

DIN EN IEC 62541-13

ICS 25.040.40; 35.200; 35.240.50

Einsprüche bis 2019-10-09
Vorgesehen als Ersatz für
DIN EN 62541-13:2015-10**Entwurf****OPC Unified Architecture –
Teil 13: Aggregation von Daten
(IEC 65E/613/CDV:2018);
Englische Fassung prEN IEC 62541-13:2018**OPC Unified Architecture –
Part 13: Aggregates
(IEC 65E/613/CDV:2018);
English version prEN IEC 62541-13:2018Architecture unifiée OPC –
Partie 13: Agrégats
(IEC 65E/613/CDV:2018);
Version anglaise prEN IEC 62541-13:2018**Anwendungswarnvermerk**

Dieser Norm-Entwurf mit Erscheinungsdatum 2019-08-09 wird der Öffentlichkeit zur Prüfung und Stellungnahme vorgelegt.

Weil die beabsichtigte Norm von der vorliegenden Fassung abweichen kann, ist die Anwendung dieses Entwurfs besonders zu vereinbaren.

Stellungnahmen werden erbeten

- vorzugsweise online im Norm-Entwurfs-Portal von DIN unter www.din.de/go/entwuerfe bzw. für Norm-Entwürfe der DKE auch im Norm-Entwurfs-Portal der DKE unter www.entwuerfe.normenbibliothek.de, sofern dort wiedergegeben;
- oder als Datei per E-Mail an dke@vde.com möglichst in Form einer Tabelle. Die Vorlage dieser Tabelle kann im Internet unter www.din.de/go/stellungnahmen-norm-entwuerfe oder für Stellungnahmen zu Norm-Entwürfen der DKE unter www.dke.de/stellungnahme abgerufen werden;
- oder in Papierform an die DKE Deutsche Kommission Elektrotechnik Elektronik Informationstechnik in DIN und VDE, Stresemannallee 15, 60596 Frankfurt am Main.

Die Empfänger dieses Norm-Entwurfs werden gebeten, mit ihren Kommentaren jegliche relevanten Patentrechte, die sie kennen, mitzuteilen und unterstützende Dokumentationen zur Verfügung zu stellen.

Gesamtumfang 94 Seiten

DKE Deutsche Kommission Elektrotechnik Elektronik Informationstechnik in DIN und VDE

Nationales Vorwort

Die Englische Fassung des europäischen Dokuments prEN IEC 62541-13:2018 „OPC Unified Architecture – Teil 13: Aggregation von Daten“ (Entwurf in der Umfrage) ist unverändert in diesen Norm-Entwurf übernommen worden.

Die Internationale Elektrotechnische Kommission (IEC) und das Europäische Komitee für Elektrotechnische Normung (CENELEC) haben vereinbart, dass ein auf IEC-Ebene erarbeiteter Entwurf für eine Internationale Norm zeitgleich (parallel) bei IEC und CENELEC zur Umfrage (CDV-Stadium) und Abstimmung als FDIS (en: Final Draft International Standard) bzw. Schluss-Entwurf für eine Europäische Norm gestellt wird, um eine Beschleunigung und Straffung der Normungsarbeit zu erreichen. Dem entsprechend ist das internationale Dokument IEC 65E/613/CDV:2018 „OPC Unified Architecture – Part 13: Aggregates“ unverändert in den Entwurf prEN IEC 62541-13:2018 übernommen worden.

Das internationale Dokument wurde vom SC 65E „Devices and integration in enterprise systems“ der Internationalen Elektrotechnischen Kommission (IEC) erarbeitet.

Bei der Abstimmung zu dem Europäischen Schluss-Entwurf bei CENELEC und dem Internationalen Schluss-Entwurf bei IEC [Final Draft International Standard (FDIS)] sind jeweils nur „JA/NEIN“-Entscheidungen möglich, wobei „NEIN“-Entscheidungen fundiert begründet werden müssen. Dokumente, die bei CENELEC als Europäische Norm angenommen und ratifiziert werden, sind unverändert als Deutsche Normen zu übernehmen.

Für dieses Dokument ist das nationale Arbeitsgremium K 931 „Systemaspekte der Automatisierung“ der DKE Deutsche Kommission Elektrotechnik Elektronik Informationstechnik in DIN und VDE (www.dke.de) zuständig.

Für den Fall einer undatierten Verweisung im normativen Text (Verweisung auf ein Dokument ohne Angabe des Ausgabedatums und ohne Hinweis auf eine Abschnittsnummer, eine Tabelle, ein Bild usw.) bezieht sich die Verweisung auf die jeweils aktuellste Ausgabe des in Bezug genommenen Dokuments.

Für den Fall einer datierten Verweisung im normativen Text bezieht sich die Verweisung immer auf die in Bezug genommene Ausgabe des Dokuments.

Der Zusammenhang der zitierten Dokumente mit den entsprechenden deutschen Dokumenten ergibt sich, soweit ein Zusammenhang besteht, grundsätzlich über die Nummer der entsprechenden IEC-Publikation. Beispiel: IEC 60068 ist als EN 60068 als Europäische Norm durch CENELEC übernommen und als DIN EN 60068 ins Deutsche Normenwerk aufgenommen.

Das Präsidium des DIN hat mit Präsidialbeschluss 1/2004 festgelegt, dass DIN-Normen, deren Inhalt sich auf internationale Arbeitsergebnisse der Informationsverarbeitung gründet, unter bestimmten Bedingungen allein in englischer Sprache veröffentlicht werden dürfen. Diese Bedingungen sind für den vorliegenden Norm-Entwurf erfüllt.

Da sich die Benutzer des vorliegenden Norm-Entwurfs der englischen Sprache als Fachsprache bedienen, wird die englische Fassung der prEN IEC 62541-13 veröffentlicht. Zu deren Abschnitt 1, der den Anwendungsbereich festlegt, und Unterabschnitt 3.1, der die Begriffe festlegt, wurde eine Übersetzung angefertigt und als informativer Nationaler Anhang NA des vorliegenden Norm-Entwurfs hinzugefügt. Für einige der verwendeten Begriffe existieren keine gebräuchlichen deutschen Benennungen, da sich die deutschen Anwender in der Regel ebenfalls der englischen Benennungen bedienen. Dieser Norm-Entwurf steht nicht in unmittelbarem Zusammenhang mit Rechtsvorschriften und ist nicht als Sicherheitsnorm anzusehen.

Es ist vorgesehen, auch bei der entsprechenden zukünftigen Deutschen Norm auf die deutsche Sprachfassung zu verzichten und diese in der englischsprachigen Fassung zu veröffentlichen.

Es wird auf die Möglichkeit hingewiesen, dass einige Elemente dieses Dokuments Patentrechte berühren können. IEC ist nicht dafür verantwortlich, einige oder alle diesbezüglichen Patentrechte zu identifizieren.

Das Original-Dokument enthält Bilder in Farbe, die in der Papierversion in einer Graustufen-Darstellung wiedergegeben werden. Elektronische Versionen dieses Dokuments enthalten die Bilder in der originalen Farbdarstellung.

Änderungen

Gegenüber DIN EN 62541-13:2015-10 wurden folgende Änderungen vorgenommen:

- a) Keine technischen Änderungen, aber zahlreiche Klarstellungen. Zudem einige Korrekturen der Beispiele.

Nationaler Anhang NA (informativ)

Auszugsweise Übersetzung

Reihenfolge und Inhalt der folgenden Abschnitte sind identisch mit denen in den jeweiligen Abschnitten der Englischen Fassung.

1 Anwendungsbereich

Dieses Dokument ist Teil der gesamten Normenreihe zur OPC Unified Architecture (vereinheitlichte Architektur der plattformunabhängigen Kommunikation) (OPC-UA)) und legt das Informationsmodell fest, welches im Zusammenhang mit Aggregaten (en: *Aggregates*) steht.

3 Begriffe und Abkürzungen

3.1 Begriffe

Für die Anwendung dieses Dokuments gelten die Begriffe nach IEC TR 62541-1, IEC 62541-3, IEC 62541-4, IEC 62541-11 und die folgenden Begriffe.

3.1.1

VerarbeitungsIntervall

(en: *ProcessingInterval*)

Zeitspanne, für die abgeleitete Werte auf der Grundlage eines bestimmten *Aggregates* (en: *Aggregate*) produziert werden

Anmerkung 1 zum Begriff: Die für AuslesungVerarbeitet (en: *ReadProcessed*) festgelegte gesamte Zeitdomäne wird durch das *VerarbeitungsIntervall* geteilt. Wird beispielsweise eine Durchschnittsfunktion von 10 min über einen Zeitraum von 12:00 bis 12:30 ausgeführt, ergibt dies einen Satz von drei Intervallen der Länge des *VerarbeitungsIntervalls*, wobei die einzelnen Intervalle eine Startzeit von 12:00, 12:10 bzw. 12:20 haben. Die Regeln, die zur Bestimmung der Grenzen des Intervalls verwendet werden, werden in 5.4.2.2 behandelt.

3.1.2

interpoliert

(en: *interpolated*)

Daten, die aus Datenstichproben berechnet werden

Anmerkung 1 zum Begriff: Datenstichproben können historisierte Daten oder gepufferte Echtzeitdaten sein. Ein *interpolierter* Wert wird aus den Datenpunkten vor und nach dem abgefragten Zeitstempel berechnet.

3.1.3

EffektivEndZeit

(en: *EffectiveEndTime*)

Zeit unmittelbar vor der *EndZeit*

Anmerkung 1 zum Begriff: Bei allen Berechnungen von *Aggregaten* (en: *Aggregate*) wird die *StartZeit* (en: *startTime*) eingeschlossen, die *EndZeit* (en: *endTime*) jedoch ausgeschlossen. Es ist jedoch manchmal notwendig, eine *interpolierte* (en: *interpolated*) Endgrenze als Wert für ein *Intervall* (en: *interval*) mit einem Zeitstempel, der innerhalb des Intervalls liegt, zurückzugeben. *Servers* müssen die Zeit unmittelbar vor der *EndZeit* verwenden, wenn die Zeitauflösung des *Servers* den genauen Wert bestimmt (dies ist nicht mit der Zeitauflösung der Hardware oder des Betriebssystems zu verwechseln). Wenn die *EndZeit* z. B. 12:01:00 ist, entspricht die Zeitauflösung 1 s und die *EffektiveEndZeit* ist 12:00:59. Siehe 5.4.2.4.

Läuft die Zeit rückwärts, müssen *Server* die Zeit unmittelbar nach der *EndZeit* verwenden, wenn die Zeitauflösung des *Servers* den genauen Wert bestimmt.

3.1.4

extrapoliert

(en: extrapolated)

Daten, die aus einem diskreten Datensatz gebildet werden, aber außerhalb dieses Datensatzes liegen

Anmerkung 1 zum Begriff: Extrapolation ist vergleichbar mit dem Prozess der Interpolation, bei dem neue Punkte zwischen bekannten Punkten gebildet werden, das Ergebnis unterliegt jedoch einer größeren Unsicherheit. *Extrapolierte* Daten werden in Fällen verwendet, in denen der abgefragte Zeitraum weiter in der Zukunft liegt als die im zugrundeliegenden System verfügbaren Daten. Siehe Beispiel in Tabelle 1.

3.1.5

Geneigte Interpolation

(en: Sloped Interpolation)

einfache lineare Interpolation

Anmerkung 1 zum Begriff: Vergleichbar mit der Kurvenanpassung mithilfe von linearen Polynomen. Siehe Beispiel in Tabelle 1.

3.1.6

Schrittweise Interpolation

(en: Stepped Interpolation)

der letzte Datenpunkt wird konstant gehalten, oder der Wert wird auf der Grundlage einer horizontalen Linienanpassung interpoliert

Anmerkung 1 zum Begriff: Es ist die folgende Tabelle 1 mit Rohwerten und *interpolierten/extrapolierten* (en: Interpolated/Extrapolated) Daten zu berücksichtigen:

Tabelle 1 – Beispiele für Interpolation

Zeitstempel	Rohwert	Geneigte Interpolation	Schrittweise 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		
		Geneigte Extrapolation	Schrittweise Extrapolation
12:00:25		35	30
12:00:27		37	30

3.1.7

Grenzwerte

(en: Bounding Values)

Werte zur *StartZeit* und *EndZeit*, die für *Aggregate* zur Berechnung des Ergebnisses benötigt werden

Anmerkung 1 zum Begriff: Wenn zur *StartZeit* und *EndZeit* keine Rohdaten vorhanden sind, muss ein Wert geschätzt werden. Es gibt zwei Möglichkeiten, *Grenzwerte* für ein Intervall zu bestimmen. Bei der ersten Möglichkeit (den sogenannten *interpolierten Grenzwerten*) (en: *Interpolated Bounding Values*) werden die ersten gültigen (en: non-Bad) Datenpunkte vor und nach dem Zeitstempel zur Schätzung der Grenze verwendet. Bei der zweiten Möglichkeit (den sogenannten *einfachen Grenzwerten*) (en: *Simple Bounding Values*) werden die Datenpunkte unmittelbar vor und nach den Grenzzeitstempeln zur Schätzung der Grenze verwendet, auch wenn diese ungültig (en: Bad) sind. In 3.1.8 und 3.1.9 werden diese beiden Ansätze ausführlicher beschrieben.

In allen Fällen wird das Flag *UnsicherAlsUngültigBehandeln* (en: *TreatUncertainAsBad*) (siehe 4.2.1.2) verwendet, um festzulegen, ob unsichere Werte ungültig oder gültig sind.

Wenn kein Rohwert ermittelt wurde und ein gültiger Grenzwert vorhanden ist, werden die *Aggregat*-Bits (siehe 5.3.3) auf „interpoliert“ gesetzt.

Bei der Berechnung von *Grenzwerten* wird die Wertemenge der Rohdaten mit dem Status Ungültig auf null gesetzt. Das bedeutet, dass die Wertemenge in keiner Berechnung verwendet wird und dass im Falle der Rückgabe des Rohwertes eine Null zurückgegeben wird. Der Status wird anhand der Regeln, die durch die Grenze oder das *Aggregat* vorgegeben sind, bestimmt.

Der Ansatz der *interpolierten Grenzwerte* (siehe 3.1.8) wird auch beim klassischen OPC-Zugriff auf historische Daten angewendet und ist für Anwendungen wichtig, die jederzeit auf nützliche Werte zugreifen können müssen, z. B. in der erweiterten Prozesssteuerung. Der Ansatz der *einfachen Grenzwerte* (siehe 3.1.9) ist neu in dieser Norm und ist für Anwendungen wichtig, die regelmäßige Berichte erzeugen müssen und keine Schätzwerte für ungültige Daten verwenden können.

3.1.8

interpolierte Grenzwerte

(en: interpolated bounding values)

Grenzwerte, die durch eine Berechnung mit dem nächstliegenden gültigen Wert bestimmt werden

Anmerkung 1 zum Begriff: Interpolierte Grenzwerte, welche die GeneigteInterpolation verwenden, werden wie folgt berechnet:

- wenn zu dem Zeitstempel ein gültiger Rohwert vorhanden ist, ist dieser als Grenzwert zu verwenden;
- es ist der erste gültige Rohwert vor dem Zeitstempel zu ermitteln;
- es ist der erste gültige Rohwert nach dem Zeitstempel zu ermitteln;
- der Wert vor und der Wert nach dem Zeitstempel sind durch eine Linie zu verbinden;
- der Punkt, an dem die Linie den Zeitstempel kreuzt, ist als Schätzwert für den Grenzwert zu verwenden.

Die Berechnung kann mit der folgenden Gleichung ausgedrückt werden:

$$V_{\text{grenze}} = (T_{\text{grenze}} - T_{\text{vor}}) \times (V_{\text{nach}} - V_{\text{vor}}) / (T_{\text{nach}} - T_{\text{vor}}) + V_{\text{vor}}$$

Dabei ist V_x ein Wert bei „x“ und T_x ist der Zeitstempel im Zusammenhang mit V_x .

Wenn keine gültigen Werte vor dem Zeitstempel vorhanden sind, ist der *StatusCode* (en: *StatusCode*) *Ungültig_KeineDaten* (en: *Bad_NoData*). Wenn zwischen dem Wert vor und dem Wert nach dem Zeitstempel ungültige Werte vorhanden sind, ist der *StatusCode* *Unsicher_DatenUnterNormal* (en: *Uncertaine_DataSubNormal*). Wenn entweder der Wert vor oder der Wert nach dem Zeitstempel unsicher ist, ist der *StatusCode* *Unsicher_DatenUnterNormal* (en: *Uncertaine_DataSubNormal*). Wenn nach dem Zeitstempel kein Wert vorhanden ist, muss der Wert vor dem Zeitstempel unter Anwendung der *GeneigtenExtrapolation* (en: *SlopedExtrapolation*) oder der *SchrittweisenExtrapolation* (en: *SteppedExtrapolation*) extrapoliert werden.

Die Dauer der Suche nach den gültigen Werten vor und nach dem Zeitstempel ist abhängig vom *Server*; wenn innerhalb einer angemessenen Zeitspanne jedoch kein gültiger Wert ermittelt werden kann, nimmt der *Server* an, dass kein gültiger Wert vorhanden ist. Der *Server* sollte mindestens eine Zeitspanne lang suchen, die der Größe des *VerarbeitungsIntervalls* (en: *ProcessingInterval*) entspricht.

Interpolierte Grenzwerte, welche die *GeneigteExtrapolation* verwenden, werden wie folgt berechnet:

- es ist der erste gültige Rohwert vor dem Zeitstempel zu ermitteln;
- es ist der zweite gültige Rohwert vor dem Zeitstempel zu ermitteln;
- diese beiden Werte sind durch eine Linie zu verbinden;
- die Linie ist bis zu dem Punkt, an dem sie den Zeitstempel kreuzt, zu verlängern;
- der Punkt, an dem die Linie den Zeitstempel kreuzt, ist als Schätzwert für den Grenzwert zu verwenden.

Die Gleichung entspricht der Gleichung, die für die *GeneigteInterpolation* verwendet wird.

Der *StatusCode* ist immer *Unsicher_DatenUnterNormal*. Wenn vor dem Zeitstempel nur ein einziger gültiger Rohwert ermittelt werden kann, wird die *SchrittweiseExtrapolation* zur Schätzung des Grenzwertes angewendet.

Interpolierte Grenzwerte, welche die *SchrittweiseInterpolation* verwenden, werden wie folgt berechnet:

- wenn zu dem Zeitstempel ein gültiger Rohwert vorhanden ist, ist dieser als Grenzwert zu verwenden;
- es ist der erste gültige Rohwert vor dem Zeitstempel zu ermitteln;
- dieser Wert ist als Schätzwert für den Grenzwert zu verwenden.

Wenn zwischen dem Wert vor dem Zeitstempel und dem Zeitstempel ungültige Werte vorhanden sind, ist der *StatusCode* *Unsicher_DatenUnterNormal*. Wenn vor dem Zeitstempel keine gültigen Rohdaten vorhanden sind, ist der *StatusCode* *Ungültig_KeineDaten*. Wenn der Wert vor dem Zeitstempel unsicher ist, ist der *StatusCode* *Unsicher_DatenUnterNormal*. Bei der *SchrittweisenInterpolation* wird der Wert nach dem Zeitstempel nicht benötigt; liegt der Zeitstempel jedoch nach dem Ende der Daten, wird der Grenzwert als extrapoliert betrachtet und der *StatusCode* ist *Unsicher_DatenUnterNormal*.

Der Ausdruck *SchrittweiseExtrapolation* beschreibt die *SchrittweiseInterpolation* für den Fall, dass ein Zeitstempel nach dem letzten Wert in der historischen Datensammlung liegt.

3.1.9

einfache Grenzwerte

(en: simple bounding values)

Grenzwerte, die durch eine Berechnung mit dem nächstliegenden Wert bestimmt werden

Anmerkung 1 zum Begriff: *Einfache Grenzwerte*, welche die *GeneigteInterpolation* verwenden, werden wie folgt berechnet:

- wenn zu dem Zeitstempel ein beliebiger Rohwert vorhanden ist, ist dieser als Grenzwert zu verwenden;
- es ist der erste Rohwert vor dem Zeitstempel zu ermitteln;
- es ist der erste Rohwert nach dem Zeitstempel zu ermitteln;
- wenn der Wert nach dem Zeitstempel ungültig ist, wird der Wert vor dem Zeitstempel als Grenzwert verwendet;
- der Wert vor und der Wert nach dem Zeitstempel sind durch eine Linie zu verbinden;
- der Punkt, an dem die Linie den Zeitstempel kreuzt, ist als Schätzwert für den Grenzwert zu verwenden.

Die Gleichung entspricht der Gleichung, die für die *GeneigteInterpolation* in 3.1.5 verwendet wird.

Wenn ein Rohwert zu dem Zeitstempel ungültig ist, ist der *StatusCode* *Ungültig_KeineDaten*. Wenn der Wert vor dem Zeitstempel ungültig ist, ist der *StatusCode* *Ungültig_KeineDaten*. Wenn der Wert vor dem Zeitstempel unsicher ist, ist der *StatusCode* *Unsicher_DatenUnterNormal*. Wenn der Wert nach dem Zeitstempel ungültig oder unsicher ist, ist der *StatusCode* *Unsicher_DatenUnterNormal*.

Einfache Grenzwerte, welche die *SchrittweiseInterpolation* verwenden, werden wie folgt berechnet:

- wenn zu dem Zeitstempel ein beliebiger Rohwert vorhanden ist, ist dieser als Grenzwert zu verwenden;
- es ist der erste Rohwert vor dem Zeitstempel zu ermitteln;
- wenn der Wert vor dem Zeitstempel gültig ist, wird er als Grenzwert verwendet.

Wenn ein Rohwert zu dem Zeitstempel ungültig ist, ist der *StatusCode* *Ungültig_KeineDaten*. Wenn der Wert vor dem Zeitstempel ungültig ist, ist der *StatusCode* *Ungültig_KeineDaten*. Wenn der Wert vor dem Zeitstempel unsicher ist, ist der *StatusCode* *Unsicher_DatenUnterNormal*.

Wenn eine der Grenzzeiten eines Intervalls hinter dem letzten Datenpunkt liegt, kann der Server Extrapolation anwenden oder einen Fehler ausgeben. Wenn der Server Extrapolation anwendet, ist der Typ der Extrapolation [*SchrittweiseExtrapolation* oder *GeneigteExtrapolation*] serverspezifisch.

In einigen Historians zeigt der letzte Rohwert nicht zwingend das Ende der Daten an. Auf der Grundlage der Kenntnisse des Historians über die Mechanismen zur Datensammlung, d. h. die Häufigkeit von Datenaktualisierungen und

Latenzzeiten, darf der Historian den letzten Wert auf eine Zeit erweitern, die bekanntermaßen abgedeckt ist. Bei der Berechnung *einfacher Grenzwerte* verhält sich der Historian so, als ob zu diesem Zeitstempel ein weiterer Rohwert vorhanden ist.

Dasselbe gilt, wenn der früheste Zeitpunkt eines Intervalls vor dem ersten Datenpunkt in der Historie beginnt und der letzte Zeitpunkt nach dem ersten Datenpunkt in der Historie liegt; dann wird das Intervall so behandelt, als ob es sich vom ersten Datenpunkt in der Historie bis zum letzten Zeitpunkt des Intervalls erstreckt, und im *StatusCode* des Intervalls wird das Teil-Bit gesetzt (siehe 5.3.3.2).

Die Dauer der Suche nach den Werten vor und nach dem Zeitstempel ist abhängig vom *Server*; wenn innerhalb einer angemessenen Zeitspanne jedoch kein Wert ermittelt werden kann, nimmt der *Server* an, dass kein Wert vorhanden ist. Der *Server* sollte mindestens eine Zeitspanne lang suchen, die der Größe des Verarbeitungsintervalls entspricht.

CONTENTS

FIGURES	5
TABLES	5
FOREWORD	2
1 Scope	9
2 Normative references	9
3 Terms, definitions, and abbreviations	9
3.1 Terms and definitions	9
3.2 Abbreviations	12
4 Aggregate Information Model	12
4.1 General	12
4.2 Aggregate Objects	12
4.2.1 General	12
4.2.2 AggregateFunction Object	13
4.3 MonitoredItem AggregateFilter	15
4.3.1 MonitoredItem AggregateFilter Defaults	15
4.3.2 MonitoredItem Aggregates and Bounding Values	15
4.4 Exposing Supported Functions and Capabilities	15
5 Aggregate specific usage of Services	16
5.1 General	16
5.2 Aggregate data handling	17
5.2.1 Overview	17
5.2.2 ReadProcessedDetails structure overview	17
5.2.3 AggregateFilter structure overview	17
5.3 Aggregates StatusCodes	17
5.3.1 Overview	17
5.3.2 Operation level result codes	18
5.3.3 Aggregate Information Bits	18
5.4 Aggregate details	19
5.4.1 General	19
5.4.2 Common characteristics	20
5.4.3 Specific Aggregated data handling	23
Annex A (informative) Aggregate Specific examples – Historical Access	54
A.1 Historical Aggregate specific characteristics	54
A.1.1 Example Aggregate data – Historian 1	54
A.1.2 Example Aggregate data – Historian 2	55
A.1.3 Example Aggregate data – Historian 3	56
A.1.4 Example Aggregate data – Historian 4	57
A.2 Interpolative	57
A.2.1 Description	57
A.2.2 Interpolative data	57
A.3 Average	59
A.3.1 Description	59
A.3.2 Average data	59
A.4 TimeAverage	60
A.4.1 Description	60

A.4.2	TimeAverage data	60
A.5	TimeAverage2	61
A.5.1	Description	61
A.5.2	TimeAverage2 data	61
A.6	Total	63
A.6.1	Description	63
A.6.2	Total data	63
A.7	Total2	64
A.7.1	Description	64
A.7.2	Total2 data	65
A.8	Minimum	66
A.8.1	Description	66
A.8.2	Minimum data	66
A.9	Maximum	66
A.9.1	Description	66
A.9.2	Maximum data	67
A.10	MinimumActualTime	67
A.10.1	Description	67
A.10.2	MinimumActualTime data	67
A.11	MaximumActualTime	68
A.11.1	Description	68
A.11.2	MaximumActualTime data	68
A.12	Range	68
A.12.1	Description	68
A.12.2	Range data	69
A.13	Minimum2	69
A.13.1	Description	69
A.13.2	Minimum2 data	69
A.14	Maximum2	70
A.14.1	Description	70
A.14.2	Maximum2 data	70
A.15	MinimumActualTime2	70
A.15.1	Description	70
A.15.2	MinimumActualTime2 data	71
A.16	MaximumActualTime2	71
A.16.1	Description	71
A.16.2	MaximumActualTime2 data	71
A.17	Range2	72
A.17.1	Description	72
A.17.2	Range2 data	72
A.18	AnnotationCount	73
A.18.1	Description	73
A.18.2	AnnotationCount data	73
A.19	Count	73
A.19.1	Description	73
A.19.2	Count data	73
A.20	DurationInStateZero	74
A.20.1	Description	74
A.20.2	DurationInStateZero data	74

A.21	DurationInStateNonZero	74
A.21.1	Description	74
A.21.2	DurationInStateNonZero data	74
A.22	NumberOfTransitions	75
A.22.1	Description	75
A.22.2	NumberOfTransitions data	75
A.23	Start	75
A.23.1	Description	75
A.23.2	Start data	75
A.24	End	76
A.24.1	Description	76
A.24.2	End data	76
A.25	StartBound	77
A.25.1	Description	77
A.25.2	StartBound data	77
A.26	EndBound	78
A.26.1	Description	78
A.26.2	EndBound data	78
A.27	Delta	78
A.27.1	Description	78
A.27.2	Delta data	78
A.28	DeltaBounds	79
A.28.1	Description	79
A.28.2	DeltaBounds data	79
A.29	DurationGood	80
A.29.1	Description	80
A.29.2	DurationGood data	80
A.30	DurationBad	80
A.30.1	Description	80
A.30.2	DurationBad data	81
A.31	PercentGood	81
A.31.1	Description	81
A.31.2	PercentGood data	81
A.32	PercentBad	82
A.32.1	Description	82
A.32.2	PercentBad data	82
A.33	WorstQuality	83
A.33.1	Description	83
A.33.2	WorstQuality data	83
A.34	WorstQuality2	84
A.34.1	Description	84
A.34.2	WorstQuality2 data	84
A.35	StandardDeviationSample	85
A.35.1	Description	85
A.35.2	StandardDeviationSample data	85
A.36	VarianceSample	85
A.36.1	Description	85
A.36.2	VarianceSample data	85
A.37	StandardDeviationPopulation	86

A.37.1	Description.....	86
A.37.2	StandardDeviationPopulation data	86
A.38	VariancePopulation	87
A.38.1	Description.....	87
A.38.2	VariancePopulation data	87

FIGURES

Figure 1 – Representation of Aggregate Configuration information in the AddressSpace	16
Figure 2 – Variable with Stepped = False and Simple Bounding Values.....	24
Figure 3 – Variable with Stepped = True and Interpolated Bounding Values	25

TABLES

Table 1 – Interpolation examples.....	10
Table 2 – AggregateConfigurationType Definition.....	12
Table 3 – Aggregate Functions Definition	13
Table 4 – AggregateFunctionType Definition	13
Table 5 – Standard AggregateType Nodes	14
Table 6 – ReadProcessedDetails	17
Table 7 – AggregateFilter structure	17
Table 8 – Bad operation level result codes	18
Table 9 – Uncertain operation level result codes	18
Table 10 – Data location	18
Table 11 – Additional information	19
Table 12 – History Aggregate interval information	20
Table 13 – Standard History Aggregate Data Type information.....	22
Table 14 – Aggregate table description	26
Table 15 – Interpolative Aggregate summary.....	27
Table 16 – Average Aggregate summary	28
Table 17 – TimeAverage Aggregate summary	28
Table 18 – TimeAverage2 Aggregate summary	30
Table 19 – Total Aggregate summary	30
Table 20 – Total2 Aggregate summary	31
Table 21 – Minimum Aggregate summary	32
Table 22 – Maximum Aggregate summary	32
Table 23 – MinimumActualTime Aggregate summary	33
Table 24 – MaximumActualTime Aggregate summary.....	34
Table 25 – Range Aggregate summary.....	34
Table 26 – Minimum2 Aggregate summary	36
Table 27 – Maximum2 Aggregate summary	36
Table 28 – MinimumActualTime2 Aggregate summary.....	37
Table 29 – MaximumActualTime2 Aggregate summary.....	38
Table 30 – Range2 Aggregate summary.....	38

Table 31 – AnnotationCount Aggregate summary	39
Table 32 – Count Aggregate summary.....	40
Table 33 – DurationInStateZero Aggregate summary	40
Table 34 – DurationInStateNonZero Aggregate Summary.....	41
Table 35 – NumberOfTransitions Aggregate summary	41
Table 36 – Start Aggregate summary	43
Table 37 – End Aggregate summary.....	43
Table 38 – Delta Aggregate summary.....	44
Table 39 – StartBound Aggregate summary	44
Table 40 – EndBound Aggregate summary.....	45
Table 41 – DeltaBounds Aggregate summary	45
Table 42 – DurationGood Aggregate summary	46
Table 43 – DurationBad Aggregate summary	47
Table 44 – PercentGood Aggregate summary	48
Table 45 – PercentBad Aggregate summary.....	48
Table 46 – WorstQuality Aggregate summary.....	49
Table 47 – WorstQuality2 Aggregate summary.....	50
Table 48 – StandardDeviationSample Aggregate summary	50
Table 49 – VarianceSample Aggregate summary	51
Table 50 – StandardDeviationPopulation Aggregate summary.....	52
Table 51 – VariancePopulation Aggregate summary.....	52

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.
- The main task of IEC technical committees is to prepare International Standards. However, a technical committee may propose the publication of a technical report when it has collected data of a different kind from that which is normally published as an International Standard, for example "state of the art".
- 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.
- The text of this technical report is based on the following documents:

Enquiry draft	Report on voting
65E/XX/DTR	65E/XX/RVC

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

This third edition cancels and replaces the second edition of IEC 62541, published in 2015.

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

- a) No technical changes but numerous clarifications. Also some corrections to the examples.

52

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

54 Throughout this document and the referenced other Parts of the series, certain document
55 conventions are used:

56 Italics are used to denote a defined term or definition that appears in the “Terms and definition”
57 clause in one of the parts of the series.

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

60 The italicized terms and names are also often written in camel-case (the practice of writing
61 compound words or phrases in which the elements are joined without spaces, with each element's
62 initial letter capitalized within the compound). For example the defined term is AddressSpace instead
63 of Address Space. This makes it easier to understand that there is a single definition for
64 AddressSpace, not separate definitions for Address and Space.

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

67 The committee has decided that the contents of this publication will remain unchanged until the
68 stability date indicated on the IEC web site under "<http://webstore.iec.ch>" in the data related to the
69 specific publication. At this date, the publication will be

- 70 • reconfirmed,
- 71 • withdrawn,
- 72 • replaced by a revised edition, or
- 73 • amended.

74

75 The National Committees are requested to note that for this publication the stability date is 2021.

76 THIS TEXT IS INCLUDED FOR THE INFORMATION OF THE NATIONAL COMMITTEES AND WILL BE DELETED AT
77 THE PUBLICATION STAGE.

78 A bilingual version of this publication may be issued at a later date.

79

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.

80

81

82

OPC Unified Architecture Specification

Part 13: Aggregates

1 Scope

This specification 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.

IEC TR 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 abbreviations

3.1 Terms and definitions

For the purposes of this document, the terms and definitions given in IEC TR 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 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 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**

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.

173 The calculation can be expressed with the following formula:

$$174 \quad V_{\text{bound}} = (T_{\text{bound}} - T_{\text{before}}) \times (V_{\text{after}} - V_{\text{before}}) / (T_{\text{after}} - T_{\text{before}}) + V_{\text{before}}$$

175 where V_x is a value at 'x' and T_x is the timestamp associated with V_x .

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

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

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

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

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

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

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

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

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

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

203 3.1.9

204 simple bounding values

205 *bounding values* determined by a calculation using the nearest value

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

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

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

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

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

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

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

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

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 <i>BaseObjectType</i> 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

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

272 *Variable* when assigning *StatusCodes*. For details on which *Aggregates* use the
273 *PercentDataBad Variable*, see the definition of each *Aggregate*. The default value is 100.

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

279 The *PercentDataGood* and *PercentDataBad* shall follow the following relationship
280 $PercentDataGood \geq (100 - PercentDataBad)$. If they are equal the result of the
281 *PercentDataGood* calculation is used. If the values entered for *PercentDataGood* and
282 *PercentDataBad* do not result in a valid calculation (e.g. Bad = 80; Good = 0) the result will
283 have a *StatusCode* of *Bad_AggregateInvalidInputs*. The *StatusCode*
284 *Bad_AggregateInvalidInputs* will be returned if the value of *PercentDataGood* or
285 *PercentDataBad* exceed 100.

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

292 4.2.2 AggregateFunction Object

293 4.2.2.1 General

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

300 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		

301
302 Each *ServerCapabilities* and *HistoricalServerCapabilities Object* shall reference an
303 *AggregateFunction Object*. In addition, each *HistoricalConfiguration Object* belonging to a
304 *HistoricalDataNode* may reference an *AggregateFunction Object* using the *HasComponent*
305 *Reference*.

306 4.2.2.2 AggregateFunctionType

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

309 Table 4 – AggregateFunctionType Definition

Attribute	Value				
BrowseName	AggregateFunctionType				
IsAbstract	False				
References	Node Class	BrowseName	DataType	Type Definition	Mod. Rule
Subtype of the <i>BaseObjectType</i> defined in IEC 62541-5					

310

For the *AggregateFunctionType*, the *Description Attribute* (inherited from the *Base NodeClass*), is mandatory. The *Description Attribute* provides a localized description of the *Aggregate*.

Table 5 specifies the *BrowseName* and *Description Attributes* for the standard *Aggregate Objects*. The description is the localized “en” text. For other locales it shall be translated.

Table 5 – Standard AggregateType Nodes

BrowseName	Description
	Interpolation Aggregate
Interpolative	At the beginning of each interval, retrieve the calculated value from the data points on either side of the requested timestamp.
Average	Retrieve the average value of the data over the interval.
TimeAverage	Retrieve the time weighted average data over the interval using <i>Interpolated Bounding Values</i> .
TimeAverage2	Retrieve the time weighted average data over the interval using <i>Simple Bounding Values</i> .
Total	Retrieve the total (time integral) of the data over the interval using <i>Interpolated Bounding Values</i> .
Total2	Retrieve the total (time integral) of the data over the interval using <i>Simple Bounding Values</i> .
Minimum	Retrieve the minimum raw value in the interval with the timestamp of the start of the interval.
Maximum	Retrieve the maximum raw value in the interval with the timestamp of the start of the interval.
MinimumActualTime	Retrieve the minimum value in the interval and the timestamp of the minimum value.
MaximumActualTime	Retrieve the maximum value in the interval and the timestamp of the maximum value.
Range	Retrieve the difference between the minimum and maximum value over the interval.
Minimum2	Retrieve the minimum value in the interval including the <i>Simple Bounding Values</i> .
Maximum2	Retrieve the maximum value in the interval including the <i>Simple Bounding Values</i> .
MinimumActualTime2	Retrieve the minimum value with the actual timestamp including the <i>Simple Bounding Values</i> .
MaximumActualTime2	Retrieve the maximum value with the actual timestamp including the <i>Simple Bounding Values</i> .
Range2	Retrieve the difference between the Minimum2 and Maximum2 value over the interval.
Count	Retrieve the number of raw values over the interval.
DurationInStateZero	Retrieve the time a Boolean or numeric was in a zero state using <i>Simple Bounding Values</i> .
DurationInStateNonZero	Retrieve the time a Boolean or numeric was in a non-zero state using <i>Simple Bounding Values</i> .
NumberOfTransitions	Retrieve the number of changes between zero and non-zero that a Boolean or numeric value experienced in the interval.
Start	Retrieve the value at the beginning of the interval.
End	Retrieve the value at the end of the interval.
Delta	Retrieve the difference between the Start and End value in the interval.
StartBound	Retrieve the value at the beginning of the interval using <i>Simple Bounding Values</i> .
EndBound	Retrieve the value at the end of the interval using <i>Simple Bounding Values</i> .
DeltaBounds	Retrieve the difference between the StartBound and EndBound value in the interval using <i>Simple Bounding Values</i> .
DurationGood	Retrieve the total duration of time in the interval during which the data is Good.
DurationBad	Retrieve the total duration of time in the interval during which the data is Bad.
PercentGood	Retrieve the percentage of data (0 to 100) in the interval which has Good <i>StatusCode</i> .
PercentBad	Retrieve the percentage of data (0 to 100) in the interval which has Bad <i>StatusCode</i> .
WorstQuality	Retrieve the worst <i>StatusCode</i> of data in the interval.
WorstQuality2	Retrieve the worst <i>StatusCode</i> of data in the interval including the <i>Simple Bounding Values</i> .
AnnotationCount	Retrieve the number of <i>Annotations</i> in the interval (applies to Historical <i>Aggregates</i> only).
StandardDeviationSample	Retrieve the standard deviation for the interval for a sample of the population ($n-1$).
VarianceSample	Retrieve the variance for the interval as calculated by the StandardDeviationSample.
StandardDeviationPopulation	Retrieve the standard deviation for the interval for a complete population (n) which includes <i>Simple Bounding Values</i> .
VariancePopulation	Retrieve the variance for the interval as calculated by the StandardDeviationPopulation which includes <i>Simple Bounding Values</i> .

4.3 MonitoredItem AggregateFilter

4.3.1 MonitoredItem AggregateFilter Defaults

The default values used for *MonitoredItem Aggregates* are the same as those used for historical *Aggregates*. They are defined in 4.2.1.2. For additional information on *MonitoredItem AggregateFilter* see IEC 62541-4.

4.3.2 MonitoredItem Aggregates and Bounding Values

When calculating *MonitoredItem Aggregates* that require the use of *Bounding Values*, the bounds may not be known. The calculation is done in the same manner as a historical read with the Partial Bit set. The historian may wait some amount of time (normally no more than one processing interval) before calculating the interval to allow for any latency in data collection and reduce the use of the Partial Bit.

A historical read done after data collection and the data from the *MonitoredItem* over the same interval may not be the same.

4.4 Exposing Supported Functions and Capabilities

Figure 1 outlines a possible representation of *Aggregate* information in the *AddressSpace*. In this example, although the *Server* at the highest level may support *Aggregate* functionality for Interpolative, Total, Average, and others, *DataVariable X* only supports Interpolative, Total and Average, while *DataVariable Y* supports Average, a vendor defined *Aggregate* and other (unstated) *Aggregates*.

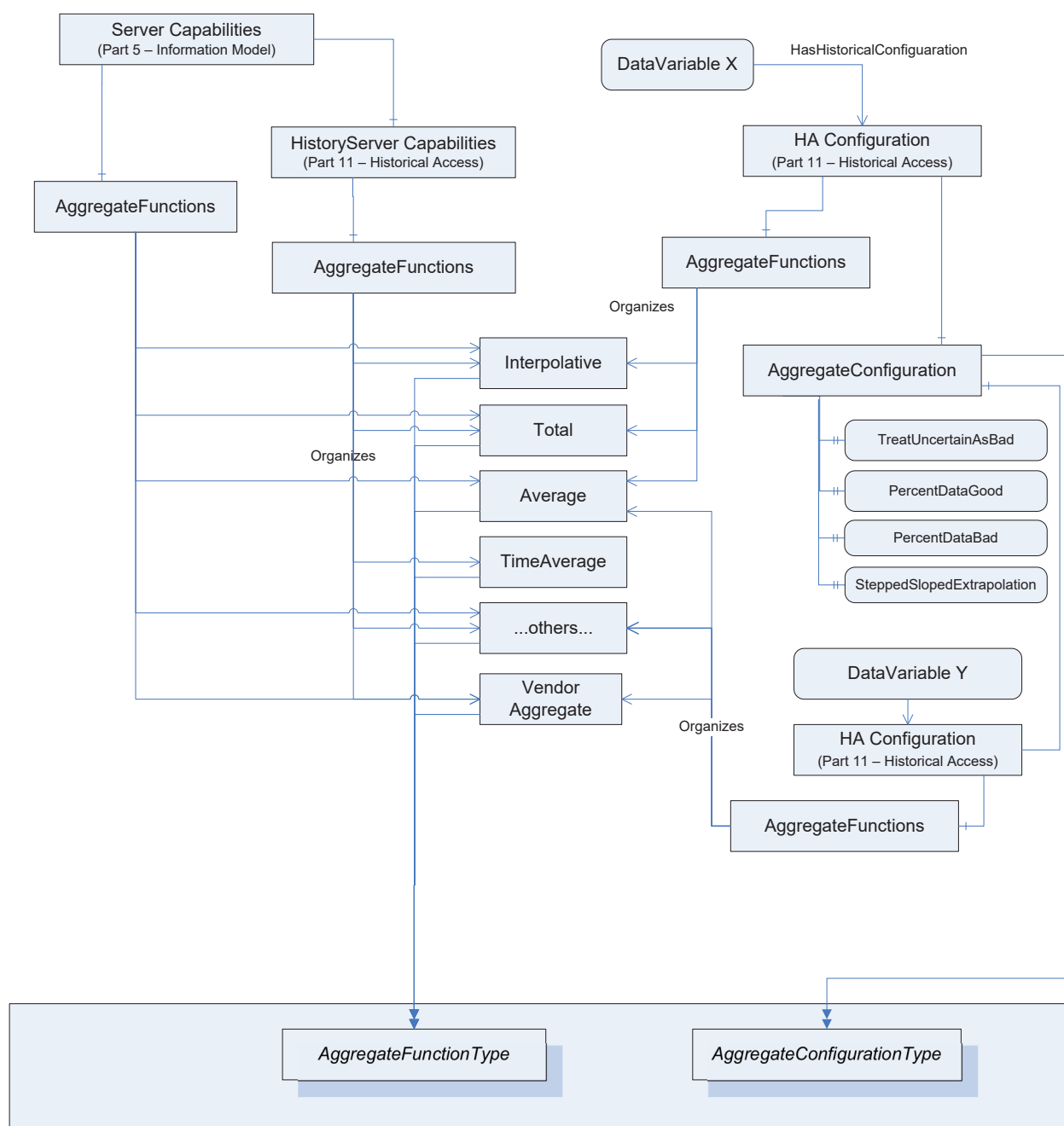


Figure 1 – Representation of Aggregate Configuration information in the AddressSpace

5 Aggregate specific usage of Services

5.1 General

IEC 62541-4 specifies all Services needed for OPC UA *Aggregates*. In particular:

- The Browse Service Set or Query Service Set to detect *Aggregates* and their configuration.
- The HistoryRead Service of the Attribute Service Set to read the aggregated history of the HistoricalNodes.
- The CreateMonitoredItems Service allows specifying a filter for each MonitoredItem to read aggregated data.

5.2 Aggregate data handling

5.2.1 Overview

The *HistoryRead* service defined in IEC 62541-4 can perform several different functions. The *historyReadDetails* parameter is an *Extensible Parameter* that specifies which function to perform. The *ReadProcessedDetails* structure is used to read aggregated data for *HistoricalDataNodes*.

The *CreateMonitoredItems* Service allows specifying a filter for each *MonitoredItem*. The *MonitoringFilter* is an extensible parameter whose structure depends on the type of item being monitored. The *AggregateFilter* structure is used to obtain aggregated data for a subscription.

5.2.2 ReadProcessedDetails structure overview

ReadProcessedDetails structure is formally detailed in IEC 62541-11. Table 6 outlines the components of the ReadProcessedDetails structure for the purposes of discussion in this document.

Table 6 – ReadProcessedDetails

Name	Description
ReadProcessedDetails	Specifies the details used to perform a "processed" history read.
startTime	Beginning of period to read.
endTime	End of period to read.
processingInterval	Interval between returned <i>Aggregate</i> values.
aggregateType[]	The <i>NodeIds</i> of the <i>AggregateFunction Objects</i> . <i>AggregateFunction Objects</i> indicate the list of <i>Aggregates</i> to be used when retrieving processed history.
aggregateConfiguration	<i>Aggregate</i> configuration structure.
useServerDefaults	If True the <i>Server's</i> default values are used and any values specified for the other parameters are ignored.
treatUncertainAsBad	See 4.2.1.2.
percentDataBad	See 4.2.1.2.
percentDataGood	See 4.2.1.2.
useSlopedExtrapolation	See 4.2.1.2.

5.2.3 AggregateFilter structure overview

The *AggregateFilter* defines the *Aggregate* function that should be used to calculate the values to be returned. The *AggregateFilter* is formally defined in IEC 62541-4. Table 7 outlines the components of the *AggregateFilter* structure for the purposes of discussion in this document.

Table 7 – AggregateFilter structure

Name	Description
AggregateFilter	
startTime	Beginning of period to calculate the <i>Aggregate</i> the first time.
aggregateType	The <i>NodeIds</i> of the <i>AggregateFunction Objects</i> that indicates the list of <i>Aggregates</i> to be used when retrieving processed data.
processingInterval	The period to be used to compute the <i>Aggregate</i> .
aggregateConfiguration	This parameter allows <i>Clients</i> to override the <i>Aggregate</i> configuration settings supplied by an <i>AggregateConfiguration Object</i> on a per monitored item basis.
useServerDefaults	If True the <i>Server's</i> default values are used and any values specified for the other parameters are ignored.
treatUncertainAsBad	See 4.2.1.2.
percentDataBad	See 4.2.1.2.
percentDataGood	See 4.2.1.2.
useSlopedExtrapolation	See 4.2.1.2.

5.3 Aggregates StatusCodes

5.3.1 Overview

Subclause 5.3 defines additional codes and rules that apply to the *StatusCode* when used for *Aggregates*.

The general structure of the *StatusCode* is specified in IEC 62541-4. It includes a set of common operational result codes which also apply to *Aggregates*.

5.3.2 Operation level result codes

In OPC UA *Aggregates* the *StatusCode* is used to indicate the conditions under which a value or *Event* was stored, and thereby can be used as an indicator of its usability. Due to the nature of aggregated data, additional information beyond the basic quality and call result code needs to be conveyed to the client. For example, whether or not the result was *Interpolated*, were all data inputs to a calculation of Good quality, etc.

In the following, Table 8 contains codes with *Bad* severity indicating a failure; Table 9 contains codes with *Uncertain* severity indicating that the value has been retrieved under sub-normal conditions. It is important to note, that these are the codes that are specific for OPC UA *Aggregates* and that they supplement the codes that apply to all types of data; they are therefore defined in IEC 62541-4, IEC 62541-8 and IEC 62541-11.

Table 8 – Bad operation level result codes

Symbolic Id	Description
Bad_AggregateListMismatch	The requested number of <i>Aggregates</i> does not match the requested number of <i>NodeIds</i> . When multiple <i>Aggregates</i> are requested, a corresponding <i>NodeId</i> is required for each <i>AggregateFunction</i> .
Bad_AggregateNotSupported	The requested <i>AggregateFunction</i> is not supported by the <i>Server</i> for the specified <i>Node</i> .
Bad_AggregateInvalidInputs	The <i>Aggregate</i> value could not be derived due to invalid data inputs, errors attempting to perform data conversions or similar situations.

Table 9 – Uncertain operation level result codes

Symbolic Id	Description
Uncertain_DataSubNormal	The value is derived from raw values and has less than the required number of Good values.

5.3.3 Aggregate Information Bits

5.3.3.1 General

These bits are set only when obtaining *Aggregate* data. They indicate where the data value came from and provide information that affects how the client uses the data value. Table 10 lists the bit settings which indicate the data location (i.e. is the value stored in the underlying data repository, or is the value the result of data aggregation). These bits are mutually exclusive.

Table 10 – Data location

StatusCode	Description
Raw	A <i>Raw data</i> value.
Calculated	A data value which was calculated.
Interpolated	A data value which was interpolated.

In the case where *Interpolated* data is requested, and there is an actual raw value for that timestamp, the *Server* should set the 'Raw' bit in the *StatusCode* of that value.

Table 11 lists the bit settings which indicate additional important information about the data values returned.

405

Table 11 – Additional information

StatusCode	Description
Partial	A calculated value that is not based on a complete interval. See 5.3.3.2.
Extra Data	If a <i>Server</i> chooses to set this bit, it indicates that a <i>Raw data</i> value supersedes other data at the same timestamp.
Multiple Values	Multiple values match the <i>Aggregate</i> criteria (i.e. multiple minimum values or multiple worst quality at different timestamps within the same <i>ProcessingInterval</i>).

406

407 The conditions under which these information bits are set depend on how the data has been
 408 requested and state of the underlying data repository.

409 5.3.3.2 Partial Information Bit

410 Partial bit is used to indicate that the interval is not a complete interval and that a client may
 411 receive a different value for the *Aggregate* if it re-fetches the interval with the same
 412 parameters.

413 The Partial Bit will be set in the following examples:

414 Assume for these examples the first stored point in the collection is 1:01:10 and the last
 415 stored point in the collection is 1:31:20. Older data may exist but is unavailable or offline at
 416 the time of the query. Newer data may be available but has not yet been stored in the history
 417 collection.

- 418 • The interval that overlaps the beginning of the history collection. If the start time is
 419 1:00:00 and end time is 1:10:00 and the interval is 2 minutes then the first interval
 420 would have a partial bit set since it has no data for the first 70 seconds. The partial bit
 421 will always be set for the first interval with data if the start time of the interval is before
 422 the first data value of the data collection. For intervals prior to the interval with a
 423 partial bit, these intervals will be flagged Bad_NoData.
- 424 • The interval that overlaps the latest point stored in the history collection. The last point
 425 in the collection is 1:31:20 and the historian was not shut down and is still running. A
 426 6-minute interval that started at 1:30:00 would have the partial bit set because the
 427 historian is expecting data, but just has not yet received anything. The partial bit will
 428 always be set for the last interval with data if the end time of the interval is after the
 429 last data value stored in the data collection. Intervals entirely after the interval with a
 430 partial bit will be flagged Bad_NoData. For those *Aggregates* with extrapolation, the
 431 partial bit may be set. See the *Aggregate* specific characteristics for more details.
- 432 • If the start/end time does not result in an even interval and there is additional data
 433 beyond the end time then the last interval will have a partial bit. If the start time is
 434 1:00:00 and end time is 1:20:00 and the interval is 6 minutes then the last interval is
 435 just 2 minutes long and will have the partial bit set. Extrapolation does not apply in this
 436 case.

437 The Partial Bit may be set with the Calculated bit when the Calculated bit is always set for the
 438 specific *Aggregate*.

439 5.4 Aggregate details

440 5.4.1 General

441 The purpose of 5.4 is to detail the requirements and behaviour for OPC UA *Servers*
 442 supporting *Aggregates*. The intent is to standardize the *Aggregates* so users can reliably
 443 predict the results of an *Aggregate* computation and understand its meaning. If users require
 444 custom functionality in the *Aggregates*, those *Aggregates* should be written as custom vendor
 445 defined *Aggregates*.

446 The standard *Aggregates* shall be as consistent as possible, meaning that each *Aggregate's*
 447 behaviour shall be similar to every other *Aggregate's* behaviour where input parameters, *Raw*
 448 *data*, and boundary conditions are similar. Where possible, the *Aggregates* should deal with
 449 input and preconditions in a similar manner.

Subclause 5.4 is divided up into two parts. Subclause 5.4.2 deals with *Aggregate* characteristics and behaviour that are common to all *Aggregates*. Subclause 5.4.3 deals with the characteristics and behaviour of *Aggregates* that are aggregate-specific.

5.4.2 Common characteristics

5.4.2.1 Description

Subclause 5.4.2 deals with *Aggregate* characteristics and behaviour that are common to all *Aggregates*.

5.4.2.2 Generating intervals

To read Historical *Aggregates*, OPC clients shall specify three time parameters:

- startTime (Start)
- endTime (End)
- *ProcessingInterval* (Int)

The OPC Server shall use these three parameters to generate a sequence of time intervals and then calculate an *Aggregate* for each interval. Subclause 5.4.2.2 specifies, given the three parameters, which time intervals are generated. Table 12 provides information on the intervals for each Start and End time combination. The range is defined to be $|End - Start|$.

All *Aggregates* return a timestamp of the start of the interval unless otherwise noted for the particular *Aggregate*.

Table 12 – History Aggregate interval information

Start/End Time	Interval	Resulting intervals
$Start = End$	$Int = \text{Anything}$	No intervals. Returns a <i>Bad_InvalidArgument StatusCode</i> , regardless of whether there is data at the specified time or not.
$Start < End$	$Int = 0$ or $Int \geq Range$	One interval, starting at <i>Start</i> and ending at <i>End</i> . Includes <i>Start</i> , excludes <i>End</i> , i.e., $[Start, End)$.
$Start < End$	$Int \neq 0$, $Int < Range$, <i>Int</i> divides <i>Range</i> evenly.	$Range/Int$ intervals. Intervals are $[Start, Start + Int)$, $[Start + Int, Start + 2 \times Int)$, ..., $[End - Int, End)$.
$Start < End$	$Int \neq 0$, $Int < Range$, <i>Int</i> does not divide <i>Range</i> evenly.	$\lceil Range/Int \rceil$ intervals. Intervals are $[Start, Start + Int)$, $[Start + Int, Start + 2 \times Int)$, ..., $[Start + (\lfloor Range/Int \rfloor - 1) \times Int, Start + \lfloor Range/Int \rfloor \times Int)$, $[Start + \lfloor Range/Int \rfloor \times Int, End)$. In other words, the last interval contains the “rest” that remains in the range after taking away $\lfloor Range/Int \rfloor$ intervals of size <i>Int</i> .
$Start > End$	$Int = 0$ or $Int \geq Range$	One interval, starting at <i>Start</i> and ending at <i>End</i> . Includes <i>Start</i> , excludes <i>End</i> , i.e., $[Start, End)$. ^a
$Start > End$	$Int \neq 0$, $Int < Range$, <i>Int</i> divides <i>Range</i> evenly.	$Range/Int$ intervals. Intervals are $[Start, Start - Int)$, $[Start - Int, Start - 2 \times Int)$, ..., $[End + Int, End)$. ^a
$Start > End$	$Int \neq 0$, $Int < Range$, <i>Int</i> does not divide <i>Range</i> evenly.	$\lceil Range/Int \rceil$ intervals. Intervals are $[Start, Start - Int)$, $[Start - Int, Start - 2 \times Int)$, ..., $[Start - (\lfloor Range/Int \rfloor - 1) \times Int, Start - \lfloor Range/Int \rfloor \times Int)$, $[Start - \lfloor Range/Int \rfloor \times Int, End)$. In other words, the last interval contains the “rest” that remains in the range after taking away $\lfloor Range/Int \rfloor$ intervals of size <i>Int</i> starting at <i>Start</i> . ^a
a In this case time is running backwards on the intervals.		

The calculation of all *Aggregates* when time flows backwards is the same as when time flows forwards with the exception that the ‘early time’ is excluded from the interval and the ‘late time’ is included. In most cases this means the value will be the same except the timestamps are shifted by one *ProcessingInterval*. E.g. when time flows forward the value at $T = n$ is the same as the value at $T = n + 1$ when time flows backward.

Note that when determining *Aggregates* with *MonitoredItem*, the interval is simply the *ProcessingInterval* parameter as defined in the *AggregateFilter* structure. See IEC 62541-4 for more details.

5.4.2.3 Data types

Table 13 outlines the valid *DataType* for each *Aggregate*. Some *Aggregates* are intended for numeric data types – i.e. integers or real/floating point numbers. Dates, strings, arrays, etc. are not supported. Other *Aggregates* are intended for digital data types – i.e. Boolean or enumerations. In addition some *Aggregates* may return results with a different *DataType* than those used to calculate the *Aggregate*. Table 13 also outlines the data type returned for each *Aggregate*.

485

Table 13 – Standard History Aggregate Data Type information

BrowseName	Valid Data Type	Result Data Type
Interpolation Aggregate		
Interpolative	Numeric	Raw Data Type
Data Averaging Aggregates		
Average	Numeric	Double
TimeAverage	Numeric	Double
TimeAverage2	Numeric	Double
Total	Numeric	Double
Total2	Numeric	Double
Data Variation Aggregates		
Minimum	Numeric	Raw data type
Maximum	Numeric	Raw data type
MinimumActualTime	Numeric	Raw data type
MaximumActualTime	Numeric	Raw data type
Range	Numeric	Raw data type
Minimum2	Numeric	Raw data type
Maximum2	Numeric	Raw data type
MinimumActualTime2	Numeric	Raw data type
MaximumActualTime2	Numeric	Raw data type
Range2	Numeric	Raw data type
Counting Aggregates		
AnnotationCount	All	Integer
Count	All	Integer
DurationInStateZero	Numeric or Boolean	Duration
DurationInStateNonZero	Numeric or Boolean	Duration
NumberOfTransitions	Numeric or Boolean	Integer
Time Aggregates		
Start	All	Raw data type
End	All	Raw data type
Delta	Numeric	Raw data type
StartBound	All	Raw data type
EndBound	All	Raw data type
DeltaBounds	Numeric	Raw data type
Data Quality Aggregates		
DurationGood	All	Duration
DurationBad	All	Duration
PercentGood	All	Double
PercentBad	All	Double
WorstQuality	All	Status Code
WorstQuality2	All	Status Code
Statistical Aggregates		
StandardDeviationSample	Numeric	Double
VarianceSample	Numeric	Double
StandardDeviationPopulation	Numeric	Double
VariancePopulation	Numeric	Double

486 **5.4.2.4 Time calculation issues**

487 The following issues may come up when calculating *Aggregates* that include time as part of
 488 the calculation.

- 489 • All *Aggregate* calculations include the *startTime* but exclude the *endTime*. However, it is
 490 sometimes necessary to return an *Interpolated* End Bound as the value for an *Interval* with
 491 a timestamp that is in the *Interval*. Servers are expected to use the time immediately
 492 before *endTime* where the time resolution of the Server determines the exact value.