SLOVENSKI STANDARD
oSIST prEN IEC 62541-13:2024

01-marec-2024

Enotna arhitektura OPC - 13. del: Zborniki

OPC Unified Architecture - Part 13: Aggregates

OPC Unified Architecture - Teil 13: Aggregation von Daten

Architecture unifiée OPC - Partie 13: Agrégats

Ta slovenski standard je istoveten z: **prEN IEC 62541-13:2024**

ICS:

25.040.40	Merjenje in krmiljenje industrijskih postopkov	Industrial process measurement and control
35.240.50	Uporabniške rešitve IT v industriji	IT applications in industry

oSIST prEN IEC 62541-13:2024 **en,fr,de**

65E/1059/CDV

COMMITTEE DRAFT FOR VOTE (CDV)

PROJECT NUMBER: **IEC 62541-13 ED3**	
DATE OF CIRCULATION: **2024-01-26**	CLOSING DATE FOR VOTING: **2024-04-19**
SUPERSEDES DOCUMENTS: **65E/984/RR**	

IEC SC 65E : DEVICES AND INTEGRATION IN ENTERPRISE SYSTEMS	
SECRETARIAT: United States of America	SECRETARY: Mr Donald (Bob) Lattimer
OF INTEREST TO THE FOLLOWING COMMITTEES:	PROPOSED HORIZONTAL STANDARD: ☐ Other TC/SCs are requested to indicate their interest, if any, in this CDV to the secretary.

FUNCTIONS CONCERNED:

☐ EMC ☐ ENVIRONMENT ☐ QUALITY ASSURANCE ☐ SAFETY

☒ SUBMITTED FOR CENELEC PARALLEL VOTING **Attention IEC-CENELEC parallel voting** The attention of IEC National Committees, members of CENELEC, is drawn to the fact that this Committee Draft for Vote (CDV) is submitted for parallel voting. The CENELEC members are invited to vote through the CENELEC online voting system.	☐ NOT SUBMITTED FOR CENELEC PARALLEL VOTING

This document is still under study and subject to change. It should not be used for reference purposes.

Recipients of this document are invited to submit, with their comments, notification of any relevant patent rights of which they are aware and to provide supporting documentation.

Recipients of this document are invited to submit, with their comments, notification of any relevant "In Some Countries" clauses to be included should this proposal proceed. Recipients are reminded that the CDV stage is the final stage for submitting ISC clauses. (SEE AC/22/2007 OR NEW GUIDANCE DOC).

TITLE:

OPC Unified Architecture - Part 13: Aggregates

PROPOSED STABILITY DATE: 2026

NOTE FROM TC/SC OFFICERS:

CONTENTS

FIGURES

TABLES

INTERNATIONAL ELECTROTECHNICAL COMMISSION

____

OPC UNIFIED ARCHITECTURE –

Part 13: Aggregates

FOREWORD

1) The International Electrotechnical Commission (IEC) is a worldwide organization for standardization comprising all national electrotechnical committees (IEC National Committees). The object of IEC is to promote international co-operation on all questions concerning standardization in the electrical and electronic fields. To this end and in addition to other activities, IEC publishes International Standards, Technical Specifications, Technical Reports, Publicly Available Specifications (PAS) and Guides (hereafter referred to as "IEC Publication(s)"). Their preparation is entrusted to technical committees; any IEC National Committee interested in the subject dealt with may participate in this preparatory work. International, governmental and non-governmental organizations liaising with the IEC also participate in this preparation. IEC collaborates closely with the International Organization for Standardization (ISO) in accordance with conditions determined by agreement between the two organizations.

2) The formal decisions or agreements of IEC on technical matters express, as nearly as possible, an international consensus of opinion on the relevant subjects since each technical committee has representation from all interested IEC National Committees.

3) IEC Publications have the form of recommendations for international use and are accepted by IEC National Committees in that sense. While all reasonable efforts are made to ensure that the technical content of IEC Publications is accurate, IEC cannot be held responsible for the way in which they are used or for any misinterpretation by any end user.

4) In order to promote international uniformity, IEC National Committees undertake to apply IEC Publications transparently to the maximum extent possible in their national and regional publications. Any divergence between any IEC Publication and the corresponding national or regional publication shall be clearly indicated in the latter.

5) IEC itself does not provide any attestation of conformity. Independent certification bodies provide conformity assessment services and, in some areas, access to IEC marks of conformity. IEC is not responsible for any services carried out by independent certification bodies.

6) All users should ensure that they have the latest edition of this publication.

7) No liability shall attach to IEC or its directors, employees, servants or agents including individual experts and members of its technical committees and IEC National Committees for any personal injury, property damage or other damage of any nature whatsoever, whether direct or indirect, or for costs (including legal fees) and expenses arising out of the publication, use of, or reliance upon, this IEC Publication or any other IEC Publications.

8) Attention is drawn to the Normative references cited in this publication. Use of the referenced publications is indispensable for the correct application of this publication.

9) Attention is drawn to the possibility that some of the elements of this IEC Publication may be the subject of patent rights. IEC shall not be held responsible for identifying any or all such patent rights.

International Standard IEC 62541-13 has been prepared by subcommittee 65E: Devices and integration in enterprise systems, of IEC technical committee 65: Industrial-process measurement, control and automation.

This fourth edition cancels and replaces the third edition published in 2020. This edition constitutes a technical revision.

This edition includes the following technical changes with respect to the previous edition:

a) Multiple fixes for the computation of aggregates

- The Raw status bit is always be set for non-bad StatusCodes for the Start and End aggregates.
- Entries in the Interpolative examples Tables A2.2 Hisotorian1, Historian2, and Historian3 have been changed from Good to Good, Raw status codes when the timestamp matches with the timestamp of the data source.
- Missing tables were added for DurationInStateZero and DurationInStateNonZero.
- The value of zero has been removed for reslts with a StatusCode of bad.
- Data Type was listed as "Status Code" when it should be "Double" for both Standard Deviation and both Variance Aggregates.
- Rounding Error in TimeAverage and TimeAverage2 have been corrected.

- The status codes have been corrected for the last two intervals and the value has been corrected in the last interval.
- The wording has been changed to be more consistent with the certification testing tool.
- UsedSlopedExtrapolation set to true for Historian2 and all examples locations needed new values or status' are modified.
- Values affected by percent good and percent bad have been updated.
- PercentGood/PercentBad are now accounted for in the calculation.
- TimeAverage must use SlopedInterpolation but the Time aggregate is incorrectly allowed to used Stepped Interpolation.
- Partial bit is now correctly calculated.
- Unclear sentence was removed.
- Examples have been moved to a CSV.
- The value and status code for Historian 3 have been updated.
- TimeAverage2 Historian1 now takes uncertain regions into account when calculating StatusCodes.
- TimeAverage2 Historian2 now takes uncertain regions into account when calculating StatusCodes.
- Total2 Historian1 now takes uncertain regions into account when calculating StatusCodes
- Total2 Historian2 now takes uncertain regions into account when calculating StatusCodes
- Maximum2 Historian1 now takes uncertain regions into account when calculating StatusCodes
- MaximumActualTime2 Historian1 now takes uncertain regions into account when calculating StatusCodes
- Minimum2 Historian1 now takes uncertain regions into account when calculating StatusCodes
- MinimumActualTime2 Historian1 now has the StatusCodes calculated while using the TreatUncertainAsBad flag.
- Range2 Historian1 now looks at TreatUncertainAsBad in the calculation of the StatusCodes.
- Clarifications were made to the text defining how PercentGood/PercentBad are used. The TimeAverage2 and Total2 aggregets had their table values and StatusCodes corrected.

The text of this International Standard is based on the following documents:

CDV	Report on voting
65E/XX/CDV	65E/XX/RVC

Full information on the voting for the approval of this International Standard can be found in the report on voting indicated in the above table.

This document has been drafted in accordance with the ISO/IEC Directives, Part 2.

Throughout this document and the other parts of the IEC 62541 series, certain document conventions are used:

Italics are used to denote a defined term or definition that appears in the "Terms and definition" clause in one of the parts of the IEC 62541 series.

Italics are also used to denote the name of a service input or output parameter or the name of a structure or element of a structure that are usually defined in tables.

The *italicized terms and names* are, with a few exceptions, written in camel-case (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). For example, the defined term is

AddressSpace instead of Address Space. This makes it easier to understand that there is a single definition for *AddressSpace*, not separate definitions for Address and Space.

A list of all parts of the IEC 62541 series, published under the general title *OPC Unified Architecture*, can be found on the IEC website.

The committee has decided that the contents of this document will remain unchanged until the stability date indicated on the IEC website under "http://webstore.iec.ch" in the data related to the specific document. At this date, the document will be

- reconfirmed,
- withdrawn,
- replaced by a revised edition, or
- amended.

IMPORTANT – The 'colour inside' logo on the cover page of this publication indicates that it contains colours which are considered to be useful for the correct understanding of its contents. Users should therefore print this document using a colour printer.

OPC Unified Architecture Specification

Part 13: Aggregates

1 Scope

This part of IEC 62541 is part of the overall OPC Unified Architecture specification series and defines the information model associated with Aggregates.

2 Normative references

The following documents, in whole or in part, are normatively referenced in this document and are indispensable for its application. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments and errata) applies.

IEC 62541-1*, OPC Unified Architecture - Part 1: Overview and Concepts*

IEC 62541-3, *OPC Unified Architecture - Part 3: Address Space Model*

IEC 62541-4, *OPC Unified Architecture - Part 4: Services*

IEC 62541-5, *OPC Unified Architecture - Part 5: Information Model*

IEC 62541-8, *OPC Unified Architecture - Part 8: Data Access*

IEC 62541-11, *OPC Unified Architecture - Part 11: Historical Access*

3 Terms, definitions, and abbreviated terms

3.1 Terms and definitions

For the purposes of this document, the terms and definitions given in IEC 62541-1, IEC 62541-3, IEC 62541-4, and IEC 62541-11 as well as the following apply.

3.1.1
ProcessingInterval
timespan for which derived values are produced based on a specified *Aggregate*

Note 1 to entry: The total time domain specified for ReadProcessed is divided by the *ProcessingInterval*. For example, performing a 10-minute Average over the time range 12:00 to 12:30 would result in a set of three intervals of *ProcessingInterval* length, with each interval having a start time of 12:00, 12:10 and 12:20 respectively. The rules used to determine the interval *Bounds* are discussed in 5.4.2.2.

3.1.2
Interpolated data
data that is calculated from data samples

Note 1 to entry: Data samples may be historical data or buffered real time data. An *interpolated* value is calculated from the data points on either side of the requested timestamp.

3.1.3
EffectiveEndTime
time immediately before *endTime*

Note 1 to entry: All *Aggregate* calculations include the *startTime* but exclude the *endTime*. However, it is sometimes necessary to return an *Interpolated* End Bound as the value for an *Interval* with a timestamp that is in the *interval*. *Servers* are expected to use the time immediately before *endTime* where the time resolution of the *Server* determines the exact value (do not confuse this with hardware or operating system time resolution). For example, if the *endTime* is 12:01:00, the time resolution is 1 second, then the *EffectiveEndTime* is 12:00:59. See 5.4.2.4.

If time is flowing backwards, *Servers* are expected to use the time immediately after *endTime* where the time resolution of the *Server* determines the exact value.

3.1.4
Extrapolated data

data constructed from a discrete data set but is outside of the discrete data set

Note 1 to entry: It is similar to the process of interpolation, which constructs new points between known points, but its result is subject to greater uncertainty. *Extrapolated* data is used in cases where the requested time period falls farther into the future than the data available in the underlying system. See example in Table 1.

3.1.5
SlopedInterpolation

simple linear interpolation

Note 1 to entry: Compare to curve fitting using linear polynomials. See example in Table 1.

3.1.6
SteppedInterpolation

Interpolation holding the last data point constant or interpolating the value based on a horizontal line fit

Note 1 to entry: Consider the following Table 1 of raw and *Interpolated*/*Extrapolated* values:

Table 1 – Interpolation examples

Timestamp	Raw Value	Sloped Interpolation	Stepped Interpolation
12:00:00	10		
12:00:05		15	10
12:00:08		18	10
12:00:10	20		
12:00:15		25	20
12:00:20	30		
		SlopedExtrapolation	SteppedExtrapolation
12:00:25		35	30
12:00:27		37	30

3.1.7
bounding values

values at the *startTime* and *endTime* needed for *Aggregates* to compute the result

Note 1 to entry: If *Raw data* does not exist at the *startTime* and *endTime* a value shall be estimated. There are two ways to determine *Bounding Values* for an interval. One way (called *Interpolated Bounding Values*) uses the first non-Bad data points found before and after the timestamp to estimate the bound. The other (called *Simple Bounding Values*) uses the data points immediately before and after the boundary timestamps to estimate the bound even if these points are Bad. Subclauses 3.1.8 and 3.1.9 describe the two different approaches in more detail.

In all cases the *TreatUncertainAsBad* (see 4.2.1.2) flag is used to determine whether Uncertain values are Bad or non-Bad.

If a Raw value was not found and a non-Bad bounding value exists the *Aggregate* Bits (see 5.3.3) are set to 'Interpolated'.

When calculating *bounding values*, the value portion of *Raw data* that has Bad status is set to null. This means the value portion is not used in any calculation and a null is returned if the raw value is returned. The status portion is determined by the rules specified by the bound or *Aggregate*.

The *Interpolated Bounding Values* approach (see 3.1.8) is the same as what is used in Classic OPC Historical Data Access (HDA) and is important for applications such as advanced process control where having useful values at all times is important. The *Simple Bounding Values* approach (see 3.1.9) is new in this standard and is important for applications which shall produce regulatory reports and cannot use estimated values in place of Bad data.

3.1.8
interpolated bounding values

bounding values determined by a calculation using the nearest Good value

Note 1 to entry: *Interpolated Bounding Values* using *SlopedInterpolation* are calculated as follows:

- if a non-Bad Raw value exists at the timestamp then it is the bounding value;
- find the first non-Bad Raw value before the timestamp;
- find the first non-Bad Raw value after the timestamp;
- draw a line between before value and after value;
- use point where the line crosses the timestamp as an estimate of the bounding value.

The calculation can be expressed with the following formula:

$$V_{\mathrm{bound}} = (T_{\mathrm{bound}} - T_{\mathrm{before}})\mathrm{x}(\ V_{\mathrm{after}} - V_{\mathrm{before}})/(\ T_{\mathrm{after}} - T_{\mathrm{before}}) + V_{\mathrm{before}}$$

where V_{x} is a value at ‘x’ and T_{x} is the timestamp associated with V_{x}.

If no non-Bad values exist before the timestamp the *StatusCode* is Bad_NoData. The *StatusCode* is *Uncertain_DataSubNormal* if any Bad values exist between the before value and after value. If either the before value or the after value are Uncertain the *StatusCode* is *Uncertain_DataSubNormal*. If the after value does not exist the before value shall be extrapolated using *SlopedExtrapolation* or *SteppedExtrapolation*.

The period of time that is searched to discover the Good values before and after the timestamp is *Server* dependent, but if a Good value is not found within some reasonable time range then the *Server* will assume it does not exist. The *Server* as a minimum should search a time range which is at least the size of the ProcessingInterval.

Interpolated Bounding Values using *SlopedExtrapolation* are calculated as follows:

- find the first non-Bad Raw value before timestamp;
- find the second non-Bad Raw value before timestamp;
- draw a line between these two values;
- extend the line to where it crosses the timestamp;
- use the point where the line crosses the timestamp as an estimate of the bounding value.

The formula is the same as the one used for *SlopedInterpolation*.

The *StatusCode* is always *Uncertain_DataSubNormal*. If only one non-Bad raw value can be found before the timestamp then *SteppedExtrapolation* is used to estimate the bounding value.

Interpolated Bounding Values using *SteppedInterpolation* are calculated as follows:

- if a non-Bad Raw value exists at the timestamp then it is the bounding value;
- find the first non-Bad Raw value before timestamp;
- use the value as an estimate of the bounding value.

The *StatusCode* is *Uncertain_DataSubNormal* if any Bad values exist between the before value and the timestamp. If no non-Bad *Raw data* exists before the timestamp then the *StatusCode* is Bad_NoData. If the value before the timestamp is Uncertain the *StatusCode* is *Uncertain_DataSubNormal*. The value after the timestamp is not needed when using *SteppedInterpolation*; however, if the timestamp is after the end of the data then the bounding value is treated as extrapolated and the *StatusCode* is *Uncertain_DataSubNormal*.

SteppedExtrapolation is a term that describes *SteppedInterpolation* when a timestamp is after the last value in the history collection.

3.1.9
simple bounding values

bounding values determined by a calculation using the nearest value

Note 1 to entry: *Simple Bounding Values* using *SlopedInterpolation* are calculated as follows:

- if any Raw value exists at the timestamp then it is the bounding value;
- find the first Raw value before timestamp;
- find the first Raw value after timestamp;
- if the value after the timestamp is Bad then the before value is the bounding value;
- draw a line between before value and after value;
- use point where the line crosses the timestamp as an estimate of the bounding value.

The formula is the same as the one used for *SlopedInterpolation* in Clause 3.1.5.

If a Raw value at the timestamp is Bad the *StatusCode* is Bad_NoData. If the value before the timestamp is Bad the *StatusCode* is Bad_NoData. If the value before the timestamp is Uncertain the *StatusCode* is *Uncertain_DataSubNormal*. If the value after the timestamp is Bad or Uncertain the *StatusCode* is *Uncertain_DataSubNormal*.

Simple Bounding Values using *SteppedInterpolation* are calculated as follows:

- if any Raw value exists at the timestamp then it is the bounding value;
- find the first Raw value before timestamp;
- if the value before timestamp is non-Bad then it is the bounding value.

If a Raw value at the timestamp is Bad the *StatusCode* is Bad_NoData. If the value before the timestamp is Bad the *StatusCode* is Bad_NoData. If the value before the timestamp is Uncertain the *StatusCode* is *Uncertain_DataSubNormal*.

If either bounding time of an interval is beyond the last data point then the *Server* may use extrapolation or return an error. If extrapolation is used by the server the type [*SteppedExtrapolation* or *SloppedExtrapolation*] of extrapolation is server specific.

In some Historians, the last Raw value does not necessarily indicate the end of the data. Based on the Historian's knowledge of the data collection mechanism, i.e. frequency of data updates and latency, the Historian may extend the last value to a time known by the Historian to be covered. When calculating *Simple Bounding Values* the Historian will act as if there is another Raw value at this timestamp.

In the same way, if the earliest time of an interval starts before the first data point in history and the latest time is after the first data point in history, then the interval will be treated as if the interval extends from the first data point in history to the latest time of the interval and the *StatusCode* of the interval will have the Partial bit set (see 5.3.3.2).

The period of time that is searched to discover the values before and after the timestamp is *Server* dependent, but if a value is not found within some reasonable time range then the *Server* will assume it does not exist. The *Server* as a minimum should search a time range which is at least the size of the ProcessingInterval.

3.2 Abbreviated terms

DA	Data Access
HA	Historical Access (access to historical data or events)
HDA	Historical Data Access
UA	Unified Architecture

4 Aggregate information model

4.1 General

IEC 62541-3 and IEC 62541-5 standards define the representation of *Aggregate* historical or buffered real time data in the OPC Unified Architecture. This includes the definition of *Aggregates* used in processed data retrieval and in historical retrieval. This definition includes both standard *Reference* types and *Object* types.

4.2 Aggregate Objects

4.2.1 General

4.2.1.1 Overview

OPC UA *Servers* can support several different functionalities and capabilities. The following standard *Objects* are used to expose these capabilities in a common fashion, and there are several standard defined concepts that can be extended by vendors.

4.2.1.2 AggregateConfigurationType

The *AggregateConfigurationType* defines the general characteristics of a *Node* that defines the *Aggregate* configuration of any *Variable* or *Property*. *AggregateConfiguration Object* represents the browse entry point for information on how the *Server* treats *Aggregate* specific functionality such as handling Uncertain data. It is formally defined in Table 2.

Table 2 – AggregateConfigurationType Definition

Attribute	**Value**				
BrowseName	AggregateConfigurationType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
Subtype of the *BaseObjectType* defined in IEC 62541-5					
HasProperty	Variable	TreatUncertainAsBad	Boolean	PropertyType	Mandatory
HasProperty	Variable	PercentDataBad	Byte	PropertyType	Mandatory
HasProperty	Variable	PercentDataGood	Byte	PropertyType	Mandatory
HasProperty	Variable	UseSlopedExtrapolation	Boolean	PropertyType	Mandatory
Conformance Units					
Aggregate Master Configuration					

The *TreatUncertainAsBad Variable* indicates how the *Server* treats data returned with a *StatusCode* severity Uncertain with respect to *Aggregate* calculations. A value of True indicates the *Server* considers the severity equivalent to *Bad*, a value of False indicates the *Server* considers the severity equivalent to *Good,* unless the *Aggregate* definition says otherwise. The default value is True. Note that the value is still treated as Uncertain when the *StatusCode* for the result is calculated.

The *PercentDataBad Variable* indicates the minimum percentage of Bad data in a given interval required for the *StatusCode* for the given interval for processed data request to be set to *Bad*. (Uncertain is treated as defined above.) Refer to 5.4.3 for details on using this *Variable* when assigning *StatusCodes*. For details on which *Aggregates* use the *PercentDataBad Variable*, see the definition of each *Aggregate*. The default value is 100.

The *PercentDataGood Variable* indicates the minimum percentage of Good data in a given interval required for the *StatusCode* for the given interval for the processed data requests to be set to *Good*. Refer to 5.4.3 for details on using this *Variable* when assigning *StatusCodes*. For details on which *Aggregates* use the *PercentDataGood Variable*, see the definition of each *Aggregate*. The default value is 100.

The following calculations are used to detemine the *StatusCode* which will be used to calculate the value of the aggregate. Refer to 5.4.3 for details on using these *Variables* when assigning *StatusCodes*. The *PercentDataGood* and *PercentDataBad* shall follow the following relationship *PercentDataGood* ≥ (100 – *PercentDataBad*). If they are equal the result of the *PercentDataGood* calculation is used. If the values entered for *PercentDataGood* and *PercentDataBad* do not result in a valid calculation (e.g. Bad = 80; Good = 0) the result will have a StatusCode of Bad_AggregateInvalidInputs The StatusCode Bad_AggregateInvalidInputs will be returned if the value of *PercentDataGood* or *PercentDataBad* exceed 100.

The *UseSlopedExtrapolation Variable* indicates how the *Server* interpolates data when no boundary value exists (i.e. extrapolating into the future from the last known value). A value of False indicates that the *Server* will use a *SteppedExtrapolation* format, and hold the last known value constant. A value of True indicates the *Server* will project the value using *UseSlopedExtrapolation* mode. The default value is False. For *SimpleBounds* this value is ignored.

4.2.2 AggregateFunction Object

4.2.2.1 General

This *Object* is used as the browse entry point for information about the *Aggregates* supported by a *Server*. The content of this *Object* is already defined by its type definition. All *Instances* of the *FolderType* use the standard *BrowseName* of ‘AggregateFunctions’. The *HasComponent Reference* is used to relate a *ServerCapabilities Object* and/or any *HistoryServerCapabilitiesType Object* to an *AggregateFunction Object*. AggregateFunctions is formally defined in Table 3.

Table 3 – Aggregate Functions Definition

Attribute	**Value**				
BrowseName	AggregateFunctions				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
HasTypeDefinition	Object Type	*FolderType*	Defined in IEC 62541-5		
Conformance Units					
Historical Access Aggregates					

Each *ServerCapabilities* and *HistoryServerCapabilitiesType Object* shall reference an *AggregateFunction Object*. In addition, each *HistoricalConfiguration Object* belonging to a *HistoricalDataNode* may reference an *AggregateFunction Object* using the *HasComponent Reference*.

4.2.2.2 AggregateFunctionType

This *ObjectType* defines an *Aggregate* supported by a UA *Server*. This *Object* is formally defined in Table 4.