error and an ID error (SDataPDU_ID, SafetyConsID).
Identifier		
DiagIdentifier	Pointer	See Table 26
Macros <..>		
<Get SDataPDU>	Macro	Instruction to take the whole SDataPDU from the OPC UA Mapper.
<use FSV>	Macro	SData_S is set to binary 0 NOTE the setting of substitute values to a different value as binary 0 is executed in the application layer or in the Machine-Specific-Interface
<use SData>	Macro	SData_S is set to SDataPDU
<Set RequestSPDU>	Macro	Instruction to transfer the whole RequestSPDU to the OPC UA Mapper
<(Re)Start WDTO_SoOPC>	Macro	WDTO_SoOPC:= 0
External Event		
Restart Cycle	Event	The external call of SafetyConsumer can be interpreted as event “Restart Cycle”

919 *) A macro is a shorthand representation for operations described in the according definition.

920 **Table 24 – SafetyConsumer driver states**

STATE NAME	STATE DESCRIPTION
Initialization	// Initial state of the SafetyConsumer driver instance. SPDU:= 0, SData_S:= 0, FSV_Activated_S:= 1, FSV_Activated_NF:=1, OAC_Req_S:= 0, OAC_Req_NF:=0, OAP_Req_S:= 0, OAC_i:=0, FaultReqOA_i:=0, TestMode_S:= 0, SComError_S:= 0, SComErrDiag_NF:= 0
S11_Wait for (Re)Start	// Safety Layer is waiting (Re)Start
S12_initialize MNR	// Use previous MNR if known - or random MNR within the allowed range (e.g. after cold start) MNR:= (previous MNR if known) or (random MNR)

STATE NAME	STATE DESCRIPTION
	If MNR<MNR_min [1] **Then** MNR:= MNR_min
S13_PrepareRequest	// Build RequestSPDU and send
S14_WaitForChangedSPDU	// Safety Layer is waiting on next SDataPDU from SafetyProvider
S15_CRCCheckSPDU	// Check CRC
S16_CheckSDataPDU	// Check SafetyConsID and SDataPDU_ID and MNR_SPDU
S17_Error	-
S18_ProvideSData_S	// Provide SData_S to the application program

Table 25 – SafetyConsumer driver transitions

TRANSITION	SOURCE STATE	TARGET STATE	GUARD CONDITION	ACTIVITY
T12	Init	S11	-	
T13	S11	S12	[Enable_C==1]	Start CountdownTimer T_ErrRateTol with T_ErrRateTol_P, **If** SDataProvID_C<>0 **Then** {SDataProvID_i:= SDataProvID_C} **Else** {SDataProvID_i:= SDataProvID_P} **If** SDataBaseID_C <>0 **Then** {SDataBaseID_i:= SDataBaseID_C} **Else** {SDataBaseID_i:= SDataBaseID_P} calculate SDataPDU_ID
T14	S12	S13	// **MNR initialized**	**Start WDTO_SoOPC**
T15	S18	S11	// **Termination** [Enable_C==0]	<use FSV>
T16	S13	S14	// **Build Request SPDU and send**	prevMNR_i:= MNR, **If** MNR== 0xFFFFFFFFF **Then** MNR:= MNR_min, **Else** MNR:= MNR + 1 Build RequestSPDU <Set RequestSPDU>
T17	S14	S15	**// New SDataPDU received** <Get SDataPDU> [MNR_SPDU<>prevMNR_i] [2]	-
T18	S14	S17	**// WDTimeout** **After:** WDTO_SoOPC >= WDTO_SoOPC_P	**<use FSV>**, FSV_Activated_S:= 1, FSV_Activated_NF, **If** OA_Necessary_P== 1 **Then** FaultReqOA_i:= 1 send diagnostic message with DiagIdentifier(CommErrTO), SComErrDiag_NF:= 1
T19	S15	S13	// **When CRC err and T_ErrRateTol==0** [(calculated CRC<>CRC_SPDU) && T_ErrRateTol== 0]	Re-Start CountdownTimer T_ErrRateTol with T_ErrRateTol_P, SComErrDiag_NF:= 1, send diagnostic message with DiagIdentifier (CRCerrIgn)

[1] a random MNR may have a value between 0 and MNR_min.

[2] Another event like “Method completion successful” can be used as guard condition “New SDataPDU received” as well

<table>
<tr><th>TRANSITION</th><th>SOURCE STATE</th><th>TARGET STATE</th><th>GUARD CONDITION</th><th>ACTIVITY</th></tr>
<tr><td>T20</td><td>S15</td><td>S17</td><td>// When CRC err and T_ErrRateTol>0
[(calculated CRC<>CRC_SPDU) && T_ErrRateTol >0]</td><td>Re-Start CountdownTimer T_ErrRateTol with T_ErrRateTol_P
SComErrDiag_NF:= 1
<use FSV>, FaultReqOA_i:= 1, FSV_Activated_S:= 1,
FSV_Activated_NF, CRCerrOA:= 1,
send diagnostic message with DiagIdentifier(CRCerrOA)</td></tr>
<tr><td>T21</td><td>S15</td><td>S16</td><td>// When CRCCheckOk
[(calculated CRC==CRC_SPDU)]</td><td>-</td></tr>
<tr><td>T22</td><td>S16</td><td>S18</td><td>// SPDU OK
[SDataPDU_ID_SPDU== SDataPDU_ID) && (SafetyConsID_SPDU== SafetyConsID && MNR_SPDU==MNR]</td><td>Stop WDTO_SoOPC,
SComErrDiag_NF:=0,
OAP_Req_S:=OAP_SF,
If ActivateFSV_SF==1
Then ActivateFSV_i:= 1
If {ActivateFSV_SF==0 && ActivateFSV_i==1 && OA_Necessary_P==1}
Then {FaultReqOA_i:=1; send diagnostic message with DiagIdentifier(FSV_C_OA)}
If ActivateFSV_SF==0
Then ActivateFSV_i:= 0
If FaultReqOA_i==1
Then {OAC_Req_S:= 1, OAC_Req_NF:=1, FaultReqOA_i:= 0}

If OAC_C==0 && OAC_Req_S==1 [1)]
Then OAC_i:= 1
If OAC_C==1 && OAC_i:= 1
Then {OAC_Req_S:=0, OAC_Req_NF:=0, OAC_i:= 0} [3]

If OAC_Req_S==0 && ActivateFSV_SF==0
Then {<use SData>, FSV_Activated_S:= 0, FSV_Activated_NF}

If OAC_Req_S==1 || ActivateFSV_SF==1
Then {<use FSV>, FSV_Activated_S:= 1. FSV_Activated_NF}
TestMode_S:= TestMode_SF</td></tr>
</table>

[3] This condition is used to accept a rising edge of OAC_C only if it occurs after OAC_Req_S is set to 1.

<table>
<tr><th>TRANSITION</th><th>SOURCE STATE</th><th>TARGET STATE</th><th>GUARD CONDITION</th><th>ACTIVITY</th></tr>
<tr><td>T23</td><td>S16</td><td>S13</td><td>// SPDU NOK and T_ErrRateTol=0

[(SDataPDU_ID_SPDU<> SDataPDU_ID) ||
(SafetyConsID_SPDU<> SafetyConsID
|| MNR_SPDU<>MNR) && T_ErrRateTol==0]</td><td>Start CountdownTimer T_ErrRateTol with T_ErrRateTol_P,
Send diagnostic message according the detected error:
DiagIdentifier (SD_IDerrIgn) or DiagIdentifier (ColDerrIgn) or DiagIdentifier (MNRerrIgn)
SComErrDiag_NF:=1</td></tr>
<tr><td>T24</td><td>S16</td><td>S17</td><td>// SPDU NOK and T_ErrRateTol>0

[(SDataPDU_ID_SPDU<> SDataPDU_ID) ||
(SafetyConsID_SPDU<> SafetyConsID
|| MNR_SPDU<>MNR) && T_ErrRateTol>0]</td><td>Re-Start CountdownTimer T_ErrRateTol with T_ErrRateTol_P
Send diagnostic message according the detected error:
DiagIdentifier (SD_IDerrOA) or DiagIdentifier (ColDerrOA) or DiagIdentifier (MNRerrOA)
SComErrDiag_NF:=1,
<use FSV>, FaultReqOA_i:= 1, FSV_Activated_S:= 1, FSV_Activated_NF</td></tr>
<tr><td>T25</td><td>S17</td><td>S18</td><td>// SPDU NOk
-</td><td>Stop SafetyConsumer Timer</td></tr>
<tr><td>T26</td><td>S18</td><td>S13</td><td>// Restart Cycle
[Enable_C==1]</td><td><(Re)Start WDTO_SoOPC></td></tr>
</table>

8.1.2.5 SafetyConsumer sequence diagram for OA

Figure 22 shows the sequence after a “a second SDataPDU error was detected before countdown timer T_ErrRateTol stops”.

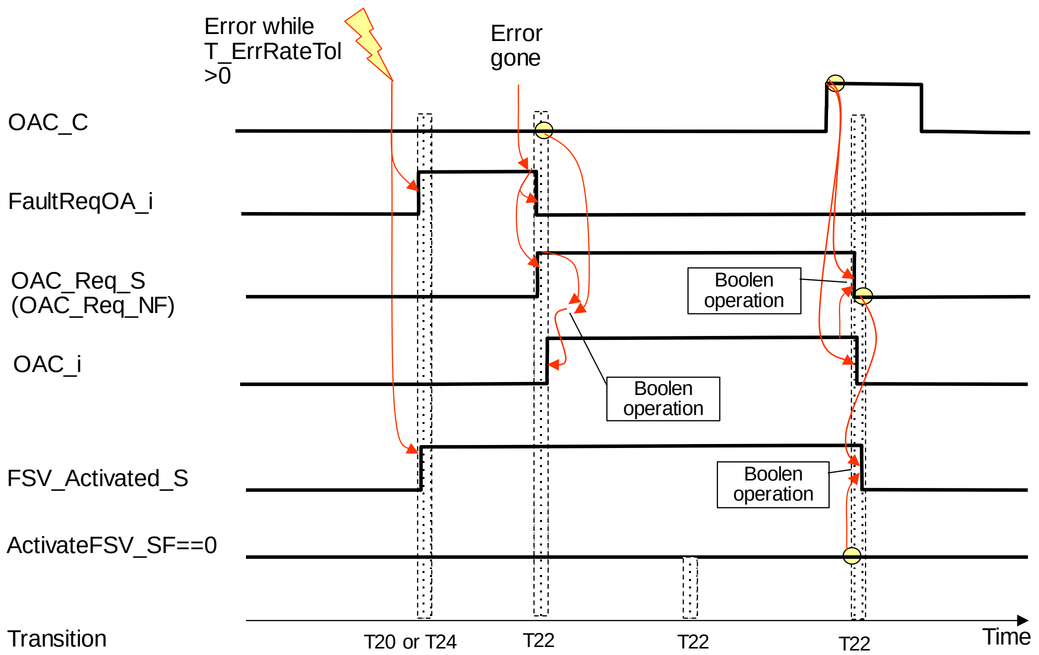


Figure 22 – Sequence diagram for OA

After the error is gone the sequence follows the logic of T22 in Table 25.

8.2 Safety communication layer protocol SafetyProsumer Classic

8.2.1 SPDU format

The SPDU format will be conform to the behavior of IEC 61784-3-3 Ed3. Its integration in OPC UA will be described in the next version of this specification.

8.2.2 SPDU structure

The SPDU structure will be conform to the behavior of IEC 61784-3-3 Ed3. Its integration in OPC UA will be described in the next version of this specification.

9 Diagnosis

The Safety over OPC UA diagnosis supports the troubleshooting of the safety communication.

Safety over OPC UA provides two features for diagnostics:

- Safety over OPC UA diagnosis messages which are generated in case of an event which is intended to inform the operator and/or commissioning engineer to improve the settings, see Table 26. The implementation and providing of the diagnosis messages are vendor specific.
- A method “ReadSDiagnosticDataV1” to get information for the service technician, see 9.2.

9.1 Diagnosis messages

Table 26 – Safety layer diagnosis messages

Classification *)						Int. identifier	Textual representation of diagnosis messages	Mandatory
1)	2)	3)	4)	5)	6)			
				x		MismBaseID	Mismatch of safety data BaseID	Yes
				x		MismPrrovID	Mismatch of safety data ProviderID	Yes
				x		MismStrID	Mismatch of safety data StructureID	Yes
x						CRCerrIgn	Communication error: CRC error tolerated	Yes
	x	x				CRCerrOA	Communication error: CRC error which requires Operator acknowledge	Yes
x						SD_IDerrIgn	Communication error on SDataPDU_ID tolerated	Yes
	x	x				SD_IDerrOA	Communication error on SDataPDU_ID which requests Operator acknowledge	Yes
x						CoIDerrIgn	Communication error on ConsumerID tolerated	Yes
	x	x				CoIDerrOA	Communication error on ConsumerID which requests operator acknowledge	Yes
x						MNRerrIgn	Communication error: Monitoring number error tolerated	Yes
	x	x				MNRerrOA	Communication error: Monitoring number error which requests operator acknowledge	Yes
	x					CommErrTO	Communication error on timeout	Yes
			x			ApplErrTO	Application error on Timeout	No
					X	FSV_C_OA	Operator acknowledge required, due to ActivateFSV_C at the provider	Yes

*) The following classification is specified:
1) Transient communication error
2) Permanent communication error
3) Transmission quality seems not to be sufficient
4) Application error
5) Parameter error
6) No error

Transmission errors shall not lead to diagnosis message flooding, therefore, only a single message should be shown, if multiple communication errors occur in sequence.

Optional Feature:
Extent diagnostic data by expected value and received value, e.g.
Incorrect ProviderID:
Expected ProviderID: 0x00000005
Received ProviderID: 0x00000007

9.2 Method ReadSDiagnosticDataV1

This method (as part of the OPC UA Mapper) is provided for each SafetyProvider serving as a diagnostic interface. For time series observation, this interface must be polled, e.g. by the diagnostic device. For details, refer to the OPC UA information model described in 6.

The diagnostic interface method takes no input parameters, and returns both the input- and output parameters of the last call of the method ReadSafeData.

Additionally, a 2-Byte sequence number is added to the diagnostic interface, which will allow detecting a missed call due to polling. It counts the number of accesses to ReadSafeData.

A best practice recommendation is to store all parameters if SComErr_diag is <> 0.

10 Safety communication layer management

10.1 SPDU parameter assignment

Export and import of SPDU parameters shall be done by exporting and importing the OPC UA information model, e.g. using XML.

10.2 Safety function response time part of communication

The part of Safety function response time which is used for a Safety over OPC UA communication $SFRT_{SoOPC}$ is specified in **Equation 1**.

Equation 1 Calculation of safety function response time part of Safety over OPC UA

$$SFRT_{SoOPC} <= WDTO\_SoOPC + WDTO\_SA$$

$SFRT_{SoOPC}$	Safety function response time part performed by Safety over OPC UA communication.
WDTO_SoOPC	SPDU WatchDog time period at SafetyConsumer from Request (T14 or T26) to response with error free SDataPDU (T22) or in case of an SPDU-error (T18).
WDTO_SA	maximum sample time period of the SafetyConsumer application. This is used in cyclic (high demand and low demand) safety applications.

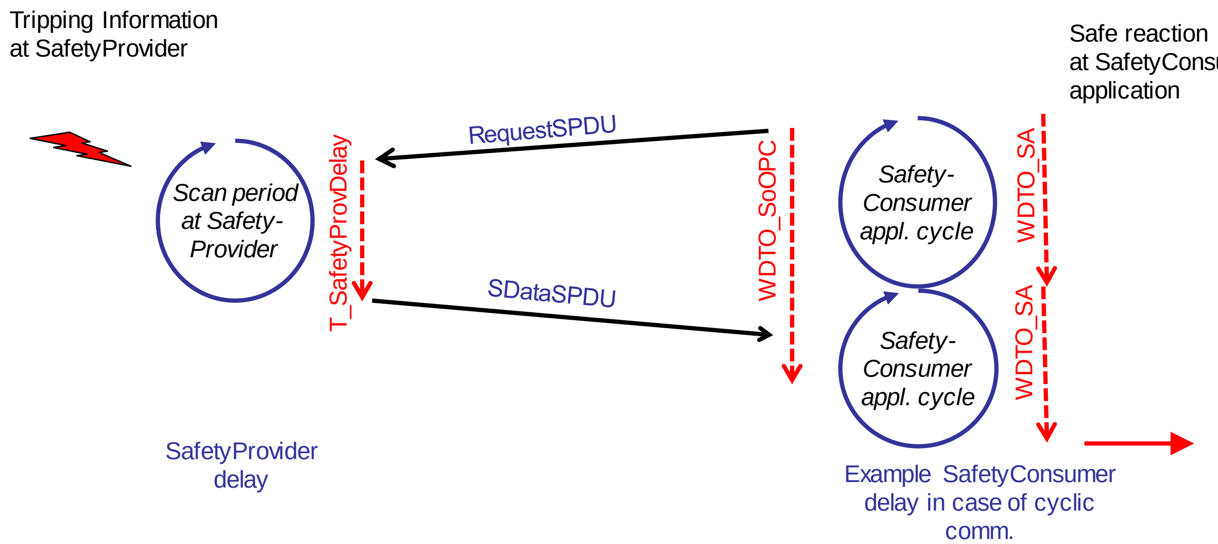


Figure 23 – principle delay times and used Watchdogs

The additional term used in Figure 23 specified as follows:

T_SDataProvDelay	worst case SafetyProvider delay in error free operation. Typically, one scan time period of the SDataProvider.

Both WDTO_SoOPC and WDTO_SA are parameters of the SafetyConsumer. The SafetyConsumer delay depends on the maximum sample time of the SafetyConsumer application. At commissioning the integrator should be advised to design it shorter than half of the target $SFRT_{SoOPC}$. If the watchdog time WDTO_SoOPC has been chosen too short, spurious trips may occur. For avoiding this, WDTO_SoOPC shall be chosen as shown in Equation 2.

Equation 2 Selection of the watchdog parameter WDTO_SoOPC

$$\text{TO\_SoOPC} >= \text{T\_CD\_RequestSPDU} + \text{T\_SDataProvDelay} + \text{T\_CD\_SDataPDU} + \text{WDTO\_SA}$$

where

T_CD_RequestSPDU: is communication delay for RequestSPDU

T_CD_SDataPDU: is communication delay for SDataPDU

NOTE to Equation 2: the reason why WDTO_SA is part of the summation, is because in a cyclic call of SafetyConsumer State S18, it may take one cycle after the asynchronous reception of SDataPDU to execute the checks.

To support the calculation of WDTO_SoOPC the SafetyProvider shall provide the T_SDataProvDelay.

System manufacturers shall provide their individual adapted calculation method if necessary.

The SafetyConsumer delay depends on the maximum sample time of the SafetyConsumer application. At commissioning the integrator should be advised to design it shorter than half of the target $SFRT_{SoOPC}$.

11 Constraints and system requirements

11.1 Constraints on SPDU-Parameter

11.1.1 SDataBaseID and SDataProvID

The pair of SDataProvID and SDataBaseID is used to check the authenticity of the SDataPDU by the SafetyConsumer. It must be ensured that these pairs of IDs are either unique for all SafetyProviders, or that the probability that any given pair of SafetyProviders is using the same ID-pair is smaller or equal to 10E-23.

On most systems, this can be achieved by generating a GUIDv4 or GUID, which already contains a random number of more than 96 bits.

The SDataProvID will be generated at engineering time.

The SDataBaseID will be generated at engineering time or at first commissioning.

The SDataBaseID and the SDataProvID shall be stored nonvolatile (i.e. persistent).

11.1.2 SafetyConsID

The SafetyConsID is a simple authenticity ID of the SData receiver. It is used to check (SafetyConsID together with the MNR_SPDU) whether the response is the answer to the request to verify the WDTO_SoOPC (SafetyConsID and MNR_SPDU as expected).

This is a random value, which can be generated at engineering time or at every start-up of the Device with the SafetyConsumer(s).

11.2 Constraints on Start value of MNR

The MNR is used to check the timeliness of transmitted data. Over the lifetime of a device, this value should take a wide range of numbers. Therefore, a random number R from which the stream of MNRs is derived shall be generated whenever the system is restarted (i.e. in state S12 of the consumer state machine). This will avoid using an identical stream of MNRs each time, which would be especially problematic, if the device is restarted very frequently.

There are no particular requirements on how the stream of MNRs is derived from R. For instance, the values R, or the MNR+1 of the last MNR sequence of this instance can be used as MNR.

A possible way to generated R is to calculate a CRC32 over the current time.

The requirements on the quality of the generator for R are low in this case. Any algorithm which fulfills the following test shall be accepted:

- The device is restarted 10 times, and R is sampled.
- If all occurrences are different, the test is passed.

11.3 Constraints on the calculation of system characteristics

11.3.1 Probabilistic considerations

The data integrity checking mechanism of the Safety over OPC UA is independent from the mechanisms of the underlying communication system, which is called a "black channel". Thus, it can be used for backplane communication channels as well.

In order to prevent a SPDU from carrying "0" only, a CRC-result of 0x00000000 is not accepted.

In order to prevent a SPDU from carrying "1" only, the SData Qualifier "ActivateFSV_SF" in Control-Word interprets "1" as fail-safe values.

According to IEC 61784-3 draft Ed4 and IEC 62280, the "properness" of the used CRC generator polynomials shall be proven. This requires calculation of the residual error probability as a function of the bit error probability for a given polynomial, here for the 32-bit version (polynomial *0xF4ACFB13*). , The residual error probability of this polynomial is calculated by the methods according [2], [3], and [4].

This polynomial is assessed as "proper" as there is no significant "humpback" curve with increasing bit error probability.

Figure 24 shows diagram calculated according [3] for the 32-bit CRC generator polynomial *0xF4ACFB13*.

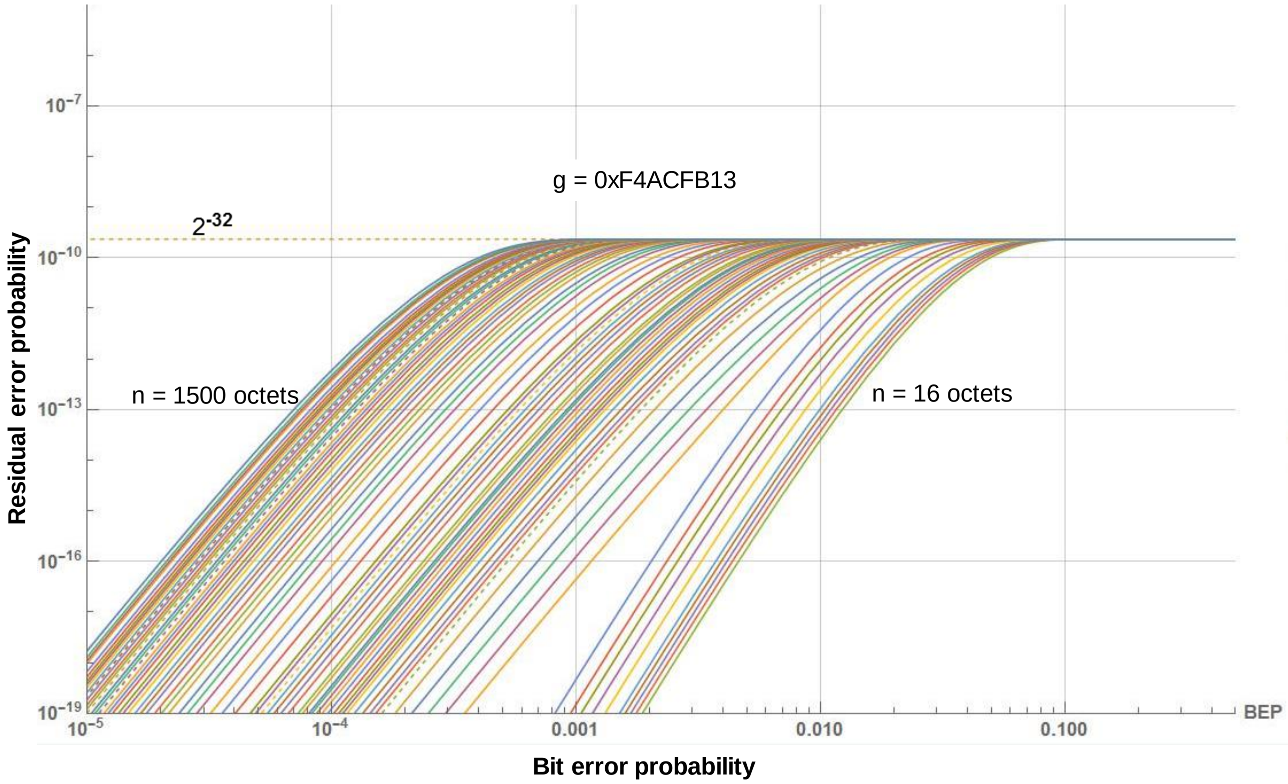


Figure 24 – Residual error probabilities for the 32-bit CRC polynomial

The terms used in Figure 24 are specified below:

g = generator polynomial

n = length of data (including the CRC signature)

11.3.2 Safety related assumptions

The boundary conditions and assumptions for safety assessments and calculations of residual error rates are listed here.

Generally:

- Number of retries in the black channel:
  No restrictions
- Black Channel CRC polynomials:
  No restrictions
- Message storing elements:
  No restrictions; any number of message storing elements is permitted
- Size of SData within one SPDU:
  ≤ 1500 bytes
- Attention of Table 28 row Operator Acknowledge

11.3.3 Non safety related constraints (availability)

- The atomic (consistent) delivery of entire SPDUs at the SafetyProvider and the SafetyConsumer shall be guaranteed.

11.4 Total residual error rate of Safety over OPC UA communication

The total residual error rate for the safety communication channel is the sum of the individual residual error rates for Timeliness, Authenticity, Data Integrity, and Masquerade.

Implementations according to Safety over OPC UA provide a communication, with the following PFH / respectively PFDavg values per logical connection of the safety function, are depending on the parameter T_ErrRateTol, see Table 27.

The SafetyConsumer limits directly the rate of detected faulty SDataPDUs and indirectly the rate of undetected faulty SDataPDUs entering the SafetyConsumer by means of the CountdownTimer T_ErrRateTol, see 8.1.2.4.

The parameter T_ErrRateTol_P effects the tolerated rate of errors (data integrity errors, incorrect sequence SPDUs, misdirected SPDUs), detected by Safety over OPC UA (error at CRC, MNR, SafetyConsID or SDataPDU_ID).

If the rate becomes greater than the maximum accepted rate, the operator is informed, see signal OAC_Req_S and safety manual, Table 28 row Operator Acknowledge.

Table 27 – The total residual error rate for the safety communication channel

T_ErrRateTol_P	Allowed for SIL range	Total Residual error rate for one logical connection of the safety function (PFH)	Total Residual error probability for one logical connection of the safety function (PFDavg)
6 Minutes	**Up to SIL 2**	$< 10^{-8}$ / h	$< 10^{-4}$
60 Minutes	**Up to SIL 3**	$< 10^{-9}$ / h	$< 10^{-5}$
600 Minutes	**Up to SIL 4**	$< 10^{-10}$ / h	$< 10^{-6}$

The requirements for the implementation of nodes are specified in the IEC 61508. The value of T_ErrRateTol influences <u>only</u> the PFH/PFD of the safety communication channel.

11.5 Safety manual

According to IEC 61508-2, Safety over OPC UA suppliers shall provide a safety manual. In case of Safety over OPC UA, the instructions, information and parameters of Table 28 shall be included.

Table 28 – Information to be included in the safety manual

	Item	Instruction and/or parameter	Remark
1	Safety handling	Instructions on how to configure, parameterize, commission, test, and lock this device safely in accordance with IEC 61508 and IEC 61784-3	
2	PFH, respectively PFDavg	The PFH, respectively PFDavg per logical connection of the safety function.	assumptions see 11.3.2 Total residual error rate, see 11.4
3	$SFRT_{SoOPC}$	At commissioning the integrator should be advised on how to design the maximum cycle time for SafetyConsumer which shall be shorter than half of the target $SFRT_{SoOPC}$.	The implementation and error reaction of WDTO_SA is in the responsibility of the vendor/integrator.

	Item	Instruction and/or parameter	Remark
4	SDataBaseID / SafetyConsID	The SafetyConsID shall be unique for all SafetyProvider with the same SDataBaseID	
5	Commissioning	At commissioning of the safety function of the SafetyConsumer Application the association to this SafetyProvider requires verification and validation according to the relevant safety manuals.	
6	Operator Acknowledge	In case of “frequent indications” of OAC_Req_S==1 with “classification transmission quality seems not to be sufficient” (see Table 26) a check of the installation (for example electromagnetic interference), network traffic load, or transmission quality should be performed. Frequent indications are defined as - more than ones per day in SIL2 and SIL 3 applications - more than ones per week in SIL4 applications	
7	Duration of demand	In safety applications where the duration of a demand signal is short (e.g. shorter than the process safety time), and it is crucial that the consumer application never misses a demand, then a bidirectional communication must be arranged and the confirmation of receiving the demand at consumer side must be implemented in the application program, by sending appropriate information within the SData.	
8	high demand and low demand applications	The SafetyConsumer shall be called cyclic within a shorter time frame than the WDTO_SoOPC.	
9	Maintenance	Specifications for system behaviour in case of device repair and replacement.	

11.6 Indicators and displays

A system with implementation of a SafetyConsumer requires the possibility of an indication of OAC_Req_S==1 either by an indicator (LED) or at an HMI.

If the indication is performed by LED is is blinking with 0,5 Hz in green mode (= safety communication ok but OA_C required).

If the indication is performed as a message at HMI the recommended text is: Operator acknowledge requested.

12 Assessment

12.1 Safety policy

In order to prevent and protect the manufacturers and vendors of Safety over OPC UA product from possibly misleading understandings or wrong expectations and gross negligence actions regarding safety-related developments and applications the following shall be observed and explained in each training, seminar, workshop and consultancy.

- Any device automatically will not be applicable for safety-related applications just by using OPC UA and a safety communication layer.
- In order to enable a product for safety-related applications, appropriate development processes according to safety standards shall be observed (see IEC 61508, IEC 61511, IEC 60204-1, IEC 62061, and ISO 13849-2) and/or an assessment from a competent assessment body shall be achieved.

- The manufacturer of a safety product is responsible for the correct implementation of the safety communication layer technology, the correctness and completeness of the product documentation and information.
- Additional important information about actual corrigendum through concluded change requests shall be considered for implementation and assessment. This information can be obtained from the organizations OPC Foundation and PI.
- The implementation of the Statemachines shall be tested by the “Automated layer test for Safety over OPC UA” at a accredited test laboratory or a notified body.

12.2 Obligations

As a rule, the international safety standards are accepted (ratified) globally. However, since safety technology in automation is relevant to occupational safety and the concomitant insurance risks in a country, recognition of the rules pointed out here is still a sovereign right. The national "Authorities" decide on the recognition of assessment reports.

NOTE Examples of such “Authorities” are the IFA (Institut für Arbeitsschutz der Deutschen Gesetzlichen Unfallversicherung / Institute for Occupational Safety and Health of the German Social Accident Insurance) in Germany, HSE (Health and Safety Executive) in UK, FM (Factory Mutual / Property Insurance and Risk Management Organization), UL (Underwriters Laboratories Inc. / Product Safety Testing and Certification Organization), or the INRS (Institut National de Recherche et de Sécurité) in France.

12.3 Automated layer test for Safety over OPC UA (informative)

For details, see Safety over OPC UA test specification.

12.3.1 Testing principle

An exemplary test principle for Safety over OPC UA is presented. The Safety over OPC UA test is a fully automated verification based on a mathematical model of the Safety over OPC UA finite states generated test patterns for all kinds of possible correct and incorrect SPDUs, parameters, and interactions with the upper interface of the SafetyProvider / SafetyConsumer driver. These test patterns together with the expected responses/stimulations are packed as an XML document and installed in the test tool software. The test tool performs the complete test patterns while connected to the Safety over OPC UA layer under test, compares the nominal with the actual reactions and is recording the results that can be printed out for the test report.

The automated Safety over OPC UA layer tester will be approved by the Notified Body.

Figure 25 shows the structure of the layer tester for SafetyProvider / SafetyConsumer.

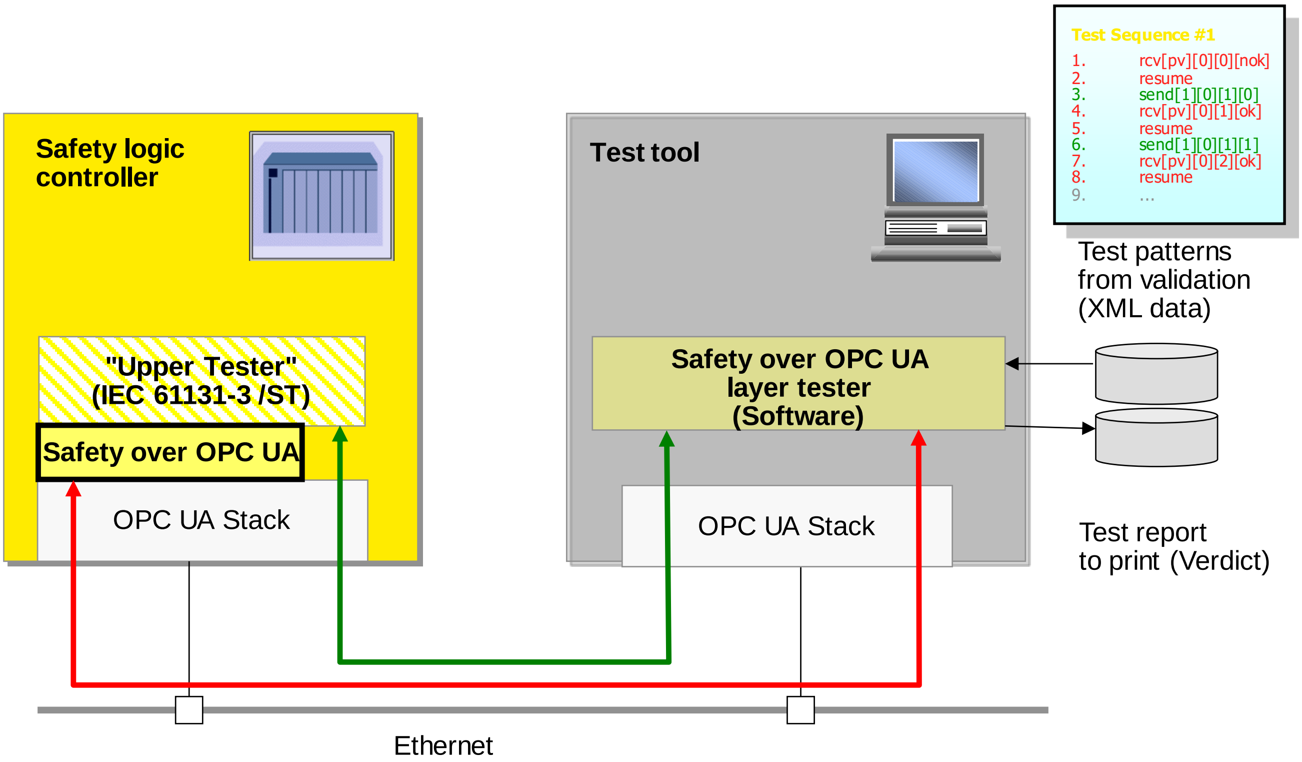


Figure 25 – Automated SafetyProvider / SafetyConsumer test

12.3.2 Test configuration

The SafetyProvider / SafetyConsumer tester "simulates" the behavior of an opposite SafetyProvider / SafetyConsumer Layer. Thus, it shall be configured according to the deployed OPC UA communication system. This can be done with the help of an XML file associated with the tester.

The support application (Upper Tester) within the SafetyProvider / SafetyConsumer as DUT (Device under Test) consists of a "Copy" application program (see Figure 26 and Figure 27) reading and writing the application interface between the SafetyProvider / SafetyConsumer driver instance.

The "Copy" application program can be implemented using standard program languages such as IEC 61131-3, Structured Text and run on the non safety part of PLC.

For the provider, testing is also possible to some extent, without such an upper tester.

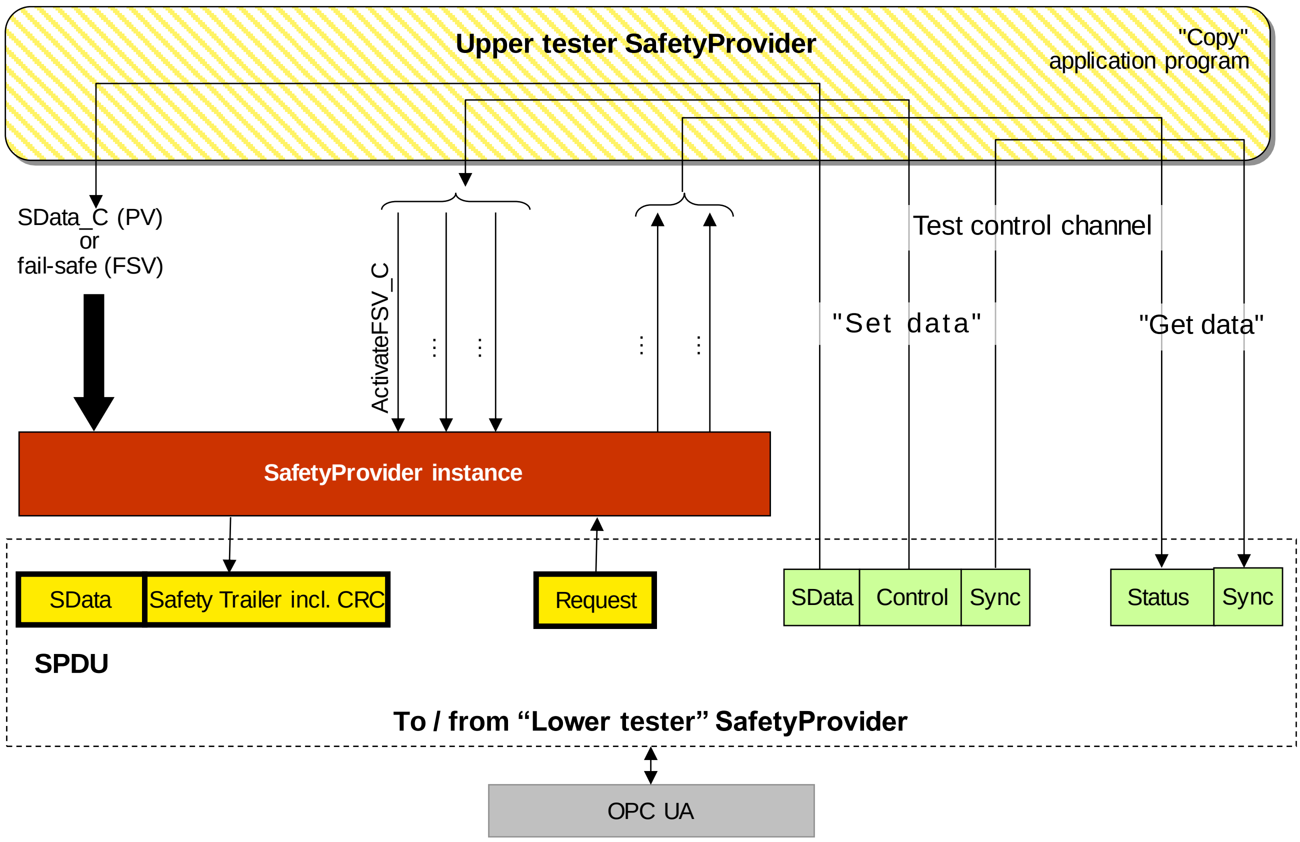


Figure 26 – Copy application program in "Upper Tester" within the SafetyProvider

The whole application program within the SafetyProvider / SafetyConsumer "Upper Tester" consists of two parts. One part, the "Set data", is responsible for the data transfer from the test control channel into the SafetyProvider / SafetyConsumer driver instance.

The second part, the "Get data", is responsible for the reverse data transfer out of the SafetyProvider / SafetyConsumer driver instance into the test control channel. In between these two activities, the SafetyProvider / SafetyConsumer driver is executing the protocol. The "Sync" data within the test control channel is looped back within the "Upper Tester".

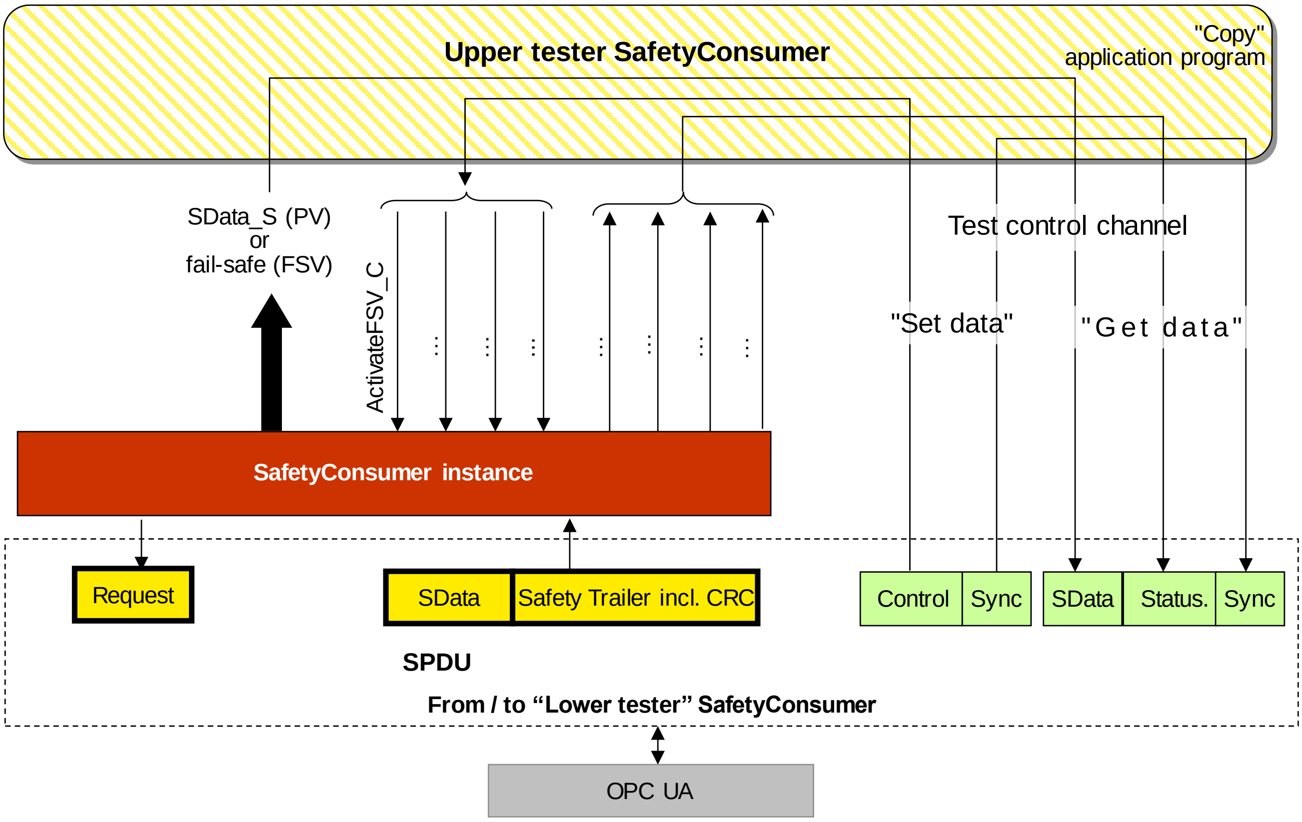


Figure 27 – Copy application program in "Upper Tester" within the SafetyConsumer

The "Lower tester" SafetyConsumer verifies the reaction on a particular test step by checking the SPDU or in case of timeout tests by checking, whether the configured timeout has been exceeded 30%.

NOTE A tolerance of 30% is acceptable as the scope of this testing is limited to the services and behavior of the state machine and not the test of precise timing of WDTO_SoOPC.

For the SafetyConsumers (see Figure 27), whether testing is also possible to some extent, without such an upper tester is still under consideration.

Suggestion to test a SafetyConsumer that is implemented in a safety device which is not programable:

- Link signal FSV_Activated_S to ConsNFlag "FSV_Activated_NF"
- Link signal OAP_Req_S to OAC_C

NOTE This SafetyConsumer type needs no control input of SDataProvID_C, SDataBaseID_C, Enable_C, and no output signal TestMode_S and OAC_Req_S.

With this rule this SafetyConsumer type can be tested.

The sequence chart in Figure 28 illustrates the data flow between SafetyProvider / Consumer layer tester, SafetyProvider / SafetyConsumer driver and Upper Tester.

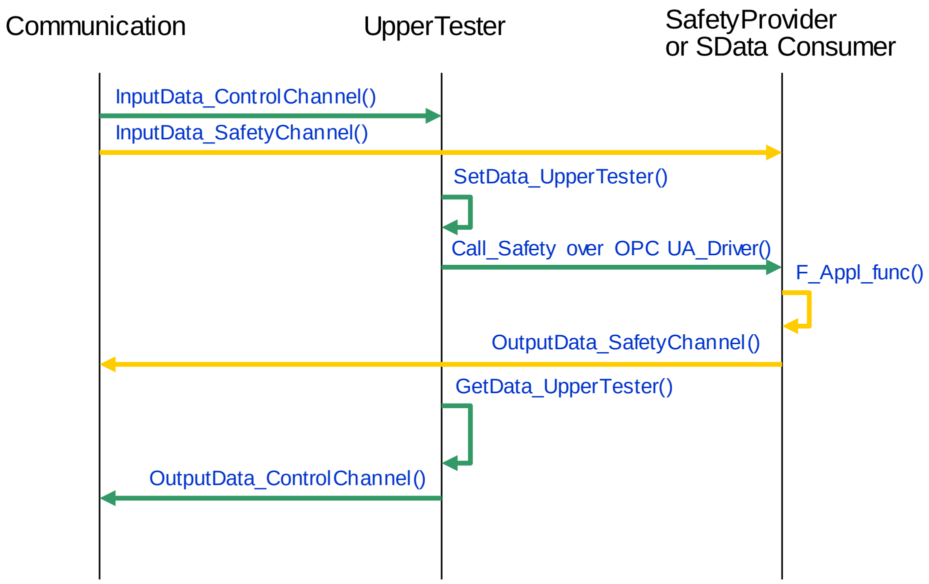


Figure 28 – Sequence chart of the "Upper Tester"

As “Upper Tester” and SafetyProvider / SafetyConsumer respond at different times and the responses are typically not synchronized, the following trigger will be used to evaluate the response from upper tester and the response from SafetyProvider / SafetyConsumer: either as soon as both are changed to the previous responses or at the end of the Timeout according WDTO_SoOPC_P.

13 Profiles and Namespaces

13.1 Namespace Metadata

Table 29 defines the namespace metadata for this specification. The *Object* is used to provide version information for the namespace and an indication about static *Nodes*. Static *Nodes* are identical for all *Attributes* in all *Servers*, including the *Value Attribute*. See Part5 for more details.

The information is provided as *Object* of type *NamespaceMetadataType*. This *Object* is a component of the *Namespaces Object* that is part of the *Server Object*. The *NamespaceMetadataType ObjectType* and its *Properties* are defined in Part5.

The version information is also provided as part of the ModelTableEntry in the UANodeSet XML file. The UANodeSet XML schema is defined in Part 6.

Table 29 – NamespaceMetadata Object for this Specification

Attribute	**Value**		
BrowseName	http://opcfoundation.org/UA/Safety		
References	**BrowseName**	**DataType**	**Value**
HasProperty	NamespaceUri	String	http://opcfoundation.org/UA/Safety
HasProperty	NamespaceVersion	String	1.0
HasProperty	NamespacePublicationDate	DateTime	2019-04-01
HasProperty	IsNamespaceSubset	Boolean	False
HasProperty	StaticNodeIdTypes	IdType[]	-
HasProperty	StaticNumericNodeIdRange	NumericRange[]	-
HasProperty	StaticStringNodeIdPattern	String	-

13.2 Conformance Units and Profiles

This chapter defines the corresponding *Profiles* and *Conformance Units* for the OPC UA Information Model for Safety. *Profiles* are named groupings of *Conformance Units*. *Facets* are *Profiles* that will be combined with other *Profiles* to define the complete functionality of an OPC UA *Server* or *Client.*

13.3 Server Facets

The following tables specify the *Facets* available for *Servers* that implement the Safety Information Model companion specification.

Table 30 defines a facet for the minimum functionality necessary for providing safe data over OPC UA.

Table 30 –Server Facet Definition

Conformance Unit	**Description**	**Optional/ Mandatory**
SafetyProvider	Supports SafetyProvider, see tbd	M

13.4 Client Facets

The following tables specify the *Facets* available for *Clients* that implement the Safety Information Model companion specification.

Table 31 defines a facet for the minimum functionality necessary for consuming safe data over OPC UA.

Table 31 –Client Facet Definition

Conformance Unit	Description	Optional/ Mandatory
SafetyConsumer	Supports SafetyConsumer see tbd	M

13.5 Handling of OPC UA Namespaces

Namespaces are used by OPC UA to create unique identifiers across different naming authorities. The *Attributes NodeId* and *BrowseName* are identifiers. A *Node* in the UA *AddressSpace* is unambiguously identified using a *NodeId*. Unlike *NodeIds*, the *BrowseName* cannot be used to unambiguously identify a *Node*. Different *Nodes* may have the same *BrowseName*. They are used to build a browse path between two *Nodes* or to define a standard *Property*.

Servers may often choose to use the same namespace for the *NodeId* and the *BrowseName*. However, if they want to provide a standard *Property*, its *BrowseName* shall have the namespace of the standards body although the namespace of the *NodeId* reflects something else, for example the *EngineeringUnits Property*. All *NodeIds* of *Nodes* not defined in this specification shall not use the standard namespaces.

Table 32 provides a list of mandatory and optional namespaces used in an Safety OPC UA *Server*.

Table 32 – Namespaces used in a Safety Server

NamespaceURI	Description	Use
http://opcfoundation.org/UA/	Namespace for *NodeIds* and *BrowseNames* defined in the OPC UA specification. This namespace shall have namespace index 0.	Mandatory
Local Server URI	Namespace for nodes defined in the local server. This may include types and instances used in an AutoID Device represented by the Server. This namespace shall have namespace index 1.	Mandatory
http://opcfoundation.org/UA/Safety	Namespace for *NodeIds* and *BrowseNames* defined in this specification. The namespace index is *Server* specific.	Mandatory
Vendor specific types	A *Server* may provide vendor-specific types like types derived from *ObjectTypes* defined in this specification in a vendor-specific namespace.	Optional
Vendor specific instances	A *Server* provides vendor-specific instances of the standard types or vendor-specific instances of vendor-specific types in a vendor-specific namespace. It is recommended to separate vendor specific types and vendor specific instances into two or more namespaces.	Mandatory

Annex A: Additional information for functional safety communication

A.1 Hash function calculation

The function in "C" programming language for the 32 bit CRC signature calculations with the help of lookup tables is shown below:

```
r = crctab32 [((r >> 24) ^ *q++) & 0xff] ^ (r << 8)
```
(A.3)

For this calculation Table A.33 is used.

Table A.33 – The CRC32 lookup table for 32 bit CRC signature calculations

CRC32 lookup table (0 to 255)							
00000000	F4ACFB13	1DF50D35	E959F626	3BEA1A6A	CF46E179	261F175F	D2B3EC4C
77D434D4	8378CFC7	6A2139E1	9E8DC2F2	4C3E2EBE	B892D5AD	51CB238B	A567D898
EFA869A8	1B0492BB	F25D649D	06F19F8E	D44273C2	20EE88D1	C9B77EF7	3D1B85E4
987C5D7C	6CD0A66F	85895049	7125AB5A	A3964716	573ABC05	BE634A23	4ACFB130
2BFC2843	DF50D350	36092576	C2A5DE65	10163229	E4BAC93A	0DE33F1C	F94FC40F
5C281C97	A884E784	41DD11A2	B571EAB1	67C206FD	936EFDEE	7A370BC8	8E9BF0DB
C45441EB	30F8BAF8	D9A14CDE	2D0DB7CD	FFBE5B81	0B12A092	E24B56B4	16E7ADA7
B380753F	472C8E2C	AE75780A	5AD98319	886A6F55	7CC69446	959F6260	61339973
57F85086	A354AB95	4A0D5DB3	BEA1A6A0	6C124AEC	98BEB1FF	71E747D9	854BBCCA
202C6452	D4809F41	3DD96967	C9759274	1BC67E38	EF6A852B	0633730D	F29F881E
B850392E	4CFCC23D	A5A5341B	5109CF08	83BA2344	7716D857	9E4F2E71	6AE3D562
CF840DFA	3B28F6E9	D27100CF	26DDFBDC	F46E1790	00C2EC83	E99B1AA5	1D37E1B6
7C0478C5	88A883D6	61F175F0	955D8EE3	47EE62AF	B34299BC	5A1B6F9A	AEB79489
0BD04C11	FF7CB702	16254124	E289BA37	303A567B	C496AD68	2DCF5B4E	D963A05D
93AC116D	6700EA7E	8E591C58	7AF5E74B	A8460B07	5CEAF014	B5B30632	411FFD21
E47825B9	10D4DEAA	F98D288C	0D21D39F	DF923FD3	2B3EC4C0	C26732E6	36CBC9F5
AFF0A10C	5B5C5A1F	B205AC39	46A9572A	941ABB66	60B64075	89EFB653	7D434D40
D82495D8	2C886ECB	C5D198ED	317D63FE	E3CE8FB2	176274A1	FE3B8287	0A977994
4058C8A4	B4F433B7	5DADC591	A9013E82	7BB2D2CE	8F1E29DD	6647DFFB	92EB24E8
378CFC70	C3200763	2A79F145	DED50A56	0C66E61A	F8CA1D09	1193EB2F	E53F103C
840C894F	70A0725C	99F9847A	6D557F69	BFE69325	4B4A6836	A2139E10	56BF6503
F3D8BD9B	07744688	EE2DB0AE	1A814BBD	C832A7F1	3C9E5CE2	D5C7AAC4	216B51D7
6BA4E0E7	9F081BF4	7651EDD2	82FD16C1	504EFA8D	A4E2019E	4DBBF7B8	B9170CAB
1C70D433	E8DC2F20	0185D906	F5292215	279ACE59	D336354A	3A6FC36C	CEC3387F
F808F18A	0CA40A99	E5FDFCBF	115107AC	C3E2EBE0	374E10F3	DE17E6D5	2ABB1DC6
8FDCC55E	7B703E4D	9229C86B	66853378	B436DF34	409A2427	A9C3D201	5D6F2912
17A09822	E30C6331	0A559517	FEF96E04	2C4A8248	D8E6795B	31BF8F7D	C513746E
6074ACF6	94D857E5	7D81A1C3	892D5AD0	5B9EB69C	AF324D8F	466BBBA9	B2C740BA
D3F4D9C9	275822DA	CE01D4FC	3AAD2FEF	E81EC3A3	1CB238B0	F5EBCE96	01473585
A420ED1D	508C160E	B9D5E028	4D791B3B	9FCAF777	6B660C64	823FFA42	76930151
3C5CB061	C8F04B72	21A9BD54	D5054647	07B6AA0B	F31A5118	1A43A73E	EEEF5C2D
4B8884B5	BF247FA6	567D8980	A2D17293	70629EDF	84CE65CC	6D9793EA	993B68F9
This table contains 32 bit values in hexadecimal representation for each value (0 to 255) of the argument a in the function crctab32 [a]. The table should be used in ascending order from top left (0) to bottom right (255).							

A.2 Use cases for Operator Acknowledge

A.2.1 Explanation

Safety over OPC UA supports Operator Acknowledge both on the SafetyProvider side, and on the SafetyConsumer side. For this purpose, both the interface of the SafetyProvider and the SafetyConsumer comprise a Boolean input called OAP_C and OAC_C, respectively. The safety application can read the status of these inputs on the consumer side via the Boolean outputs OAC_Req_S and OAP_Req_S, respectively.

The following sections show some examples on how to use these inputs and outputs. Dashed lines indicate that the corresponding input or output are not used in this use case. For details, see 7.3 and 7.4.

A.2.2 Use case 1: unidirectional comm. and OA on the SafetyConsumer side

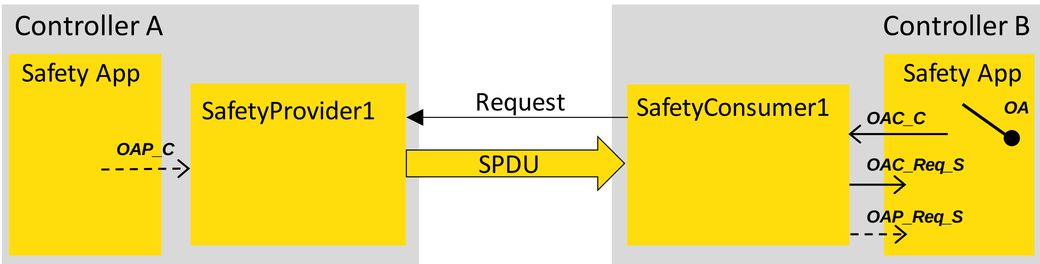


Figure 29 – OA in unidirectional safety communication

A.2.3 Use case 2: bidirectional comm. and dual OA

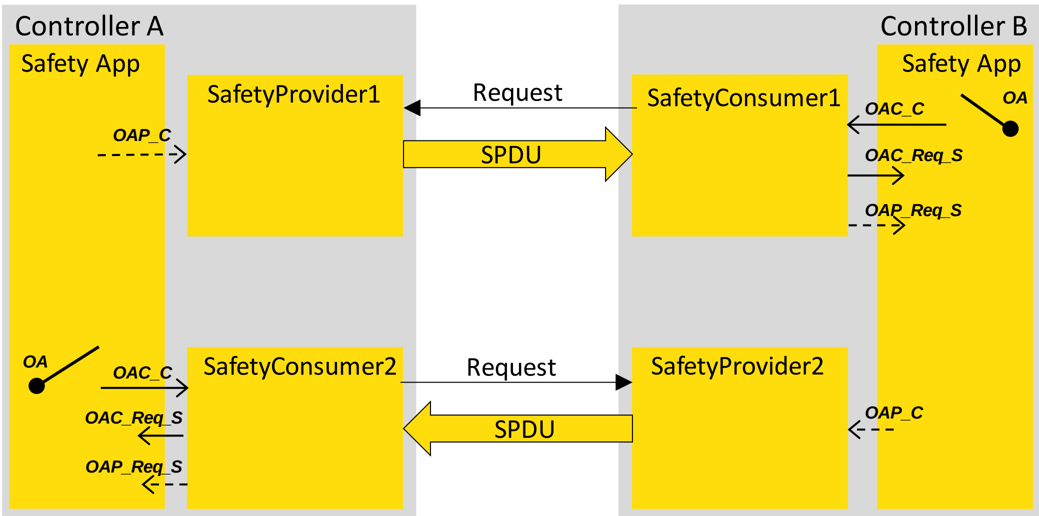


Figure 30 – Two-sided OA in bidirectional safety communication

In this scenario, operator acknowledge is done independently for both directions. Only after both sides have been acknowledged, process values are transferred and delivered in both directions.

A.2.4 Use case 3: bidirectional comm. and single, one-sided OA

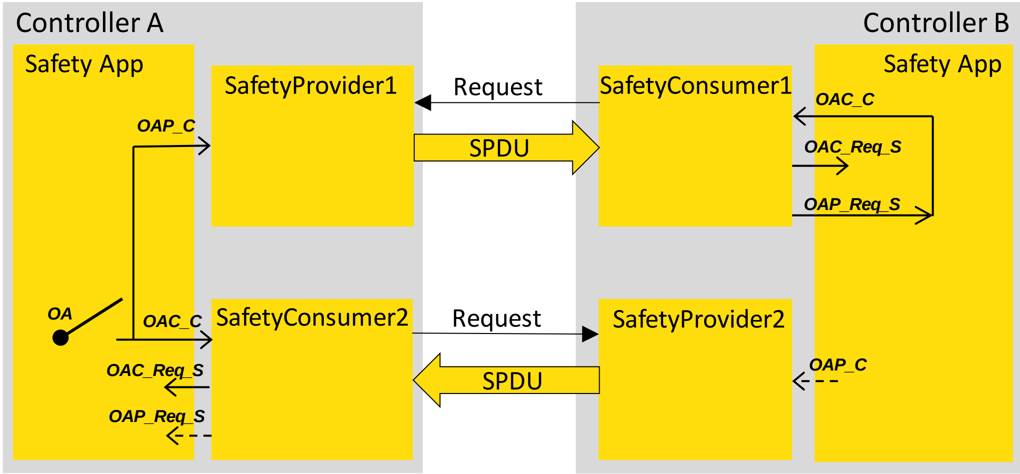


Figure 31 – One sided OA in bidirectional safety communication

In this scenario (see Figure 31), an operator acknowledge activated at controller A suffices for re-establishing the bidirectional connection. Both sides will cease delivering fail-safe values and continue sending process values. This is accomplished by connecting OAP_Req_S with OAC_C at the SafetyConsumer of controller B. Activating operator acknowledge at controller B is not possible in this scenario.

A.2.5 Use case 4: bidirectional comm. and single, two-sided OA

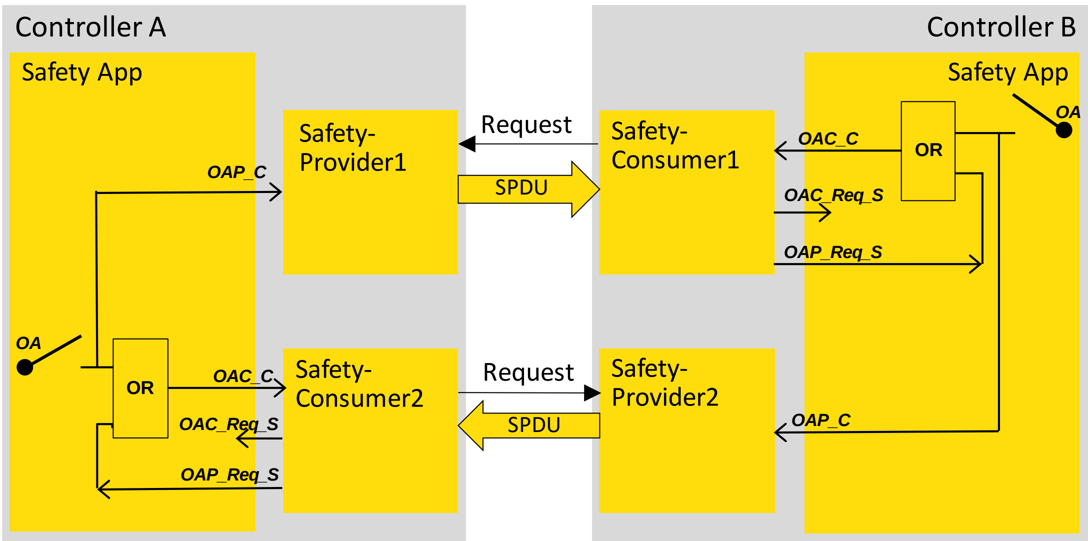


Figure 32 – One sided OA on each side is possible

In this scenario (see Figure 32), an operator acknowledge activated at controller A or controller B suffices for re-establishing the bidirectional connection. Both sides will cease delivering fail-safe values and continue sending process values. This is accomplished by the logic circuit shown in Figure 32.

Annex B (normative): Safety Namespace and mappings

B.1 Namespace and identifiers for Safety Information Model

This appendix defines the numeric identifiers for all of the numeric *NodeIds* defined in this specification. The identifiers are specified in a CSV file with the following syntax:

```
<SymbolName>, <Identifier>, <NodeClass>
```

Where the *SymbolName* is either the *BrowseName* of a *Type Node* or the *BrowsePath* for an *Instance Node* that appears in the specification and the *Identifier* is the numeric value for the *NodeId*.

The *NamespaceUri* for all *NodeIds* defined here is http://opcfoundation.org/UA/Safety

The CSV released with this version of the specification can be found here:
http://www.opcfoundation.org/UA/schemas/Safety/1.0/NodeIds.csv

NOTE The latest CSV that is compatible with this version of the specification can be found here:
http://www.opcfoundation.org/UA/schemas/Safety/NodeIds.csv

A computer processible version of the complete Information Model defined in this specification is also provided. It follows the XML Information Model schema syntax defined in Part 6.
The Information Model Schema released with this version of the specification can be found here:
http://www.opcfoundation.org/UA/schemas/Safety/1.0/Opc.Ua.Safety.NodeSet2.xml

NOTE The latest Information Model schema that is compatible with this version of the specification can be found here:
http://www.opcfoundation.org/UA/schemas/Safety/Opc.Ua.Safety.NodeSet2.xml

B.2 Profile URIs for Safety Information Model

Table 34 defines the Profile URIs for the Safety Information Model companion specification.

Table 34 – Profile URIs

Profile	Profile URI
SafetyProvider	http://opcfoundation.org/UA-Profile/External/Safety/SafetyProvider
SafetyConsumer	http://opcfoundation.org/UA-Profile/External/Safety/SafetyConsumer

Annex C: Bibliography

[1] BRUCE P. DOUGLASS, *Doing Hard Time: Developing Real-Time Systems with UML, Objects, Frameworks, and Patterns*, 2011, Addison-Wesley, ISBN-13: 978-0321774934

[2] SCHILLER F and MATTES T: An Efficient Method to Evaluate CRC-Polynomials for Safety-Critical Industrial Communication, Journal of Applied Computer Science, Vol. 14, No 1, pp. 57-80, Technical University Press, Łódź,Poland, 2006

[3] GUY E. CASTAGNOLI, STEFAN BRÄUER, and MARTIN HERRMANN, Optimization of Cyclic Redundancy-Check Codes with 24 and 32 Parity Bits, June 1993, IEEE Transactions On Communications, Volume 41, Issue 6

[4] A. KUZNETSOV, Francis SWARTS, Hendrik C FERREIRA, et al, On the undetected error probability of linear block codes on channels with memory, Information Theory, IEEE Transactions on, 42(1):303-309, 1996