

DIN EN 62541-13

ICS 35.200; 35.240.50

Einsprüche bis 2014-09-18

Entwurf

**OPC Unified Architecture –
Teil 13: Aggregation von Daten
(IEC 65E/379/CDV:2014);
Englische Fassung FprEN 62541-13:2014**

OPC Unified Architecture –
Part 13: Aggregates
(IEC 65E/379/CDV:2014);
English version FprEN 62541-13:2014

Architecture unifiée OPC –
Partie 13: Agrégats
(CEI 65E/379/CDV:2014);
Version anglaise FprEN 62541-13:2014

Anwendungswarnvermerk

Dieser Norm-Entwurf mit Erscheinungsdatum 2014-07-18 wird der Öffentlichkeit zur Prüfung und Stellungnahme vorgelegt.

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

Stellungnahmen werden erbeten

- vorzugsweise online im Norm-Entwurfs-Portal des DIN unter www.entwuerfe.din.de 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@din.de möglichst in Form einer Tabelle. Die Vorlage dieser Tabelle kann im Internet unter www.din.de/stellungnahme 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 im DIN und VDE (Hausanschrift: 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 100 Seiten

DKE Deutsche Kommission Elektrotechnik Elektronik Informationstechnik im DIN und VDE

Nationales Vorwort

Die Englische Fassung des europäischen Dokuments FprEN 62541-13:2014 „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/379/CDV:2014 „OPC Unified Architecture – Part 13: Aggregates“ unverändert in den Entwurf FprEN 62541-13:2014 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 DKE Deutsche Kommission Elektrotechnik Elektronik Informationstechnik im DIN und VDE (www.dke.de) zuständig.

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

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

Der Zusammenhang der zitierten Normen mit den entsprechenden Deutschen Normen 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 Normentwurf erfüllt.

Da sich die Benutzer des vorliegenden Normentwurfs der englischen Sprache als Fachsprache bedienen, wird die Englische Fassung der EN 62541-13 veröffentlicht. Zu deren Abschnitt 3, Symbole und Abkürzungen, wurde eine Übersetzung angefertigt und als informativer Nationaler Anhang NA des vorliegenden Normentwurfs hinzugefügt. Dieser Normentwurf steht nicht in unmittelbarem Zusammenhang mit Rechtsvorschriften und ist nicht als Sicherheitsnorm anzusehen.

Es wird auf die Möglichkeit hingewiesen, dass einige Elemente dieses Dokuments Patentrechte berühren können. ISO [und/oder] IEC sind 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.

Nationaler Anhang NA (informativ)

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 und IEC 62541-11 sowie die folgenden Begriffe.

3.1.1

VerarbeitungsIntervall

(en: **ProcessingInterval**)

Zeitspanne, für die abgeleitete Werte auf der Grundlage eines bestimmten Aggregates produziert werden

ANMERKUNG Die für *AuslesungVerarbeitet* festgelegte gesamte Zeitdomäne wird durch das *VerarbeitungsIntervall* geteilt. Wird beispielsweise eine Durchschnittsfunktion von 10 Minuten 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 Datenstichproben können historische Daten oder gepufferte Echtzeitdaten sein. Ein *interpolierter* Wert wird aus den Datenpunkten vor und nach dem abgefragten Zeitstempel berechnet.

3.1.3

EffektiveEndZeit

(en: **EffectiveEndTime**)

Zeit unmittelbar vor der *EndZeit*

ANMERKUNG Bei allen Aggregatberechnungen wird die *StartZeit* eingeschlossen, die *EndZeit* jedoch ausgeschlossen. Es ist jedoch manchmal notwendig, eine *interpolierte* Endgrenze als Wert für ein *Intervall* mit einem Zeitstempel, der innerhalb des *Intervalls* liegt, zurückzugeben. *Server* 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 Sekunde 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 Extrapolation ist mit dem Prozess der Interpolation, bei dem neue Punkte zwischen bekannten Punkten gebildet werden, vergleichbar, 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

GeneigteInterpolation

(en: **SlopedInterpolation**)

einfache lineare Interpolation

ANMERKUNG 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 Es sind die folgenden Tabellen mit Rohwerten und *interpolierten/extrapolierten* 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		
		<i>Geneigte Extrapolation</i>	<i>Schrittweise Extrapolation</i>
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 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*) werden die ersten gültigen Datenpunkte vor und nach dem Zeitstempel zur Schätzung der Grenze verwendet. Bei der zweiten Möglichkeit (den sogenannten *einfachen Grenzwerten*) werden die Datenpunkte unmittelbar vor und nach den Grenzzeitstempeln zur Schätzung der Grenze verwendet, auch wenn diese ungültig sind. In 3.1.80 und 3.1.9 werden diese beiden Ansätze ausführlicher beschrieben.

In allen Fällen wird das Flag *UnsicherAlsUngültigBehandeln* (siehe 4.2.1.1) verwendet, um festzulegen, ob unsichere Werte ungültig oder gültig sind.

Wenn kein Rohwert ermittelt wurde und ein gültiger Grenzwert vorhanden ist, werden die *Aggregate*-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 *Aggregate* 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 Spezifikation 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 *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 Formel ausgedrückt werden:

$$V_{\text{Grenze}} = (T_{\text{Grenze}} - T_{\text{vor}}) * (V_{\text{nach}} - V_{\text{vor}}) / (T_{\text{nach}} - T_{\text{vor}}) + V_{\text{vor}}$$

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

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

Die Dauer der Suche nach den gültigen Werten vor und nach dem Zeitstempel ist serverabhängig; 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* entspricht.

Interpolierte Grenzwerte, welche die *GeneigteInterpolation* 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 Formel entspricht der Formel, 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 *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 Formel entspricht der Formel, 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 für die Daten die Extrapolation angewendet werden, wenn der Server dies als dienlich einschätzt.

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 5.3.3.1].

Die Dauer der Suche nach Werten vor und nach dem Zeitstempel ist serverabhängig; 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

	Page
1 Scope	1
2 Normative references	1
3 Terms, definitions, and abbreviations	1
3.1 Terms and definitions	1
3.2 Abbreviations and symbols	4
4 Aggregate Information Model	5
4.1 General	5
4.2 Aggregate Objects	5
4.2.1 General	5
4.2.1.1 AggregateConfigurationType	5
4.2.2 AggregateFunction Object	6
4.2.2.1 AggregateFunctionType	7
4.3 MonitoredItem AggregateFilter	9
4.3.1 MonitoredItem AggregateFilter Defaults	9
4.3.2 MonitoredItem Aggregates and Bounding Values	9
4.4 Exposing Supported Functions and Capabilities	9
5 Aggregate specific usage of Services	11
5.1 General	11
5.2 Aggregate data handling	11
5.2.1 Overview	11
5.2.2 ReadProcessedDetails structure overview	11
5.2.3 AggregateFilter structure overview	12
5.3 Aggregates StatusCodes	12
5.3.1 Overview	12
5.3.2 Operation level result codes	12
5.3.3 Aggregate Information Bits	13
5.3.3.1 Partial Information Bit	13
5.4 Aggregate Details	14
5.4.1 General	14
5.4.2 Common characteristics	14
5.4.2.1 Description	14
5.4.2.2 Generating intervals	14
5.4.2.3 Data types	15
5.4.2.4 Time calculation issues	16
5.4.3 Specific Aggregated data handling	17
5.4.3.1 StatusCode calculation	17
5.4.4 Aggregate specific characteristics	20
5.4.4.1 Description	20
5.4.4.2 Interpolative	21
5.4.4.3 Average	22
5.4.4.4 TimeAverage	22
5.4.4.5 TimeAverage2	23
5.4.4.6 Total	24
5.4.4.7 Total2	25
5.4.4.8 Minimum	26
5.4.4.9 Maximum	26

50	5.4.4.10	MinimumActualTime	27
51	5.4.4.11	MaximumActualTime	28
52	5.4.4.12	Range	29
53	5.4.4.13	Minimum2.....	29
54	5.4.4.14	Maximum2.....	30
55	5.4.4.15	MinimumActualTime2	31
56	5.4.4.16	MaximumActualTime2	32
57	5.4.4.17	Range2	33
58	5.4.4.18	AnnotationCount.....	33
59	5.4.4.19	Count	34
60	5.4.4.20	DurationInStateZero	34
61	5.4.4.21	DurationInStateNonZero.....	35
62	5.4.4.22	NumberOfTransitions.....	36
63	5.4.4.23	Start.....	36
64	5.4.4.24	End	37
65	5.4.4.25	Delta	38
66	5.4.4.26	StartBound	38
67	5.4.4.27	EndBound	39
68	5.4.4.28	DeltaBounds.....	39
69	5.4.4.29	DurationGood	40
70	5.4.4.30	DurationBad	41
71	5.4.4.31	PercentGood	42
72	5.4.4.32	PercentBad	42
73	5.4.4.33	WorstQuality.....	43
74	5.4.4.34	WorstQuality2.....	44
75	5.4.4.35	StandardDeviationSample	45
76	5.4.4.36	VarianceSample	45
77	5.4.4.37	StandardDeviationPopulation	46
78	5.4.4.38	VariancePopulation	47
79	Annex A (informative): Aggregate Specific Examples – Historical Access		49
80	A.1	Historical Aggregate specific characteristics	49
81	A.1.1	Example <i>Aggregate</i> data – Historian 1	49
82	A.1.2	Example <i>Aggregate</i> data – Historian 2	50
83	A.1.3	Example <i>Aggregate</i> data – Historian 3	51
84	A.1.4	Example <i>Aggregate</i> data – Historian 4	52
85	A.2	Interpolative.....	53
86	A.2.1	Description	53
87	A.2.2	Interpolative data	53
88	A.3	Average.....	55
89	A.3.1	Description	55
90	A.3.2	Average data	55
91	A.4	TimeAverage	56
92	A.4.1	Description	56
93	A.4.2	TimeAverage data.....	56
94	A.5	TimeAverage2	57
95	A.5.1	Description	57
96	A.5.2	TimeAverage2 data.....	57
97	A.6	Total.....	59
98	A.6.1	Description	59

99	A.6.2	Total data	59
100	A.7	Total2	60
101	A.7.1	Description	60
102	A.7.2	Total2 data	60
103	A.8	Minimum	62
104	A.8.1	Description	62
105	A.8.2	Minimum data	62
106	A.9	Maximum	62
107	A.9.1	Description	62
108	A.9.2	Maximum data	62
109	A.10	MininumActualTime	63
110	A.10.1	Description	63
111	A.10.2	MinimumActualTime data	63
112	A.11	MaximumActualTime	64
113	A.11.1	Description	64
114	A.11.2	MaximumActualTime data	64
115	A.12	Range	65
116	A.12.1	Description	65
117	A.12.2	Range data	65
118	A.13	Minimum2	65
119	A.13.1	Description	65
120	A.13.2	Minimum2 data	65
121	A.14	Maximum2	66
122	A.14.1	Description	66
123	A.14.2	Maximum2 data	66
124	A.15	MinimumActualTime2	67
125	A.15.1	Description	67
126	A.15.2	MinimumActualTime2 data	67
127	A.16	MaximumActualTime2	67
128	A.16.1	Description	67
129	A.16.2	MaximumActualTime2 data	67
130	A.17	Range2	68
131	A.17.1	Description	68
132	A.17.2	Range2 data	68
133	A.18	AnnotationCount	69
134	A.18.1	Description	69
135	A.18.2	AnnotationCount data	69
136	A.19	Count	69
137	A.19.1	Description	69
138	A.19.2	Count data	69
139	A.20	DurationInStateZero	70
140	A.20.1	Description	70
141	A.20.2	DurationInStateZero data	70
142	A.21	DurationInStateNonZero	71
143	A.21.1	Description	71
144	A.21.2	DurationInStateNonZero data	71
145	A.22	NumberOfTransitions	71
146	A.22.1	Description	71
147	A.22.2	NumberOfTransitions data	71

148	A.23 Start	72
149	A.23.1 Description	72
150	A.23.2 Start data	72
151	A.24 End	73
152	A.24.1 Description	73
153	A.24.2 End data	73
154	A.25 StartBound	73
155	A.25.1 Description	73
156	A.25.2 StartBound data	73
157	A.26 EndBound	74
158	A.26.1 Description	74
159	A.26.2 EndBound data	74
160	A.27 Delta	75
161	A.27.1 Description	75
162	A.27.2 Delta data	75
163	A.28 DeltaBounds	75
164	A.28.1 Description	75
165	A.28.2 DeltaBounds data	76
166	A.29 DurationGood	76
167	A.29.1 Description	76
168	A.29.2 DurationGood data	76
169	A.30 DurationBad	77
170	A.30.1 Description	77
171	A.30.2 DurationBad data	77
172	A.31 PercentGood	78
173	A.31.1 Description	78
174	A.31.2 PercentGood data	78
175	A.32 PercentBad	79
176	A.32.1 Description	79
177	A.32.2 PercentBad data	79
178	A.33 WorstQuality	80
179	A.33.1 Description	80
180	A.33.2 WorstQuality data	80
181	A.34 WorstQuality2	81
182	A.34.1 Description	81
183	A.34.2 WorstQuality2 data	81
184	A.35 StandardDeviationSample	82
185	A.35.1 Description	82
186	A.35.2 StandardDeviationSample data	82
187	A.36 VarianceSample	82
188	A.36.1 Description	82
189	A.36.2 VarianceSample data	83
190	A.37 StandardDeviationPopulation	83
191	A.37.1 Description	83
192	A.37.2 StandardDeviationPopulation data	83
193	A.38 VariancePopulation	84
194	A.38.1 Description	84
195	A.38.2 VariancePopulation data	84
196	Annex B (informative): Revision Highlights	85

FIGURES

Figure 1 – Representation of Aggregate Configuration information in the AddressSpace.....	10
---	----

TABLES

Table 1 – Interpolation Examples	2
Table 2 – AggregateConfigurationType Definition.....	5
Table 3 – Aggregate Functions Definition	6
Table 4 – AggregateFunctionType Definition	7
Table 5 – Standard AggregateType Nodes	8
Table 6 – ReadProcessedDetails	11
Table 7 – AggregateFilter structure	12
Table 8 – Bad operation level result codes	12
Table 9 – Uncertain operation level result codes	12
Table 10 – Data Location	13
Table 11 – Additional Information	13
Table 12 – History Aggregate Interval Information	15
Table 13 – Standard History Aggregate Data Type Information	16
Table 14 – Aggregate Table Description.....	20
Table 15 – Interpolative Aggregate Summary	21
Table 16 – Average Aggregate Summary	22
Table 17 – TimeAverage Aggregate Summary.....	23
Table 18 – TimeAverage2 Aggregate Summary.....	23
Table 19 – Total Aggregate Summary	24
Table 20 – Total2 Aggregate Summary.....	25
Table 21 – Minimum Aggregate Summary	26
Table 22 – Maximum Aggregate Summary	27
Table 23 – MinimumActualTime Aggregate Summary	27
Table 24 – MaximumActualTime Aggregate Summary	28
Table 25 – Range Aggregate Summary	29
Table 26 – Minimum2 Aggregate Summary	29
Table 27 – Maximum2 Aggregate Summary	30
Table 28 – MinimumActualTime2 Aggregate Summary	31
Table 29 – MaximumActualTime2 Aggregate Summary	32
Table 30 – Range2 Aggregate Summary	33
Table 31 – AnnotationCount Aggregate Summary	33
Table 32 – Count Aggregate Summary	34
Table 33 – DurationInStateZero Aggregate Summary	35
Table 34 – DurationInStateNonZero Aggregate Summary.....	35
Table 35 – NumberOfTransitions Aggregate Summary	36
Table 36 – Start Aggregate Summary	37

Table 37 – End Aggregate Summary	37
Table 38 – Delta Aggregate Summary	38
Table 39 – StartBound Aggregate Summary	38
Table 40 – EndBound Aggregate Summary	39
Table 41 – DeltaBounds Aggregate Summary	40
Table 42 – DurationGood Aggregate Summary	40
Table 43 – DurationBad Aggregate Summary	41
Table 44 – PercentGood Aggregate Summary	42
Table 45 – PercentBad Aggregate Summary	43
Table 46 – WorstQuality Aggregate Summary	43
Table 47 – WorstQuality2 Aggregate Summary	44
Table 48 – StandardDeviationSample Aggregate Summary	45
Table 49 – VarianceSample Aggregate Summary	46
Table 50 – StandardDeviationPopulation Aggregate Summary	47
Table 51 – VariancePopulation Aggregate Summary	47

INTERNATIONAL ELECTROTECHNICAL COMMISSION

OPC Unified Architecture –

Part 13: Aggregates

FOREWORD

- 1) The International Electrotechnical Commission (IEC) is a worldwide organization for standardization comprising all national electrotechnical committees (IEC National Committees). The object of IEC is to promote international co-operation on all questions concerning standardization in the electrical and electronic fields. To this end and in addition to other activities, IEC publishes International Standards, Technical Specifications, Technical Reports, Publicly Available Specifications (PAS) and Guides (hereafter referred to as "IEC Publication(s)"). Their preparation is entrusted to technical committees; any IEC National Committee interested in the subject dealt with may participate in this preparatory work. International, governmental and non-governmental organizations liaising with the IEC also participate in this preparation. IEC collaborates closely with the International Organization for Standardization (ISO) in accordance with conditions determined by agreement between the two organizations.
- 2) The formal decisions or agreements of IEC on technical matters express, as nearly as possible, an international consensus of opinion on the relevant subjects since each technical committee has representation from all interested IEC National Committees.
- 3) IEC Publications have the form of recommendations for international use and are accepted by IEC National Committees in that sense. While all reasonable efforts are made to ensure that the technical content of IEC Publications is accurate, IEC cannot be held responsible for the way in which they are used or for any misinterpretation by any end user.
- 4) In order to promote international uniformity, IEC National Committees undertake to apply IEC Publications transparently to the maximum extent possible in their national and regional publications. Any divergence between any IEC Publication and the corresponding national or regional publication shall be clearly indicated in the latter.
- 5) IEC itself does not provide any attestation of conformity. Independent certification bodies provide conformity assessment services and, in some areas, access to IEC marks of conformity. IEC is not responsible for any services carried out by independent certification bodies.
- 6) All users should ensure that they have the latest edition of this publication.
- 7) No liability shall attach to IEC or its directors, employees, servants or agents including individual experts and members of its technical committees and IEC National Committees for any personal injury, property damage or other damage of any nature whatsoever, whether direct or indirect, or for costs (including legal fees) and expenses arising out of the publication, use of, or reliance upon, this IEC Publication or any other IEC Publications.
- 8) Attention is drawn to the normative references cited in this publication. Use of the referenced publications is indispensable for the correct application of this publication.
- 9) Attention is drawn to the possibility that some of the elements of this IEC Publication may be the subject of patent rights. IEC shall not be held responsible for identifying any or all such patent rights.

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

The text of this standard is based on the following documents:

FDIS	Report on voting
65E/XX/FDIS	65E/XX/RVD

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

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

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 publication will remain unchanged until the stability date indicated on the IEC web site under "<http://webstore.iec.ch>" in the data related to the specific publication. At this date, the publication will be

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

The National Committees are requested to note that for this publication the stability date is 2017.

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

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 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. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

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

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

IEC 62541-9, *OPC Unified Architecture – Part 9: Alarms & Conditions*

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

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

NOTE The total time domain specified for ReadProcessed is divided by the ProcessingInterval. For example, performing a 10 minute Average over the time range 12:00-12:30 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 interval *Bounds* are discussed in Clause 5.4.2.2.

3.1.2

Interpolated

Data that is calculated from data samples

NOTE 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 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 Clause 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 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 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 Consider the following table 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		
		<i>SlopedExtrapolation</i>	<i>SteppedExtrapolation</i>
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 If *Raw data* does not exist at the *startTime* and *endTime* a value must 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. Clauses 3.1.8 and 3.1.9 describe the two different approaches in more detail.

In all cases the *TreatUncertainAsBad* (see Clause 4.2.1.1) 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 Clause 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 Clause 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 Clause 3.1.9) is new in this specification and is important for applications which must 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 Interpolated Bounding Values using SlopedInterpolation are calculated as follows:

- If a non-Bad Raw value exists at the timestamp then it is the bounding value;
- Find the first non-Bad Raw value before the timestamp;
- Find the first non-Bad Raw value after the timestamp;
- Draw a line between before value and after value;
- Use point where the line crosses the timestamp as an estimate of the bounding value;

The calculation can be expressed with the following formula:

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

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

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

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

Interpolated Bounding Values using SlopedExtrapolation are calculated as follows:

- Find the first non-Bad Raw value before timestamp;
- Find the second non-Bad Raw value before timestamp;
- Draw a line between these two values;
- Extend the line to where it crosses the timestamp;
- Use the point where the line crosses the timestamp as an estimate of the bounding value;

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

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

Interpolated Bounding Values using SteppedInterpolation are calculated as follows:

- If a non-Bad Raw value exists at the timestamp then it is the bounding value;
- Find the first non-Bad Raw value before timestamp;
- Use the value as an estimate of the bounding value;

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

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

3.1.9

Simple Bounding Values

Bounding Values determined by a calculation using the nearest value

NOTE Simple Bounding Values using SlopedInterpolation are calculated as follows:

- If any Raw value exists at the timestamp then it is the bounding value;
- Find the first Raw value before timestamp;
- Find the first Raw value after timestamp;
- If the value after the timestamp is Bad then the before value is the bounding value;
- Draw a line between before value and after value;
- Use point where the line crosses the timestamp as an estimate of the bounding value;

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

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

Simple Bounding Values using SteppedInterpolation are calculated as follows:

- If any Raw value exists at the timestamp then it is the bounding value;
- Find the first Raw value before timestamp;
- If the value before timestamp is non-Bad then it is the bounding value;

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

If either bounding time of an interval is beyond the last data point then extrapolation may be used if the Server feels it is appropriate for the data.

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 Clause 5.3.3.1].

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 and symbols

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

The OPC UA Aggregates specification defines 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

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.

4.2.1.1 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 Clause 5.4.3 for details on using this *Variable* when assigning *StatusCodes*. For details on which *Aggregates* use the *PercentDataBad Variable*, see the definition of each *Aggregate*. The default value is 100.

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

The *PercentDataGood* and *PercentDataBad* must follow the following relationship $PercentDataGood \geq (100 - PercentDataBad)$. If they are equal the result of the *PercentDataGood* calculation is used. If the values entered for *PercentDataGood* and *PercentDataBad* do not result in a valid calculation (e.g. *Bad*=80; *Good*=0) the result will have a *StatusCode* of *Bad_AggregateInvalidInputs*.

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

4.2.2 AggregateFunction Object

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

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		

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

4.2.2.1 AggregateFunctionType

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

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					

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.
	Data Averaging and Summation Aggregates
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> .
	Data Variation Aggregates
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.
	Counting Aggregates
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.
	Time Aggregates
Start	Retrieve the value at the beginning of the interval using <i>Interpolated Bounding Values</i> .
End	Retrieve the value at the end of the interval using <i>Interpolated Bounding Values</i> .
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.
	Data Quality Aggregates
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> .
	Historical Annotation Aggregates
AnnotationCount	Retrieve the number of <i>Annotations</i> in the interval. (applies to Historical Aggregates only)
	Statistical Aggregates
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.

StandardDeviation Population	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> .

248

249 **4.3 MonitoredItem AggregateFilter**250 **4.3.1 MonitoredItem AggregateFilter Defaults**

251 The default values used for *MonitoredItem Aggregates* are the same as those used for historical
252 *Aggregates*. They are defined in clause 4.2.1.1. For additional information on *MonitoredItem*
253 *AggregateFilter* see IEC 62541-4.

254 **4.3.2 MonitoredItem Aggregates and Bounding Values**

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

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

261 **4.4 Exposing Supported Functions and Capabilities**

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

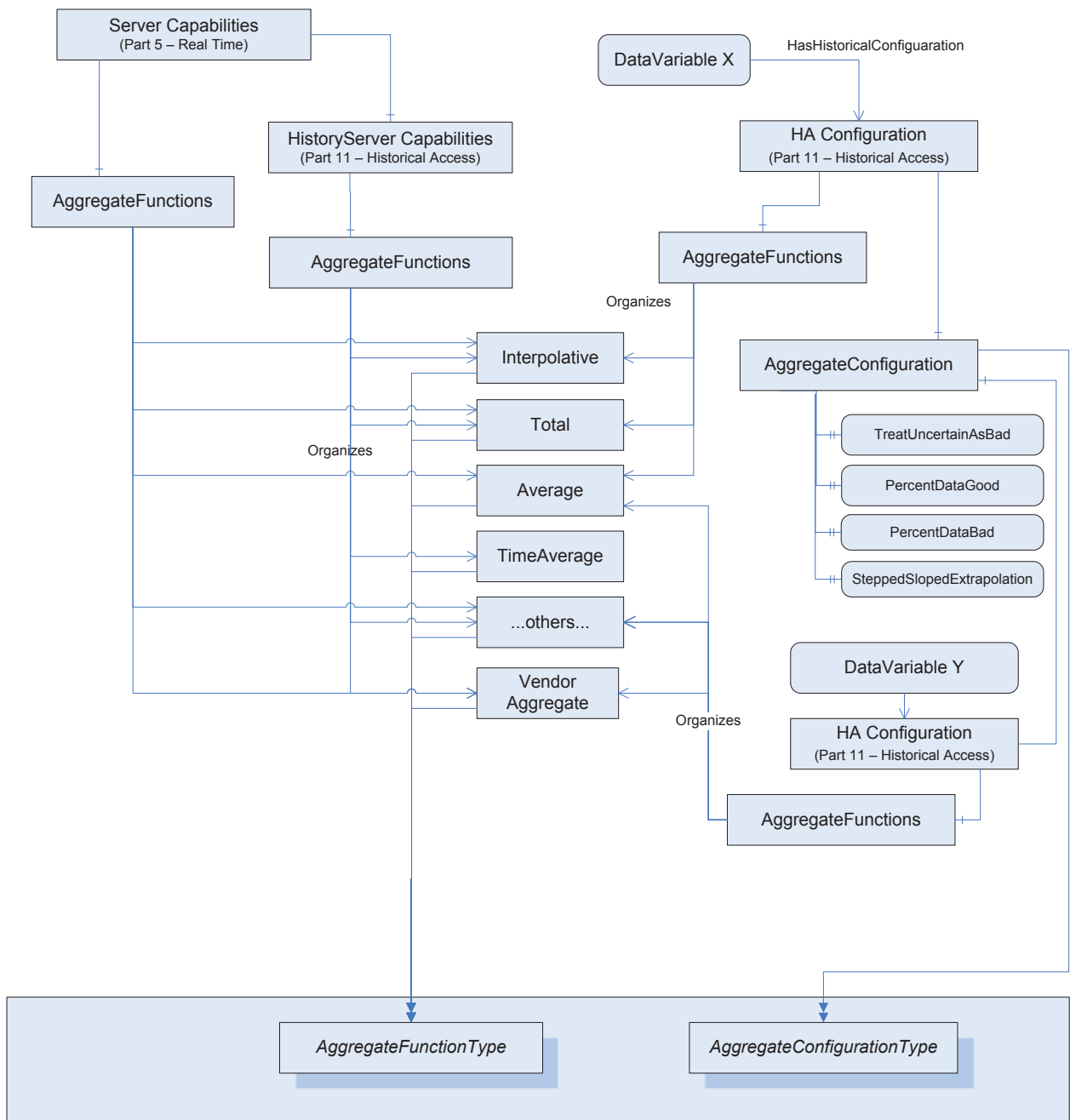


Figure 1 – Representation of Aggregate Configuration information in the AddressSpace.

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 aggregated history of *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	Aggregate configuration structure
useServerDefaults	If True the Server's default values are used and any values specified for the other parameters are ignored.
treatUncertainAsBad	See Clause 4.2.1.1
percentDataBad	See Clause 4.2.1.1
percentDataGood	See Clause 4.2.1.1
useSlopedExtrapolation	See Clause 4.2.1.1

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 aggregate 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 aggregate.
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 Server's default values are used and any values specified for the other parameters are ignored.
treatUncertainAsBad	See Clause 4.2.1.1
percentDataBad	See Clause 4.2.1.1
percentDataGood	See Clause 4.2.1.1
useSlopedExtrapolation	See Clause 4.2.1.1

5.3 Aggregates StatusCodes

5.3.1 Overview

This section 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 supplement the codes that apply to all types of data and 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 support by the server for the specified <i>Node</i> .
Bad_AggregateInvalidInputs	The aggregate 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 <u>Good</u> values.

314

315 **5.3.3 Aggregate Information Bits**

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

320

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.

321

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

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

326

327

Table 11 – Additional Information

StatusCode	Description
Partial	A calculated value that is not based on a complete interval. See Clause 5.3.3.1
Extra Data	If a Server 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 aggregate criteria (i.e. multiple minimum values or multiple worst quality at different timestamps within the same <i>ProcessingInterval</i>)

328

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

331 **5.3.3.1 Partial Information Bit**

332 Partial bit is used to indicate that the interval is not a complete interval and that a client may receive a
333 different value for the aggregate if it re-fetches the interval with the same parameters.

334

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

336

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

340

- 341 • The interval that overlaps the beginning of history collection. If the start time is 1:00:00 and end
342 time is 1:10:00 and the interval is 2 minutes then the first interval would have a partial bit set
343 since it has no data for the first 70 seconds. The partial bit will always be set for the first
344 interval with data if the start time of the interval is before the first data value of data collection.
345 For intervals prior to the interval with a partial bit, these intervals will be flagged *Bad_NoData*.
- 346 • The interval that overlaps the latest point stored in the history collection. The last point in the
347 collection is 1:31:20 and the historian was not shut down and is still running. A 6 minute interval
348 that started at 1:30:00 would have the partial bit set because the historian is expecting data, but
349 just has not yet received anything. The partial bit will always be set for the last interval with data
350 if the end time of the interval is after the last data value stored in the data collection. Intervals
351 entirely after the interval with a partial bit will be flagged *Bad_NoData*. For those aggregates

352 with extrapolation, the partial bit may be set. See the *Aggregate* specific characteristics for
353 more details.

- 354 • If the start/end time does not result in an even interval and there is additional data beyond the
355 end time then the last interval will have a partial. If the start time is 1:00:00 and end time is
356 1:20:00 and the interval is 6 minutes then the last interval is just 2 minute long and will have the
357 partial bit set. Extrapolation does not apply in this case.

358 The Partial Bit may be set with the Calculated bit when the Calculated bit is always set for the specific
359 *Aggregate*.
360

361 5.4 Aggregate Details

362 5.4.1 General

363 The purpose of this section is to detail the requirements and behaviour for OPC UA Servers supporting
364 *Aggregates*. The intent is to standardize the *Aggregates* so users can reliably predict the results of an
365 *Aggregate* computation and understand its meaning. If users require custom functionality in the
366 *Aggregates*, those *Aggregates* should be written as custom vendor defined *Aggregates*.

367 The standard *Aggregates* must be as consistent as possible, meaning that each *Aggregate*'s behaviour
368 must be similar to every other *Aggregate*'s behaviour where input parameters, *Raw data*, and boundary
369 conditions are similar. Where possible, the *Aggregates* should deal with input and preconditions in a
370 similar manner.

371 This section is divided up into two parts. The first sub section deals with *Aggregate* characteristics and
372 behaviour that are common to all *Aggregates*. The remaining sub sections deal with the characteristics
373 and behaviour of *Aggregates* that are aggregate-specific.

374 5.4.2 Common characteristics

375 5.4.2.1 Description

376 This subsection deals with *Aggregate* characteristics and behaviour that are common to all *Aggregates*.

377 5.4.2.2 Generating intervals

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

- 379 - *startTime* (Start)
380 - *endTime* (End)
381 - *ProcessingInterval* (Int)

382 The OPC server must use these three parameters to generate a sequence of time intervals and then
383 calculate an *Aggregate* for each interval. This section specifies, given the three parameters, which time
384 intervals are generated. Table 12 provides information on the intervals for each Start and End time
385 combination. Range is defined to be |End - Start|.

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

388

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 $[Start, Start + Int)$, $[Start + Int, Start + 2 * Int)$, ..., $[End - Int, End)$.
$Start < End$	$Int \neq 0$, $Int < Range$, <i>Int</i> does not divide <i>Range</i> evenly.	$\lceil Range/Int \rceil$ intervals. Intervals are $[Start, Start + Int)$, $[Start + Int, Start + 2 * Int)$, ..., $[Start + (\lfloor Range/Int \rfloor - 1) * Int, Start + \lfloor Range/Int \rfloor * Int)$, $[Start + \lfloor Range/Int \rfloor * Int, End)$. 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., $[Start, End)$ Note: in this case time is running backwards on the intervals
$Start > End$	$Int \neq 0$, $Int < Range$, <i>Int</i> divides <i>Range</i> evenly.	$Range/Int$ intervals. Intervals are $[Start, Start - Int)$, $[Start - Int, Start - 2 * Int)$, ..., $[End + Int, End)$. Note in this case time is running backwards on the intervals
$Start > End$	$Int \neq 0$, $Int < Range$, <i>Int</i> does not divide <i>Range</i> evenly.	$\lceil Range/Int \rceil$ intervals. Intervals are $[Start, Start - Int)$, $[Start - Int, Start - 2 * Int)$, ..., $[Start - (\lfloor Range/Int \rfloor - 1) * Int, Start - \lfloor Range/Int \rfloor * Int)$, $[Start - \lfloor Range/Int \rfloor * Int, End)$. 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> . Note: in this case time is running backwards on the intervals

389

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

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

397 5.4.2.3 Data types

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