

DIN EN IEC 62541-10

ICS 35.200; 35.240.50

EntwurfEinsprüche bis 2019-10-09
Vorgesehen als Ersatz für
DIN EN 62541-10:2015-11**OPC Unified Architecture –
Teil 10: Programme
(IEC 65E/611/CDV:2018);
Englische Fassung prEN IEC 62541-10:2018**OPC Unified Architecture –
Part 10: Programs
(IEC 65E/611/CDV:2018);
English version prEN IEC 62541-10:2018Architecture unifiée OPC –
Partie 10: Programmes
(IEC 65E/611/CDV:2018);
Version anglaise prEN IEC 62541-10:2018**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 51 Seiten

DKE Deutsche Kommission Elektrotechnik Elektronik Informationstechnik in DIN und VDE

Nationales Vorwort

Die Englische Fassung des europäischen Dokuments prEN IEC 62541-10:2018 „OPC Unified Architecture – Teil 10: Programme“ (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/611/CDV:2018 „OPC Unified Architecture – Part 10: Programs“ unverändert in den Entwurf prEN IEC 62541-10:2018 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 62541-10 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 62541-10:2015-11 wurden folgende Änderungen vorgenommen:

- a) ProgramType zu ProgramStateMachineType geändert. Diese entspricht KnotenEinrichtung (und somit den Implementierungen). In ProgrammDiagnoseDatenTyp wurde die Definition von LetzteEingabeArgumente und LetzteAusgabeArgumente geändert, und es wurden zwei zusätzliche Felder für Argumentwerte hinzugefügt. Außerdem wurde StatusErgebnis zu StatusCode geändert. Neue Version des Typs für ProgrammDiagnose2DatenTyp erstellt.
- b) Optionale Modellierungsregel für Programmsteuerungsmethoden zu OptionalerPlatzhalter geändert. Nach der Klarstellung in Teil 3 erlaubt dies Untertypen (oder Instanzen), so dass Argumente hinzugefügt werden können.

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* (en: *information model*) fest, welches im Zusammenhang mit *Programmen* steht. Dies schließt die Festlegung der *KnotenKlassen* (en: *NodeClasses*), genormter *Eigenschaften* (en: *Properties*), *Methoden* (en: *Methods*) und *Ereignisse* (en: *Events*) und zugehöriges Verhalten und Informationen für *Programme* ein.

Das vollständige Adressraummodell, einschließlich aller *KnotenKlassen* und *Attribute* (en: *Attributes*) ist in IEC 62541-3 festgelegt. Die *Dienste* (en: *Services*), wie beispielsweise jene zum Aufrufen der *Methoden*, welche zur Verwaltung von *Programmen* verwendet werden, sind in IEC 62541-4 festgelegt.

3 Begriffe und Konventionen

3.1 Begriffe

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

ISO und IEC stellen terminologische Datenbanken für die Anwendung in der Normung unter den folgenden Adressen bereit:

- IEC Electropedia: verfügbar unter <http://www.electropedia.org/>
- ISO Online Browsing Platform: verfügbar unter <http://www.iso.org/obp>

3.1.1

Funktion

(en: Function)

programmtechnische Aufgabenstellung, ausgeführt durch einen *Server* oder ein Gerät, üblicherweise durch Ausführung eines Maschinencodes

3.1.2

Endlicher Zustands Automat

(en: Finite State Machine)

Sequenz von Zuständen und gültigen Zustandsübergängen zusammen mit den Auslösern und Wirkungen dieser Zustandsübergänge, welche die Handlungen eines *Programms* im Sinne von diskreten Phasen definieren

3.1.3

ProgrammStatusMaschinenTyp

(en: ProgramStateMachineType)

Typdefinition eines *Programms* und ein Subtyp des *EndlicherZustandsAutomatTyp*

3.1.4

Programmsteuerungsmethode

(en: Program Control Method)

Methode mit spezifischer Semantik und angelegt zur Steuerung eines *Programms* durch Auslösen eines Zustandsübergangs

3.1.5

Programmaufruf

(en: Program Invocation)

eindeutige *Objekt*-Instanz eines *Programms*, die auf einem *Server* existiert

Anmerkung 1 zum Begriff: Ein *Programmaufruf* unterscheidet sich von anderen *Objekt*-Instanzen desselben *ProgrammStatusMaschinenTyps* durch den eindeutigen Suchpfad des *ObjektKnotens*.

– Leerseite –

CONTENTS

FIGURES	3
TABLES	3
FOREWORD	2
1 Scope	1
2 Normative references	1
3 Terms, definitions and conventions	1
3.1 Terms and definitions	1
3.2 Abbreviations	2
4 Concepts	2
4.1 General	2
4.2 Programs	3
4.2.1 Overview	3
4.2.2 Security considerations	3
4.2.3 Program Finite State Machine	4
4.2.4 Program states	4
4.2.5 State transitions	5
4.2.6 Program state transition stimuli	5
4.2.7 Program Control Methods	5
4.2.8 Program state transition effects	7
4.2.9 Program result data	7
4.2.10 Program lifetime	8
5 Model	8
5.1 General	8
5.2 ProgramStateMachineType	9
5.2.1 Overview	9
5.2.2 ProgramStateMachineType Properties	10
5.2.3 ProgramStateMachineType components	11
5.2.4 ProgramStateMachineType causes (Methods)	15
5.2.5 ProgramStateMachineType effects (Events)	16
5.2.6 AuditProgramTransitionEventType	19
5.2.7 FinalResultData	19
5.2.8 ProgramDiagnostic2 DataType	20
5.2.9 ProgramDiagnostic2Type VariableType	20
Annex A (informative) Program example	22
A.1 Overview	22
A.2 DomainDownload Program	22
A.2.1 General	22
A.2.2 DomainDownload states	22
A.2.3 DomainDownload transitions	23
A.2.4 DomainDownload Methods	24
A.2.5 DomainDownload Events	25
A.2.6 DomainDownload model	25

FIGURES

Figure 1 – Automation facility control	2
Figure 2 – Program illustration	3
Figure 3 – Program states and transitions	4
Figure 4 – Program Type	9
Figure 5 – Program FSM References	11
Figure 6 – ProgramStateMachineType causes and effects	15
Figure A.1 – Program example	22
Figure A.2 – DomainDownload state diagram	23
Figure A.3 – DomainDownloadType partial state model	29
Figure A.4 – Ready To Running model	32
Figure A.5 – Opening To Sending To Closing model	34
Figure A.6 – Running To Suspended model	35
Figure A.7 – Suspended To Running model	36
Figure A.8 – Running To Halted – Aborted model	37
Figure A.9 – Suspended To Aborted model	38
Figure A.10 – Running To Completed model	39
Figure A.11 – Sequence of operations	40

TABLES

Table 1 – Program Finite State Machine	4
Table 2 – Program states	5
Table 3 – Program state transitions	5
Table 4 – Program Control Methods	7
Table 5 – ProgramStateMachineType	10
Table 6 – Program states	12
Table 7 – Program transitions	13
Table 8 – ProgramStateMachineType causes	15
Table 9 – ProgramTransitionEventType	17
Table 10 – ProgramTransitionEvents	18
Table 11 – AuditProgramTransitionEventType	19
Table 12 – ProgramDiagnostic2DataType structure	20
Table 13 – ProgramDiagnostic2DataType definition	20
Table 14 – ProgramDiagnostic2Type VariableType	21
Table A.1 – DomainDownload states	24
Table A.2 – DomainDownload Type	25
Table A.3 – Transfer State Machine Type	26
Table A.4 – Transfer State Machine – states	27
Table A.5 – Finish State Machine Type	27
Table A.6 – Finish State Machine – states	28
Table A.7 – DomainDownload Type Property Attributes variable values	28
Table A.8 – Additional DomainDownload transition types	30

Table A.9 – Start Method additions	32
Table A.10 – StartArguments	33
Table A.11 – IntermediateResults Object	34
Table A.12 – Intermediate result data Variables	35
Table A.13 – FinalResultData	38

INTERNATIONAL ELECTROTECHNICAL COMMISSION

OPC UNIFIED ARCHITECTURE –

Part 10: Programs

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 62541-10 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 several clarifications and in addition the following significant technical changes with respect to the previous edition:

a) Changed ProgramType to ProgramStateMachineType. This is in line with the NodeSet (and thus implementations). In ProgramDiagnosticDataType - changed the definition of lastInputArguments and lastOutputArguments and added two additional fields for the argument values. Also changed StatusResult into StatusCode. Created new version of the type to ProgramDiagnostic2DataType.

b) Changed Optional modelling rule to OptionalPlaceHolder for Program control Methods. Following the clarification in Part 3 this now allows subtypes (or instances) to add arguments.

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

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

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

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

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

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

The committee has decided that the contents of this 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 2021.

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

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

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

OPC Unified Architecture Specification

Part 10: Programs

1 Scope

This specification is part of the overall OPC Unified Architecture (OPC UA) standard series and defines the *information model* associated with *Programs*. This includes the description of the *NodeClasses*, standard *Properties*, *Methods* and *Events* and associated behaviour and information for *Programs*.

The complete address space model including all *NodeClasses* and *Attributes* is specified in IEC 62541-3. The *Services* such as those used to invoke the *Methods* used to manage *Programs* are specified in IEC 62541-4.

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*

3 Terms, definitions and conventions

3.1 Terms and definitions

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

3.1.1

Function

programmatic task performed by a *Server* or device, usually accomplished by computer code execution

3.1.2

Finite State Machine

sequence of states and valid state transitions along with the causes and effects of those state transitions that define the actions of a *Program* in terms of discrete stages

3.1.3

ProgramStateMachineType

type definition of a *Program* and is a subtype of the *FiniteStateMachineType*

3.1.4

Program Control Method

Method having specific semantics designed for the control of a *Program* by causing a state transition

3.1.5

Program Invocation

unique *Object* instance of a *Program* existing on a *Server*

Note 1 to entry: A *Program Invocation* is distinguished from other *Object* instances of the same *ProgramStateMachineType* by the object node's unique browse path.

3.2 Abbreviations

DA Data Access

FSM Finite State Machine

HMI Human Machine Interfaces

UA Unified Architecture

4 Concepts

4.1 General

Integrated automation facilities manage their operations through the exchange of data and the coordinated invocation of system *Functions* as illustrated in Figure 1. *Services* are required to perform the data exchanges and to invoke the *Functions* that constitute system operation. These *Functions* may be invoked through Human Machine Interfaces, cell controllers, or other supervisory control and data acquisition type systems. OPC UA defines *Methods* and *Programs* as an interoperable way to advertise, discover, and request these *Functions*. They provide a normalizing mechanism for the semantic description, invocation, and result reporting of these *Functions*. Together *Methods* and *Programs* complement the other OPC UA *Services* and *ObjectTypes* to facilitate the operation of an automation environment using a client-server hierarchy.

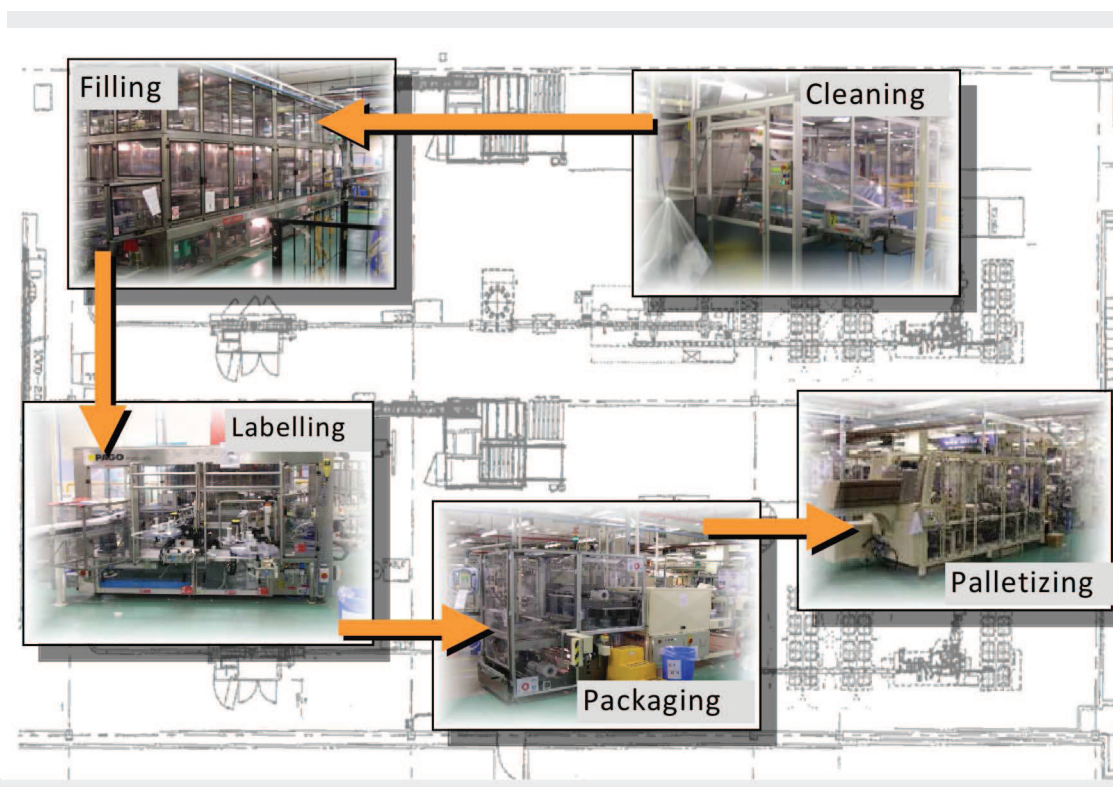


Figure 1 – Automation facility control

Methods and *Programs* model *Functions* typically have different scopes, behaviours, lifetimes, and complexities in OPC Servers and the underlying systems. These *Functions* are not normally characterized by the reading or writing of data which is accomplished with the OPC UA *Attribute* service set.

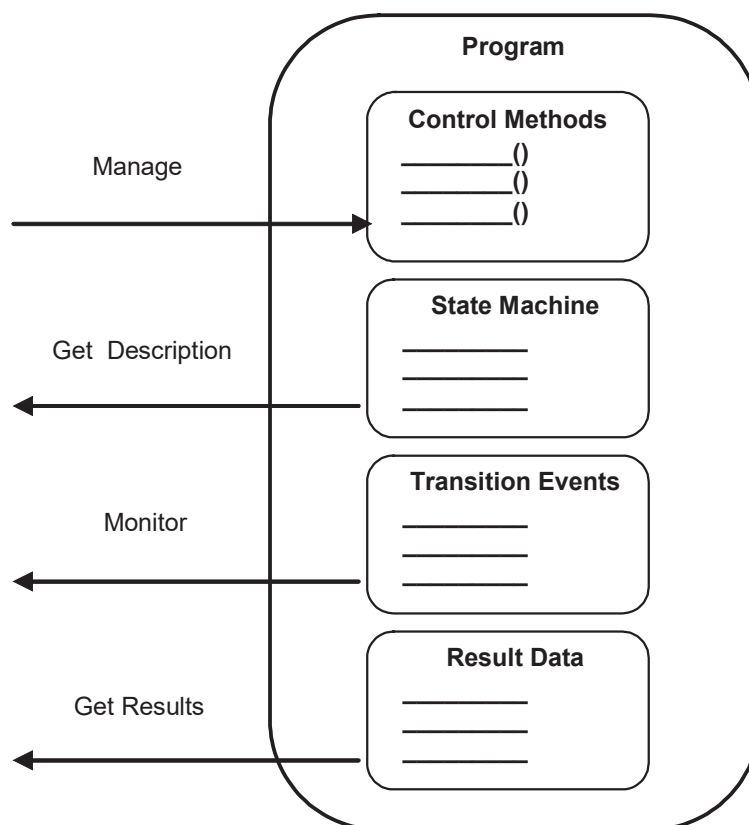
165 *Methods* represent basic *Functions* in the *Server* that can be invoked by a *Client*. *Programs*,
 166 by contrast, model more complex and stateful functionality in the system. For example, a
 167 method call may be used to perform a calculation or reset a counter. A *Program* is used to run
 168 and control a batch process, execute a machine tool part program, or manage a domain
 169 download. *Methods* and their invocation mechanism are described in IEC 62541-3 and
 170 IEC 62541-4.

171 This standard describes the extensions to, or specific use of, the core capabilities defined in
 172 IEC 62541-5 as required for *Programs*.

173 4.2 Programs

174 4.2.1 Overview

175 *Programs* are complex *Functions* in a *Server* or underlying system that can be invoked and
 176 managed by a *Client*. *Programs* can represent any level of functionality within a system or
 177 process in which *Client* control or intervention is required and progress monitoring is desired.
 178 Figure 2 illustrates the model.



179
180 **Figure 2 – Program illustration**

181 *Programs* are stateful and transition through a prescribed sequence of states as they execute.
 182 Their behaviour is defined by a *Program Finite State Machine (PFSM)*. The elements of the
 183 PFSM describe the phases of a *Program's* execution in terms of valid transitions between a
 184 set of states, the stimuli or causes of those transitions, and the resultant effects of the
 185 transitions.

186 4.2.2 Security considerations

187 Since *Programs* can be used to perform advanced control algorithms or other actions, their
 188 use should be restricted to personnel with appropriate access rights. It is recommended that
 189 *AuditUpdateMethodEvents* are generated to allow monitoring the number of running *Programs*
 190 in addition to their execution frequency.

4.2.3 Program Finite State Machine

The states, transitions, causes and effects that compose the *Program Finite State Machine* are listed in Table 1 and illustrated in Figure 3.

Table 1 – Program Finite State Machine

No.	Transition name	Cause	From state	To state	Effect
1	HaltedToReady	Reset Method	Halted	Ready	Report Transition 1 Event/Result
2	ReadyToRunning	Start Method	Ready	Running	Report Transition 2 Event/Result
3	RunningToHalted	Halt Method or Internal (Error)	Running	Halted	Report Transition 3 Event/Result
4	RunningToReady	Internal	Running	Ready	Report Transition 4 Event/Result
5	RunningToSuspended	Suspend Method	Running	Suspended	Report Transition 5 Event/Result
6	SuspendedToRunning	Resume Method	Suspended	Running	Report Transition 6 Event/Result
7	SuspendedToHalted	Halt Method	Suspended	Halted	Report Transition 7 Event/Result
8	SuspendedToReady	Internal	Suspended	Ready	Report Transition 8 Event/Result
9	ReadyToHalted	Halt Method	Ready	Halted	Report Transition 9 Event/Result

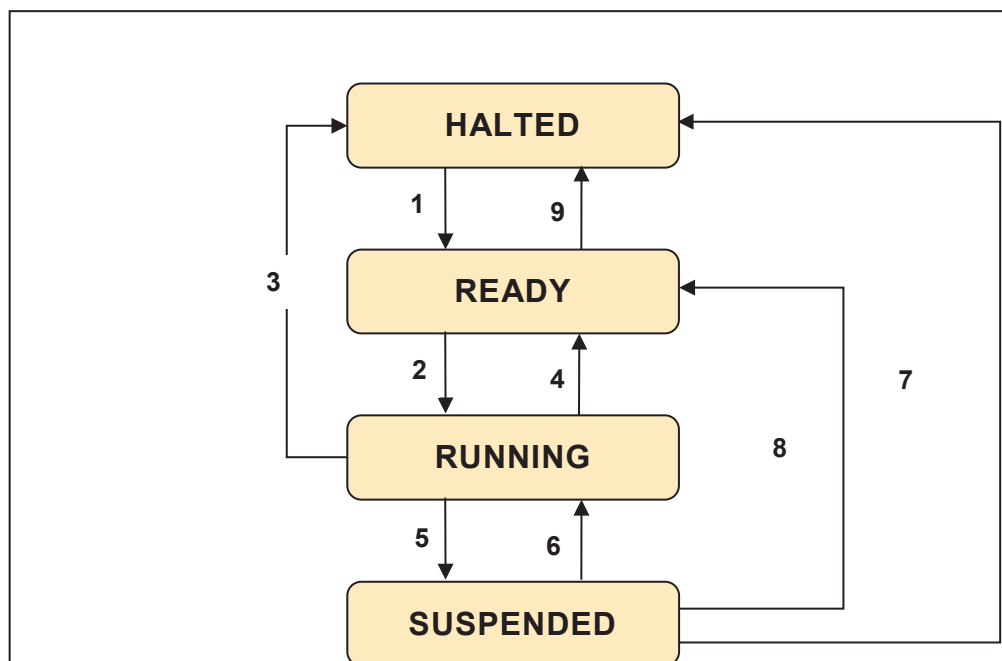


Figure 3 – Program states and transitions

4.2.4 Program states

A standard set of base states is defined for *Programs* as part of the *Program Finite State Machine*. These states represent the stages in which a *Program* can exist at an instance in time as viewed by a *Client*. This state is the *Program's* current state. All *Programs* shall support this base set. A *Program* may or may not require a *Client* action to cause the state to change. The states are formally defined in Table 2.

204

Table 2 – Program states

State	Description
Ready	The <i>Program</i> is properly initialized and may be started.
Running	The <i>Program</i> is executing making progress towards completion.
Suspended	The <i>Program</i> has been stopped prior to reaching a terminal state but may be resumed.
Halted	The <i>Program</i> is in a terminal or failed state, and it cannot be started or resumed without being reset.

205

206 The set of states defined to describe a *Program* can be expanded. *Program* sub states can be
 207 defined for the base states to provide more resolution of a process and to describe the cause
 208 and effect(s) of additional stimuli and transitions. Standards bodies and industry groups may
 209 extend the base *Program Finite State Model* to conform to various industry models. For
 210 example, the Halted state can include the sub states “Aborted” and “Completed” to indicate if
 211 the *Function* achieved a successful conclusion prior to the transition to Halted. Transitional
 212 states such as “Starting” or “Suspending” might also be extensions of the Running state, for
 213 example.

214 4.2.5 State transitions

215 A standard set of state transitions is defined for the *Program Finite State Machine*. These
 216 transitions define the valid changes to the *Program’s* current state in terms of an initial state
 217 and a resultant state. The transitions are formally defined in Table 3.

218

Table 3 – Program state transitions

Transition no.	Transition name	Initial state	Resultant state
1	HaltedToReady	Halted	Ready
2	ReadyToRunning	Ready	Running
3	RunningToHalted	Running	Halted
4	RunningToReady	Running	Ready
5	RunningToSuspended	Running	Suspended
6	SuspendedToRunning	Suspended	Running
7	SuspendedToHalted	Suspended	Halted
8	SuspendedToReady	Suspended	Ready
9	ReadyToHalted	Ready	Halted

219

220 4.2.6 Program state transition stimuli

221 The stimuli or causes for a *Program’s* state transitions can be internal to the *Server* or
 222 external. The completion of machining steps, the detection of an alarm condition, or the
 223 transmission of a data packet are examples of internal stimuli. *Methods* are an example of
 224 external stimuli. Standard *Methods* are defined which act as stimuli for the control of a
 225 *Program*.

226 4.2.7 Program Control Methods

227 *Clients* manage a *Program* by calling *Methods*. The *Methods* impact a *Program’s* behaviour by
 228 causing specified state transitions. The state transitions dictate the actions performed by the
 229 *Program*. This standard defines a set of standard *Program Control Methods*. These *Methods*
 230 provide sufficient means for a *Client* to run a *Program*.

231 Table 4 lists the set of defined *Program Control Methods*. Each *Method* causes transitions
 232 from specified states and shall be called when the *Program* is in one of those states.