ISaGRAF
creates synergy

# IEC 61499 Frequently Asked Questions

## Why not use a general-purpose distributed object model, like DCOM or CORBA?

Implementation of the features specified for these information-technology models would be too expensive, and their performance would almost always be too slow, for use in a distributed real-time industrial-process measurement and control system (IPMCS).

Additionally, there is no standard, easily understood graphical model for representing the interconnections of events and data among these kinds of objects in distributed applications.

## Are data connections and event connections a kind of object?

The **declarations** of data and event connections contained in **library elements** can be considered objects to be manipulated by software tools, as shown in Annex C of IEC 61499-1. Additionally, data and event connections are regarded as **managed objects** within **resources**, as shown in Annex C of IEC 61499-1.

## Why are there no GLOBAL or EXTERNAL variables?

All variables are encapsulated. There is no guarantee that there will be an implicit global distribution mechanism available. When such mechanisms are available, they can always be mapped to service interface function blocks.

## How can "contained parameters" be accessed?

External access to internal variables of function blocks is contrary to the principles of good software design. Nonetheless, to accommodate exceptional cases and previous practice, the READ and WRITE services and associated access paths to internal variables are defined for management function blocks. However, this practice may substantially degrade system performance, reliability, maintainability and safety, especially if used instead of the standard PUBLISH/SUBSCRIBE or CLIENT/SERVER services for high-rate, periodic access to variables.

## Why use function blocks to model device or resource management applications?

The benefits of this approach are:

- A consistent model of all applications in the system, including management applications
- A consistent means of encapsulating and reusing all functions, including management functions
- Reuse of existing data types
- Use of existing, standardized means for the definition of required new data types and specification of management messages.

## 1.1 The event-driven model

### Why an event-driven model?

Any execution control strategy (cyclic, time scheduled, etc.) can be represented in terms of an event-driven model, but the converse is not necessarily true. IEC 61499 opts for the more general model in order to provide maximum flexibility and descriptive power to compliant standards and systems.

### How can a change in a data value generate an event?

`E_R_TRIG` and `E_F_TRIG` function block types, defined in Annex A of IEC 61499-1, propagate an input event when the value of a Boolean input rises or falls, respectively, between successive occurrences of the input event. Instances of this type may be combined with other function blocks to produce rising- and falling-edge triggers, threshold detection events, etc.

### What are event types? What are they used for?

An **event type** is an identifier associated with an event input or an event output of a function block type, assigned as part of the event input or output declaration, as described in IEC 61499-1-2.2.1.1. It can be used by software tools to assure that event connections are not improperly mixed, for instance to assure that an event output that is intended to be used for initialization is not connected to an event input that is intended to be used for alarm processing.

Event types cannot be detected by execution control charts (ECCs), which control the execution of algorithms in basic function block types as defined in IEC 61499-1-2.2, so the type of an event cannot be used to influence the processing of events in such function block types.

IEC 61499 does not define any standard event types other than the default type EVENT.

### How can this model accommodate sampled-data systems?

The major issues in sampled-data systems such as those used in motion, robotic and continuous process control are:

- how to achieve synchronized sampling of process or machine inputs;
- how to assure that all data has arrived in time for processing by control algorithms;
- how to assure that all outputs are present and ready for sampling before beginning the next cycle of sampling and execution.

Solutions to these problems typically require specialized communication and operating system services, which can be represented in terms of the IEC 61499 model by **service interface function blocks**. The inputs and outputs to be sampled can likewise be represented by service interface function blocks, while the algorithms to be performed will typically be encapsulated in basic function blocks. The relationships between the system services, input and output sampling, and algorithm execution can then be expressed as event connections and data connections.

### What happens if a critical event is lost by the communication subsystem?

This issue is common to all distributed control systems and its solutions are well known, e.g., via periodic communication, missing event detection and/or positive acknowledgment protocols. The IEC 61499 model provides for notification of abnormal operation via the `IND-`, `CNF-` and `INITO-` service primitives and `STATUS` output of service interface function blocks.

### Is there any way to distinguish between "process events" and "execution scheduling events"?

These events can be distinguished from each other by the event type mechanism. These can be used by software tools to show only the event types of interest and to restrict connections to be only among compatible event types.

### How can a function block respond to faults and exceptions?

In the IEC 61499 model, faults and exceptions are modeled as event outputs and associated data outputs of service interface function blocks. These outputs can be connected to appropriate event inputs and associated data inputs of any function blocks which must respond to the fault or exception, e.g., by changing their operational modes.

### Where can event handling function blocks (`E_CYCLE`, `E_RESTART`, etc.) be instantiated?

The standard does not restrict the context for instantiation of event handling function blocks or service interface function blocks in general. They can in principle be used wherever any other function block type may be instantiated, for instance as components of composite function blocks or as part of a resource configuration.

### Are there any mechanisms to prioritize execution of algorithms or events?

There is just some description of priority attributes for algorithms in Annex H of IEC 61499-1; however, this description is not normative.

### How can IEC 61131-3 tasks with priorities and cycling time be converted to IEC 61499?

The `E_CYCLE` function block is intended to be used mainly for the control of cyclic execution like the cyclic tasks of IEC 61131-3. See the previous question for a discussion of priorities.

## 1.2 Engineering methodologies

### How can I use IEC 61499 to design and implement state-machine control?

There are various formal and informal methodologies for the design of state-machine control systems. A general outline of these methods and how IEC 61499 models and software tools can be used is as follows:

1. Define the desired sequence of operations for the controlled machine or process, as well as possible abnormal sequences which may occur. This may be done informally in ordinary language, or using more formal notations such as Petri nets or IEC 61131-3 Sequential Function Charts (SFCs).
2. Define the actuators that are to be used to implement the desired behaviors, the sensors that are to be used to determine the actual state of the controlled machine or process, and their interfaces to the state-machine controller(s). IEC 61499 service interface function block types may be used for this purpose; such type definitions will typically be provided for this purpose by the supplier of the 61499-compliant sensor and actuator devices.
3. Model the behaviors of the machine or process in response to commands (events plus data) given to the actuators, and the resulting, observable time-dependent outputs (events plus data) from the sensors. This modeling may be done formally using notations such as Petri nets, or informally using simulation models (which may be implemented with appropriate IEC 61499 models).

4. Define the appropriate state-machine controllers, typically as the ECCs and algorithms of basic function block types. Methodologies for doing this step may be informal, based on engineering experience, or more formal, using for example Petri-net theory. Best of all is to reuse existing, proven state-machine controllers from an IEC 61499 function block library!
5. Validate the proposed state-machine controllers versus the model of the controlled machine or process. In order to catch possible design or specification errors, this would usually be done using simulation; in addition, more formal validation methods may be used if available.
6. Finally, replace the simulated sensor and actuator interfaces with the service interfaces to the actual machine or process to be controlled. This installation is normally to be done piecewise for testing purposes.

### What is an ECC? Why should I use it and when?

An ECC (Execution Control Chart) is a specialized state machine to enable multiple events to trigger multiple algorithms in a basic function block type, possibly dependent on some internal state. You should use it when you need maximum flexibility in algorithm selection and scheduling, or when you need a high-performance, event-driven state machine. Otherwise you can just use the mechanisms in Annex D of IEC 61499-1 for converting IEC 61131-3 function blocks to 61499-style blocks.

### Why can't a composite function block have internal variables?

Internal variables are not required in composite function blocks, since all possible sources of data are accounted for as input or output variables of the composite function block or of its component function blocks.

### Are extensible user-defined function blocks permitted?

IEC 61499 follows the usage of IEC 61131-3 and does not define a specific syntax for specification of extensible inputs and outputs of user-defined function blocks. The use of the ellipsis (...) and accompanying textual descriptions is considered adequate for the specification of extensible inputs and outputs of standard and service interface function block types.

### Can alternative graphical representations be used?

Software tools can use alternative graphical, textual or tabular representations, so long as accompanying documentation specifies an unambiguous mapping to the graphical elements and associated textual syntax defined in IEC 61499-1.

### How can "trend" and "view" objects be created?

In some process control terminologies, a view is a collection of data values from various sources, arranged for remote access. .This function is performed by standard IEC 61499 communication function blocks.

> NOTE This usage of the term "View" differs from the well-known Model/View/Controller (MVC) model for user interface applications.

A trend is a sequence of data values from the same source. The function of collecting trend data can be implemented as an algorithm of a basic function block, and remote access to the data so collected can be provided by standard communication function blocks. See subclauses 3.1.1 and 3.1.2 for further information.

### Why and when should I use the WITH construct?

When you are designing function block types, you use the WITH construct to specify an association between an event input and a set of input variables, or between an event output and a set of output variables.

IEC 61499-1 states that the WITH construct is used to determine:

- which input variables to sample when an event occurs at the associated event input of an instance of the type;
- which output events are used to indicate when values of associated output variables change.

In either case, it is expected that this information will be used by software tools to assist the user to ensure that:

- The data used by an algorithm in one function block is consistent with the data produced by an algorithm in another function block and delivered over one or more data connections associated with one or more event connections.
- The messages transmitted over communication connections between resources in a distributed application carry consistent data and events between the function block instances in the application in the way intended by the application designer.

### Are the "sequences" of service interface function blocks (SIFBs) only for documentation, or can they be used for programming purposes?

The use of service sequences for SIFBs was intended for documentation purposes, and especially to show a direct mapping for well-specified services which follow the ISO TR 8509 conventions. Even in documentation terms, however, they should be considered as imposing constraints on the externally observable operational sequences of the corresponding SIFBs. Software tools may, but are not required to, use these specifications to generate skeleton code for an implementation language such as IEC 61131-3 ST, C++ or Java.

## 1.3 Applications

### Why doesn't 61499 define applications as instances of application types?

In liaison with IEC SC65C/WG7, TC65/WG6 determined that an implementation-independent specification of an application type would be identical to a composite function block type; however, this view has now evolved to that of a **sub-application type**. The configuration of applications could then be carried out according to the following process:

1. Create and interconnect one or more instances of the sub-application types (and possibly additional function block types) representing the application.
2. Create instances of the service interface function block types representing the process interfaces of the application.
3. Create appropriate event connections and data connections between the process interface function blocks and the sub-applications representing the application.
4. Remove the encapsulation around the sub-applications, exposing their component function blocks (which may also be sub-applications) as distributable elements of the application.
5. Remove the encapsulation of all of the newly exposed sub-applications, repeating as necessary.
6. Distribute the application by allocating its function block instances to appropriate resources.

7. Create and configure appropriate instances of communication function block types to maintain the event and data flows of the application.

NOTE *Software tools* would typically provide means of automating many of the operations in this process, especially in steps 4, 5 and 7.

### Can an application contain "sub-applications"?

Yes. See the above description as well as subclause 2.4 of IEC 61499-1.

### Can an application interface with other applications?

Not directly, since there is no **application type** within which an interface could be defined. However, applications may communicate with each other by means of **communication service interface function blocks**. Also, **sub-applications** may interface with each other via event connections and data connections.

### How is the loading and starting of applications managed?

Software tools for this purpose will take as input a system configuration and generate sequences of commands to management function blocks to perform loading and starting of applications, either locally or remotely via communication function blocks. Details of this functionality will typically be contained in **compliance profiles** following the rules given in IEC 61499-4.