



DPC: 24 / 30484432 DC

BSI Group Headquarters

389 Chiswick High Road London W4 4AL

Tel: +44 (0)20 8996 9000

Fax: +44 (0)20 8996 7400

www.bsigroup.com

Date: 29 January 2024

Origin: International

Latest date for receipt of comments: 20 March 2024

Project No. 2023/02975

Responsible committee: GEL/65/5 Devices and integration in enterprise systems

Interested committees:

Title: BS EN IEC 62541-11 OPC Unified Architecture

Part 11: Historical access

Please notify the secretary if you are aware of any keywords that might assist in classifying or identifying the standard or if the content of this standard

- i) has any issues related to 3rd party IPR, patent or copyright
- ii) affects other national standard(s)
- iii) requires additional national guidance or information

**WARNING: THIS IS A DRAFT AND MUST NOT BE REGARDED OR USED AS A BRITISH STANDARD.
THIS DRAFT IS NOT CURRENT BEYOND 20 March 2024**

This draft is issued to allow comments from interested parties; all comments will be given consideration prior to publication. No acknowledgement will normally be sent. **See overleaf for information on the submission of comments.**

No copying is allowed, in any form, without prior written permission from BSI except as permitted under the Copyright, Designs and Patent Act 1988 or for circulation within a nominating organization for briefing purposes. Electronic circulation is limited to dissemination by e-mail within such an organization by committee members.

Further copies of this draft may be purchased from BSI Shop <http://shop.bsigroup.com> or from BSI Customer Services, Tel: +44(0) 20 8996 9001 or email cservices@bsigroup.com. British, International and foreign standards are also available from BSI Customer Services.

Information on the co-operating organizations represented on the committees referenced above may be obtained from <http://standardsdevelopment.bsigroup.com>

Responsible Committee Secretary: Ms Charmaine Mukherjee (BSI)

Direct tel:

E-mail: charmaine.mukherjee@bsigroup.com

Introduction

This draft standard is based on national and international discussions. Your comments on this draft are invited and will assist in the preparation of the consequent standard.

For international standards, comments will be reviewed by the relevant UK national committee before sending the consensus UK vote and comments to the international committee, which will then decide appropriate action. If the international standard is approved, it is usual for the text to be published as a British Standard.

For national standards, comments will be reviewed by the relevant UK national committee and the resulting standards published as a British Standard.

UK Vote

Please indicate whether you consider the UK should submit a negative (with supporting technical reasons) or positive vote on this draft. Please indicate if you are aware of any reason why this draft standard should not be published as a British Standard.

Submission of Comments

- **Annotated drafts are not acceptable and will be rejected.**
- All comments should be submitted online at <https://standardsdevelopment.bsigroup.com/>. You will need to register in order to comment.

Template for comments and secretariat observations

Date: xx/xx/20xx	Document: ISO/DIS xxxx
------------------	------------------------

1	2	(3)	4	5	(6)	7
MB	Clause No./ Subclause No./Annex (e.g. 3.1)	Paragraph/Figure/ Table/Note	Type of comment	Commend (justification for change) by the MB	Proposed change by the MB	Secretariat observations on each comment submitted
	3.1	Definition 1	ed	Definition is ambiguous and needs clarifying.	Amend to read '...so that the mains connector to which no connection...'	
	6.4	Paragraph 2	te	The use of the UV photometer as an alternative cannot be supported as serious problems have been encountered in its use in the UK.	Delete reference to UV photometer.	



COMMITTEE DRAFT FOR VOTE (CDV)

PROJECT NUMBER: IEC 62541-11 ED3	
DATE OF CIRCULATION: 2024-01-26	CLOSING DATE FOR VOTING: 2024-04-19
SUPERSEDES DOCUMENTS: 65E/983/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 11: Historical Access

PROPOSED STABILITY DATE: 2026

NOTE FROM TC/SC OFFICERS:

Copyright © 2024 International Electrotechnical Commission, IEC. All rights reserved. It is permitted to download this electronic file, to make a copy and to print out the content for the sole purpose of preparing National Committee positions. You may not copy or "mirror" the file or printed version of the document, or any part of it, for any other purpose without permission in writing from IEC.

CONTENTS

1			
2			
3			
4	FOREWORD	vi	
5	1 Scope	1	
6	2 Normative references	1	
7	3 Terms, definitions, and abbreviations	1	
8	3.1 Terms and definitions	1	
9	3.2 Abbreviated terms	3	
10	4 Concepts	3	
11	4.1 General	3	
12	4.2 Data architecture	3	
13	4.3 Historians and interruption of data collection	4	
14	4.4 Modification of Historical Data/Events	5	
15	4.5 Timestamps	5	
16	4.6 Bounding Values and time domain	6	
17	4.7 Changes in AddressSpace over time	7	
18	5 Historical Information Model	8	
19	5.1 HistoricalNodes	8	
20	5.1.1 General	8	
21	5.1.2 Annotations Property	8	
22	5.2 HistoricalDataNodes	9	
23	5.2.1 General	9	
24	5.2.2 HistoricalDataConfigurationType	9	
25	5.2.3 Attributes	11	
26	5.2.4 Historical Data Configuration Object	11	
27	5.3 References	11	
28	5.3.1 Overview	11	
29	5.3.2 HasHistoricalConfiguration ReferenceType	12	
30	5.3.3 HasCurrentData ReferenceType	12	
31	5.3.4 HasCurrentEvent ReferenceType	13	
32	5.4 HistoricalEventNodes	13	
33	5.4.1 General	13	
34	5.4.2 HistoricalEventFilter Property	14	
35	5.4.3 HistoricalEventConfigurationType	14	
36	5.4.4 HistoricalEventNode Attributes	15	
37	5.5 External History sources	15	
38	5.5.1 Introduction	15	
39	5.5.2 External Historical Event Node	15	
40	5.6 Example Object Models in Historian Servers (informative)	16	
41	5.6.1 Overview	16	
42	5.6.2 HistoricalDataNodes Address Space Model	16	
43	5.6.3 Historical data	17	
44	5.6.4 HistoricalEventNodes Address Space Model	17	
45	5.6.5 Historical Events	18	
46	5.7 Exposing supported functions and capabilities	19	
47	5.7.1 General	19	
48	5.7.2 HistoryServerCapabilitiesType	20	
49	5.7.3 Default Configuration	22	

50	5.8	Historical Audit Events	23
51	5.8.1	General	23
52	5.8.2	AuditHistoryEventUpdateEventType	23
53	5.8.3	AuditHistoryValueUpdateEventType	24
54	5.8.4	AuditHistoryAnnotationUpdateEventType	25
55	5.8.5	AuditHistoryDeleteEventType	25
56	5.8.6	AuditHistoryRawModifyDeleteEventType	26
57	5.8.7	AuditHistoryAtTimeDeleteEventType	27
58	5.8.8	AuditHistoryEventDeleteEventType	27
59	5.8.9	AuditHistoryConfigurationChangeEvent	28
60	5.8.10	AuditHistoryBulkInsertEventType	28
61	6	Historical Access specific usage of Services	29
62	6.1	General	29
63	6.2	Historical Nodes StatusCodes	29
64	6.2.1	Overview	29
65	6.2.2	Operation level result codes	29
66	6.2.3	Semantics changed	31
67	6.3	Continuation Points	31
68	6.4	Arrays, index ranges and substrings	32
69	6.5	HistoryReadDetails parameters	33
70	6.5.1	Overview	33
71	6.5.2	ReadEventDetails structure	34
72	6.5.3	ReadRawModifiedDetails structure	37
73	6.5.4	ReadProcessedDetails structure	39
74	6.5.5	ReadAtTimeDetails structure	41
75	6.5.6	ReadAnnotationDataDetails structure	42
76	6.6	HistoryData parameters returned	42
77	6.6.1	Overview	42
78	6.6.2	HistoryData type	42
79	6.6.3	HistoryModifiedData type	43
80	6.6.4	HistoryEvent DataType	43
81	6.6.5	HistoryModifiedEvent DataType	44
82	6.6.6	Annotation DataType	44
83	6.7	HistoryUpdateType Enumeration	45
84	6.8	PerformUpdateType Enumeration	45
85	6.9	HistoryUpdateDetails parameter	46
86	6.9.1	Overview	46
87	6.9.2	UpdateDataDetails structure	47
88	6.9.3	UpdateStructureDataDetails structure	49
89	6.9.4	UpdateEventDetails structure	50
90	6.9.5	DeleteRawModifiedDetails structure	52
91	6.9.6	DeleteAtTimeDetails structure	53
92	6.9.7	DeleteEventDetails structure	53
93	Annex A (informative)	Client conventions	55
94	A.1	How clients may request timestamps	55
95	A.2	Determining the first or last historical data point	56
96	Bibliography		57
97			

FIGURES

98		
99		
100	Figure 1 – Possible OPC UA Server supporting Historical Access	4
101	Figure 2 – ReferenceType hierarchy	12
102	Figure 3 – Historical Variable with Historical Data Configuration and Annotations	16
103	Figure 5 – Representation of an Event with History in the AddressSpace	18
104	Figure 7 – Server and HistoryServer Capabilities	20

TABLES

106		
107		
108	Table 1 – Bounding Value examples	6
109	Table 2 – Annotations Property	8
110	Table 3 – HistoricalDataConfigurationType definition	9
111	Table 4 – ExceptionDeviationFormat Items	10
112	Table 5 – ExceptionDeviationFormat definition	11
113	Table 6 – Historical Access configuration definition	11
114	Table 7 – HasHistoricalConfiguration ReferenceType	12
115	Table 8 – HasCurrentData ReferenceType	13
116	Table 9 – HasCurrentEvent ReferenceType	13
117	Table 10 – Standard Historical Events Properties	14
118	Table 11 – HistoricalEventConfigurationType definition	14
119	Table 12 – HistoricalExternalEventSourceType definition	15
120	Table 13 – HistoryServerCapabilitiesType Definition	21
121	Table 16 – AuditHistoryEventUpdateEventType definition	24
122	Table 17 – AuditHistoryValueUpdateEventType definition	24
123	Table 18 – AuditHistoryAnnotationUpdateEventType definition	25
124	Table 19 – AuditHistoryDeleteEventType definition	26
125	Table 20 – AuditHistoryRawModifyDeleteEventType definition	26
126	Table 21 – AuditHistoryAtTimeDeleteEventType definition	27
127	Table 22 – AuditHistoryEventDeleteEventType definition	27
128	Table 23 – AuditHistoryConfigurationChangeEvent definition	28
129	Table 24 – AuditHistoryBulkInsertEventType definition	28
130	Table 25 – Bad operation level result codes	30
131	Table 26 – Good operation level result codes	30
132	Table 27 – HistoryReadDetails parameter Symbolic Names	33
133	Table 28 – HistoryReadDetails definition	34
134	Table 29 – ReadEventDetails Structure	34
135	Table 30 – ReadEventDetails definition	35
136	Table 31 – ReadEventDetails2 Structure	36
137	Table 32 – ReadEventDetails2 definition	36
138	Table 33 – ReadRawModifiedDetails structure	37
139	Table 34 – ReadRawModifiedDetails definition	37
140	Table 35 – ReadProcessedDetails structure	39

141	Table 36 – ReadProcessedDetails definition	40
142	Table 37 – ReadAtTimeDetails structure	41
143	Table 38 – ReadAtTimeDetails definition	41
144	Table 39 – ReadAnnotationDataDetails Structure	42
145	Table 40 – ReadAnnotationDataDetails definition	42
146	Table 41 – HistoryData structure	42
147	Table 42 – HistoryData definition	43
148	Table 43 – HistoryModifiedData structure	43
149	Table 44 – HistoryModifiedData definition	43
150	Table 45 – HistoryEvent structure	43
151	Table 46 – HistoryEvent definition	44
152	Table 47 – HistoryModifiedEvent structure	44
153	Table 48 – HistoryModifiedEvent definition	44
154	Table 49 – Annotation Structure	44
155	Table 50 – <i>Annotation</i> definition	45
156	Table 51 – HistoryUpdateType Items	45
157	Table 52 – HistoryUpdateType definition	45
158	Table 53 – PerformUpdateType Items	45
159	Table 54 – PerformUpdateType definition	46
160	Table 55 – HistoryUpdateDetails parameter Symbolic Names	46
161	Table 56 – HistoryUpdateDetails Structure	47
162	Table 57 – HistoryUpdateDetails definition	47
163	Table 58 – UpdateDataDetails Structure	48
164	Table 59 – UpdateDataDetails definition	48
165	Table 60 – UpdateStructureDataDetails Structure	49
166	Table 61 – UpdateStructureDataDetails definition	49
167	Table 62 – UpdateEventDetails Structure	50
168	Table 63 – UpdateEventDetails definition	51
169	Table 64 – DeleteRawModifiedDetails Structure	52
170	Table 65 – DeleteRawModifiedDetails definition	52
171	Table 66 – DeleteAtTimeDetails Structure	53
172	Table 67 – DeleteAtTimeDetails definition	53
173	Table 68 – DeleteEventDetails Structure	53
174	Table 69 – DeleteEventDetails definition	54
175	Table A.1 – Time keyword definitions	56
176	Table A.2 – Time offset definitions	56
177		
178		

INTERNATIONAL ELECTROTECHNICAL COMMISSION

OPC UNIFIED ARCHITECTURE –

Part 11: Historical Access

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 62541-11 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 significant technical changes with respect to the previous edition:

- a) Added functionality to support retrieving of modified events.
- b) Added Event to indicate when a backfill occurred.
- c) Defined a new ReferenceType that can be used to indicate an external node.
- d) Improved text to better explain the concept of annotation and remove conflicting explanations.
- e) Defined a default historian configuration (and where to find it).
- f) Added HistoricalEventConfigurationType, which provides general configuration information about the historical Event storage.
- g) Updated text and added optional fields to HA configuration object to allow configuration to be defined for periodic data collection, not just for exception-based collection.

- h) Provided ObjectType that can be used for external event collection and also provided example how histoians can be configured.

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

Part 11: Historical Access

1 Scope

This document is part of the OPC Unified Architecture standard series and defines the *Information Model* associated with Historical Access (HA). It particularly includes additional and complementary descriptions of the *NodeClasses* and *Attributes* needed for Historical Access, additional standard *Properties*, and other information and behaviour.

The complete *AddressSpace* Model including all *NodeClasses* and *Attributes* is specified in IEC 62541-3. The predefined *Information Model* is defined in IEC 62541-5. The *Services* to detect and access historical data and events, and description of the *ExtensibleParameter* types are specified in IEC 62541-4.

This standard includes functionality to compute and return *Aggregates* like minimum, maximum, average etc. The *Information Model* and the concrete working of *Aggregates* are defined in IEC 62541-13.

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 62541-7, OPC Unified Architecture - Part 7: Profiles

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

IEC 62541-13, OPC Unified Architecture - Part 13: Aggregates

3 Terms, definitions, and abbreviations

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, IEC 62541-13 and the following apply.

3.1.1

Annotation

metadata associated with an item at a given instance in time

Note 1 to entry: An *Annotation* is metadata that is associated with an item at a given instance in time.

3.1.2

BoundingValues

values associated with the starting and ending time

Note 1 to entry: *BoundingValues* are the values that are associated with the starting and ending time of a *ProcessingInterval* specified when reading from the historian. *BoundingValues* may be required by *Clients* to determine the starting and ending values when requesting *RawData* over a time range. If a *RawData* value exists at the start or end point, it is considered the bounding value even though it is part of the data request. If no *RawData*

value exists at the start or end point, then the *Server* will determine the boundary value, which may require data from a data point outside of the requested range. See 4.6 for details on using *BoundingValues*.

3.1.3

Historian

application storing time series data and/or time series events

3.1.4

HistoricalNode

Object, Variable, Property or View in the *AddressSpace* where a *Client* can access historical data or *Events*

Note 1 to entry: A *HistoricalNode* is a term used in this document to represent any *Object, Variable, Property or View* in the *AddressSpace* for which a *Client* may read and/or update historical data or *Events*. The terms "*HistoricalNodes's history*" or "history of a *HistoricalNodes*" will refer to the time series data or *Events* stored for this *HistoricalNode*. The term *HistoricalNode* refers to both *HistoricalDataNodes* and *HistoricalEventNodes*.

3.1.5

HistoricalDataNode

Variable or Property in the *AddressSpace* where a *Client* can access historical data

Note 1 to entry: A *HistoricalDataNode* represents any *Variable or Property* in the *AddressSpace* for which a *Client* may read and/or update historical data. "*HistoricalDataNode history*" or "history of a *HistoricalDataNode*" refers to the time series data stored for this *HistoricalNode*. Examples of such data are:

- device data (like temperature sensors)
- calculated data
- status information (open/closed, moving)
- dynamically changing system data (like stock quotes)
- diagnostic data

The term *HistoricalDataNodes* is used when referencing aspects of the standard that apply to accessing historical data only.

3.1.6

HistoricalEventNode

Object or View in the *AddressSpace* for which a *Client* can access historical *Events*

Note 1 to entry: "*HistoricalEventNode's history*" or "history of a *HistoricalEventNode*" refers to the time series *Events* stored in some historical system. Examples of such data are:

- *Notifications*
- system *Alarms*
- operator action *Events*
- system triggers (such as new orders to be processed)

The term *HistoricalEventNode* is used when referencing aspects of the standard that apply to accessing historical *Events* only.

3.1.7

ModifiedValues

HistoricalDataNode's value that has been changed (or manually inserted or deleted) after it was stored in the historian

Note 1 to entry: For some *Servers*, a lab data entry value is not a *modified value*, but if a user corrects a lab value, the original value would be considered a *modified value*, and would be returned during a request for *ModifiedValues*. Also manually inserting a value that was missed by a standard collection system may be considered a *modified value*. Unless specified otherwise, all historical *Services* operate on the current, or most recent, value for the specified *HistoricalDataNode* at the specified timestamp. Requests for *ModifiedValues* are used to access values that have been superseded, deleted or inserted. It is up to a system to determine what is considered a *modified value*. Whenever a *Server* has modified data available for an entry in the historical collection it shall set the *ExtraData* bit in the *StatusCode*.

3.1.8

RawData

data that is stored within the historian for a *HistoricalDataNode*

Note 1 to entry: The data may be all data collected for the *DataValue* or it may be some subset of the data depending on the historian and the storage rules invoked when the item's values were saved.

3.1.9

StartTime/EndTime

bounds of a history request which define the time domain

Note 1 to entry: For all requests, a value falling at the end time of the time domain is not included in the domain, so that requests made for successive, contiguous time domains will include every value in the historical collection exactly once.

3.1.10

TimeDomain

interval of time covered by a particular request, or response

Note 1 to entry: In general, if the start time is earlier than or the same as the end time, the time domain is considered to begin at the start time and end just before the end time; if the end time is earlier than the start time, the time domain still begins at the start time and ends just before the end time, with time "running backward" for the particular request and response. In both cases, any value which falls exactly at the end time of the *TimeDomain* is not included in the *TimeDomain*. See the examples in 4.6. *BoundingValues* effect the time domain as described in 4.6.

All timestamps which can legally be represented in a *UtcTime DataType* are valid timestamps, and the *Server* may not return an invalid argument result code due to the timestamp being outside of the range for which the *Server* has data. See IEC 62541-3 for a description of the range and granularity of this *DataType*. *Servers* are expected to handle out-of-bounds timestamps gracefully, and return the proper *StatusCodes* to the *Client*.

3.1.11

StructuredHistoryData

structured data stored in a history collection where parts of the structure are used to uniquely identify the data within the data collection

Note 1 to entry: Most historical data applications assume only one current value per timestamp. Therefore, the timestamp of the data is considered the unique identifier for that value. Some data or meta data such as *Annotations* may permit multiple values to exist at a single timestamp. In such cases the *Server* would use one or more parameters of the *StructuredHistoryData* entry to uniquely identify each element within the history collection. *Annotations* are examples of *StructuredHistoryData*.

3.2 Abbreviated terms

DA Data Access
HA Historical Access
HDA Historical Data Access
UA Unified Architecture

4 Concepts

4.1 General

This document defines the handling of historical time series data and historical *Event* data in the OPC Unified Architecture (in a Historian). Included is the specification of the representation of historical data and *Events* in the *AddressSpace*.

4.2 Data architecture

A *Server* supporting Historical Access provides *Clients* with transparent access to different historical data and/or historical *Event* sources (e.g., process *Historians*, event *Historians*, etc.).

The historical data or *Events* may be in a proprietary data collection, database or a short-term buffer within memory. A *Server* supporting Historical Access will provide historical data and *Events* for all or a subset of the available *Variables*, *Objects*, *Properties* or *Views* within the *Server AddressSpace*.

Figure 1 illustrates how the *AddressSpace* of a UA *Server* might consist of a broad range of different historical data and/or historical *Event* sources.

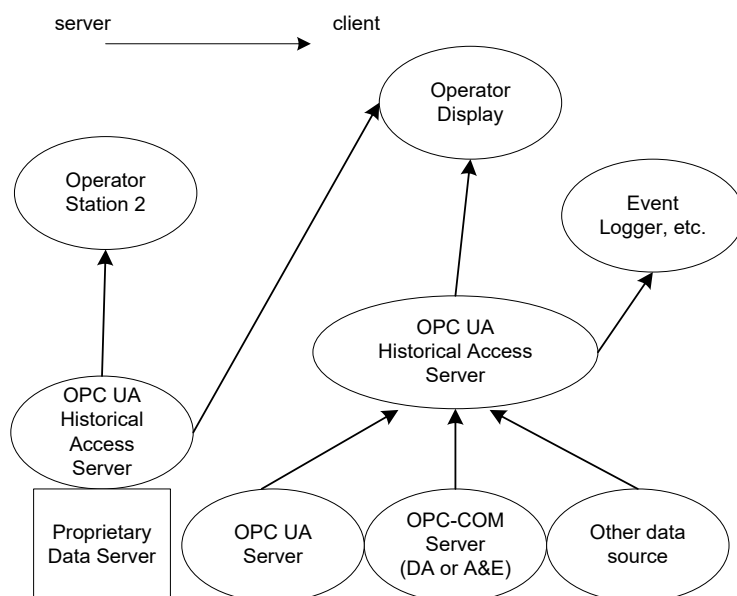


Figure 1 – Possible OPC UA Server supporting Historical Access

The *Historian* may be implemented as a standalone OPC UA Server that collects data from another OPC UA Server or another data source. The *Historian* may also just aggregate historical data from underlying *Historians*. The *Client* that references the OPC UA Server supporting Historical Access for historical data may be simple trending packages that desire values over a given time frame or they may be complex reports that require data in multiple formats.

There are general requirements for *Historians*, but *Historians* can vary in functionality. A consistent requirement for all *Historians* is that they store Historical data including a timestamp. All historical data should include status information for each value, but a *Historian* may compress this to only storing status information that indicates a problem (Bad status) and/or status change, instead of storing a status for every time series data item. The status of historical data can be complex. What is required is that the values returned as part of the timeseries raw data match the data that would have been observed if the *Value* was subscribed to at that point in time.

Historical *Events* are more complicated. In a stream of *Events* each Event can have a different list of fields. *EventTypes* are defined in a hierarchical manner, where each *EventType* inherits fields from its parent type and can add additional fields. Some of these additional fields may be mandatory and are required to understand or process the given *EventType*. A *Historian* that stores *Events*, shall be configurable to store all mandatory fields for any *EventTypes* that it historizes. If it receives for storage an *EventType* that it does not support all mandatory fields for, it may store it as one of its supertype *EventTypes* (one that it does support all mandatory fields for), but then it shall not claim that it supports historizing of that *EventType* (see 5.4.3). The *Historian* shall also provide information about the fields that are currently being historized (see 5.4.3).

4.3 Historians and interruption of data collection

When an *Historian* is collecting and storing data, the data collection may be interrupted. The interruption may have been for collecting the current values of data or for an event stream. The interruption may have been due to an interruption in the source of a value or an interruption of the forwarding of historical data from an underlying *Historian*. The interruption might also have been due to an action that stopped the collection of *HistoricalData* or historical *Events*. Some of these interruptions may recover with no loss of data, others may result in data gaps. The *Historian* shall report any gaps when a client is accessing the stored historical data with an error code of *Bad_DataLost*.

For example, if a subscription for data breaks, and the historian recovers the subscription after several minutes, it can check the *SourceTimestamp* of the initial values in the subscription. If the initial value *SourceTimestamp* matches the last stored value *SourceTimestamp*, then no data was lost and nothing needs to be stored indicating the given *HistoricalDataNode*'s data collection was interrupted. But if the *SourceTimestamp* of the initial value is later than the *SourceTimestamp* of the last stored value, then the historian has no way of knowing if any data was lost and it shall record a bad status for the value with the timestamp of when the connection was broken.

For *Event* collection, in addition to what is described for a data collection the following also has to be addressed. If a *Subscription* for *Event* is interrupted for long enough that the *Subscription* buffer reports an overflow (instance of *EventQueueOverflowEventType*), then the *Event* storage shall report this error. This can be accomplished by storing the instance of *EventQueueOverflowEventType* event or by recording a *Bad_DataLost*. If the lost data is indicated by *EventQueueOverflowEventType*, this event shall always be returned for any filter, just as it is in an *Event Subscription*.

4.4 Modification of Historical Data/Events

A *Historian* collects values (data or *Events*) and provides long term storage of these values. Occasionally data that is collected is incorrect. It may be incorrect for many reasons, such as a failed sensor, or failed communication or even an out of calibration sensor. Sometimes the correct values are obtained from another source and the historical value will need to be updated. In many *Historians*, the act of updating a historical value needs to be tracked. This specification provides two aspects for tracking:

- Recording the original value and information related to who made the change (see section 6.9 for *HistoryUpdateDetails*).
- Generating an audit event describing the history edit action (see section 5.8).

Some *Historians* obtain data from other sources (for example not an OPC UA *Subscription*). This data may be from lab data analysis package or other off-line activities. This data is typically provided in bulk and with *SourceTimestamps* that are in the past. For this type of data, even though it may be inserted using an *HistoryUpdateDetails* service call, the *Historian* does not have to create a modification record for the initial insertion.

Modification of *Events* is different than modification of data, in that in a single event stream there may be many *Events* with the same *Timestamp*. *Events* need to be identified by their *EventId*. The *EventId* is generated as part of normal *Event* processing in a *Server* and is used to correlate actions related to the *Event*. For example, when an Alarm (which is represented by an *Event*) is acknowledged, the acknowledgement is related to the specific *EventId* that represent that event is time. A *Historian* that historizes *Events* shall store the *EventId* that it receives; Modification of *Events* shall not allow modification to *EventIds*.

4.5 Timestamps

The nature of OPC UA Historical Access requires that a single timestamp reference be used to relate the multiple data points, and the *Client* may request which timestamp will be used as the reference. See IEC 62541-4 for details on the *TimestampsToReturn* enumeration. An OPC UA *Server* supporting Historical Access will treat the various timestamp settings as described below. A *HistoryRead* with invalid settings will be rejected with *Bad_TimestampsToReturnInvalid* (see IEC 62541-4).

For *HistoricalDataNodes*, the *SourceTimestamp* is used to determine which historical data values are to be returned.

The request is in terms of *SourceTimestamp* but the reply could be in *SourceTimestamp*, *ServerTimestamp* or both timestamps. If the reply has the *Server* timestamp the timestamps could fall outside of the range of the requested time.

SOURCE Return the *SourceTimestamp*.

SERVER Return the *ServerTimestamp*.

BOTH Return both the *SourceTimestamp* and *ServerTimestamp*.

NEITHER This is not a valid setting for any HistoryRead accessing HistoricalDataNodes.

Any reference to timestamps in this context throughout this standard will represent either *ServerTimestamp* or *SourceTimestamp* as dictated by the type requested in the HistoryRead Service. Some Servers may not support historizing both *SourceTimestamp* and *ServerTimestamp*, but it is expected that all Servers will support historizing *SourceTimestamp* (see IEC 62541-7 for details on *Server Profiles*).

If a request is made requesting both *ServerTimestamp* and *SourceTimestamp* and the Server is only collecting the *SourceTimestamp*, the Server shall return *Bad_TimestampNotSupported*. Some *Historians* may aggregate data from underlying *Historians*. These Servers might have a mix of data, some with *ServerTimestamp* some without. The *Historian* may not know if a *HistoricalDataNode* supports *ServerTimestamps*. As a result of this uncertainty, the *Bad_TimestampNotSupported* may be returned on the service level or at the operation level.

For *HistoricalEventNodes* this parameter does not apply. This parameter is ignored since the entries returned are dictated by the *Event Filter*. See IEC 62541-4 for details.

4.6 Bounding Values and time domain

When accessing *HistoricalDataNodes* via the *HistoryRead Service*, requests can set a flag, *returnBounds*, indicating that *BoundingValues* are requested. For a complete description of the *Extensible Parameter HistoryReadDetails* that include *StartTime*, *EndTime* and *NumValuesPerNode*, see 6.5. The concept of Bounding Values and how they affect the time domain that is requested as part of the *HistoryRead* request is further explained in 4.4. 4.4 also provides examples of *TimeDomains* to further illustrate the expected behaviour.

When making a request for historical data using the *HistoryRead Service*, the required parameters include at least two of these three parameters: *startTime*, *endTime* and *numValuesPerNode*. What is returned when *BoundingValues* are requested varies according to which of these parameters are provided. For a *Historian* that has values stored at 5:00, 5:02, 5:03, 5:05 and 5:06, the data returned when using the Read Raw functionality is given by Table 1. In the table, FIRST stands for a tuple with a value of null, a timestamp of the specified *StartTime*, and a *StatusCode* of *Bad_BoundNotFound*. LAST stands for a tuple with a value of null, a timestamp of the specified *EndTime*, and a *StatusCode* of *Bad_BoundNotFound*.

In some cases, attempting to locate bounds, particularly FIRST or LAST points, may be resource intensive for Servers. Therefore, how far back or forward to look in history for Bounding Values is Server dependent, and the Server search limits may be reached before a bounding value can be found. There are also cases, such as reading *Annotations* or *Attribute* data where Bounding Values may not be appropriate. For such use cases it is permissible for the Server to return a *StatusCode* of *Bad_BoundNotSupported*.

Table 1 – Bounding Value examples

Start Time	End Time	numValuesPerNode	Bounds	Data Returned
5:00	5:05	0	Yes	5:00, 5:02, 5:03, 5:05
5:00	5:05	0	No	5:00, 5:02, 5:03
5:01	5:04	0	Yes	5:00, 5:02, 5:03, 5:05
5:01	5:04	0	No	5:02, 5:03
5:05	5:00	0	Yes	5:05, 5:03, 5:02, 5:00
5:05	5:00	0	No	5:05, 5:03, 5:02
5:04	5:01	0	Yes	5:05, 5:03, 5:02, 5:00
5:04	5:01	0	No	5:03, 5:02
4:59	5:05	0	Yes	FIRST, 5:00, 5:02, 5:03, 5:05
4:59	5:05	0	No	5:00, 5:02, 5:03
5:01	5:07	0	Yes	5:00, 5:02, 5:03, 5:05, 5:06, LAST
5:01	5:07	0	No	5:02, 5:03, 5:05, 5:06
5:00	5:05	3	Yes	5:00, 5:02, 5:03

Start Time	End Time	numValuesPerNode	Bounds	Data Returned
5:00	5:05	3	No	5:00, 5:02, 5:03
5:01	5:04	3	Yes	5:00, 5:02, 5:03
5:01	5:04	3	No	5:02, 5:03
5:05	5:00	3	Yes	5:05, 5:03, 5:02
5:05	5:00	3	No	5:05, 5:03, 5:02
5:04	5:01	3	Yes	5:05, 5:03, 5:02
5:04	5:01	3	No	5:03, 5:02
4:59	5:05	3	Yes	FIRST, 5:00, 5:02
4:59	5:05	3	No	5:00, 5:02, 5:03
5:01	5:07	3	Yes	5:00, 5:02, 5:03
5:01	5:07	3	No	5:02, 5:03, 5:05
5:00	UNSPECIFIED	3	Yes	5:00, 5:02, 5:03
5:00	UNSPECIFIED	3	No	5:00, 5:02, 5:03
5:00	UNSPECIFIED	6	Yes	5:00, 5:02, 5:03, 5:05, 5:06, LAST ^a
5:00	UNSPECIFIED	6	No	5:00, 5:02, 5:03, 5:05, 5:06
5:07	UNSPECIFIED	6	Yes	5:06, LAST
5:07	UNSPECIFIED	6	No	NODATA
UNSPECIFIED	5:06	3	Yes	5:06,5:05,5:03
UNSPECIFIED	5:06	3	No	5:06,5:05,5:03
UNSPECIFIED	5:06	6	Yes	5:06,5:05,5:03,5:02,5:00, FIRST ^b
UNSPECIFIED	5:06	6	No	5:06, 5:05, 5:03, 5:02, 5:00
UNSPECIFIED	4:48	6	Yes	5:00, FIRST
UNSPECIFIED	4:48	6	No	NODATA
4:48	4:48	0	Yes	FIRST,5:00
4:48	4:48	0	No	NODATA
4:48	4:48	1	Yes	FIRST
4:48	4:48	1	No	NODATA
4:48	4:48	2	Yes	FIRST,5:00
5:00	5:00	0	Yes	5:00,5:02 ^c
5:00	5:00	0	No	5:00
5:00	5:00	1	Yes	5:00
5:00	5:00	1	No	5:00
5:01	5:01	0	Yes	5:00, 5:02
5:01	5:01	0	No	NODATA
5:01	5:01	1	Yes	5:00
5:01	5:01	1	No	NODATA

a The timestamp of LAST cannot be the specified End Time because there is no specified End Time. In this situation the timestamp for LAST will be equal to the previous timestamp returned plus one second.
 b The timestamp of FIRST cannot be the specified End Time because there is no specified Start Time. In this situation the timestamp for FIRST will be equal to the previous timestamp returned minus one second.
 c When the Start Time = End Time (there is data at that time), and Bounds is set to True, the start bounds will equal the Start Time and the next data point will be used for the end bounds.

4.7 Changes in AddressSpace over time

Clients use the browse *Services* of the *View Service Set* to navigate through the *AddressSpace* to discover the *HistoricalNodes* and their characteristics. These *Services* provide the most current information about the *AddressSpace*. It is possible and probable that the *AddressSpace* of a *Server* will change over time (i.e. *TypeDefinitions* may change; *NodeIds* may be modified, added or deleted).

Server developers and administrators need to be aware that modifying the *AddressSpace* may impact a *Client's* ability to access historical information. If the history for a *HistoricalDataNodes*

is still required, but the *HistoricalDataNodes* is no longer historized, then the *Object* should be maintained in the *AddressSpace*, with the appropriate *AccessLevel Attribute* and *Historizing Attribute* settings (see IEC 62541-3 for details on access levels).

If the *HistoricalConfiguration* of a *HistoricalDataNode* in the *AddressSpace* is changed and audit events are supported, an audit *Event* of *AuditHistoryConfigurationChangeEvent* shall be generated.

A *Historian* shall expose the *Historizing* attribute on any nodes that are currently collecting historical data. They shall also set the *AccessLevel HistoryRead* and optionally the *HistoryWrite* bits for *HistoricalNodes* (those with *Historical* data).

If a *HistoricalConfiguration Object* is added to or deleted from a *Server* or a *Reference* to a *HistoricalConfiguration Object* is added to or deleted from a *Node*, a *GeneralModelChangeEvent* shall be generated. If an *Annotation Property* is added to or deleted from a *Node*, a *GeneralModelChangeEvent* shall be generated.

5 Historical Information Model

5.1 HistoricalNodes

5.1.1 General

The Historical Access model defines additional *Properties* that are applicable for both *HistoricalDataNodes* and *HistoricalEventNodes*.

5.1.2 Annotations Property

The *DataVariable* or *Object* that has *Annotation* data will add the *Annotations Property* as shown in Table 2.

Table 2 – Annotations Property

Name	Use	Data Type	Description
Annotations	O	Annotation	The <i>Annotations Property</i> is used to indicate that the history collection exposed by a <i>HistoricalDataNode</i> supports <i>Annotation</i> data. <i>Annotation DataType</i> is defined in 6.6.6.
ConformanceUnits			
Historical Access Annotations			

Since it is not allowed for *Properties* to have *Properties*, the *Annotations Property* is only available for *DataVariables* or *Objects*.

The *Annotations Property* shall be present on every *HistoricalDataNode* that supports modifications, deletions, or additions of *Annotations* whether or not *Annotations* currently exist. *Annotation* data is accessed using the standard *HistoryRead* functions. *Annotations* are modified, inserted or deleted using the standard *HistoryUpdate* functions and the *UpdateStructuredDataDetails* structure. The presence of the *Annotations Property* does not indicate the presence of *Annotations* on the *HistoricalDataNode*.

A *Server* shall add the *Annotations Property* to a *HistoricalDataNode* only if it will also support *Annotations* on that *HistoricalDataNode*. See IEC 62541-4 for adding *Properties* to *Nodes*. A *Server* shall remove all *Annotation* data if it removes the *Annotations Property* from an existing *HistoricalDataNode*.

An *Annotation* can be accessed in two manners. It can be accessed via *ReadAnnotationDataDetails* or *ReadRawModifiedDetails* structure in the *HistoryRead* service.

The *ReadAnnotationDataDetails* is described in section 6.5.6. The *ReadAnnotationDataDetails* contains an element *reqTimes* that is a list of *Timestamps*. These *Timestamps* can be obtained from an invocation of the *HistoryRead* service call with a *ReadRawModifiedDetails* parameter for the *HistoricalDataNode* of the *Annotations Property*. In the returned *dataValues* array, values with the *ExtraData* flag set provide the *SourceTimestamp*.

The *ReadRawModifiedDetails* structure is described in 6.5.3. For the *ReadRawModifiedDetails*, *Annotation* is returned by passing an instance of the *Annotation Property NodeId* as the *NodeId* in the *NodesToRead* parameter.

Annotation data does not have a separate *HistoricalDataConfigurationType* object instance. The historical configuration defined on the parent node applies to the *Annotations Property*.

As with all *HistoricalNodes*, modifications, deletions or additions of *Annotations* will raise the appropriate historical audit *Event* with the corresponding *NodeId*.

5.2 HistoricalDataNodes

5.2.1 General

The Historical Data model defines additional *ObjectTypes* and *Objects*. These descriptions also include required use cases for *HistoricalDataNodes*.

5.2.2 HistoricalDataConfigurationType

The Historical Access Data model extends the standard type model by defining the *HistoricalDataConfigurationType*. This *Object* defines the general characteristics of a *Node* that defines the historical configuration of any *HistoricalDataNode* that is defined to contain history. It is formally defined in Table 3.

Instances of the *HistoricalDataConfigurationType* use the standard *BrowseName* as defined in 5.2.4.

Table 3 – HistoricalDataConfigurationType definition

Attribute	Value				
BrowseName	HistoricalDataConfigurationType				
IsAbstract	False				
References	NodeClass	BrowseName	Data Type	TypeDefinition	Modelling Rule
HasComponent	Object	AggregateConfiguration	--	AggregateConfigurationType	Mandatory
HasComponent	Object	AggregateFunctions	--	FolderType	Optional
HasProperty	Variable	Stepped	Boolean	PropertyType	Mandatory
HasProperty	Variable	Definition	String	PropertyType	Optional
HasProperty	Variable	MaxTimeInterval	Duration	PropertyType	Optional
HasProperty	Variable	MinTimeInterval	Duration	PropertyType	Optional
HasProperty	Variable	ExceptionDeviation	Double	PropertyType	Optional
HasProperty	Variable	ExceptionDeviationFormat	Exception DeviationFormat	PropertyType	Optional
HasProperty	Variable	StartOfArchive	UtcTime	PropertyType	Optional
HasProperty	Variable	StartOfOnlineArchive	UtcTime	PropertyType	Optional
HasProperty	Variable	ServerTimestampSupported	Boolean	PropertyType	Optional
HasProperty	Variable	MaxTimeStoredValues	Duration	PropertyType	Optional
HasProperty	Variable	MaxCountStoredValues	UInt32	PropertyType	Optional
ConformanceUnits					
Historical Access Exception Storage					
Historical Access Periodic Storage					
Historical Access Default HistoricalData Configuration					
Historical Access Archive Period					

AggregateConfiguration Object represents the browse entry point for information on how the *Server* treats *Aggregate* specific functionality such as handling *Uncertain data*. This *Object* is required to be present even if it contains no *Aggregate* configuration *Objects*. *Aggregates* are defined in IEC 62541-13.

AggregateFunctions is an entry point to browse to all *Aggregate* capabilities supported by the *Server* for Historical Access. All *HistoryAggregates* supported by the *Server* should be able to be browsed starting from this *Object*. *Aggregates* are defined in IEC 62541-13.

The *Stepped Variable* specifies whether the historical data was collected in such a manner that it should be displayed as *SlopedInterpolation* (sloped line between points) or as *SteppedInterpolation* (vertically-connected horizontal lines between points) when *RawData* is examined. This *Property* also effects how some *Aggregates* are calculated. A value of True indicates the stepped interpolation mode. A value of False indicates *SlopedInterpolation* mode. The default value is False.

The *Definition Variable* is a vendor-specific, human readable string that specifies how the value of this *HistoricalDataNode* is calculated. Definition is non-localized and will often contain an equation that can be parsed by certain *Clients*.

Example: *Definition::*“(TempA – 25) + TempB”

The *MaxTimeInterval Variable* specifies the maximum interval between data points in the history repository regardless of their value change (see IEC 62541-3 for definition of *Duration*).

The *MinTimeInterval Variable* specifies the minimum interval between data points in the history repository regardless of their value change (see IEC 62541-3 for definition of *Duration*).

If *MaxTimeInterval*=*MinTimeInterval* and they do not equal 0 then data is collected based on a periodic sampling rate at the defined interval.

The *ExceptionDeviation Variable* specifies the minimum amount that the data for the *HistoricalDataNodes* shall change in order for the change to be reported to the history database.

The *ExceptionDeviationFormat Variable* specifies how the *ExceptionDeviation* is determined. Its values are defined in Table 4.

The *StartOfArchive Variable* specifies the date before which there is no data in the archive either online or offline.

The *StartOfOnlineArchive Variable* specifies the date of the earliest data in the online archive.

The presence of the optional *Variable ServerTimestampSupported* indicates the *Server* supports the storage of *ServerTimestamps*. A value of True indicates the *Server* (and this configuration) stores *ServerTimestamps* in addition to *SourceTimestamp*. The default is False.

The *MaxTimeStoredValues* is the maximum time period that is available for historical data. This is set if the historian is a circular buffer based on time (e.g. it stores the last 6 hours of data).

The *MaxCountStoredValues* is the maximum number of values that can be stored for a single point assigned to this configuration.

Typically, either *MaxTimeStoredValues* or *MaxCountStoredValues* is provided, if both are provided the smaller of the two applies. E.g. if 3 hours are specified and 10,000 values, then for a slowly changing point the 3 hour would be the cut off, but for a point that change more than once a second the 10,000 values would be encountered first.

Table 4 – ExceptionDeviationFormat Items

Name	Value	Description
ABSOLUTE_VALUE	0	<i>ExceptionDeviation</i> is an absolute <i>Value</i> .
PERCENT_OF_VALUE	1	<i>ExceptionDeviation</i> is a percentage of <i>Value</i> .

PERCENT_OF_RANGE	2	<i>ExceptionDeviation</i> is a percentage of <i>InstrumentRange</i> (see IEC 62541-8).
PERCENT_OF_EU_RANGE	3	<i>ExceptionDeviation</i> is a percentage of <i>EURange</i> (see IEC 62541-8).
UNKNOWN	4	<i>ExceptionDeviation</i> type is <i>Unknown</i> or not specified.

Its representation in the AddressSpace is defined in Table 52.

Table 5 – ExceptionDeviationFormat definition

Attribute		Value			
BrowseName		ExceptionDeviationFormat			
IsAbstract		False			
References	NodeClass	BrowseName	DataType	TypeDefinition	Other
Subtype of the 0:Enumeration type defined in IEC 62541-5					
0:HasProperty	Variable	0:EnumStrings	0:LocalizedText []	0:PropertyType	
Conformance Units					
Historical Access Exception Storage					

5.2.3 Attributes

Subclause 5.2.6 lists the *Attributes* of *Variables* that have particular importance for historical data. They are specified in detail in IEC 62541-3.

- AccessLevel
- Historizing

5.2.4 Historical Data Configuration Object

This *Object* is used as the browse entry point for information about *HistoricalDataNode* configuration. The content of this *Object* is already defined by its type definition in Table 3. It is formally defined in Table 6. If a *HistoricalDataNode* has configuration defined then one instance shall have a *BrowseName* of 'HA Configuration'. Additional configurations may be defined with different *BrowseNames*. It is highly recommended that display names for historical configuration historical configuration are chosen that clearly describe the historical configuration e.g. "1 Second Collection", "Long Term Configuration" etc. All historical configuration *Objects* shall be referenced using the *HasHistoricalConfiguration ReferenceType*.

Table 6 – Historical Access configuration definition

Attribute	Value				
BrowseName	HA Configuration				
References	Node Class	BrowseName	DataType	TypeDefinition	ModellingRule
HasTypeDefinition	Object Type	HistoricalDataConfigurationType	Defined in Table 3		

5.3 References

5.3.1 Overview

Figure 2 illustrate the references defined by this specification (outlined in red). Each reference type is describe in there own sections.