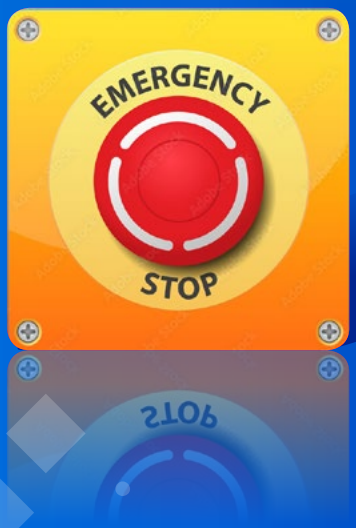
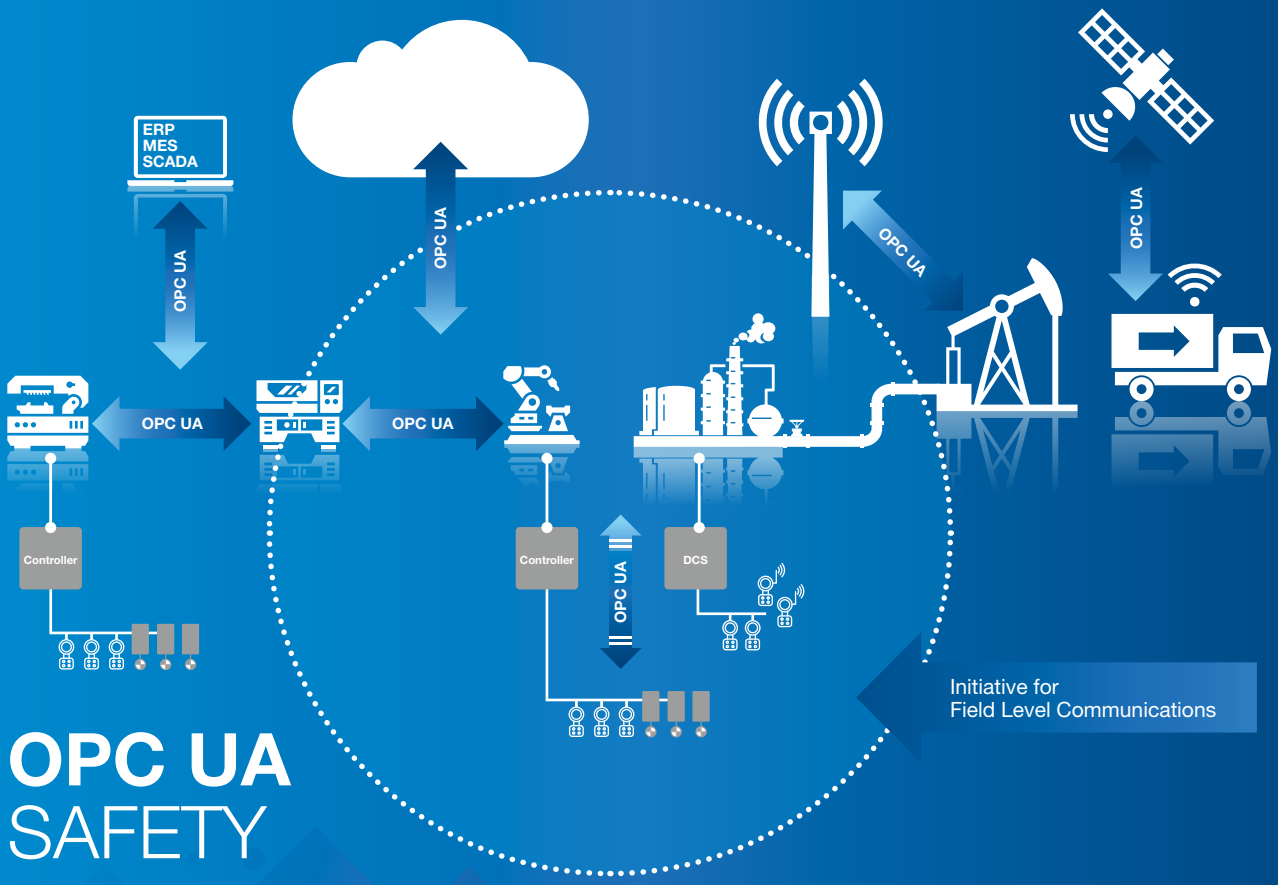
OPC UA Safety:
Functional Safety Communication with OPC UA

Version 1 // September 2022



Technical Paper

Functional Safety Communication with OPC UA



OPC UA
SAFETY

Contents

OPC UA Safety: Functional Safety Communication with OPC UA

Safe Communication for Modular Machines and Dynamic Systems

OPC UA Safety (OPC 10000-15 Unified Architecture Part 15) specifies a functional safety layer for communication between industrial controllers via a standardized, vendor-independent interface. It supports both unicast and multicast at the application layer, as well as arbitrarily structured safety data with a length of up to 1500 bytes. The safety measures comply with all relevant safety standards. OPC UA Safety makes it possible to build modular machines in which the safety functions adapt to the actual configuration of such a machine. It is even possible to realize safety functions requiring a change of communication partners during runtime, for instance in the context of autonomous mobile robots using OPC UA Safety.

Milestones

→ February 2018: Kick-Off Meeting of “PROFIsafe over OPC UA”, a joint working group between OPC Foundation and Profibus & Profinet International

→ April 2019: The FLC (Field-Level Communications) Initiative of the OPC Foundation decides to make of it for the safe exchange of data, and the working group is moved into that initiative

→ July 2019: The specification is renamed “OPC UA Part 15: Safety” and becomes part of the OPC core set of specifications

→ October 2019: Publication of Release 1.04 (Evaluated against IEC 61784-3 by TÜV Süd)

→ October 2019: FLC Initiative funds the development of an OPC UA Safety test tool

→ March 2020: FLC Initiative funds development of a stack and invites other companies to participate

→ July 2020: Publication of Release 1.04

→ November 2021: Publication of Release 1.05

Introduction

The state of the art in functional safety communication is described in IEC 61508 [1], IEC 62280 [2] and IEC 61784-3 [3]. These standards describe the transmission of safety-related messages via a standard (non-safe) communication channel. This can only be done if all transmission errors are detected in the receiver in a safety-related manner with a guaranteed probability of detection. Today, almost all fieldbus protocols offer an associated profile for safe communication. This allows safety-related controllers to communicate via standard communication channels with the sensors and actuators belonging to the safety function. A separate communication channel for functional safety-related data exchange is no longer required. Yet, communication between machines demands a safety protocol that supports communication from controller to controller since each machine or even each machine module is usually represented by a controller. For this type of safety communication, however, no manufacturer-independent, open standard exists to date. This gap is now being closed by the OPC UA Safety specification. Functional, safe communication between machines (or between modules of machines) is of relevance to a variety of different scenarios. Typical examples are transfer lines, electrified monorail systems and machine tools with modular loading and unloading systems. Others include autonomous mobile robots (AMR) that dock on to machines and then perform a common safety function. A common safety function is given, say, where the machine with resident AMR needs to be stopped in response to pressing the emergency stop button on the AMR itself.

OPC UA Safety addresses scenarios such as these. In particular, it offers the following features and properties:

- → Unidirectional and bidirectional communication, as well as multicast, at the application layer
- → Any network topology: star, line, ring, grid, etc.
- → Up to 1500 bytes of safety user data
- → Dynamic connection setup at runtime.

From the aspect of functional safety, OPC UA Safety is based on the widely used PROFIsafe protocol [4]. Because OPC UA Safety is built on OPC UA, it inherits properties such as:

- → Safety-related communication and standard communication on a single transmission channel
- → Arbitrary data rates
- → No safety-related requirements whatsoever on the non-safety-related nodes in the network
- → No safety-related requirements on network components (e.g., switches)
- → No requirements on safe clock synchronization.

Some of the unique benefits of OPC UA Safety are:

- → Support of interoperable multi-vendor Controller-to-Controller (C2C), Controller-to-Device (C2D) and Device-to-Device (D2D) communication
- → Highly flexible – applicable in transportation, process & factory automation, motion control
- → Scalable - from SIL 1 to SIL 4
- → Support of dynamic application scenarios – dynamic reconfiguration of machinery and plants
- → Large data payload size – scalable from simple to complex, high capacity applications
- → Ability to traverse routers – scalable from single machines to plant-wide operation
- → Simple deployment in resource-constrained end-points (amplified by the availability of a commercial stack)

This brochure first provides an overview (chapter 1) and then describes the architecture of OPC UA Safety (chapter 2). The last part (chapter 3) summarizes the basic reasoning for verifying protocol safety.

1. Overview

1.1 Services

The basic principle behind the safe exchange of data with OPC UA Safety is a direct point-to-point connection with the SafetyProvider (data source) and SafetyConsumer (data sink) endpoints. As defined by the part it plays, the SafetyProvider receives user data from the local safety application and makes it available via OPC UA services. The SafetyConsumer uses OPC UA services to retrieve the user data and makes them available to the local safety application. As such, this is a request/response procedure, as shown in Figure 1.

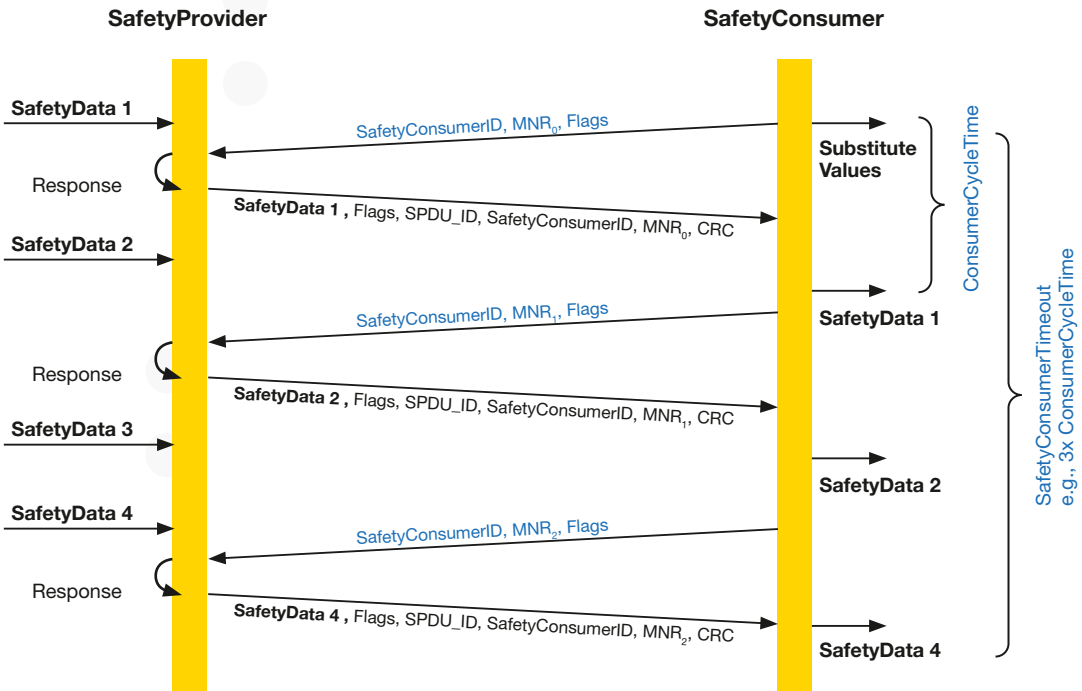


Figure 1: Sequence diagram of an OPC UA Safety connection to illustrate the request/response procedure.

1.2 Safety Measures

To detect transmission errors, relevant additional information is added to the safety-related request and response messages. Essentially, this involves system-wide unique identifiers (IDs) for checking the respective provider, a monitoring number (MNR) for checking the correct sequence, and a CRC signature for detecting data corruption. Sending a timestamp is not necessary since punctuality of the ResponseSPDU can be checked by the Safety-Consumer using its local clock alone. This means there is no need to synchronize the SafetyProvider and SafetyConsumer clocks.

1.3 Unique Identifiers (IDs)

OPC UA Safety uses different IDs to detect addressing errors. Each SafetyProvider is assigned a unique ProviderID, which is made known to the SafetyConsumer to begin with. Together with other information (e.g., signature showing the structure of the user data sent), an SPDU_ID is calculated from the ProviderID, which is included in each ResponseSPDU. Based on the SPDU_ID, the SafetyConsumer can check whether the SPDU received originates from the expected SafetyProvider. In larger-type systems, assigning unique IDs may come with a high level of administrative input. This is particularly true if different parts of a plant are built by different integrators. Using several machines of the same type and the associated cloning of automation projects initially also lead to the co-existence of identical IDs, which would then have to be changed manually to rule out an ID occurring twice. To reduce the input required to manage the IDs, each SafetyProvider additionally contains a BaseID, which is also included in the SPDU_ID and checked by the SafetyConsumer. Not every SafetyProvider needs a unique BaseID; instead, these are assigned jointly for entire plant sections or machines. For example, if two integrators are working on the construction of a plant, each will use a different BaseID. It is sufficient for the ProviderIDs assigned by each integrator to be unique. When cloning projects for volume-production machines, it is sufficient to regenerate a separate BaseID for each machine. Since the ProviderIDs are not changed, it is not necessary to check for the uniqueness of all IDs after cloning. The BaseID is a 128-bit value that can be created using a random number generator. In practice, for example, this is where a Universal Unique Identifier (UUID, [8]) can be generated. It can be proven with sufficient probability that no two identical BaseIDs will occur. Therefore, it is not necessary to make an explicit check as to whether all plant parts have different BaseIDs. In some applications, it may make sense to inform the SafetyProvider of the Safety-Consumer it is currently communicating with. For this purpose, the Safety-Consumer also receives a relevant ConsumerID. However, since it is sufficient to check only the identity of the SafetyProvider to react in a safe manner, the ConsumerID is not safety-relevant.

1.4 Communication Patterns

In practice, bidirectional and point-to-multipoint connections (multicast) occur in safety applications in addition to direct point-to-point connections. These are achieved in OPC UA Safety by using multiple SafetyProvider/SafetyConsumer pairs, as illustrated in Figure 2. Given the SafetyProviders' multiple instantiation, multicast involves a certain overhead over a solution that would use any existing multicast mechanisms in the lower layers. OPC UA Safety, however, comes with a specification that enables the SafetyProvider to be implemented in a way that is highly efficient both in terms of memory and computing capacity.

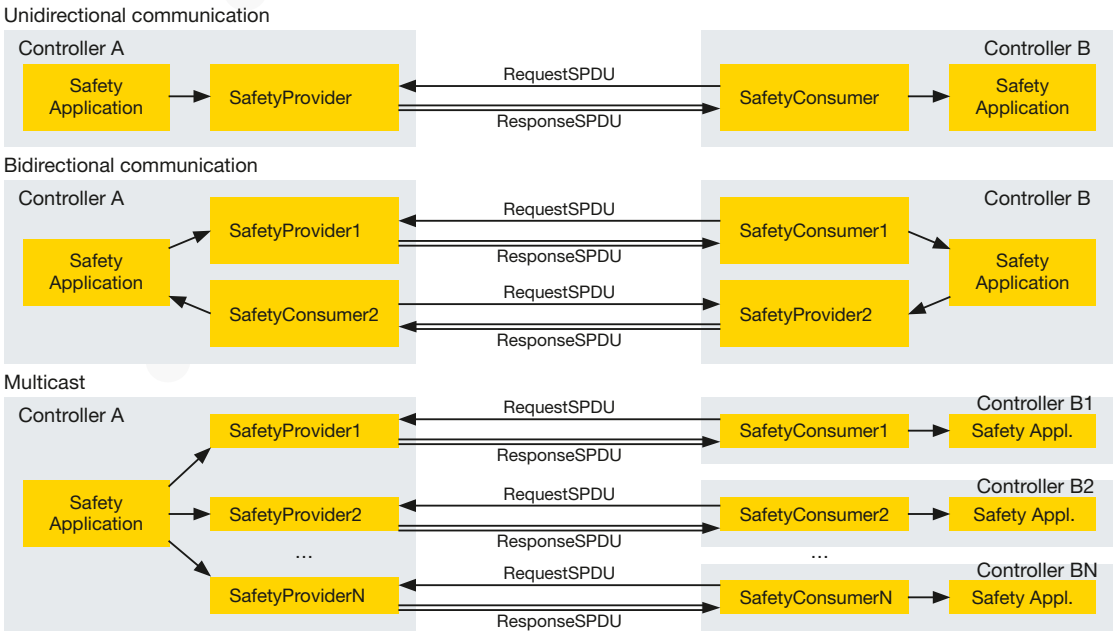


Figure 2: Communication relationships: Unidirectional and bidirectional unicast, as well as multicast.

2. Architecture

2.1 Overview and Interfaces

OPC UA Safety follows the approach recommended in IEC 61784-3 (Functional Safety for Fieldbus, [2]), as illustrated in Figure 3. Accordingly, OPC UA Safety is a layer that is inserted between the safety application and the lower network layer that is implemented as standard (non-safe) communication channel. The OPC UA Safety stack layer has the function of checking the integrity of all safety messages communicated on the standard channel and, on this basis, of detecting communication errors and delivering only correct user data to the safety application. This provides the capability of remaining in control of all communication errors, but not of errors in the end nodes themselves. Consequently, OPC UA Safety (SafetyProvider and SafetyConsumer) must be implemented in compliance with IEC 61508 [1]. In particular, this involves measures to overcome random hardware errors on the one hand, and systematic hardware and software errors on the other.

Making it easier to accommodate different underlying communication services, connection to the OPC UA stack takes place by what is referred to as an OPC UA Mapper. Not being part of the safety-relevant communication layer, this mapper can be adapted without the need for any re-assessment of implementing OPC UA Safety. The OPC UA Mapper currently supports remote method calls (OPC UA Client/Server) as well as OPC UA PubSub. The interface to the safety application (Safety Application Interface, SAPI) is in part application-specific because different applications also exchange different data. As usual in OPC UA, however, this interface can also be defined on a manufacturer-independent basis – e.g., industry-specific – by means of so-called companion specifications. In addition to SAPI and connection to OPC UA via the OPC UA Mapper, SafetyProvider and SafetyConsumer each have a Safety Parameter Interface (SPI) and a diagnostic interface. Exchanged between SafetyProvider and SafetyConsumer, the safety protocol data unit (SPDU), is defined by the structure of the RequestSPDU and ResponseSPDU on the one hand, and by the SafetyProvider and SafetyConsumer state machines on the other.

The definitions of the SafetyProvider and Safety-Consumer are described in more detail on the following pages.

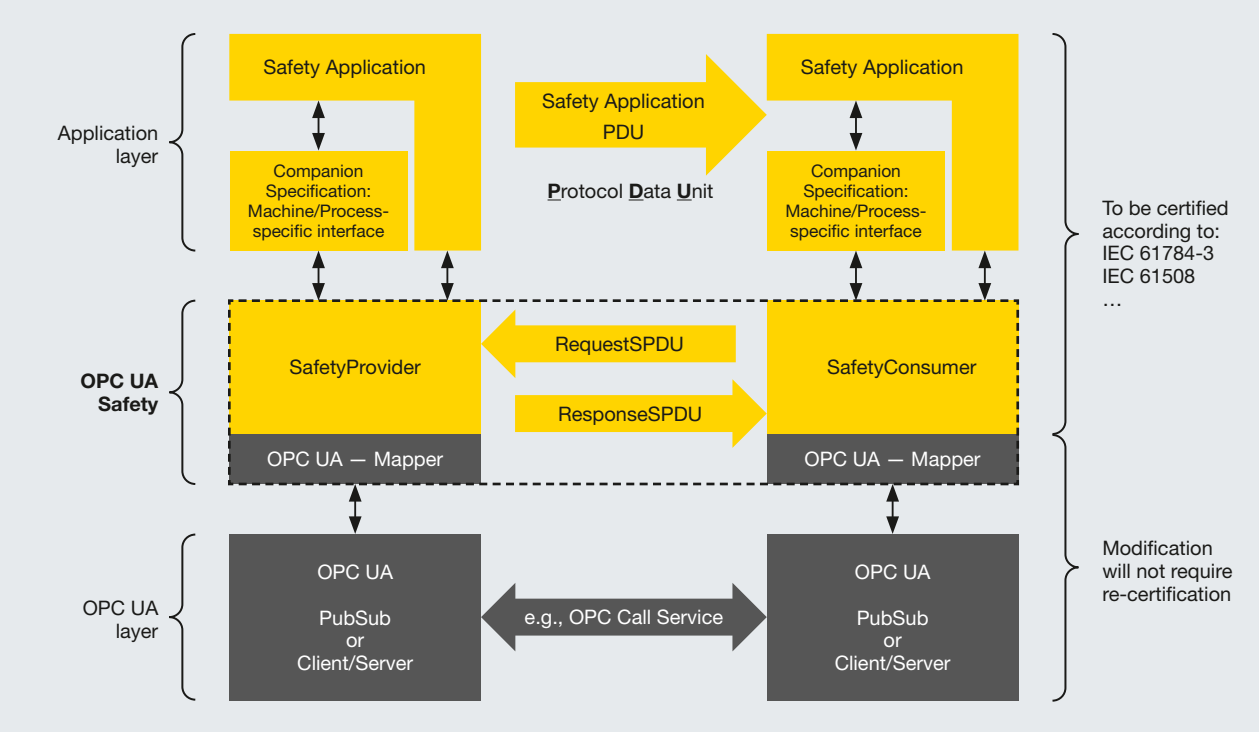


Figure 3: Overview of the OPC UA Safety Architecture

2.2 SafetyProvider

The SafetyProvider receives the safe user data (SafetyData) from the safety application through its Safety Application Programming Interface (SAPI). Moreover, non-safety-related data (NonSafetyData) can also be sent in consistent fashion (see Figure 4). This means that both safe and non-safe data sampled at the same time by the SafetyProvider are also delivered together to the SafetyConsumer. The SafetyProvider's behavior, and indirectly that of the SafetyConsumer, can be influenced by the safety application via control inputs. For example, the ActivateFSV input can be used to make the SafetyConsumer deliver safe substitute values instead of the actual process values to its safety application. The SafetyProvider is parameterized via the Safety Parameter Interface (SPI) at commissioning time. SafetyBaseID and SafetyProviderID together define a globally (sufficiently) unique ID for this instance of the SafetyProvider. The SafetyStructureSignature is a checksum across the structure and type identifier of the safe user data sent. This is also checked by the SafetyConsumer. If, for example, a programming error makes the SafetyConsumer connect a three-dimensional vector (identifier for the type e.g., “vec3D”) with a SafetyProvider that provides orientation in the form of three Euler angles (identifier for the type e.g., “orientation”), the signatures will not match up, and the Safety-Consumer will not deliver this data to its safety application.

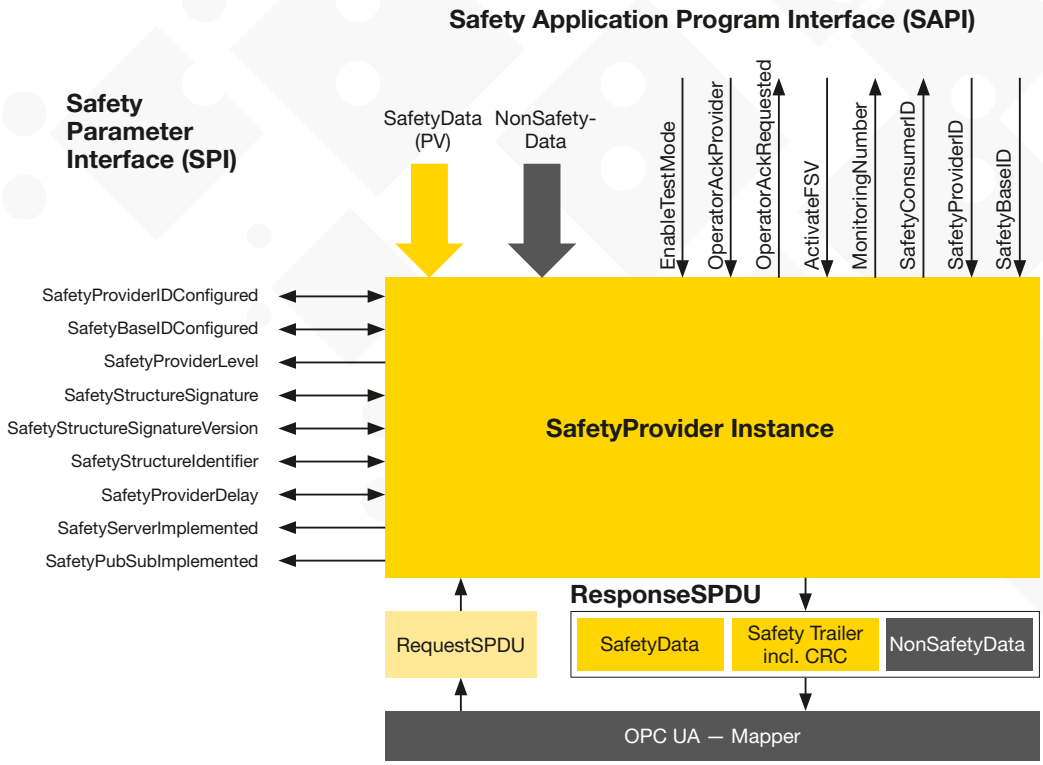


Figure 4: Interfaces of the SafetyProvider: SAPI, SPI and SPDUs.

The SafetyProvider's state machine is trivial and comprises only two states (see Figure 6). In the “WaitForRequest” state, the provider waits for a RequestSPDU. In the “PrepareSPDU” state, a ResponseSPDU is generated with the help of the RequestSPDU, and the data currently present at the SAPI, and is transferred to the OPC UA Mapper. This means implementation of the SafetyProvider is practically without any state. It is not necessary to establish any connection at safety level, and identity of the SafetyConsumer does not have to be made known to the SafetyProvider.

A SafetyProvider can serve several SafetyConsumers one after another. To avoid availability problems, the SafetyConsumers must in this case implement a procedure that prevents simultaneous access to one and the same SafetyProvider. However, this can be done in a non-safe manner since any data corruption caused by concurrent access would be safely detected.

Each instance of the SafetyProvider is represented in the OPC UA information model by an object of type SafetyProviderType (see Figure 5). Additionally, each OPC UA Server implementing OPC UA Safety gets the SafetyACSet node with a fixed, known NodeID, which organizes all SafetyProvider objects. This makes it easy to reach all SafetyProviders while commissioning and during runtime.

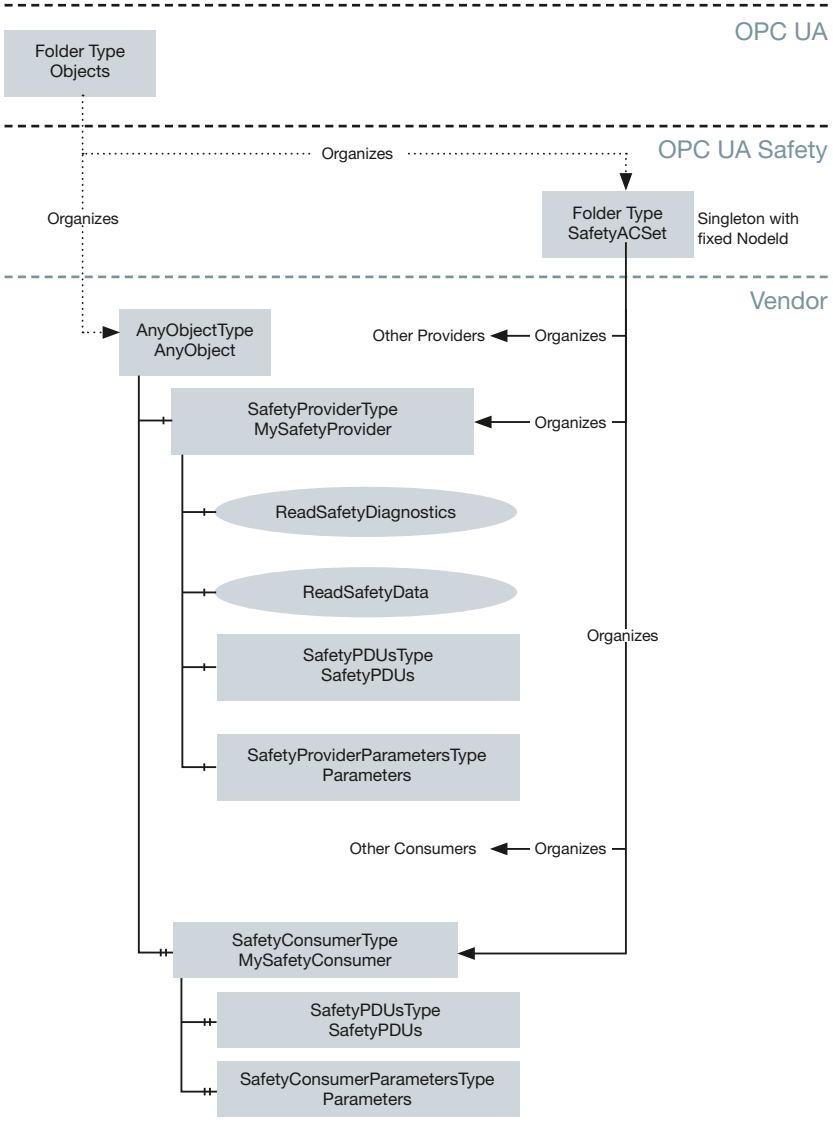


Figure 5: OPC UA information model, consisting of the node SafetyACSet (with fixed NodeID) and one or more SafetyProviders.

2.3 SafetyConsumer

In the SafetyConsumer, the safe user data is taken from the ResponseSPDU, checked for validity and delivered to the safety application together with the non-safe user data via the SAPI (see Figure 7). In addition, the safety application receives information about the validity of this data via this interface (output FSV_Activated). The SafetyConsumer is parameterized via the SPI. In particular, the anticipated SafetyBaseID and SafetyProviderID are now set and, in the same way as the SafetyProvider, a signature saved that reveals the safe user data structure and identifier(s). In the event if an error, for example, an operator acknowledgement (OperatorAckNecessary) is defined which also determines the length of time the SafetyConsumer is required to wait before the SafetyProvider's responds and triggers a timeout (SafetyConsumerTimeout).

The SafetyConsumer's state machine is shown in Figure 8. Operating in a fault-free state, the SafetyConsumer cycles through states S13, S14, S15, S16 and S18. In state S13 the RequestSPDU is sent and in state S14 the ResponseSPDU is being awaited. In state S15 the CRC signature of the ResponseSPDU is checked and in state S16 the origin and timeliness are checked. Failure of any of these checks or the occurrence of any timeout in state S14 induces state S17, whereupon an error message is generated for diagnostic purposes.

A safety-related response may be required when OPC UA Safety detects an error in the non-safety-related communication layers. Depending on circumstances, however, OPC UA Safety is capable of tolerating such an error. One of these being that the error occurs sporadically, i.e., the subsequent RequestSPDU is re-rendered error-free. On top of this, the last error to occur must not have occurred before a SafetyErrorIntervalLimit. This SafetyConsumer parameter determines the minimum interval tolerable between sporadic errors, and is set according to the desired Safety Integrity Level. For SIL2, a value of six minutes or greater must be set. And for SIL3, a value of sixty minutes or greater must be set. In this case, sporadic errors occurring less frequently than every six minutes (for SIL2) do not necessarily lead to any transition to a safe state. Nonetheless, the next, correct RequestSPDU must come in before the SafetyConsumerTimeout expires. A tolerated error makes the request-response cycle

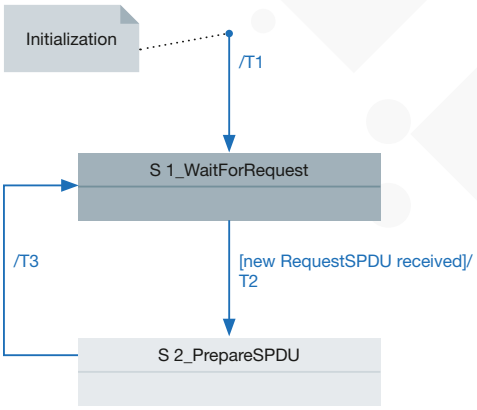


Figure 6: State diagram of the SafetyProvider.

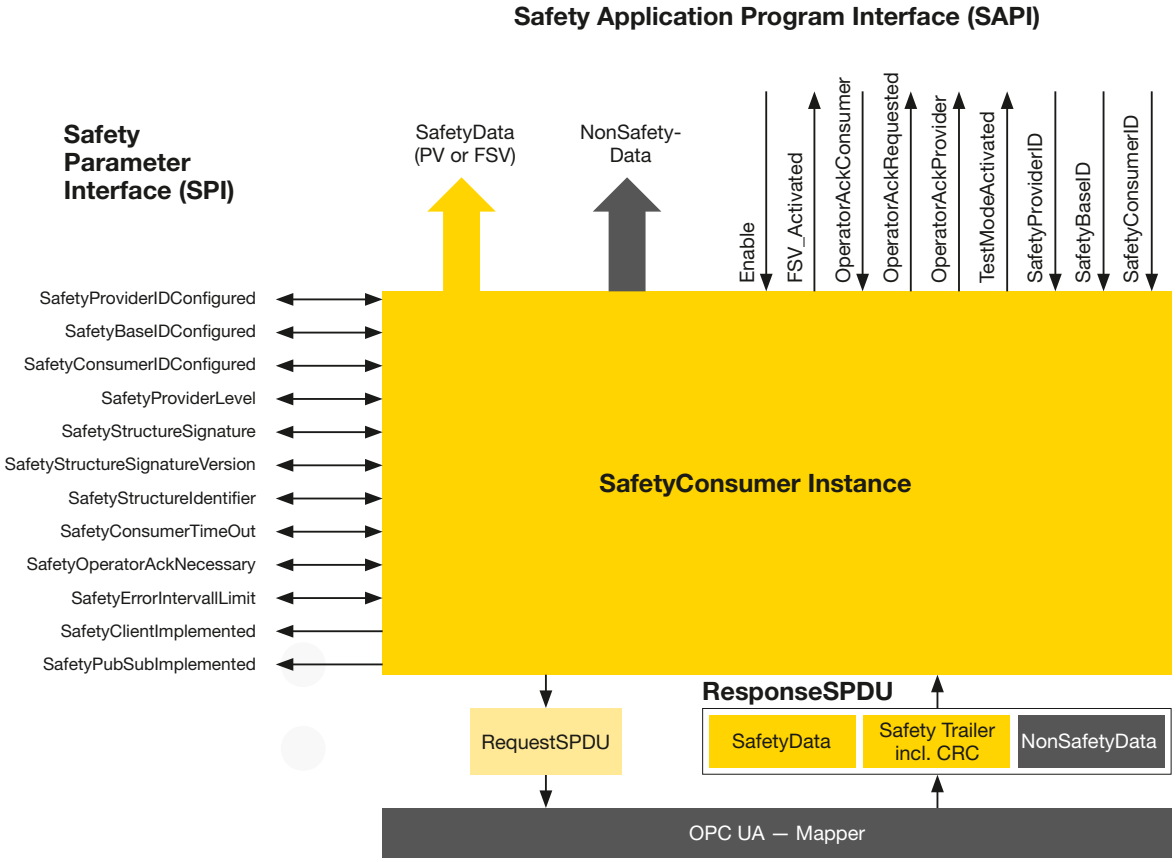


Figure 7: Interfaces of the SafetyConsumer: SAPI, SPI and SPDUs.

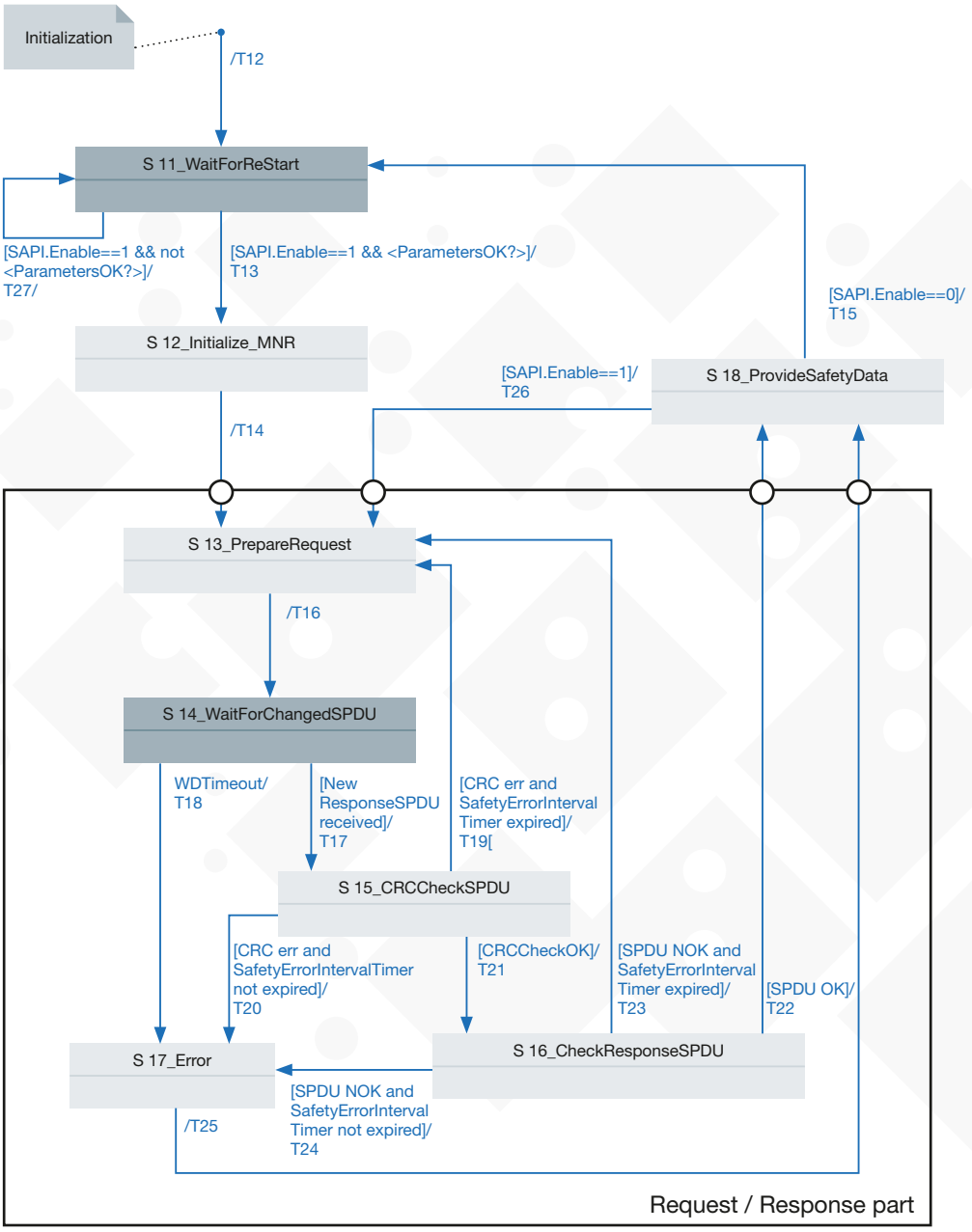


Figure 8: State diagram of the SafetyConsumer.

to run through a second time. It must be noted that new user data cannot be made available to the safety application while this is taking place. A safe response must be given if an error cannot be tolerated (because it is a permanent error, or transient errors occur too frequently). In such cases, the SafetyConsumer uses the FSV_Activated variable to tell the safety application that no valid process values exist. In this case, the user data is set to zero bit by bit.

Depending on the type of error that has occurred, operator acknowledgement may be required to return to operation with real process values. Some applications always demand operator acknowledgement, this can be given via the OperatorAckNecessary parameter (see Figure 9).

Any operator acknowledgement required is indicated via the OperatorAckRequested output as soon as any error has been eliminated. A rising edge at the OperatorAckConsumer input subsequently resumes regular operation, i.e., the output of process values rather than safe substitute values. In the simplest case, the OperatorAckConsumer input is connected in the safety application of the SafetyConsumer, e.g., to a push button or an element in a human-machine interface. Nonetheless, this does not rule out acknowledgement scenarios of a more complex nature. Figure 9, for instance, shows a bidirectional OPC UA Safety connection with one SafetyProvider and SafetyConsumer in either direction. In this example, operator acknowledgement (OA) can take place on both sides. For this purpose, the signal is connected to the OperatorAckConsumer input of the respective SafetyConsumer on the one hand, and to the OperatorAckProvider input of the SafetyProvider on the other. The latter's result sets the relevant SafetyConsumer's Operator-AckProvider output. Connecting this output to the OperatorAckConsumer input acknowledges the SafetyConsumer at the connection's other end.

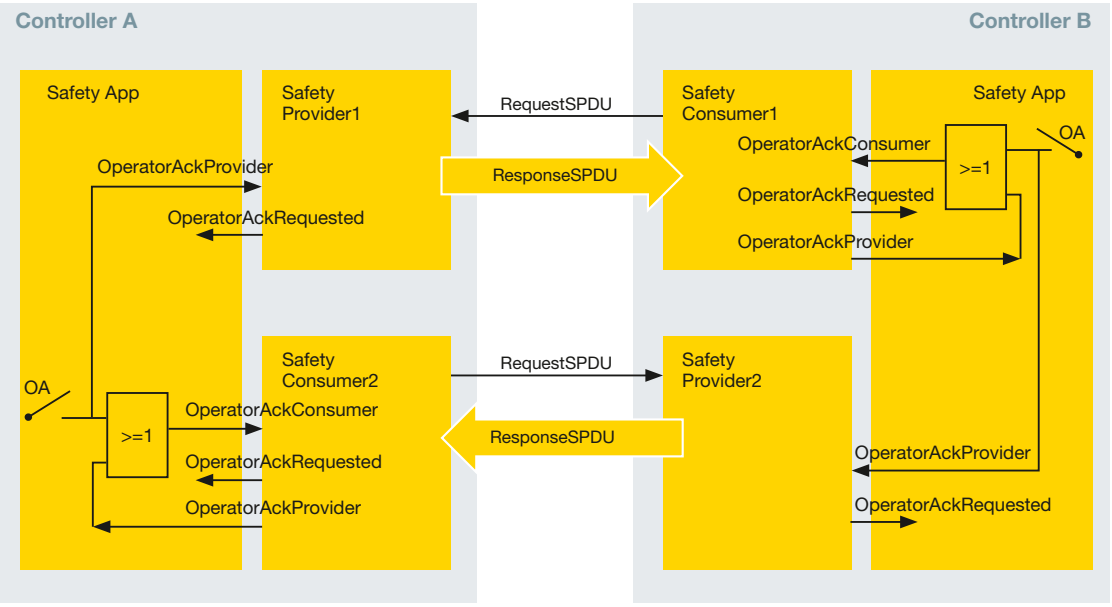


Figure 9: Bidirectional communication where operator acknowledgment is possible on both sides.

2.4 OPC UA Safety for Field-Level Communications

The OPC Foundation is developing the Field eXchange (OPC UA FX or simply UAFX) specifications to extend and enhance the OPC UA capabilities to cover requirements of field-level applications. These are now available for Controller-Controller applications and will soon be available for Controller-Device applications. The OPC UA Safety protocol is transported within standard UAFX connections using the inter-channel principles described in IEC 61784-3 to transmit safety data payload between automation components in addition to the standard data payload exchange. This principle cuts assessment input to the safe transmission functions, i.e., to a level that means the underlying UAFX connections need no additional functional safety assessment. Safety Functional Entities may include non-safe and safe input and output variables. Safety application inside the FE must also be developed in a safe workflow. The safety application is connected directly with SafetyProvider / SafetyConsumer, which exchange

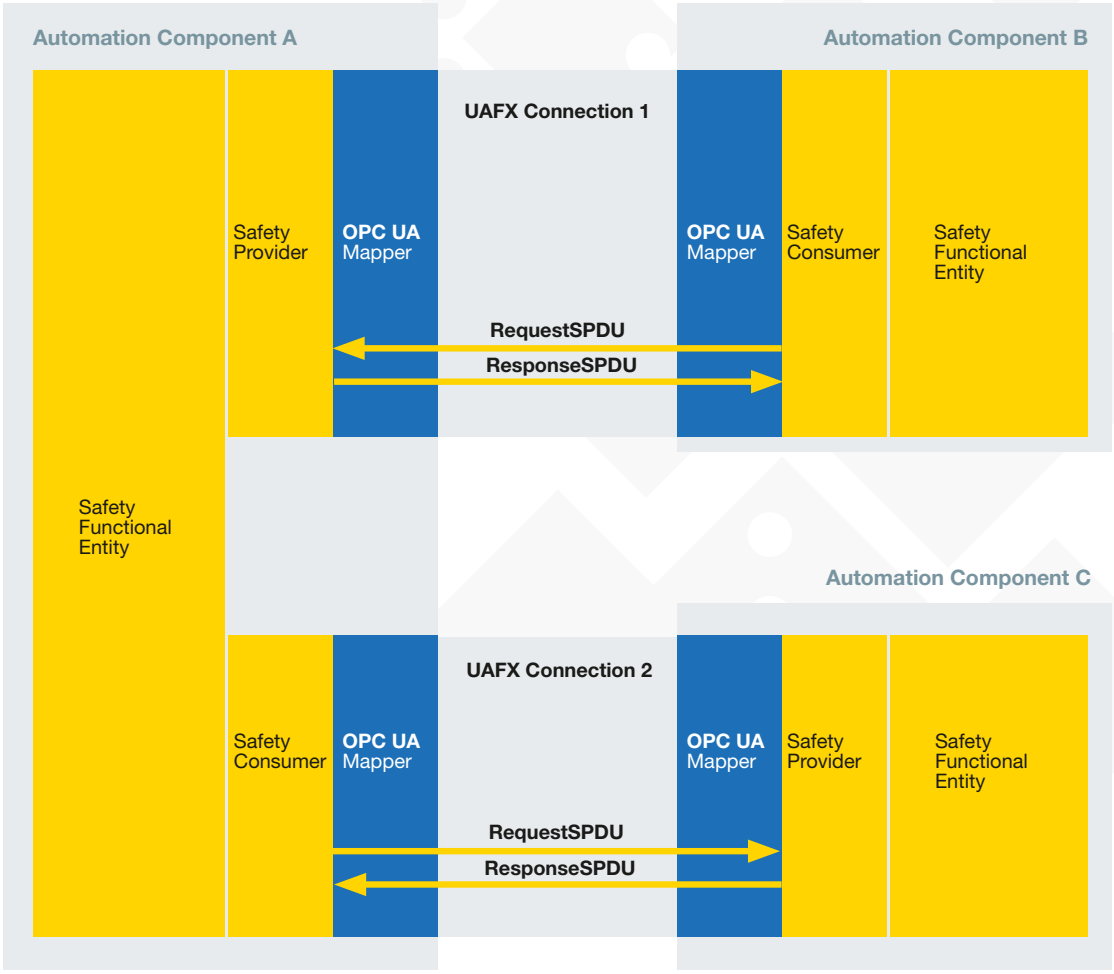


Figure 10: Safety connections between Automation Components

data via the safety protocol (see Figure 10). The OPC UA Mapper is used to interface both the safety layer and underlying communication. and supports the channel between SafetyProvider and SafetyConsumer. The most basic type of safety communication is bidirectional communication, where a safety application on one AC A sends data to a safety application on another AC B. The SafetyConsumer initiates communication with the Request SPDU. The SafetyProvider mirrors the received ID and counters, adds the requested safety data and secures all data via a checksum before responding with the ResponseSPDU (see Figure 11).
One AC can be SafetyConsumer and SafetyProvider at one and the same time. Connection between SafetyProvider and SafetyConsumer can be established and terminated during runtime, allowing different consumers to connect to the same SafetyProvider at different times.

SafetyProvider State Diagram
The SafetyProvider has a very simple state machine to implement. It simply waits for a request and, if the request is received, the safety message is sent out. All safety checks take place on the SafetyConsumer side.

SafetyConsumer State Diagram
SafetyConsumer initiates the safe exchange of data, waits for the response, and checks for potential communication errors (integrity, promptness, authenticity, in line with IEC61784-3). Once done, SafeData is provided to the safety application inside the AC. If a communication error occurs, failsafe substitute values are instead passed to the safety application, resulting in error indication.

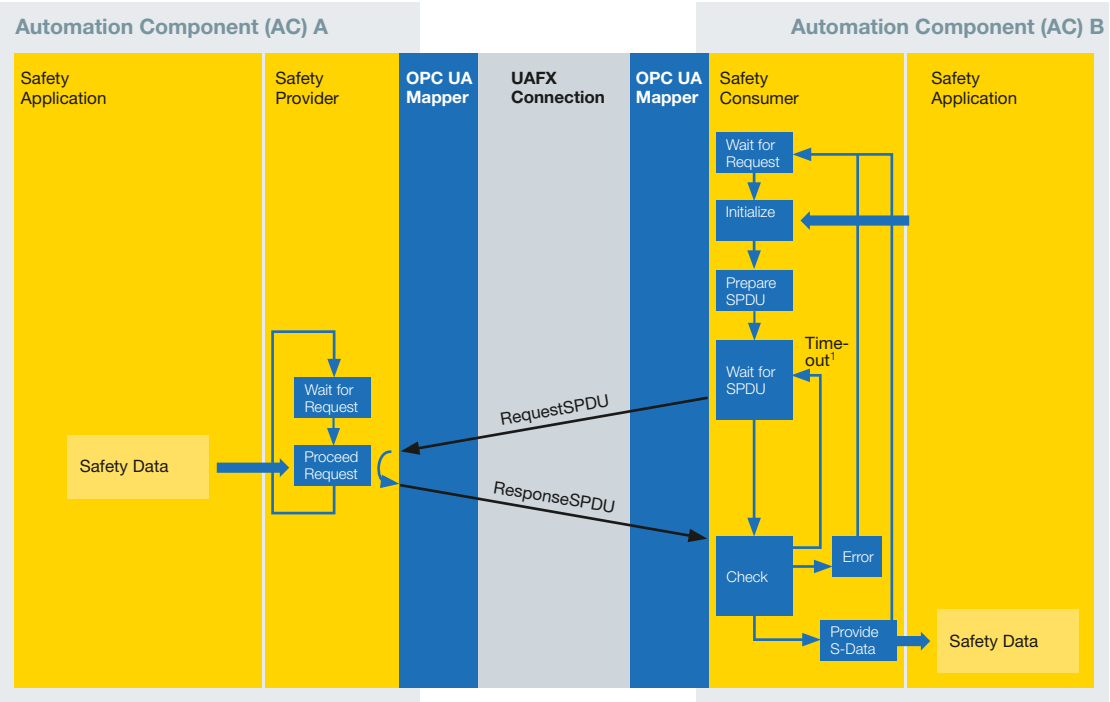


Figure 11: SafetyProvider and Consumer State Machines

[1] To avoid running into safety timeout, SPDUs may also be protected by end-to-end latency guarantee.

3. Safety Argumentation

3.1 Qualitative Reasoning

Following international standard IEC 61784-3 [2], OPC UA Safety must be able to handle all communication errors that can occur in the lower network layers. These are depicted in Table 1, the mechanisms specified in OPC UA Safety being shown in each case. The mechanisms cover errors in the (RequestSPDU) request and in the (ResponseSPDU) response, despite only the response being checked for errors by the SafetyConsumer. This works because the SafetyProvider copies all data contained in the RequestSPDU to the ResponseSPDU, making it possible to detect an error later in the SafetyConsumer.

Communication Error	Safety Measures			
	MonitoringNumber	Timeout with receipt	Set of IDs for SafetyProvider	Data integrity check
Corruption	–	–	–	x
Unintended repetition	x	x	–	–
Incorrect sequence	x	–	–	–
Loss	x	x	–	–
Unacceptable delay	–	x	–	–
Insertion	x	–	–	–
Masquerade	x	–	x	x
Addressing	–	–	x	–

Table 1: Communication errors protection measures taken according to IEC 61784-3.

3.2 Quantitative Evaluation

The 32-bit CRC method used in OPC UA has a conditional residual error probability of 4.0 x 10^{-10} or better. This means that out of 2.5 x 10^{9} faulty RequestSPDUs, on average less than one is not detected as faulty. Figure 12 shows the residual error probability for all SPDU lengths from 1 to 1,521 bytes and all relevant bit error probabilities. Each of the values were calculated using the dual code (e.g., Castagnoli method [6]). The need to iterate over all possible user data lengths is shown in Figure 13, this showing the computed conditional residual error probability for selected user data spanning a more prolonged period. In these cases, the residual error probability exceeds the desired value of (4 x 10^{-10}) by several orders of magnitude. The figure also shows that the residual error probability must be calculated for all bit error probabilities. Calculation of the "worst case" for bit error probability (p = 0.5) does not necessarily lead to the highest conditional residual error probability.

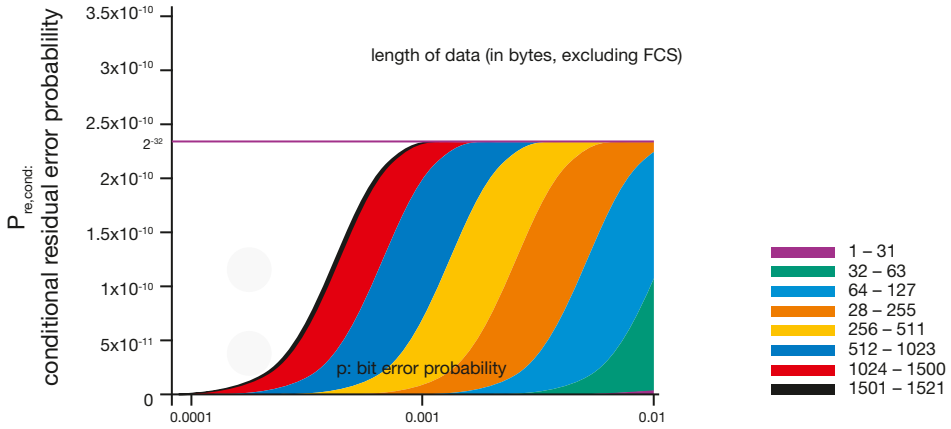


Figure 12: Conditional residual error probability of the CRC procedure used in OPC UA for message lengths of 1-1,521 bytes and all relevant bit error probabilities p.

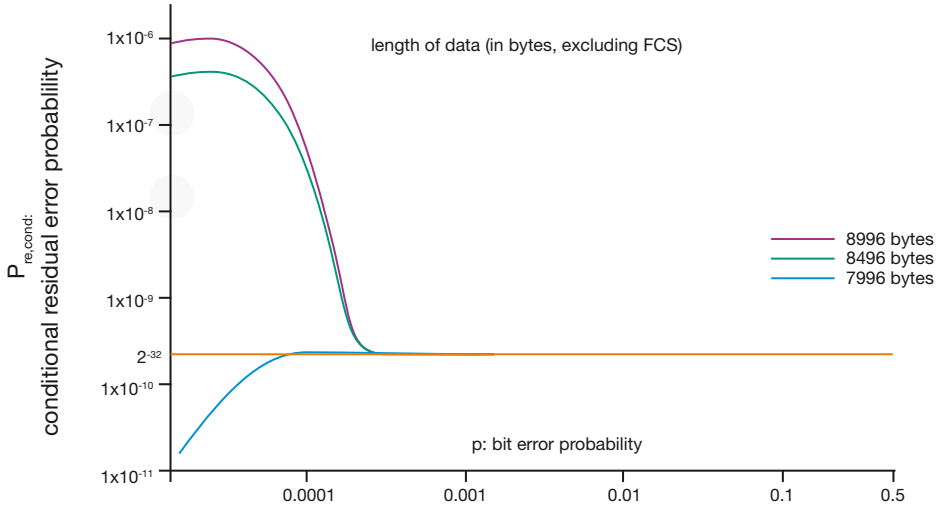


Figure 13: Counter-example: selected message lengths where the conditional residual error probability is much larger than 2^{-32}.

Summary

For the first time, OPC UA Safety specifies a standardized exchange of functional safety-related data between industrial controllers and devices from different manufacturers. Proceeding from OPC UA Client/Server and OPC UA PubSub, which are part of the non-safety related communication layers, it is possible to define a safe communication protocol that meets all IEC 61508 [1] requirements. Unlike safe fieldbus protocols, there is no distinction between "controllers" and "devices". All communication users have equal rights and can implement any number of data sources (SafetyProvider) and data sinks (SafetyConsumer). This permits the creation of complex communication relationships and network topologies.

OPC UA Safety provides the capability of structuring user data to suit any application. Capability of providing user data lengths of between one and 1,500 bytes. IEC 61784-3, OPC UA Safety uses a set of IDs to check at runtime level whether the data originates from the expected source or whether data was supplied from an incorrect source, e.g., in response to an addressing error. Unlike the known safe communication protocols, OPC UA Safety can select these IDs by the safety application in runtime. Basically, this makes it possible to use one and the same safety connection for different communication users. This, in turn, is the prerequisite for challenging scenarios in the field of modular machines and AMRs, which call for dynamic connection setups of this type.

References

[1] IEC 61508. (2010).
Functional safety of electrical/electronic/programmable electronic safety-related systems.
IEC: www.iec.ch

[2] IEC 62280 (2014). Railway applications – Communication, signalling and processing systems – Safety-related communication in transmission systems.
IEC: www.iec.ch

[3] EC 61784-3. (2021). Industrial communication networks – Profiles – Part 3:
Functional safety fieldbuses – General rules and profile definitions.
IEC: www.iec.ch

[4] IEC 61784-3-3. (2021). Industrial communication networks – Profiles – Part 3:
Functional safety fieldbuses – Additional specifications for CPF 3.
IEC: www.iec.ch

[5] IEC 62541. (2016). OPC 10000 Unified Architecture.
IEC: www.iec.ch

[6] OPC 10000-15 OPC Unified Architecture (2021) –
Part 15: Safety, www.opcfoundation.org

[7] Castagnoli, G., Brauer, S., & Herrmann, M. (1993).
Optimization of cyclic redundancy-check codes with 24 and 32 parity bits.
IEEE Transactions on Communications, 41(6), 883-892.

[8] Leach, P. (2005). A Universally Unique Identifier (UUID) URN Namespace,
Request for Comments: 4122, The Internet Society, 2005.

[9] Walter M., Barthel H. (2020). Funktional sichere Kommunikation mit OPC UA Part 15:
Safety: atp!info, Vulkan-Verlag GmbH, 2020. www.vulkan-shop.de

OPC FOUNDATION HEADQUARTERS
OPC Foundation
16101 N. 82nd Street, Suite 3B
Scottsdale, AZ 85260-1868 USA
Phone: 480 483-6644
office@opcfoundation.org

OPC FOUNDATION EUROPE
opceurope@opcfoundation.org

OPC FOUNDATION CHINA
opcchina@opcfoundation.org

OPC FOUNDATION JAPAN
opcjapan@opcfoundation.org

OPC FOUNDATION KOREA
opckorea@opcfoundation.org

OPC FOUNDATION ASEAN
opcasean@opcfoundation.org

OPC FOUNDATION INDIA
opcindia@opcfoundation.org