IEC 62541-10

Edition 3.0 2020-07
REDLINE VERSION

INTERNATIONAL STANDARD

colour
inside

OPC unified architecture –
Part 10: Programs

IEC Central Office
3, rue de Varembé
CH-1211 Geneva 20
Switzerland

Tel.: +41 22 919 02 11
info@iec.ch
www.iec.ch

IEC 62541-10

Edition 3.0 2020-07
REDLINE VERSION

INTERNATIONAL STANDARD

colour
inside

OPC unified architecture –
Part 10: Programs

INTERNATIONAL
ELECTROTECHNICAL
COMMISSION

ICS 25.040.40; 35.100.05

ISBN 978-2-8322-8638-8

Warning! Make sure that you obtained this publication from an authorized distributor.

CONTENTS

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.

This redline version of the official IEC Standard allows the user to identify the changes made to the previous edition. A vertical bar appears in the margin wherever a change has been made. Additions are in green text, deletions are in strikethrough red text.

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.

This third edition cancels and replaces the second edition 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 IEC 62541-3, this now allows subtypes (or instances) to add arguments.

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

FDIS	Report on voting
65E/719/FDIS	65E/735/RVD

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

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

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

Italics are used to denote a defined term or definition that appears in Clause 3 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, with a few exceptions, written in camel-case (the practice of writing compound words or phrases in which the elements are joined without spaces, with each element's initial letter capitalized within the compound). For example the defined term is *AddressSpace* instead of *Address Space*. This makes it easier to understand that there is a single definition for *AddressSpace*, not separate definitions for Address and Space.

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

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

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

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

OPC UNIFIED ARCHITECTURE –

Part 10: Programs

1 Scope

This part of IEC 62541 ~~is part of the overall OPC Unified Architecture (OPC UA) standard series and~~ defines the *information model* associated with *Programs* in the OPC Unified Architecture. 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~~:2015~~, *OPC Unified Architecture – Part 3: Address Space Model*

IEC 62541-4~~:2015~~, *OPC Unified Architecture – Part 4: Services*

IEC 62541-5~~:2015~~, *OPC Unified Architecture – Part 5: Information Model*

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

3 Terms, definitions and ~~conventions~~ abbreviated terms

3.1 Terms and definitions

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

ISO and IEC maintain terminological databases for use in standardization at the following addresses:

- IEC Electropedia: available at http://www.electropedia.org/
- ISO Online browsing platform: available at http://www.iso.org/obp

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
~~**ProgramType**~~
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 ~~*ProgramType*~~ *ProgramStateMachineType* by the object node's unique browse path.

3.2 Abbreviated terms

DA	data access
FSM	finite state machine
HMI	human–machine interface
~~PCM~~	~~Program Control Method~~
~~PGM~~	~~Program~~
~~PI~~	~~Program Invocation~~
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.

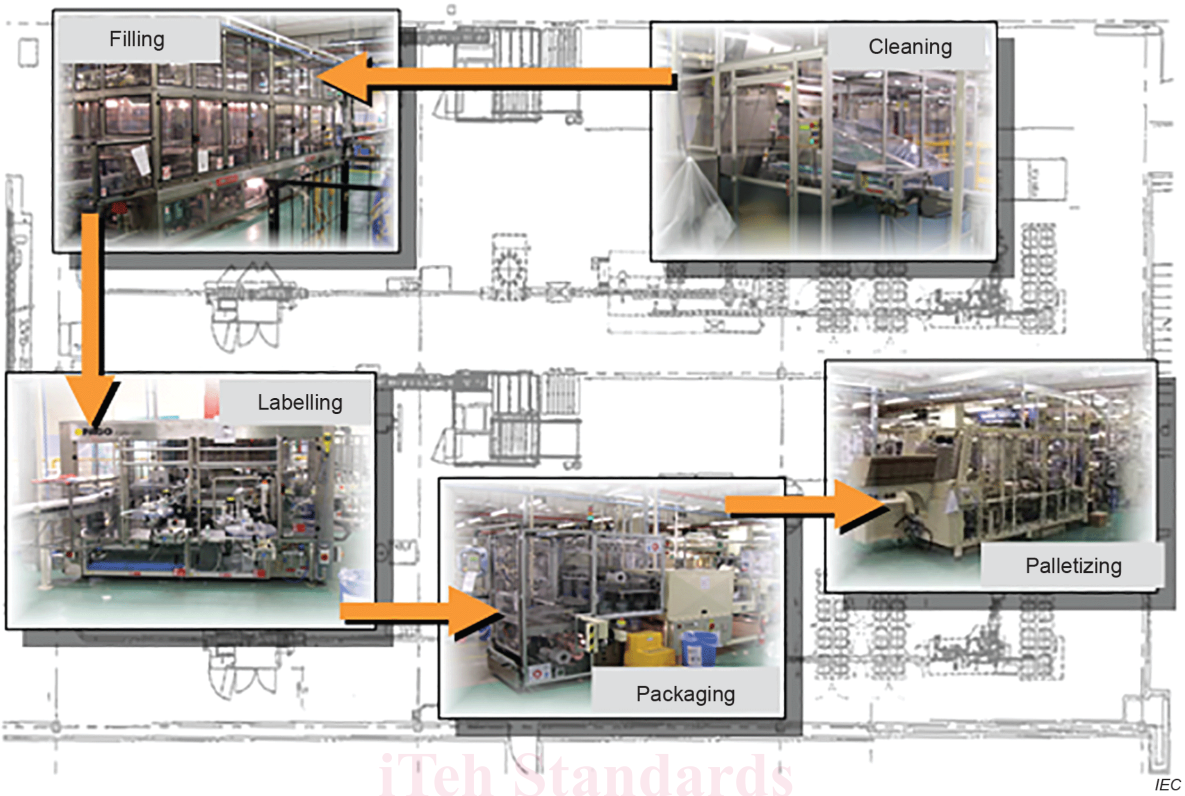

Figure 1 – Automation facility control

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.

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

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

4.2 Programs

4.2.1 Overview

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

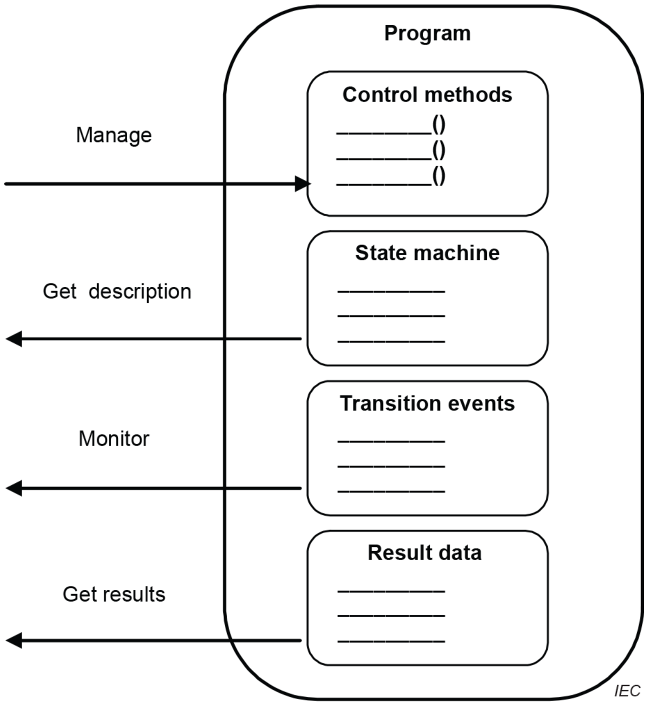

Figure 2 – Program illustration

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

4.2.2 Security considerations

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

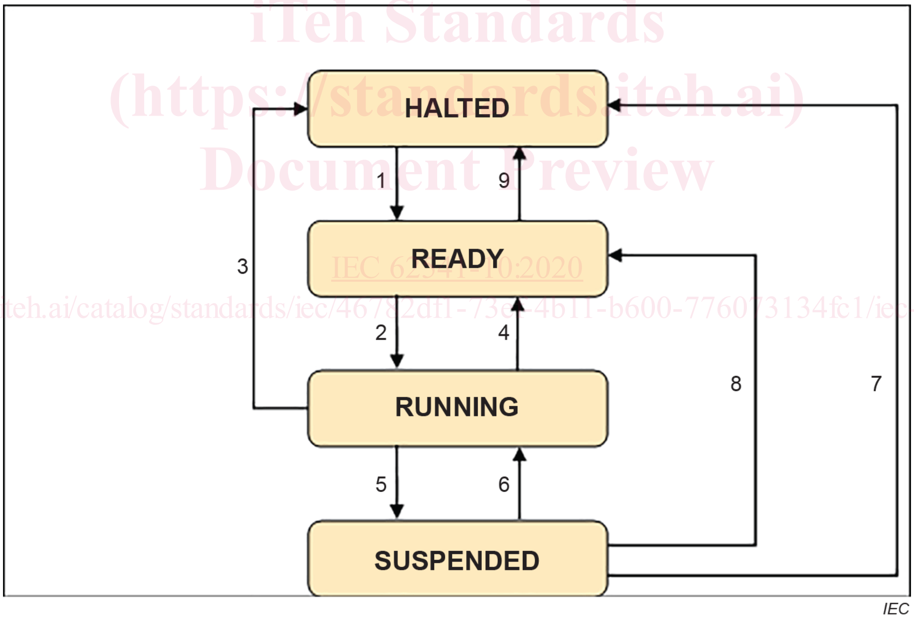

Figure 3 – Program states and transitions

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~~ instant 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.

Table 2 – Program states

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

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

4.2.5 State transitions

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

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

4.2.6 Program state transition stimuli

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

4.2.7 Program Control Methods

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