ARCHITECTURE DESCRIPTION

Aspect Object Architecture Overview

SUMMARY

This document provides an overview of ABB’s Aspect Object architecture. It is intended for those who need a high level understanding of how an Industrial IT system is organized, and as an introduction to the more detailed architecture definition provided in other documents.
## CONTENTS

1 **INTRODUCTION** .......................................................................................................................... 3
  1.1 Purpose and scope........................................................................................................................ 3
  1.2 Conventions and abbreviations used in this document .......................................................... 3
  1.3 Related documents..................................................................................................................... 3

2 **BACKGROUND** .......................................................................................................................... 4

3 **SYSTEM TOPOLOGY** .................................................................................................................. 5
  3.1 Overview.................................................................................................................................. 5
  3.2 Control and client/server networks ........................................................................................... 6
  3.3 Windows domains ..................................................................................................................... 7
  3.4 Fieldbus configurations............................................................................................................. 8

4 **THE ASPECT OBJECT MODEL** .............................................................................................. 9
  4.1 Aspect Objects....................................................................................................................... 9
  4.2 Object structures and plant modeling ....................................................................................... 12
  4.3 Aspect Object types................................................................................................................ 14

5 **THE ASPECT OBJECT ARCHITECTURE** ............................................................................. 15
  5.1 Overview................................................................................................................................ 15
  5.2 Client/server model.................................................................................................................. 16
  5.3 Afw OPC ................................................................................................................................ 17
  5.4 Security ................................................................................................................................... 21
  5.5 Aspect systems......................................................................................................................... 22
  5.6 Service and workplace deployment ......................................................................................... 24
  5.7 Example – Graphics................................................................................................................. 27

6 **INTEGRATION OF CONTROLLERS AND DEVICES** ............................................................. 28
  6.1 Overview................................................................................................................................ 28
  6.2 Connectivity packages.............................................................................................................. 28
  6.3 Control network devices.......................................................................................................... 30
  6.4 Fieldbus devices....................................................................................................................... 30
  6.5 Generic devices......................................................................................................................... 31
  6.6 Web devices............................................................................................................................. 31

7 **THE ASPECT INTEGRATOR PLATFORM** ............................................................................... 32
1 INTRODUCTION

1.1 Purpose and scope

This document is an introduction to and an overview of ABB’s Aspect Object architecture. It is intended for those who need a high level understanding of how the Industrial IT system is organized, and as an introduction to the more detailed architecture definition documents listed in section 1.3.

It should be noted that while this document describes the Aspect Object architecture, it does not provide any information as to what functionality is available in specific products at different times. For such information, it is necessary to refer to relevant product documentation for each product respectively.

1.2 Conventions and abbreviations used in this document

The following conventions are used in this document:

- **Shall or must** are used to indicate qualities or solutions that are requirements.
- **Should** is used to describe qualities or solutions that are desirable.
- **Could** is used to indicate possible solutions.
- **May** is used to describe qualities or solutions that are acceptable, but not necessarily desirable.

Special terms are written in *italic* font when first introduced. When later used, they are written in plain font.

[x] denotes a reference to a document listed in section 1.3.

1.3 Related documents

Different perspectives of the Aspect Object architecture are described in the following documents:

1. Aspect Object Architecture Overview, 3BSE022283 (this document)
2. Aspect Object Architecture, 3BSE012770
3. Aspect Object Model, 3BDS007014
5. User Interface Style Guide, <id>
6. Control Architecture, <id>
7. Batch architecture, <id>
8. Fieldbus – Scope and Architectural Overview, 3BDS008303
9. Engineering Concept, 3BDS007294
10. Basic Object Types, <id>
11. Information Management, <id>
12. On-line Help and User Documentation, <id>
13. Aspect Integrator Platform, Integration Guideline, 3BSE023960
14. Industrial IT Certification, Overview, 3BSE023905
   Industrial IT Certification, Document Structure, 3BSE025565
15. Industrial IT Dictionary, 3BSE025675.
2 BACKGROUND

Industrial IT is a concept for real-time integration of automation, information, and collaborative business systems across the enterprise. It supports application reuse for higher quality and lower engineering costs, and simpler operation, maintenance and training. It includes functionality ranging from field devices to business systems, focused on supporting decisions and improving customer productivity and asset utilization, from the first phases of design, through installation, commissioning, operation, and asset optimization.

![Diagram of Industrial IT concepts](image)

**Figure 1** Industrial IT seamlessly links plant automation, asset optimization, and collaborative business processes

The Aspect Object architecture supports this concept by defining concepts and rules for the design of Industrial IT compliant products. The implementation of these concepts and rules make up a framework called the Aspect Framework. This framework is packaged together with basic system functions, development tools, and documentation, in the Aspect Integrator Platform.

There are several important requirements that the Aspect Object architecture is designed to meet:

- Make it possible to build a system that provides functionality for process automation, asset optimization and collaborative business processes, yet is easily understood and efficient to use
- Be optimized for the performance, predictability, reliability and availability that is required for high performance, real-time process control
- Provide security mechanisms that allow all operations to be access controlled and logged to comply with regulatory requirements
- Scale competitively from very small to very large
- Provide strong support for building reusable application solutions
- Allow software and equipment of different origin and with different internal implementation technologies to be integrated and work together as one consistent and integrated system
- Allow new functionality to be incrementally added to and integrated with the system without changing or recompiling existing software
- Efficiently support development by independent groups in a distributed organization.
3 SYSTEM TOPOLOGY

3.1 Overview

The Aspect Object architecture assumes a system of computers and devices that communicate with each other over different types of communication networks, as illustrated conceptually in Figure 2 below.

![Conceptual communication network configuration](image)

**Figure 2**  Conceptual communication network configuration

The **client/server network** is used for communication between **servers**, and between servers and **workplaces**. Servers run software that provides system functionality, workplaces run software that provides various forms of user interaction – for small systems, server and workplace software may be installed and run on the same machine.

The **control network** is a local area network (LAN) that is optimized for high performance and reliable communication with predictable response times in real time. It is used to connect **controllers** to the servers. Controllers are nodes that run control software.

**Fieldbuses** are used to interconnect field devices, such as I/O modules, smart sensors and actuators, variable speed drives, PLCs, or small single loop devices, and to connect these devices to the system, either via a controller, as indicated in Figure 2 above, or directly to a server.

Via a router, the client/server network can be connected to a plant intranet, and via a firewall to the Internet. For performance and integrity reasons, connection of foreign systems directly to the control and client/server networks should be avoided.
3.2 Control and client/server networks

The control and client/server networks are based on TCP/IP over Ethernet – the control network may also be extended with PPP (Internet Point-to-Point Protocol) segments for remote communication. The routing protocol that is used on the control and client/server networks is RNRP (Redundant Network Routing Protocol). This protocol supports redundant network configurations based on standard network components. All network components, including interface cards, can be duplicated. Switch over to the redundant network takes less than one second, with no loss or duplication of data.

For small and medium sized systems the control and client/server networks can be implemented on the same physical network:

![Diagram of control and client/server network](image)

**Figure 3 Combined control and client/server network**

The Aspect Object architecture scales from very small to very large systems. The smallest system configuration consists of only one node, with workplace and server functionality running on the same computer. Also control functionality can be run on the same machine, with I/O connected through a fieldbus.

![Diagram of small system configuration examples](image)

**Figure 4 Small system configuration examples**
In larger systems, several servers may be required to provide the same functionality, for example data access to controllers. Further, to ensure high availability it is often required that a server is redundant. Servers that in this way perform the same function with respect to the same partition of the system, e.g. the same set of controllers, form a server group. An overview of how server functionality is deployed to servers, and of how different workplaces are associated with different servers, is provided in section 5.6.

For very large systems, and for systems with several control networks or where different types of control networks are mixed, the client/server and control networks are preferably implemented as separate networks. In such configurations each control network is connected to one or several server groups:

![Server Group Diagram]

**Figure 5** Separate control networks connect to different server groups

### 3.3 Windows domains

The control and client/server networks define one Windows domain. If the system is connected to a plant intranet, this domain may be defined as a sub-domain in the overall network configuration. To ensure system integrity and security, passwords and security settings should still be maintained separately in this sub-domain.
3.4 Fieldbus configurations

Fieldbuses are connected to controllers as illustrated in Figure 2, or directly to servers as shown in Figure 6 below:

![Fieldbus configurations diagram](image)

**Figure 6 Fieldbus connected directly to a server**

Fieldbuses that use the TCP/IP/Ethernet protocol stack, such as Foundation Fieldbus High Speed Ethernet (FF HSE), can share the same medium as the control network, as shown in Figure 7 below.

![Fieldbus and control network diagram](image)

**Figure 7 Control network and FF HSE on same medium**

Fieldbuses may be redundant to the extent that this is supported by each fieldbus specification respectively.
4 THE ASPECT OBJECT MODEL

4.1 Aspect Objects

A central problem in plant operations as well as asset lifecycle management is that you need a way to keep together, manage, and have access to information about all different aspects of a great number of plant and process entities. These entities, or real world objects, are of many different kinds. They can be physical process objects, like a valve or motor, or more complex entities, like a reactor. They can be immaterial, like recipes, manufacturing orders, and customer accounts. Other examples are products, raw material, production batches, etc.

In a system that integrates automation, information, and collaborative business processes across the enterprise, each of these real world objects needs to be described from several different perspectives. Each perspective defines a piece of information, and a set of functions to create, access, and manipulate this information. We call this an aspect of the object.

![Figure 8](image1.png) Different aspects of an object

It is for many reasons necessary to be able to implement these aspects using many different applications, existing and new ones, from ABB, third parties and customers, now and in the future. It is desirable to be able to do this without changing the way these applications work internally. It is not reasonable to require that all different applications are aware of each other. Still, the applications must be able to cooperate to provide an integrated view and functionality of the object.

Aspect Objects provide a solution to this problem. In this concept, rather than creating one single object or data model in the system to represent the real world object, each aspect is modeled separately. An Aspect Object is a container that includes these independent models.

![Figure 9](image2.png) An Aspect Object is a container of aspects
Different kinds of Aspect Objects have different sets of aspects. Figure 10 below shows some examples: a device level object (e.g. a valve or a motor), a unit level object (e.g. a reaction vessel), and a manufacturing order object.

Aspects are implemented by software systems known as aspect systems, each of which stores, manages and presents its information in its own optimal way. The environment in which aspect systems are integrated is called the Aspect Framework. This framework provides mechanisms by which the aspects systems can cooperate and share data, to provide an integrated view and functionality of the object, and one time data entry.

Figure 10  Aspect examples

Figure 11  Different aspects are implemented by different aspect systems
An aspect system defines one or several **aspect types**, each representing the implementation of a certain aspect. Of each aspect type, one or more **aspect categories** can be defined as different specializations. For example, the aspect system Graphics implements the aspect types Graphic Display, Faceplate, and Display Element, where the aspect type Graphic Display includes the categories Overview, Group, and Object Display. See Figure 12:

![Diagram of Aspect System, Type, Category, Implementation, Specialization, Instance]

**Figure 12** Relations between aspects and aspect systems

Aspect systems present their functionality through COM objects. The internal implementation of an aspect system, however, is not restricted to COM; it can use any suitable implementation technology.

The Aspect Framework is based on Microsoft’s Component Object Model (COM). COM in itself does not provide the application independence that is required – to use the functions of a COM object a client application must know the identity of that object. In the Aspect Object architecture, aspect systems cooperate with other aspect systems without knowing which they are, or even how many they are. To make this possible, they do not interact directly with each other, but only with the Aspect Framework.

The framework includes an **Aspect Directory**, where all Aspect Objects and their aspects are registered, and also all aspect systems and the operations they support. To perform an operation on an Aspect Object, an application (e.g. an aspect system) invokes a framework interface for that operation. Using information in the Aspect Directory, the framework then invokes the corresponding interfaces of all aspect systems that are concerned by that particular operation for that particular object. Thus, to copy and paste an object, for example, all aspect systems that implement aspects that are defined for that object are involved and perform their part of the operation.

![Diagram of Aspect Framework, Client Application, Aspect Directory, Control, Process Graphics, Reports, etc.]

**Figure 13** The Aspect Framework provides application independence

The result is a system of integrated but independent software systems. It is open, so that new software systems that were not even anticipated from start can be added without changing or recompiling the ones that are already in place. Users work with concepts and entities (objects) that are familiar to them, i.e., valves, reactors, products, manufacturing orders, customers, etc., rather than with the implementation objects that realize the individual aspects.
4.2 Object structures and plant modeling

A very natural way to represent relations between different entities is to organize them in a structure. Depending on from which perspective we look at it, the same entity fits naturally in several different structures. For example, a certain piece of process equipment has a certain position in a functional structure depending on the functional breakdown of the plant. It is also physically placed somewhere, and thus it has a place in a location structure. The same piece of equipment may currently be allocated to a certain production batch, so it belongs in the batch structure. Because it is used to produce a certain product, it fits in a product structure. And so on.

Thus, there is a very obvious need to organize Aspect Objects in structures. It must be possible for one Aspect Object to be represented in several structures at the same time. For certain applications it must be possible to dynamically move an object between different positions in the same or a different structure, for example representing products moving through an assembly line, or production orders being allocated to different process equipment.

The concept of structures is central in the Aspect Object architecture. A number of structures are used to represent different types of information related to the system. All Aspect Object structures are hierarchical, i.e., the structures are defined by parent-child relations between Aspect Objects. Standards dealing with structural relationships between entities, such as IEC 61346 and S88, are supported.

The relation to a certain structure is represented as an aspect. By adding several structure aspects to an Aspect Object, the object can be placed in several structures, or even in several positions within the same structure. By dynamically adding, deleting, and changing structure aspects, the object can be inserted in, deleted from, or moved to different positions in various structures.

By means of Aspect Objects, and by arranging Objects in various structures, it is thus possible to effectively model many and various types of plants, equipment, products, processes and procedures.

![Figure 14 Multiple object structures](image-url)
The *Plant Explorer* is used for navigation in and between object structures. It is the default tool to create and manage Aspect Objects and to browse for information. It is based on the Windows Explorer metaphor, but instead of folders and files it deals with Aspect Objects and aspects.

**Figure 15  The Plant Explorer**

An Aspect Object can inherit aspects from its parent or a higher-level ancestor in each structure where it is placed. The child object inherits those aspects of the ancestor object that are marked “to-be-inherited”. This *structure inheritance* is different from traditional forms of inheritance in object oriented systems, in that it is not defined in terms of object class hierarchies, but in terms of structural relationships between object instances of unrelated classes. For example, a control valve may be part of a flow control loop for a mixing unit. The valve is placed as a child to the flow control loop in the functional structure, and thus inherits the security settings that apply to the mixing unit.

**Figure 16  Structure inheritance**
Structure inheritance is dynamic. When an Aspect Object is inserted in a particular structure, it inherits aspects from its ancestors. When it is deleted from a structure, it loses the aspects that were inherited through that structure. When it is again inserted in a different position in the same structure, or in a different structure, it inherits from its new ancestors.

4.3 Aspect Object types

An Aspect Object type defines certain characteristics that are common to several Aspect Object instances, such as a basic set of common aspects. This makes it possible to create and efficiently re-use standardized solutions to frequently recurring problems. For example, rather than building an Aspect Object from scratch for every valve in a plant, you can define a set of valve types, and then create all valve objects as instances of these types.

When an instance of an object type is created, the aspects that are defined in the object type are instantiated and associated with it. You can add aspects to a specific instance, or replace inherited aspects with instance specific aspects of the same type, but it is not possible to delete aspects that were inherited from the object type.

An object type has rules associated with it. These are either aspect rules that control what aspects can be associated with an instance of that type, or child rules that control what objects can be placed as children under an instance of that type, in a particular structure.

Object types can be created as specializations of other types. A specialized object type inherits aspects and other characteristics from the type it is derived from, the super type. For example, from a generic valve type that has a certain set of aspects, you can create specializations for block valves, control valves, etc., adding aspects and other characteristics that are specific to those types.

A simple object type describes one object; each time it is instantiated, precisely one object is created. A composite object type describes a set of objects organized in a structure, with a parent object and one or several child objects. The children in a composite object type are called formal instances, because they inherit from object types defined elsewhere in the Object Type Structure, but they are not actual instances. Only when a composite object is instantiated are actual instances created for these child objects. This is illustrated in Figure 17.

Object types are placed in the object type structure. They can be packaged, delivered, and installed as object type libraries.

![Figure 17 Aspect Object types](image-url)
5 THE ASPECT OBJECT ARCHITECTURE

5.1 Overview

The Aspect Object architecture divides the system topology described in chapter 3 into separate functional layers for control, server, and workplace functions respectively. The architecture deals with the workplace and server layers, and defines how the control layer is interfaced. Understanding these layers is essential to understanding the Aspect Object architecture.

Two main concepts are central to the Aspect Object architecture: the concept of Aspect Objects, and the concept of Afw Services. Based on these concepts, a framework is defined, with rules and conventions for how to add functionality to the server and workplace layers, and for how to connect functions in the control layer to the server layer. This framework is known as the Aspect Framework (Afw).

Functionality in the server and workplace layers is provided by software components referred to as system applications (or just applications). An important feature of Industrial IT is that information and functions are centered on Aspect Objects. To participate in Aspect Object operations, an application must present itself as an aspect system (or possibly as several aspect systems). In essence this means that the application provides COM objects called aspect system objects, which support certain framework-defined interfaces, through which the application can initiate and participate in common operations on objects and aspects.

Functionality in the control layer is connected to the server layer through connectivity components. A connectivity component provides Aspect Object types, access to real time data, and various forms of supporting functionality for different types of controllers and devices. See chapter 5.7.

---

**Figure 18** Functionality layers

**Figure 19** The Afw Aspect Framework
5.2 Client/server model

Applications are implemented as client applications or services. A service is an entity that provides a certain set of functions in the system. Services run in the server layer.

Client applications are applications that utilize the functionality provided by one or more services, e.g. to present some information to a user. A client application can also support thin clients by providing an interface to an Internet Information Server.

Figure 20 illustrates how client applications and services are deployed in a system.

Services must conform to the Afw Service model. An Afw Service is designed to run around the clock. It is partitioned into several service groups, each group handling part of the scope of the service (e.g. part of the object space). For redundancy each group can contain several service providers running on different servers. A service manager initiates and supervises the execution of Afw services.

An Afw Service normally provides a service handler. This is a COM object that a client includes and runs as an in-process object to access the service. Using information from the Aspect Directory, the service handler provides one uniform interface to the service, relieving the client from having to deal with issues such as how the service is partitioned, redundant service providers, etc.

![Diagram of client/server model and Afw Service model](image-url)
5.3 Afw OPC

The OPC (OLE for Process Control) specification is a non-proprietary technical specification that defines a set of standard interfaces based on Microsoft’s COM technology. The purpose of the OPC standard interface is to facilitate interoperability between automation and control applications, field devices, and business and office applications.

Traditionally, each application developer was required to write different custom interfaces to exchange data with different data sources. OPC eliminates this need by defining a common, high performance interface that permits this work to be done once, as an OPC server that can be used by many different applications. Different OPC servers are available for different brands of controller, fieldbuses, and other data sources.

OPC plays a vital role in the Aspect Object architecture – it is through OPC compliant interfaces that applications have access to real-time data, historical data, and alarm and event data from controllers, field devices, and other software applications.

The Afw OPC server concept allows many different OPC servers to be placed under a one common server that unifies access to data from different sources. This common server provides one set of OPC compliant interfaces, eliminating the need for client applications to know which OPC server to use for each data item.

![Diagram of Afw OPC server concept]

There are three Afw OPC servers: Afw OPC/DA for real-time data access; Afw OPC/HDA for access to historical data; and Afw OPC/AE for access to alarm and event data.

The Afw OPC Servers are designed as Afw Services, and thus consist of two parts:

- An Afw service handler that runs as an in-process COM object with the client application, and provides a unified OPC interface to all data sources in the system
- Server components that are accessed by the client component, and perform the actual access to data provided by different data sources.

For more convenient use with certain types of applications, such as calculation and report tools, real time data, historical data, and alarm and event data are also accessible through OLE/DB interfaces.
Afw OPC/DA

Aspect Objects can have properties. A property is a named data item that is related to an Aspect Object and owned by an aspect system. Through framework-defined interfaces, each aspect system supplies information about the properties it exposes for the aspects that it implements. A control aspect, for example, may expose properties such as VALUE, SETPOINT, and OUTPUT for a control function. The Name aspect provides the properties NAME and DESCRIPTION.

Data access is provided by the Afw OPC/DA Server. This function unifies a client’s access to all data sources, by splitting a request into separate requests for separate data sources, and merging the responses. Common data sources are controllers and other types of devices, but also applications and even entire systems can serve as data sources for the Afw OPC Server.

**Figure 23  The Afw OPC Server (arrows indicate direction of data flow)**

The server component consists of a connector, which includes common functionality, and an adaptor that provides the necessary adaptations for a particular data source. While the connector is a shared component provided by the Aspect Integrator Platform, the adaptor is specific for each type of data source.

Adaptors are easier to implement than OPC servers, because much of the required OPC functionality is provided by the client component and the connector. For data sources where OPC/DA servers are already available, the platform provides a common OPC/DA adaptor component.

Applications can publish Aspect Object properties by including a connector component and supporting a framework-defined interface for data subscription, as illustrated in Figure 23 above.
Afw OPC/HDA

The Afw OPC/HDA Server provides basic functionality for logging and access to historical data. It is modeled on the OPC History Data Access (OPC/HDA) specification.

The server part of the Afw OPC HDA Server performs two main functions. It collects and stores historical data in logs, and it provides access to down-stream history servers. A down-stream history server is a history server that provides its own collection and storage functions, but wants to make its data accessible through the same unified OPC/HDA access mechanism as the rest of the system.

A client’s access to historical data is handled by the Afw OPC/HDA Handler. This object runs as an in-process COM object with the client application, and provides a unified OPC/HDA interface to all data sources in the system. It finds out which service group is storing the requested historical data, and passes the request to that service group. A service provider in that group handles the request, including forwarding requests for data from downstream servers if needed, and returns the result to the client.

![Afw OPC/HDA Diagram](image)

**Figure 24 Afw OPC/HDA (arrows indicate direction of data flow)**

The server component includes collector components, which provide the necessary adaptations for different data sources. Collectors are easier to implement than OPC/HDA servers, because much of the required OPC functionality is provided by the client component and the History server. For data sources where OPC/DA or OPC/HDA servers are already available, the platform provides common OPC/DA and OPC/HDA collector components.
Afw OPC/AE

The Afw OPC/AE Server provides functions for logging and access to various forms of alarms and events. It is modeled on the OPC Alarm and Event (OPC/AE) specification.

The server part of the Afw OPC Alarm & Event Server collects and stores event notifications from many different OPC/AE servers. It stores the current status internally, and passes alarm and event messages on to the System Messages server, which stores all types of messages in the system in chronological order.

A client’s access to alarm and event data is handled by an in-process OPC/AE+ handler object. It broadcasts the request to all service groups. A service provider in the service group that supports the particular object handles the request and returns the result to the client. Besides standard OPC/AE interfaces, the OPC/AE+ handler supports an interface that allows the client to specify a time interval to access an extract of the logged messages, e.g. for presentation in a list.

![Diagram of Afw OPC/AE](image)

Figure 25  Afw OPC/AE (arrows indicate direction of data flow)
5.4 Security

Security in the Aspect Object architecture is based on Windows security, adding certain features and capabilities that allow products and systems built on the architecture to comply with relevant regulatory requirements.

Security controls a user’s authority to perform different operations on Aspect Objects, depending on several parameters:

- The user’s credentials, as provided by Windows.
- The node where the user is logged in. This makes it possible to give a user different authority depending on where he or she is located, e.g. close to the process equipment, in a control room, or at home accessing the system through Internet.
- The operation the user wants to perform.
- The Aspect Object that the user wants to perform the operation on.

These parameters are checked against a security descriptor provided by a security aspect held by the Aspect Object. A security descriptor includes an access control list, where each entry specifies an access mask (permission) and the users and/or user groups that are granted (or denied) access. The security descriptor also includes an audit control list, specifying which operations shall be logged for auditing purposes.

Security aspects can be inherited from a parent object through a structure (see structure inheritance in section 4.2). It is thus possible to define the security settings for all Aspect Objects in a certain area or other division of a plant, or for other collections of Aspect Objects, simply by defining security aspect with the appropriate settings for the top object in the corresponding substructure. Security aspects can actually be inherited through multiple structures; in such cases a system defined precedence order is used to determine which setting applies.

Afw defines a list of operations that can be performed on Aspect Objects. Additional operations can be defined for specific aspect types – this is part of the information that an aspect system registers with the Aspect Directory. Operations are mapped into permissions per aspect category. For each aspect type it is thus possible to create several categories with different security settings.

An end user of the system normally works only with permissions, configuring the security settings for each object or group of objects. It is only when you design new categories that you need to consider setting up new or different mappings into operations.

Each aspect system is responsible for performing relevant security checks on the operations it provides. Afw provides a set of functions for this purpose. The designer of an aspect system must decide which operations to provide, register these with the aspect directory, and include relevant calls at appropriate places in the aspect system code to verify that the current user is granted the right to use the operation. Similarly, he or she must include calls to log the operation.

The Afw OPC Server performs security checks and audit logging on all OPC accesses. This relieves aspect systems designers from having to implement security checks and audit logging on read and write operations on properties that are published through OPC.
5.5 Aspect systems

To participate in Aspect Object operations, an application must present itself as an *aspect system* (or possibly as several aspect systems). Aspect systems provide the functionality that is defined for Aspect Objects. Examples are Control, Graphics, Alarm & Event, History, Reports, Documentation, Simulation, Asset Optimization, Material Tracking, Production Scheduling, etc.

An aspect system provides implementations for one or several aspect types through COM objects, referred to as *aspect system objects* (ASO). These objects interact with the Aspect Framework through different sets of framework-defined interfaces for common object and aspect operations (see [2]). ASOs may also expose object properties through OPC.

A group of tightly related aspect systems may agree to use specific interfaces for special purposes. This possibility should be used restrictively, and only for interaction that does not require that other aspect systems participate or are informed.

The Aspect Framework provides COM representations of Aspect Objects and aspects. Aspect systems provide their functionality through aspect system objects. Users identify and access aspect system objects through Aspect Objects and Aspects.

![Diagram of Aspect Objects, Aspects, and Aspect Systems](image)

**Figure 26** COM representations of Aspect Objects, Aspects, and Aspect Systems

Aspect system objects can use any suitable means for interacting with the applications they represent. Thus, it is possible to implement different aspects using applications that internally use very different implementation technologies — the aspect system objects serve as a translation layer between the Aspect Object architecture and whatever architecture is used internally by the aspect system. Since applications do not need to interact directly with each other in order to participate in Aspect Object operations, they do not need to be aware of each other, and thus it is possible to add new applications without changing existing ones.

Each aspect system is responsible for storing and maintaining its own data. However, in many cases data must be shared among a group of aspect systems. Unless handled in a correct way, this may cause data consistency problems. The Aspect Object architecture offers two ways to avoid that:

- A shared data item is stored in only one copy by one aspect system, and other aspect systems access it from there, through framework defined interfaces.
- Several aspect systems hold their own copies of a shared data item. When an aspect system updates its copy, it must inform the framework, which in turn informs other aspects systems to update their copies.

An aspect system may provide one or more user interfaces, implemented as an ActiveX, ASP/HTML page, Active Document, OLE Server, or Windows application. All user interface components should follow the User Interface Style Guide [5]. This includes colors, fonts, symbols, icons, menus, dialogs, graphics displays, etc.
Native language support should be provided by means of the NLS services that are provided by the platform.

Many aspect systems provide some degree of configurability, typically by means of some form of configuration tools. To ensure that engineering can be done in an efficient and consistent way, there are certain rules defined for how aspect systems shall handle and share configuration data. These rules are defined in [9].

Different aspect systems can be more or less well integrated into the system. To be able to easily describe how well integrated an aspect system is, different integration levels are defined. These are described in overview below. Please refer to [14] for details.

0. **User Interface Wrapping** – The application provides its user interface as an ActiveX, ASP/HTML page, Active Document, OLE Server, or Windows application. This is the lowest level of integration, allowing a user to access the application’s user interface through Aspect Objects, but providing no other integration benefits.

1. **User Interface Integration** – The application recognizes itself as an aspect system, and it supports basic aspect operations, such as Create/Delete. When you navigate to the aspect you end up in a context that is relevant to the current Aspect Object.

2. **Navigation Integration** – The aspect system recognizes the fact that there are also other aspect systems. It supports context menus, making it possible to navigate to other aspects directly from within the application. It also provides contents for other aspect systems to include in context menus.

3. **Engineering Integration** – The aspect system supports all aspect operations, including Copy/Paste, Export/Import, Inheritance and Version Handling. The aspect system has knowledge about object type libraries and structures.

4. **Administration Integration** – The aspect system supports integrated administration, including install, backup/restore, and NLS translation.

5. **Data Management Integration** – The aspect system supports life-cycle management, and transaction handling with rollback.

Depending on which integration level is selected, the aspect system objects must support different sets of framework-defined interfaces for common object and aspect operations. See [2]. The requirements at a higher level include all requirements defined for lower levels.
5.6 Service and workplace deployment

The Aspect Object architecture assumes a system of nodes and networks as described in section 3. This chapter describes general principles for the deployment of services and workplace software on such a system. Please note that for exact information on configuration rules, performance and capacity limitations, etc., it is necessary to refer to relevant product documentation for each product respectively.

From an architectural perspective, any Afw Service can run on any server node in such a system. While this is very flexible, it can also be quite confusing. Therefore, to make it possible to create simple configuration rules, and to describe, test and verify various supported configurations, three classes of servers are defined:

- **Aspect servers** run the ‘central’ intelligence in the system, including the aspect directory and other services related to object management, names, security etc. This is normally also where Windows Domain Controller is run.

- **Connectivity servers** provide access to controllers and other data sources. Several groups of connectivity servers may exist in a system, each serving one set of data sources. Examples of services that run on a connectivity server are OPC related services (DA, AE, and HDA), SysMsg, etc.

- **Application Servers** run various types of system applications, for example Batch, Asset Optimization, Simulation and Optimization, Enterprise Historian, AIP Web Server, etc.

Server classes are deployed on **nodes** - a node is a network addressable machine. For medium sized and large systems, one server class is typically deployed on several nodes. In small systems, several server classes can be deployed on the same node.

A formalized graphical notation is used to illustrate the deployment of server classes to nodes. A node is represented with a rectangle, with the server class that is deployed to it as a heading. When required for clarity, specific services or groups of services (e.g. products) are listed below the heading. Redundant servers are drawn as overlapping rectangles, with the type of redundancy stated as 1o2 or 2o3 (one-out-of-two or two-out-of-three):

![Graphical representation of the deployment of server classes and services to nodes](image)

*Figure 27  Graphical representation of the deployment of server classes and services to nodes*
The figures below show examples of server deployment for small, medium sized and large systems.

**Figure 28** Server deployment – small system

**Figure 29** Server deployment – medium sized system

**Figure 30** Server deployment – large system
Workplaces can be configured to use any available server. However, to be able to control how the server capacity is utilized, e.g. to ensure that operators always have good response times, it is possible to tie groups of workplaces to specific groups of servers. This grouping is called affinity, and is also used to describe how workplaces shall be reconnected to different servers in various failure situations. See Figure 31.

![Figure 31 Example - Workplace connection to servers](image)
5.7 Example – Graphics

This section briefly describes the aspect system Graphics, as an example of how applications that are built on the Aspect Object architecture can be structured and deployed.

Graphics provides functionality for creating and presenting graphics with dynamic information from Aspect Object properties. It implements aspects such as process graphics and faceplates, but is also useful in many other cases where a graphical user interface with dynamic information is required.

Graphics is a client application. It uses the services of the Afw OPC/DA server for real-time access to Aspect Object properties, and of the system function File Set Distribution (FSD) for storage and distribution of graphics pages.

A new graphics display page is created by means of the Graphics Builder. When the display page is built it is transferred to FSD. From there, Graphics clients have access to it.

Display pages are cached locally with a Graphics client when they are brought up, thus allowing previously displayed pages to come up quicker. The client can be configured to always keep certain frequently used pages in cache for fast access.

![Diagram](image)

**Figure 32 Application example – Graphics (arrows indicate direction of data flow)**
6 INTEGRATION OF CONTROLLERS AND DEVICES

6.1 Overview

Functionality in the control layer is provided by controllers and other types of devices. In the Aspect Object architecture, these are entities that in some form of dedicated environment provide part of the functionality of certain aspects, typically acting as sources for real time data, historical data, and alarm and event data.

Depending on how they are connected to the server layer, devices are classified as:
- Control network devices, i.e., devices connected through a control network. This includes controllers, robots, variable speed drives, etc.
- Fieldbus devices, i.e., devices connected through a fieldbus. Examples are remote I/O, and smart sensors and actuators, but also controllers, robots, variable speed drives, etc., when these devices are connected through a fieldbus.
- Web devices, i.e., devices that include a web server through which functionality of the device is accessed. Using the application layer protocol HTTP, web devices can be connected in different ways, although TCP/IP/Ethernet is the most common.
- Generic devices, i.e., devices that are connected to an IIIT system through other means than IIIT supported control networks, fieldbuses, or web technology. Examples are devices connected through Modbus and similar protocols.

![Diagram of different ways to connect devices](image)

Figure 33 Different ways to connect devices

6.2 Connectivity packages

Controllers and devices are connected to the server layer through connectivity components. These provide access to real time data, historical data, and alarm and event data, from different types of controllers and devices. Different connectivity components allow workstation and server level functions to be used with many and varying types of control systems and devices.

---

1 Modbus is a trademark of AEG Schneider Automation, Inc.
Access to real time data, historical data, and alarm and event data is provided through Afw OPC. For other types of access, e.g. tools’ access to configuration data, relevant underlying application protocol services are used (e.g. MMS, FF HSE, etc.). Figure 34 below illustrates this for a controller:

Figure 34 Connectivity example

A connectivity package is a bundle of all the components that are needed to support the integration of a certain type of controller or device. Connectivity packages are typically provided for controllers and devices, but can be used for integrating any type of system or function.

The Aspect Object architecture defines three levels of controller and device integration. For each level certain integration requirements apply. These requirements are described in overview below. Please refer to [14] for details

Level 1:
- An OPC/DA server, or a connector/adaptor for the Afw OPC server, for real time data access.
- Support for OPC browsing (or an up-loader, see below), to make it possible to build an Aspect Object representation of the device, and for other applications to access device data.

Level 2:
- Supporting aspect systems, e.g. for configuration.
- Aspect Object types that represent the device and the functionality it provides. These object types must be derived from base types defined in [10], to allow other applications to use them in an intelligent way. The object types shall include
  – Support for System Status and Security, including a defined default setting
  – Graphical aspects, such as faceplates, display elements, object display, etc., as relevant
- If Aspect Objects are not populated from the configuration tool, an upload function that does this from information in the device is required.
- OPC /HDA and /AE servers shall be provided if the device supports local collection of historical data and alarm and event detection respectively.
Level 3:
- Configuration tools must be well integrated (level 3 or 4, see section 5.3), and must be able to automatically populate Aspect Objects to represent the device and its functionality. It must be possible to perform all device engineering from the Plant Explorer workplace, and to do parameter driven bulk configuration of devices (see [9] for details).
- Object trend, documentation aspects, and other relevant aspects must be provided for the object types.
- Extended property information must be provided, describing how different device properties are related, e.g., which property holds engineering units, min/max ranges etc. for a certain other property (see [2] for details).

For all integration levels, relevant documentation and online help shall be provided.

6.3 Control network devices

Control network devices are connected through the control network. This typically includes controllers, but also robots, variable speed drives, etc., may be connected in this way. All requirements that are described for the different integration levels in section 6.2 are relevant.

6.4 Fieldbus devices

Fieldbus devices are connected through one of the Industrial IT supported fieldbuses. These include Profibus, Foundation Fieldbus, and HART. Examples of fieldbus devices are smart sensors and actuators and remote I/O, but also controllers, robots, variable speed drives, etc., when these types of devices are connected through a fieldbus.

All requirements that are described for the different integration levels in section 6.2 are relevant. For supported fieldbuses, much of this functionality is already provided through the fieldbus integration, such as protocol stack, OPC server or adaptor, configuration tools, and Aspect Object types. Thus, in many cases only a few Aspect Object types need to be added to integrate a new fieldbus device type.

A Device Type Manager (DTM) is a software component that packages device specific knowledge, such as how to configure device specific functionality, or how to access device specific information. The DTM can be plugged into a standardized framework to provide a host system with device specific functions. In the Aspect Object architecture, a fieldbus aspect system serves as this framework. For devices connected to fieldbuses where this technology is used, such as Profibus, a DTM must be provided for integration at levels 2 and 3. For further details see [8].

![Accessing device specifics through a DTM](Figure 35)

**Figure 35  Accessing device specifics through a DTM**
6.5 Generic devices

Generic devices are connected through other means than IIT supported control networks, fieldbuses, or web technology. Examples are devices connected through Modbus\(^1\) and similar protocols. Although in many cases a relatively low integration level is aimed for, there are situations where a high integration level is desirable also for these kinds of devices.

The effort that is required to integrate a generic device depends on whether the communication link is already supported or not, and whether it is connected directly to a server or through a controller or other control network device.

Cases where a generic device is connected directly to the server layer are very similar to integration of control network devices, and the same requirements apply for the three integration levels.

When a generic device is connected to a controller (or other control network device) at integration level 1, the device is seen as an extension of the controller. Variables in the generic device are represented as a subset of the variable space of the controller, and configuration of the data exchange between the controller and the generic device is the responsibility of the controller configuration tool. At higher integration levels, generic device integration becomes similar to fieldbus device integration, and it becomes an issue to determine whether the communication link should be promoted to the status of a supported fieldbus.

6.6 Web devices

<Web devices will be connected through OPC/XML. Details will be defined when relevant specs from OPC Foundation are available.>

\(^1\) Modbus is a trademark of AEG Schneider Automation, Inc.
THE ASPECT INTEGRATOR PLATFORM

Based on the Aspect Object architecture and the Aspect Framework, the Aspect Integrator Platform (AIP) is a collection of software components and documentation. Defined as a fully supported product in itself, AIP provides the development and execution environment for applications and products that build on the Aspect Object architecture.

From a development, packaging, and installation point-of-view, products built on AIP are of two kinds:

- **System Products** include Aspect Systems and other software components and selected platform run-time components
- **System Product Add-ons** include Aspect Systems and other software components that are packaged for installation on existing System Products.

AIP includes tools for development of aspect systems and connectivity components, and to build System Products and System Product Extensions:

- **Aspect Automation** is a programming model that allows a user to create scripts using platform functionality.
- **Aspect Express** is a tool integrated with Visual Basic, for easy and fast integration of standard Active-X controls as aspect systems.
- **Aspect Studio** is a powerful toolset integrated with Visual Studio, for creating aspect systems and system products.

The Aspect Integrator Platform contains a large number of functions, divided into the following groups:

- **Basic functions:** This group contains the functionality that is necessary to do consistent NLS handling (mainly text strings), networking components, message generation, file distribution, test and debug logging, basic data type definitions etc.
- **Object Management functions:** This group includes that Aspect Directory and the Object Manager, the different ‘basic’ aspect systems to support naming, structures etc. All library, object type, import/export and drag-and-drop functionality is also placed in this group.
- **Service functions:** This group contains the necessary framework components to define and deploy services. The support for location transparency, redundancy and scalability, start-up/shutdown and configuration is placed in this group.
- **Optional functions:** This group includes View Handling and Navigation, Graphics, Data Subscription, Basic History, System Supervision and Alarm and Event.

Depending on the type of functionality that is required, different subsets of this platform functionality are needed in different system products.

The Aspect Integrator platform, the tools, and the processes for development of applications and products based on AIP, are described in [13].
<table>
<thead>
<tr>
<th>Rev. ind.</th>
<th>Page (P) Chapt.(C)</th>
<th>Description</th>
<th>Date Dept./Init.</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td></td>
<td>New document</td>
<td>2001-11-20 ATIC/PA/A TP</td>
</tr>
</tbody>
</table>