



SEMI E39-95

OBJECT SERVICES STANDARD: CONCEPTS, BEHAVIOR, AND SERVICES

CONTENTS

1 Purpose

2 Scope

3 Referenced Documents

4 Definitions

4.1 Requirements

4.2 Objects

4.3 Services

4.4 Data Type

5 Conventions

5.1 OMT Object Information Model

5.2 Object Attribute Representation

5.3 Service Message Representation

6 Background

7 Applicable Documents

8 Basic Concepts and Behavior

8.1 Object Attributes

8.1.1 Use of Text

8.1.2 Object Type

8.1.3 Object Identifier

8.1.4 Object Handle

8.2 Object Definition

9 Object Relationships

9.1 Object Hierarchy

9.2 Scope and Ownership

10 Services and Scenarios

10.1 Scope

10.2 Filtering

10.3 Object Services Parameter Dictionary

10.4 Get and Set Attributes

10.5 GetType

10.6 GetAttrName

11 Applications

11.1 Get All of an Object's Attributes

11.2 Determine All Objects of a Specific Type and with Specific Characteristics

11.3 Determine Specific Attributes of a Specific Object Instance

11.4 Determine Types of Subordinate Objects

11.5 Determine Names of Attributes of Subordinate Objects

12 Requirements for Compliance

12.1 Fundamental Requirements

12.2 Additional Capabilities

12.2.1 Filters

12.2.2 Owner Objects

Appendix 1

A1-1 Overview of Object Terminology

A1-2 Object Modeling Technique (OMT) Notation

A1-2.1 Basic Notation

A1-2.2 Associations

A1-3 Generalization and Inheritance

A1-3.1 Object Composition and Containment

List of Tables

Table 8.1 Top Object Attribute Definition

Table 10.1 Object Services

Table 10.2 Object Services Parameter Dictionary

Table 10.3 GetAttr Service

Table 10.4 SetAttr Service

Table 10.5 GetType Service

Table 10.6 GetAttrName Service



NOTES

SUPERSEDED

This page intentionally left blank.



SEMI E39-95

OBJECT SERVICES STANDARD: CONCEPTS, BEHAVIOR, AND SERVICES

1 Purpose

The purpose of the Object Services Standard (OSS) standard is to provide general terminology, conventions, and notation for describing behavior and data in terms of objects and object attributes. In addition, it provides basic services for reading object attributes, setting their values, and for asking for an object's contents. This standard is intended to be referenced by other standards which define specific objects to reduce redundancy.

2 Scope

The scope of this standard is to provide concepts, behavior, and services common to a variety of public objects.

Object models are common to multiple standards. Object Services provide basic object-related definitions, and basic services for getting object attributes and setting attribute values, that can be used by all standards defining public objects. These services allow basic management of data based on objects.

The object services defined in this document may be included in the services provided by other standards. They may also be provided independently of such other standards.

3 Referenced Documents

James Rumbaugh, Michael Blaha, William Premerlani, Frederick Eddy, William Lorensen, Object-Oriented Modeling and Design, Englewood Cliffs, New Jersey: Prentice-Hall, 1991.

ISO 9595¹ — Common Management Information Service (CMIS)

4 Definitions

The following basic definitions are provided in a logical order.

4.1 Requirements

fundamental requirements — The requirements for information and behavior that must be satisfied for compliance with a standard. Fundamental requirements apply to specific areas of application, objects, or services.

fundamental compliance — Conformance to all fundamental requirements for an object or service resource.

4.2 Objects

*object*² — An entity with a specific set of data and behaviors. Objects may be physical or conceptual.

object model — A static graphic model of objects to show structure — the identity of objects, their attributes and operations, and their relationships with one another.

object type — A formal classification of a group of similar objects. Synonym: *object class*. Examples: *equipment, wafer, carrier*.

object instance — An instance of an object type. An object type is like a template, while an object instance is the actual object. *Example: an actual and specific optical stepper installed in a particular fab is an instance of the type "Optical Stepper."*

object attribute (attribute) — Information concerning an object. *Examples for object type "equipment": manufacturer, model, serial number.*

Attributes are classified in various ways, according to their visibility (public/private), optionality, and access:

- *private attribute* — An attribute that is used strictly for internal purposes and is unknown (invisible) through public services.
- *public attribute* — An attribute that is known (visible) and whose current value is provided as a service to other entities upon request.
- *fundamental attribute* — An attribute that is required for fundamental compliance with a standard service.

¹ ISO, International Organization for Standardization, C.P. 56, CH-1211 Geneva 20, Switzerland

² The term "object" may refer to either an object class or an instance of a class.



— *optional attribute* — An attribute that is required only in support of one or more optional standard services.

— *read-only attribute (RO)* — May not be changed through public services.

— *read/write attribute (RW)* — May be changed through public services.

attribute name — The formal name of the attribute that is used to identify it. The names (and data types) of public attributes are included with the object's definition and are unique for that object.

object identifier — A set of one or more items of information, concerning a particular instance (instantiation) of an object of a given type, that together uniquely distinguish that instance from all other instances of that object within a defined scope. NOTE: An object may have more than one identifier. An identifier may be *simple* (consist of only one attribute) or *complex* (consist of more than one attribute). *Example: The combination of equipment's manufacturer, model, and serial number serve as an identifier that uniquely identifies a specific installation.*

object handle — A numeric or binary identifier assigned by an application for internal use. NOTE: A handle may also be available as a *public attribute* but can not generally be guaranteed to be either persistent or unique outside of the current relationship between service user and service provider. The persistence of the handle is specified by the standard that defines the object.

operation — A function performed by, or inherent to, an object. *Example: for equipment, "run," "stop," "abort."*

aggregation object — An object that is composed (made up) of other objects. An aggregation may lose some degree of integrity if one of its components is missing.

component object — An object that is part of an aggregation.

container object — An object that is intended to hold other types of objects. The contents may or may not be ordered.

contents — An object that is in a container. *Examples: a wafer in a cassette, a book in a library.*

owner object — An object that is an aggregation, container, or supervisor of another object. The owner object is said to *own* the other object.

owned object — An object that is a component of, contained in, or supervised by, another object. The owned object is said to be *owned by* the other object.

An object may have multiple owners. For example, it may be a shared resource.

4.3 Services

scope — The specification of one or more objects that starts with a specific owner object and proceeds downward through a hierarchical sequence of "owns" relationships.

service (or message service) — Represents a function offered to a user by a provider. A service consists of a sequence of service primitives, each described by a list of parameters. A service excludes definition of message structure and protocol.

notification service — Initiated by the service provider and sent to the service consumer/subscriber. No response is expected.³

Notifications consist of two service primitives — a message from the sender to the communications facility and an indication to the receiver from the communications facility.

request service — Initiated by the service consumer. Requests ask for data or for an activity (operation) from the provider. Requests expect a specific response message.

A request consists of a message that requires a response from the receiver. The primitives for a request are the same as those of the notification, while the response defines additional primitives called the *response* and the *confirmation*.

service resource — A logical group of one or more services within a specific area of functionality.

4.4 Data Type

form — Type of data: positive integer, unsigned integer, integer, enumerated, boolean, text, formatted text, structure, list, ordered list.

³ This does not preclude the implementation of a response message in the message protocol.



positive integer — May take the value of any positive whole number. Messaging protocol may impose a limit on the range of possible values.

unsigned integer — May take the value of any positive integer or zero. Messaging protocol may impose a limit on the range of possible values.

integer — May take on the value of any negative or unsigned integer. Messaging protocol may impose a limit on the range of possible values.

floating point — May take on any single (real) numeric value, positive or negative. Messaging protocol may impose a limit on the range of possible values.

enumerated — May take on one of a limited set of possible values. These values may be given logical names, but they may be represented by any single-item data type.

boolean — May take on one of two possible values, equating to TRUE or FALSE.

text — A text string. Messaging protocol may impose restrictions, such as length or ASCII representation.

formatted text — A text string with an imposed format. This could be by position, by use of special characters, or both.

structure — A complex structure consisting of a specific set of items, of possibly mixed data types, in a specified arrangement.

list — A set of one or more items that are all of the same form (one of the above forms).

ordered list — A list for which the order in which items appear is significant.

5 Conventions

- Defined terms are presented in boldface when introduced for the first time.
- Formally reserved text strings, such as attribute names, are underlined.

5.1 OMT Object Information Model — The object models are presented using the Object Modeling Technique (OMT) developed by Rumbaugh, James, et al, in Object-Oriented Modeling and Design, Prentice Hall, Englewood Cliffs, NJ, c1991.⁴

⁴ For a full description of the Object Modeling Technique, see *Object-Oriented Modeling and Design*, Rumbaugh, Blaha, Premerlani, Eddy and Lorenson, Prentice Hall, 1991.

Overviews of this notation are provided in the Appendix. A brief discussion of terminology is also provided in the Appendix.

5.2 Object Attribute Representation — The object information models for standardized objects will be supported by an attribute definition table with the following column headings:

Attribute Name	Definition	Access	Rqmt	Form
The formal text name of the attribute	Description of the information contained.	RO or RW	Y or N	(see 4.4)

The Access column uses RO (Read Only) or RW (Read and Write) to indicate the access that users of the service have to the attribute.

A 'Y' or 'N' in the Requirement (Rqmt) column indicates if this attribute must be supported in order to meet fundamental compliance for the service.

The Form column is used to indicate the format of the attribute. (See Section 4.4 for definitions.)

5.3 Service Message Representation

Service Resource Definition — A service resource definition table defines the specific set of messages for a given service group, as shown in the following table:

Message Service Name	Type	Description
Message name	N or R	The intent of the service.

Type can be either N = Notification or R = Request.

Notification type messages are initiated by the service provider, and the provider does not expect to get a response from the consumer/subscriber.

Request messages are initiated by a service consumer or subscriber. Request messages ask for data or an activity from the provider. Request messages expect a specific response message (no presumption on the message content).

Service Parameter Dictionary — A service parameter dictionary table defines the parameters for one or more services, as shown in the following table:

Parameter	Form	Description
Parameter X	Data type	A parameter called X is B in A.



A row is provided in the table for each parameter of the service. The first column contains the name of the parameter. This is followed by columns describing the form and contents of the corresponding primitive.

The Form column is used to indicate the type of data contained in a parameter. (See Section 4.4 for definitions.)

The Description column in the Service Parameter Dictionary table describes the meaning of the parameter, the values it can assume, and any interrelationships with other parameters.

To prevent the definition of numerous parameters named "XxxList," this document adopts the convention of referring to the list as "(List of) Xxx." In this case, the definition of the variable Xxx will be given, not of the list. The term "list" indicates a collection (or set) of zero or more items of the same data type. Where a list is used in both the request and the response, the list order in the request is retained in the response. A list must contain at least one element unless zero elements are specifically allowed.

Service Message Definition — A service message definition table defines the parameters used in a service, as shown in the following table:

Parameter	Req/Ind	Rsp/Conf	Description
Parameter X	(see below)	(see below)	A description of the service.

The columns labeled Req/Ind and Rsp/Conf link the parameters to the direction of the message. The message sent by the initiator is called the "Request." The receiver terms this message the "Indication" or the request. The receiver may then send a "Response," which the original sender terms the "Confirmation."

The following codes appear in the Req/Ind and Rsp/Conf columns and are used in the definition of the parameters (e.g., how each parameter is used in each direction):

"M" — *Mandatory Parameter* — must be given a valid value.

"C" — *Conditional Parameter* — may be defined in some circumstances and undefined in others. Whether a value is given may be completely optional or may depend on the values of other parameters.

"U" — *User-Defined Parameter*

"-" — The parameter is not used.

"=" — (for response only) Indicates that the value of this parameter in the response must match that in the primary (if defined).

6 Background

During the development of proposals for several other service standards, it was discovered that they shared a common need for terminology and services related to objects. Originally each proposal included a service-specific message for getting the values of an object's attributes, and some also provided a service-specific message for setting these values.

The decision was made to eliminate this redundancy through provision of a standard to provide those definitions and services used by the other separate service standards.

7 Applicable Documents

SEMI Book of Standards, Equipment Automation/Software Volumes 1 and 2⁵

ISO/TR 8509:1987,⁶ Information Processing Systems, Open Systems Interconnection — Service Conventions

William Stallings, Networking Standards, *A Guide to OSI, ISDN, LAN, and MAN Standards*, Addison-Wesley Publishing Company

8 Basic Concepts and Behavior

This section defines the concepts and behavior that are common to public objects.

Object-oriented analysis is a widely accepted tool, both for applications that are implemented using object-oriented technologies and those using traditional technologies. Object information models provide a powerful method for describing relationships that can be intuitively understood.

For those unfamiliar with object-oriented terminology, a brief discussion of terms is provided in the Appendix.

8.1 Object Attributes — An *object attribute* is a data value that is held by all instances of a given object

⁵ This set of documents may be obtained from SEMI, 805 East Middlefield Road, Mountain View, CA 94043.

⁶ ISO, International Organization for Standardization, C.P. 56, CH-1211 Geneva 20, Switzerland



type. For communication purposes, public attributes are assigned a logical *name* that is unique for that object. An object's attributes may be individually referenced either by an enumerated numeric value or by a reserved *attribute name* that conforms to the use of text defined in Section 8.1.1.

An attribute that can be requested or referenced through formal public services is a *public attribute*. Other attributes may exist in specific implementations but are invisible through public services and are called *private attributes*.

Specific attributes may be designated as *fundamental* for a specific object. These shall be supported by all implementations that use or reference the specified object. Attributes that are not fundamental are *optional*.

8.1.1 Use of Text — Values defined as text, when given in ASCII, are subject to certain restrictions. This includes object attribute values, attribute names, and service parameters.

Text in ASCII is restricted to the characters between 20₁₆ and 7E₁₆, excluding the question mark "?," the asterisk "*", and the tilde "~."

The question mark and asterisk are reserved for use as "wild characters" in filters and searches, while the tilde is reserved to allow systems that cannot use spaces to convert spaces to tildes for internal use.

Text used in specific contexts may have additional restrictions.

Unless otherwise stated, case is not significant for purposes of comparison. However, case is used to improve readability and should be preserved whenever possible.

8.1.2 Object Type — An object always knows its type. For this reason, the attribute for object type, ObjType, is required for all public objects. ObjType is a text string containing the formal classification of an object. The text string shall conform to the convention defined in Section 8.1.1, with the additional restriction that the "greater than" symbol ">" and the colon ":" are also excluded. The text string shall not start or end with a space.

Object types (the values for ObjType) for standardized objects are reserved. Types reserved are specific to individual standards.

8.1.3 Object Identifier — Every instance of an object shall have one or more attributes that together uniquely distinguish that instance from all other instances of objects of the same object type. An object may have more than one identifier. The identifier(s) are defined as part of the object definition.

ObjID is a text-based fundamental attribute of all public objects and provides an attribute of a known form to serve as an identifier for an object of any type. The text string shall conform with the convention defined in Section 8.1.1 and with the additional restriction that the "greater than" symbol ">" and the colon ":" are also excluded. The text string shall not start or end with a space.

The object definition for each object shall specify the particular identifier used for general access. For objects that normally use an identifier with a numeric value as an identifier (such as "handle," described below), the value of ObjID is a numeric text string set to the value of that attribute.

An object may also have other identifiers in addition to ObjID.

8.1.4 Object Handle — An object's *handle* is an attribute with a numeric value that is assigned by the application that created the object. The handle may be used for the object's identifier. It is generally intended for local use and may or may not be defined as a public attribute in the object's formal definition.

Where used, the handle may be guaranteed to be unique only within a specific context and within a single association between service-provider and service-user for a specific service resource. The handle may or may not be persistent beyond a certain context within that association or beyond it. Persistence of the handle is specified as part of the definition of the object and is beyond the scope of OSS.

8.2 Object Definition — An object definition includes an attribute definition table described in Section 5.3.

The object that is the super-type of all public objects is called the *top object*. The attributes and operations of the top object, shown in Figure 8.1, are fundamental requirements for all public objects. That is, all objects shall respond to the object attributes ObjType and text strings, and all public objects shall recognize and respond to the operations *get attributes* and *set attributes*.