IEC 61970-404

Edition 1.0 2007-08

INTERNATIONAL STANDARD

Energy management system application program interface (EMS-API) – Part 404: High Speed Data Access (HSDA)

IEC 61970-404:2007(E)

IEC Central Office
3, rue de Varembé
CH-1211 Geneva 20
Switzerland
Email: inmail@iec.ch
Web: www.iec.ch

About the IEC

The International Electrotechnical Commission (IEC) is the leading global organization that prepares and publishes International Standards for all electrical, electronic and related technologies.

About IEC publications

The technical content of IEC publications is kept under constant review by the IEC. Please make sure that you have the latest edition, a corrigenda or an amendment might have been published.

- Catalogue of IEC publications: www.iec.ch/searchpub

The IEC on-line Catalogue enables you to search by a variety of criteria (reference number, text, technical committee,…). It also gives information on projects, withdrawn and replaced publications.

- IEC Just Published: www.iec.ch/online_news/justpub

Stay up to date on all new IEC publications. Just Published details twice a month all new publications released. Available on-line and also by email.

- Electropedia: www.electropedia.org

The world's leading online dictionary of electronic and electrical terms containing more than 20 000 terms and definitions in English and French, with equivalent terms in additional languages. Also known as the International Electrotechnical Vocabulary online.

- Customer Service Centre: www.iec.ch/webstore/custserv

If you wish to give us your feedback on this publication or need further assistance, please visit the Customer Service Centre FAQ or contact us:

Email: csc@iec.ch
Tel.: +41 22 919 02 11
Fax: +41 22 919 03 00

IEC 61970-404

Edition 1.0 2007-08

INTERNATIONAL STANDARD

Energy management system application program interface (EMS-API) – Part 404: High Speed Data Access (HSDA)

INTERNATIONAL
ELECTROTECHNICAL
COMMISSION

PRICE CODE S

ICS 33.200

ISBN 2-8318-9251-1

CONTENTS

INTERNATIONAL ELECTROTECHNICAL COMMISSION

ENERGY MANAGEMENT SYSTEM APPLICATION PROGRAM INTERFACE (EMS-API) –

Part 404: High Speed Data Access (HSDA)

FOREWORD

1) The International Electrotechnical Commission (IEC) is a worldwide organization for standardization comprising all national electrotechnical committees (IEC National Committees). The object of IEC is to promote international co-operation on all questions concerning standardization in the electrical and electronic fields. To this end and in addition to other activities, IEC publishes International Standards, Technical Specifications, Technical Reports, Publicly Available Specifications (PAS) and Guides (hereafter referred to as "IEC Publication(s)"). Their preparation is entrusted to technical committees; any IEC National Committee interested in the subject dealt with may participate in this preparatory work. International, governmental and non-governmental organizations liaising with the IEC also participate in this preparation. IEC collaborates closely with the International Organization for Standardization (ISO) in accordance with conditions determined by agreement between the two organizations.
2) The formal decisions or agreements of IEC on technical matters express, as nearly as possible, an international consensus of opinion on the relevant subjects since each technical committee has representation from all interested IEC National Committees.
3) IEC Publications have the form of recommendations for international use and are accepted by IEC National Committees in that sense. While all reasonable efforts are made to ensure that the technical content of IEC Publications is accurate, IEC cannot be held responsible for the way in which they are used or for any misinterpretation by any end user.
4) In order to promote international uniformity, IEC National Committees undertake to apply IEC Publications transparently to the maximum extent possible in their national and regional publications. Any divergence between any IEC Publication and the corresponding national or regional publication shall be clearly indicated in the latter.
5) IEC provides no marking procedure to indicate its approval and cannot be rendered responsible for any equipment declared to be in conformity with an IEC Publication.
6) All users should ensure that they have the latest edition of this publication.
7) No liability shall attach to IEC or its directors, employees, servants or agents including individual experts and members of its technical committees and IEC National Committees for any personal injury, property damage or other damage of any nature whatsoever, whether direct or indirect, or for costs (including legal fees) and expenses arising out of the publication, use of, or reliance upon, this IEC Publication or any other IEC Publications.
8) Attention is drawn to the Normative references cited in this publication. Use of the referenced publications is indispensable for the correct application of this publication.
9) Attention is drawn to the possibility that some of the elements of this IEC Publication may be the subject of patent rights. IEC shall not be held responsible for identifying any or all such patent rights.

International Standard IEC 61970-404 has been prepared by IEC Technical Committee 57: Power systems management and associated information exchange.

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

FDIS	Report on voting
57/887/FDIS	57/906/RVD

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

A list of all parts of the IEC 61970 series, under the general title *Energy Management System Application Program Interface (EMS-API)*, can be found on the IEC website.

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

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

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

INTRODUCTION

This part of IEC 61970 is part of the IEC 61970 series that defines Application Program Interfaces (APIs) for an Energy Management System (EMS). The IEC 61970-4XX and IEC 61970-5XX series documents comprise Component Interface Specifications (CISs). The IEC 61970-4XX series CIS are specified as Platform Independent Models (PIMs), which means they are independent of the underlying technology used to implement them. PIM specifications are also referred to as Level 1 specifications. The IEC 61970-5XX series CIS, on the other hand, are specified as Platform Specific Models (PSMs). PSM specifications are also referred to as Level 2 specifications.

IEC 61970-4XX CISs specify the functional requirements for interfaces that a component (or application) should implement to exchange information with other components (or applications) and/or to access publicly available data in a standard way. The component interfaces describe the specific event types and message contents that can be used by applications for this purpose.

IEC 61970-404 specifies an interface for the efficient transfer of data in a distributed environment. Small amounts of data are transferred with short delay but also large amounts of data are transferred in short time but with possibly longer delay. This is a typical requirement for a SCADA system that acts as a real time data provider to other sub-systems. Other systems than SCADA may also benefit from the characteristics of HSDA. When short delay times as well as bulk data transfer is required, HSDA is a good fit.

These component interface specifications refer to entity objects for the power system domain that is defined in the IEC 61970-3XX series, including IEC 61970-301.

ENERGY MANAGEMENT SYSTEM APPLICATION PROGRAM INTERFACE (EMS-API) –

Part 404: High Speed Data Access (HSDA)

1 Scope

The IEC 61970-404 High Speed Data Access (HSDA) specification specifies a generalized interface for efficient exchange of data. The specification takes into account the latencies caused by a Local Area Network (LAN) providing efficient data exchange also over Local Area Networks.

IEC 61970-404 is derived from the Object Management Group (OMG) Data Acquisition from Industrial Systems section Data Access (DAIS DA) specification. OMG DAIS DA relies on the OMG Data Access Facility (DAF) and OPC Data Access (DA) specifications. OMG DAIS DA is a Platform Specific Model (PSM) with CORBA as the platform and OPC DA is a PSM with Microsoft COM as the platform. IEC 61970-404 describes the functionality of these PSMs in a technology independent way (i.e., as a Platform Independent Model (PIM)). Hence it explains the functionality to a level that can be used to create additional PSMs or act as an introduction to existing PSMs, i.e. DAIS DA and OPC DA. Implementers wanting an introduction to OMG DAIS DA and OPC DA shall read these documents.

The HSDA interface is intended to interoperate with other IEC 61970 based interfaces. Hence, it is possible to use information retrieved from an other interface to access the same information using this interface, for example:

- object identifiers,
- attribute names or identifiers,
- class names or identifiers.

Subclause 4.6 provides a generic mapping for the CIM classes and attributes.

The way data is organized in a server implementing the HSDA interface can be seen by using the browse interfaces for data and meta data. It is also possible to use the data access interface directly without using the browse interfaces if the client has an *a priori* knowledge of object, class and attribute identifiers. Object identifiers may be retrieved using data from other interfaces, for example a CIMXML file or the IEC 61970-403 interface. Information on what classes and attributes are available will be described in IEC 61970-45X documents, for example SCADA data, State Estimator results etc.

IEC 61970-1 provides the EMS-API reference model upon which this standard is based. In that reference model, the terminology used in This part of IEC 61970 is introduced and the role of the CIS is explained.

IEC 61970-401 provides an overview and framework for the CIS (IEC 61970-4XX) standards.

The mapping of IEC 61970-404 to implementation specific technologies or PSMs is further described in a separate series of documents, i.e. the future IEC 61970-5XX. For actual implementations, the future IEC 61970-5XX, OMG DAIS DA, OMG DAF or OPC DA are used.

2 Normative references

The following referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies:

IEC 61970-1, *Energy management system application program interface (EMS-API) – Part 1: Guidelines and general requirements*.

IEC/TS 61970-2, *Energy management system application program interface (EMS-API) – Part 2: Glossary*

IEC 61970-301:2005, *Energy management system application program interface (EMS-API) – Part 301: Common Information Model (CIM) base*

IEC 61970-401, *Energy management system application program interface (EMS-API) – Part 401: Component Interface Specification (CIS) Framework*

IEC 61970-402, *Energy management system application program interface (EMS-API) – Part 402: Component Interface Specification (CIS) – Common Services*

Data Acquisition from Industrial Systems section Data Access (DAIS DA), OMG Adopted Specification Version 1.1, formal/05-06-01 June 2005 (Referred herein as 'OMG DAIS DA')

Utility Management System (UMS) Data Access Facility (DAF), OMG Adopted Specification, Version 2.0.1, formal/05-06-03, July 2005 (Referred to herein as 'OMG DAF')

OPC Data Access Custom Interface Specification, Version 2.05, OPC file: opcda205_cust.doc, OPC Foundation, December 17, 2000 (Referred to herein as 'OPC DA')

3 Terms, definitions and identification conventions

3.1 Terms and definitions

For the purposes of this part of IEC 61970, the terms and definitions given in IEC/TS 61970-2 apply.

3.2 Conventions

A convention used in this document to uniquely identify a UML attribute is to concatenate the class name and the attribute name with a period in between, for example, the attribute "id" in the class "Node" will then be named "Node.id". For attributes in sub structures multiple attribute names may be used, for example "Item.id.node_id" where the "node_id" is a part in the structure "Item.id".

4 CIS Specification

4.1 Background (informative)

For historical reasons, control systems for different industrial processes have evolved along different lines. Control systems for power systems have evolved on a UNIX base and control systems for most other industrial processes have evolved on a Windows base. For Windows based control systems, OPC has become the dominating standard. For UNIX based systems, the DAIS API defined in Common Object Request Broker Architecture (CORBA) Interface Definition Language (IDL) has been developed. DAIS is based on OPC to benefit from the success of OPC and enable easy bridging to OPC. With this intent Object Management Group (OMG) started in 1997 to develop a CORBA based interface with the same functionality as

OPC. HSDA has the functionality from OMG DAIS DA and OPC DA described in a technology neutral way, hence HSDA is intended to be a Platform Independent Model (PIM).

4.2 SCADA use case (informative)

A major component in a utility operational system is a SCADA system that acts as a real time data provider to other components or clients. Such a system typically has the following components:

- Process instrumentation making sensor data and actuation capabilities available.
- Remote Terminal Units (RTUs), process control or substation control systems reading sensor data and controlling actuators.
- Process communication units connecting to RTUs or substation control systems. Remote communication is typically solved by specialized RTU or field bus protocols (e.g. IEC 60870-5). The IEC 61850 series is a new standard for communication with and within substations. The interfaces defined in this part of IEC 61970 can be used as a standard API to encapsulate such communication solutions at the client side.
- SCADA subsystem making processed sensor data and control capabilities available to operators, applications or other systems.
- An Energy Management System (EMS) using the SCADA subsystem for extended processing and control.
- Graphical User Interfaces visualizing process data.

SCADA and EMS systems can be regarded as having a server part where data processing is performed and a Graphical User Interface (GUI) part where visualization and command dialogues are made. The SCADA and EMS may have common or different GUI's. The architecture is shown in Figure 1 below.

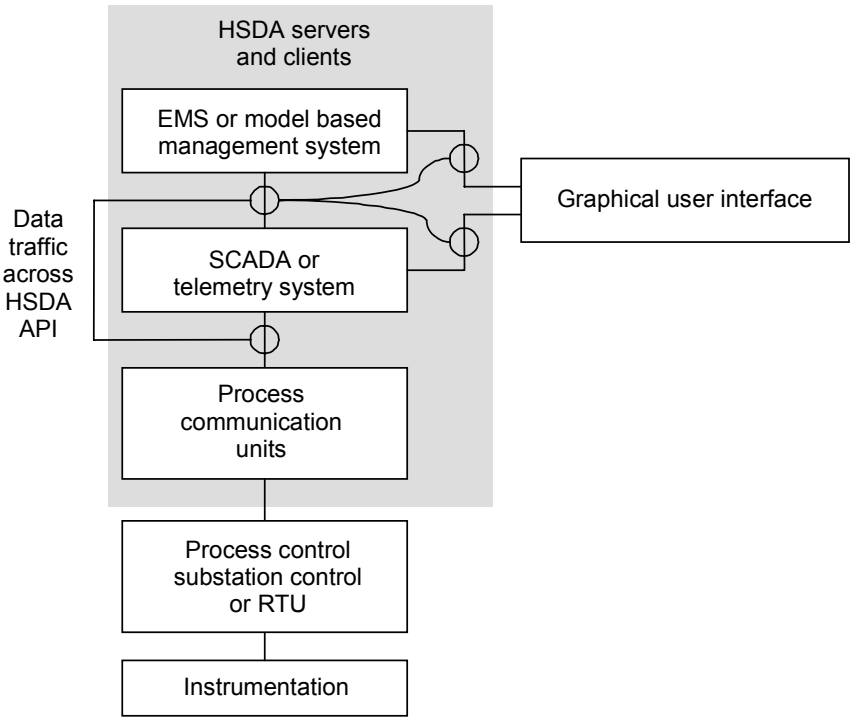

Figure 1 – Control system structure

The shaded area in Figure 1 is where it is most appropriate to use HSDA to transfer data. The components in Figure 1 interact as follows:

- The Process Communication Unit has an HSDA server that provides a SCADA HSDA client with data.
- The SCADA component has an HSDA server that provides its GUI and the EMS HSDA client with data.
- The EMS component has an HSDA server that provides its GUI with calculation results.

HSDA supports both subscription and read/write operations. The concept of subscription is described in Figure 2.

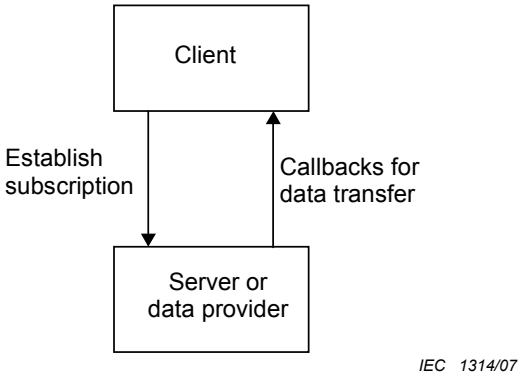

Figure 2 – Data subscription

A subscription involves a server that publishes data and clients that subscribe to receive the data. The server has no *a priori* knowledge of its clients; they become known to the server when a client creates a subscription. Once a subscription is established, the server calls the client back when data becomes available or is updated.

Data provided by a HSDA server can be of many kinds:

- Real-time data collected from the power system.
- Calculated data representing quantities in the power system.
- Parameters describing properties about the power system or equipment in the power system.
- Parameters controlling the processing of real-time or calculated data.
- Controls that can be issued to the power system.

4.3 Data model (normative)

The HSDA data model describes how data seen through the HSDA interface appears to be organized within a server. Internally, a server implementation may organize data differently, but a client using HSDA will see a data model as shown in Figure 3. A description of how the data model maps to IEC 61970-301 is given in 4.6.

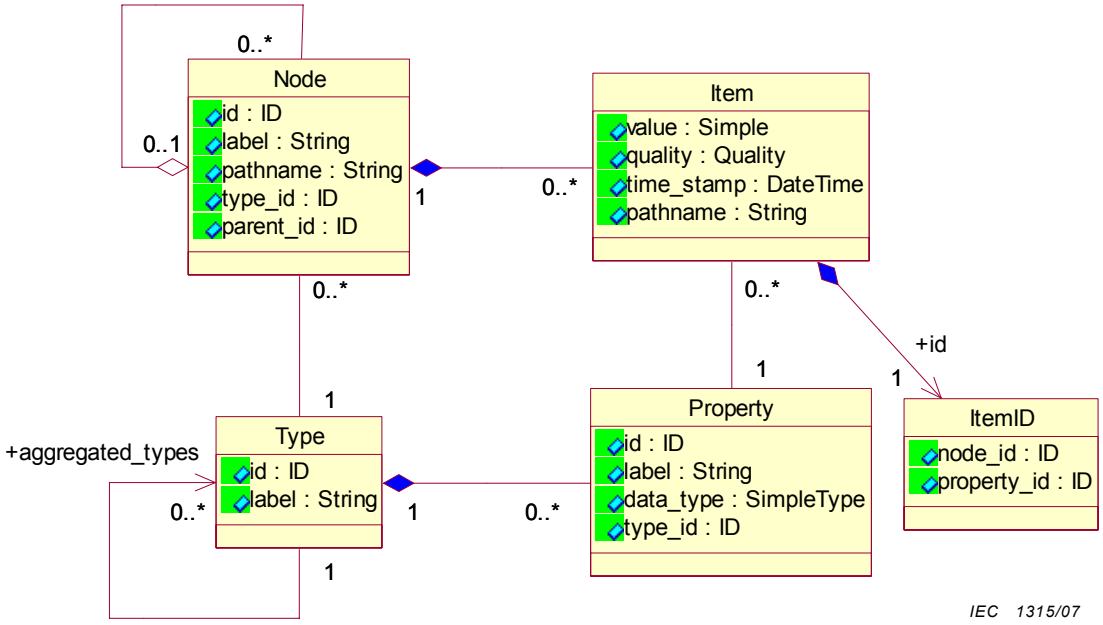

Figure 3 – DAIS Data Access Data Model

Figure 3 is a UML diagram showing the HSDA object types appearing in the interfaces:

- Type: describes the Nodes; hence Type is meta data.
- Property: describes the Items; hence Property is meta data.
- Node: describes data objects that can be accessed with HSDA. Nodes may contain other Nodes in a hierarchical structure. Nodes may also contain Items.
- Item: describes the data values that can be accessed with HSDA. Items are always contained by a Node.

The meanings of the UML attributes in Figure 3 are:

- Type.id, Property.id and Node.id are system unique identification codes of type ID. The ID type is a string or number with a range sufficient to enable globally unique identifiers. However, the id is not required to be globally unique. The id is intended for use by machines.
- Item.id of type ItemID is a system-unique Item identification that consists of the ItemID.node_id that refer to a Node and the ItemID.property_id that refer to a Property. The Item.id is intended for use by machines.
- Type.label, Property.label and Node.label are human readable names. The Type.label or Property.label uniquely identifies a Type or Property. A Node.label is unique only among the Nodes that are children of the same parent Node. The label is intended for use by humans.
- Node.pathname is a name unique within a system. It consists of all labels from the Node in question to the root. This is the same idea as for a file path. The exact layout of the Node.pathname is platform and implementation specific. However, it is suggested that whenever possible an implementation shall use existing standards, for example XPath from W3C. The pathname is intended for use by humans.
- Item.pathname is a name unique within a system. It consists of all names from the Node in question to the root and ends with the Property.label for the Property describing the Item. The exact layout of the Item.pathname is platform and implementation specific. However, it is suggested that whenever possible an implementation shall use existing standards, for example XPath from W3C. The pathname is intended for use by humans.

- Property.data_type describes the Item.value data type, for example, any string, number, boolean, etc.
- Property.type_id refers to the Type a Property belongs to.
- Type.aggregated_types enumerates the IDs of the Type of Nodes that may be children to a Node of this Type. This is used to restrict what Type of Nodes may appear as children to other Nodes in the Node hierarchy, for example, if a Node of Type Substation may contain Nodes of Type Bay or Measurement, Type.aggregated_types enumerates the Types Bay and Measurement.
- Node.type_id is the Type.id of the Type that a Node has. It implements the Node.type reference.
- Node.parent_id refers to a parent Node it exists. It creates a hierarchical structure of Nodes that is the basis for creation of the pathnames from the labels in the path to the root Node. The reference may be implemented as a Node.parent_id attribute of type ID.
- Item.value is a value of the data type Property.data_type. The data type Simple is any type not divisible into sub types, for example any type of number, string or boolean.
- Item.quality is the Quality code of the Item.value. The Quality code tells if the Item.value is valid and if not valid it supplies the reason why the Item.value is invalid. The default Quality code is "Good", i.e. a Item.value that has no particular Quality code associated with it will always appear as "Good" seen through the HSDA interface. A large number specifications defining Quality codes exists, for example RTU protocols, ICCP, ELCOM, OPC DA, OMG DAIS DA etc. This specification does not specify a unified Quality code system, but relies on the Quality codes defined in OMG DAIS DA.
- Item.time_stamp is the time when the Item.value was last updated. If no time has been assigned to an Item.value, the Item.time_stamp is left unspecified. For configuration parameters, if available, the time when the Item.value was entered or updated may be used as Item.time_stamp.

An HSDA server implementing the IEC 61970-301 will have Types, Properties, Nodes and Items recognized from the IEC 61970-301 and IEC 61970-402. Some examples are:

- Types can be IEC 61970-301 Measurement, Breaker, Substation, etc. Types are also presented by the ClassView from IEC 61970-402.
- Properties can be IEC 61970-301 Measurement.normalValue, Breaker.ampRating, etc.
- A Node corresponds to an instantiated IEC 61970-301 class, for example, a particular Measurement, Breaker, Substation, etc. The Node hierarchy is also presented by the PhysicalView from IEC 61970-402.
- Item corresponds to a property value, for example, a Measurement.normalValue equal to 500, Breaker.ampRating equal to 60, etc. Item.values may have a quality and time stamp assigned to it within a server implementation. If so, a HSDA server returns them as Item.quality and Item.time_stamp. Otherwise, the server returns unspecified Item.time_stamp and “Good” Item.quality.

Figure 4 shows an example Node-Item structure or the IEC 61970-402 PhysicalView.

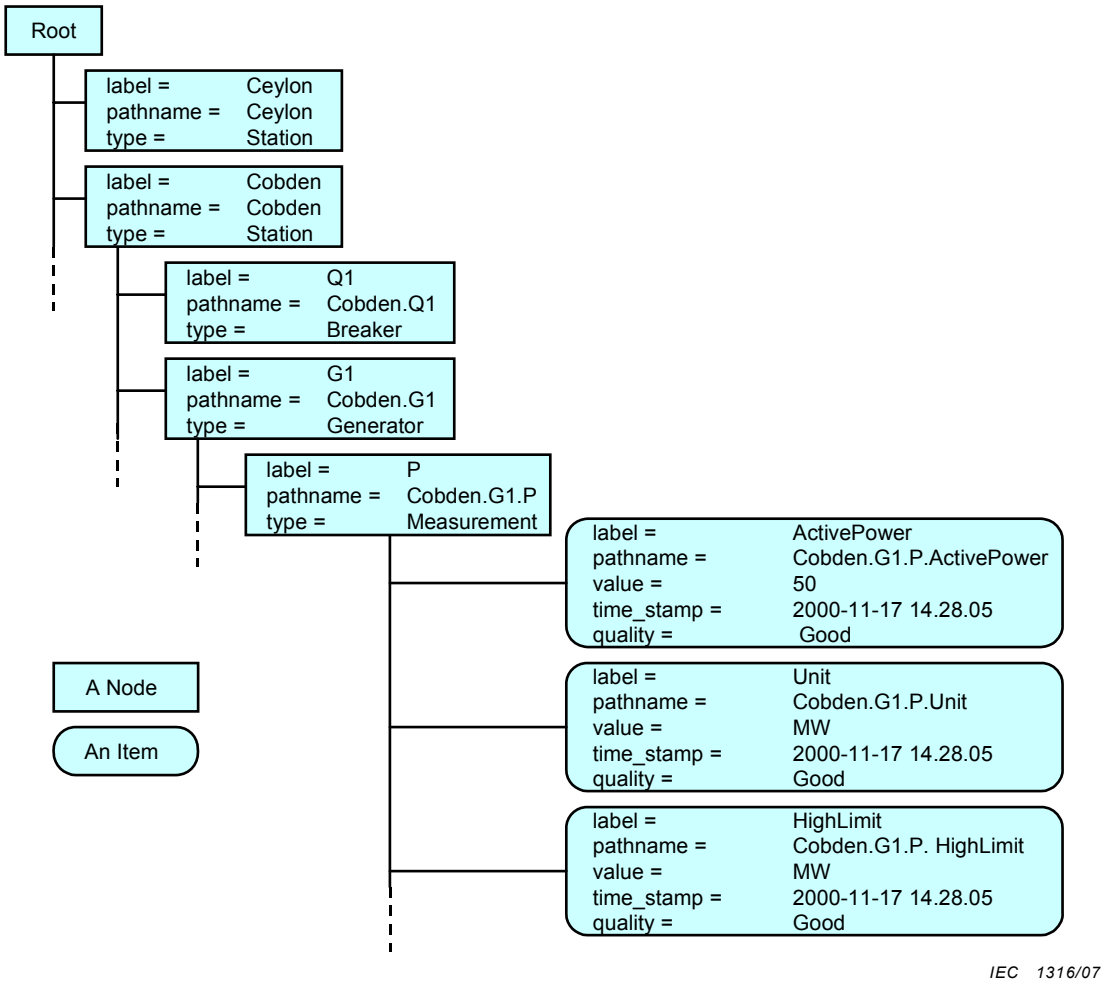

Figure 4 – Example HSDA Node and item structure

How HSDA relates to IEC 61970-301 and IEC 61970-402 is further described in 4.6.

4.4 Messages (normative)

The payload in the data messages from the server to the client contains arrays each of which consists of:

- An Item identification that uniquely identifies the Item within the server.
- A data value (the Item.value in Figure 3).
- The quality (the Item.quality in Figure 3) of the data value, i.e. if the value is reliable. If the value is bad, the quality also indicates the reason why the value is bad. The default quality code is "Good", i.e. the value is valid.
- A timestamp (the Item.time_stamp in Figure 3) that indicates when the item value was last updated.
- An Item identification that uniquely identifies the Item within the client.

4.5 Interface (normative)

4.5.1 Objects and interfaces

The object types (e.g. Server, Client, Session etc.) described in this Subclause shall all exist in an implementation if not otherwise noted. An implementation is allowed to create more than those object types described below, for example instead of letting an object type implement multiple interfaces a specific object type may implement each interface as in OMG DAIS DA.

The HSDA interfaces, objects and their relations are shown in Figure 5.

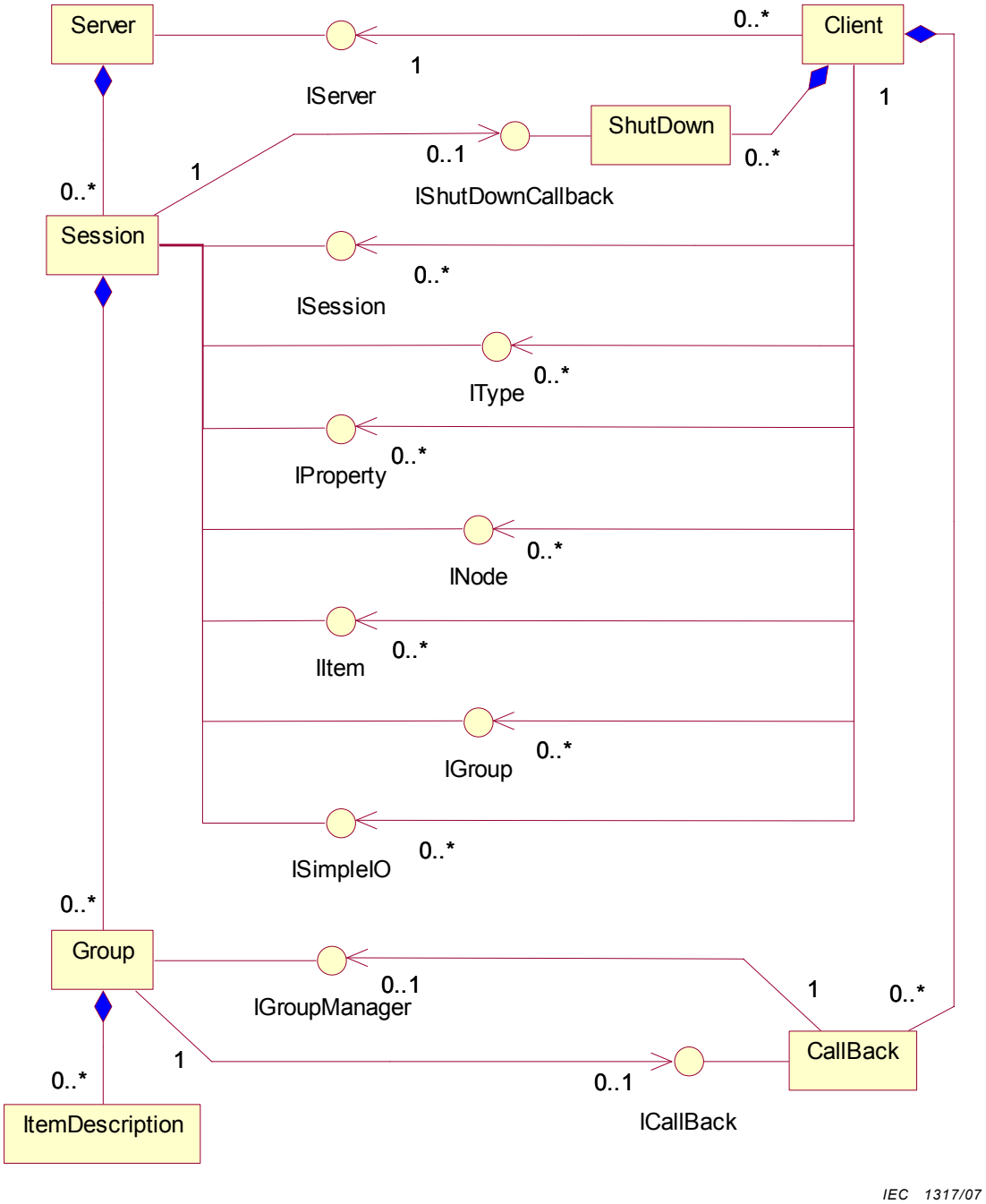

Figure 5 – HSDA objects and interfaces

Figure 5 indicates the objects and the interfaces a HSDA server and client implements. The diagram uses the UML 1 notation where interfaces are shown as small circles attached with a line to the class implementing the interface.

The Server is an object that implements the IServer interface and may have any number of Clients using it. The Server object has a number of Session objects. The Server and Session objects may be combined into one object as in OPC DA. In OPC DA each server object is also a session object.

The browse interfaces (i.e. Type, Property, Node and Item) directly correspond to the objects in the data model shown in Figure 3.

The Session object has a number of interfaces:

- ISession, which is used to manage the session.
- IType, which is a browser that is used to find meta-data about the Node data objects implemented by the HSDA server.
- IProperty, which is a browser that is used to find meta-data about the Item data implemented by the HSDA server.
- INode browser, which is used to find Node data objects that have been instantiated within the HSDA server. Among other data, the object name is the most important information.
- IItem browser, which is used to find Item values that exist for an Node object. The most important or interesting Item values are the ones with quality coded and time stamped values. But also values without a time stamp and/or quality code are also returned as Item values.
- IGroup, which is an administration interface for creation of Group objects.
- ISimpleIO, which is a data access interface that is used for direct access without need for Group objects. ISimpleIO is useful for single read or write operations.

The Session object has a number of Group objects that implements the IGroupManager interface. The Group object is a data access object used for repeated access of Item values or for management of subscriptions. Each data Item has an ItemDescription within the Group object. The data Items may have been recognized using the Node and Item browsers. The Group object implements methods for subscription, read and write. It also implements methods to populate the Group object with the Items that shall later be accessed. Group objects are created from the Session object.

The CallBack object implements the ICallBack interface and is implemented by the Client. The CallBack object is used by the Server to deliver data according to subscriptions or asynchronous calls from the Client. Each Group object may have an associated Callback object and a Client may have as many Callback objects as the number of Group objects it has created.

A Group object is used when the client wants to do repeated read calls, write calls or create a subscription. A single call for a set of items is better made using the ISimpleIO interface at the Session object.

The Group object has a few subscription parameters used to tune the transfer of call back data so that latency and network load is minimized. Such parameters are:

- Update rate, i.e., at a minimum, how often data shall be sent to the Client.
- Dead band, i.e., how large the relative change of value must be prior to sending a data item to the Client.

The Client may also have a ShutDown object that implements the IShutDownCallback interface and is used by Server to inform the Client about the Server shutting down. Each Session may have one ShutDown object registered. If a client has multiple Sessions, one ShutDown object is created for each Session.

A typical interaction sequence between a Server and a Client object is shown in Figure 6.

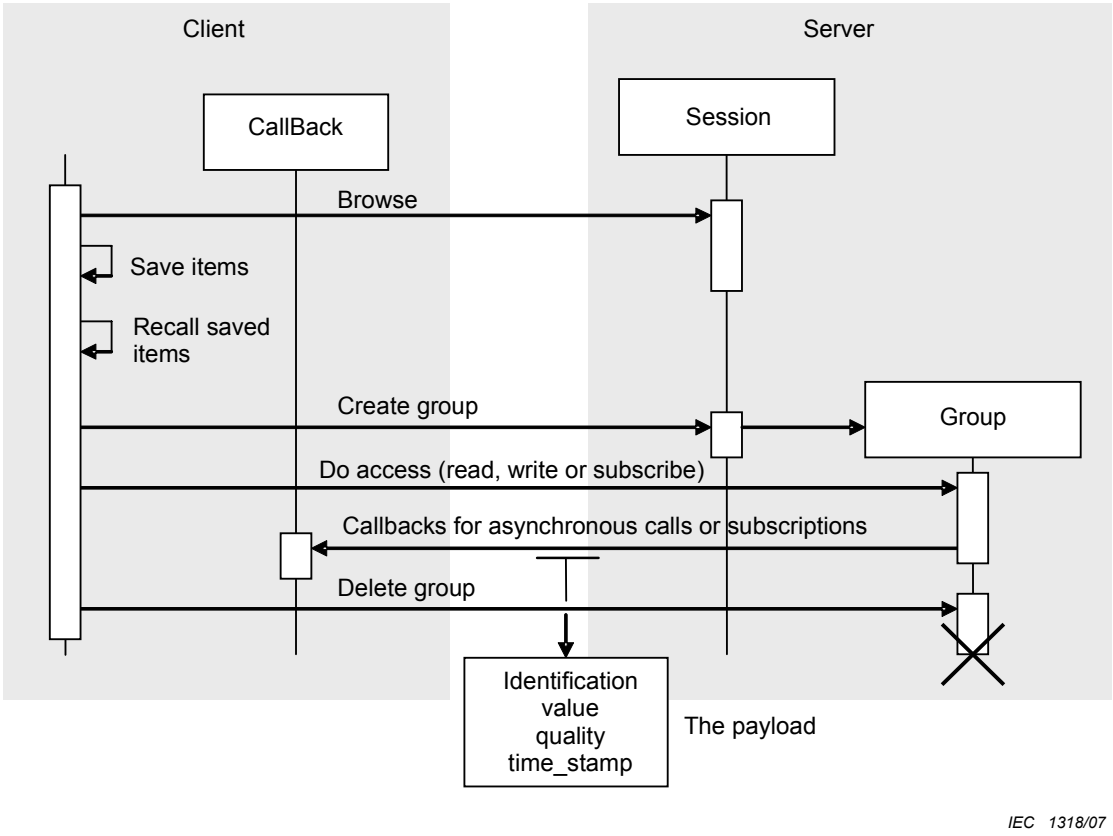

Figure 6 – Typical interaction between the HSDA objects

A Client will typically start to browse the server to find out what data is available. The browse interfaces (IType, IProperty, INode and IItem) are used for this. The Client selects a subset of the Items found when browsing and saves them for later use. The Client that does the browse can be an editor for display building, dialogue building or data base generation.

At a later time, for example, at picture call up, the saved Items will be recalled and used to populate ItemDescriptions for one or more Group objects created by the Client. When Groups are created, they can be used to read, write or subscribe for data. A subscription or asynchronous read or write call results in the Server making a callback at the Callback object. For a subscription, the callbacks will continue until the Client terminates them. The criteria used by the Server to decide when to send callbacks are controlled with the subscription parameters described above.

4.5.2 Server and session interfaces

The IServer interface has the following attributes and methods:

- a read only attribute describing the Server status, for example, server health, start time, current time, vendor information, etc.
- create_data_access_session(), which is used to create a Session object.
- create_data_access_session_for_view(), which is used to create a Session object. This method can be used if a server supports multiple hierarchies. Each hierarchy then corresponds to a view.
- find_views(), which returns the views that are supported by the server.
- a read only attribute that describes which functions supported by the interface.

The IDASession interface has the following attributes and methods:

- a read only status attribute that describes the status of the Session, for example, name, time when started, current time and number of groups.
- an attribute holding the optional ShutDown object.

4.5.3 Browse interfaces

The Browse interfaces are used to show

- Instance data, i.e. the INode and IItem interfaces.
- Meta data, i.e. the IType and IProperty interfaces.

The INode and IItem interfaces have the following methods:

- find(), which returns more information about one Node or Item specified by it's id's.
- find_each(), which returns more information about a number of Nodes or Items specified by their id's.
- find_by_parent(), which returns all children for a parent with a specified id.
- find_by_type(), which recursively returns all children with a given Type.id for a parent with a specified id.
- get_pathnames(), which translates a number of id's to the correspondingpathnames.
- get_ids(), which translates a number of i pathnames toid's.

The IType interface has the following methods:

- find(), which returns more information about a Type specified by its id.
- find_by_schema(), which returns the Type.id's for all Types that belong to a schema specified by a given id.

The IProperty interface has the following methods:

- find(), which returns more information about a number of Properties specified by their id's.
- find_by_node(), which returns all Properties for a Node with a specified id.
- find_by_type(), which returns all Properties for a Type with a specified id.

4.5.4 Group management interfaces

The IGroup interface has the following methods:

- find_public_groups(), which returns more information about all server side stored groups.
- find(), which returns more information about a server side stored group specified by its id.
- create_group(), which creates a new empty group.
- create_group_from_public(), which creates a new empty group from a server side stored Group description.
- remove_public_group(), which removes the specified server side group description from the server.

The server side groups are a service for clients to save a group description in the Server.

The IGroupManager interface has the following methods:

- create_entries(), which is used to add Items.
- validate_entries(), which is used to check if a specified number of items “link”, i.e., are present in the server.
- remove_entries(), which removes a specified number of items from the group.
- clone() that create a copy of the group.
- clone_to_public(), which saves a description of the group at the server.

4.5.5 IO interfaces

The IGroupManager is an interface at the Group object. Hence, a Group object shall be created to use this interface. The IGroupManager interface has the following IO methods:

- sync_read(), which returns data for specified items within the group.
- sync_write(), which updates data for specified items within the group.
- async_read(), which returns data for specified items within the group at the Callback object.
- async_write(), which updates data for specified items within the group. When the write is complete this is reported at the Callback object.
- refresh(), which asks for all items to be sent on the Callback object.
- cancel(), which is used to interrupt an asynchronous read, write or refresh.

The sync_read() and sync_write() method does not require a CallBack object while the others do.

The ISimpleIO is an interface at the Session object, hence no Group object is needed. It has the following methods for access that does not require a Group object:

- read(), which is used to read a number of Items specified by their ids.
- write(), which is used to update a number of Items specified by their ids.
- write_with_qt(), which is used to update a number of Item values including quality code specified by the Item.ids.

4.5.6 Client interfaces

To support the asynchronous calls and subscriptions, the Client shall, for each Session object implement a Callback object that has the ICallback interface with following methods:

- on_data_change(), which is used to receive subscription callbacks.
- on_read_complete(), which is used to receive asynchronous read responses.
- on_write_complete(), which is used to receive asynchronous write responses.
- on_cancel_complete(), which is used to receive asynchronous cancel responses.

4.6 Mapping of HSDA (normative)

The HSDA interfaces may convey data originating from an IEC 61970-301 compliant data source. For an IEC 61970-301 compliant server supporting HSDA, the interfaces shall be used as follows:

- The INode browse interface exposes objects defined by the classes in IEC 61970-301, for example Station, Bay, Measurement etc. The objects are hierarchically organized in a containment structure. This is the PhysicalView as defined in IEC 61970-402. The actual hierarchy is transparent to the INode browse interface and other hierarchical structures can be exposed.
- The IItem browse interface exposes property values defined by the classes in IEC 61970-301, for example Station.name="Cobden", Bay.breakerConfiguration="HalfBay", Measurement. normalValue ="500" etc.
- The IType browse interface expose the classes (meta data) defined in IEC 61970-301, for example Station, Bay, Measurement etc. Only classes from IEC 61970-301 instantiated as object are exposed by the IType interface. The classes may be exposed in a flat structure, i.e. the inheritance between classes is not exposed as a hierarchical structure. This is the ClassView as defined in IEC 61970-402.
- The IProperty browse interface exposes the class attributes (meta data) defined for a class in IEC 61970-301, for example Station.name, Bay.breakerConfiguration, Measurement.normal Value etc.

DAIS DA or OPC DA implements HSDA. The OPC DA API differs slightly from HSDA and this requires some mapping. Table 1 describes the mapping between HSDA UML attributes, the IEC 61970-301UML attributes and OPC DA API parameters.

Table 1 – HSDA to IEC 61970-301 mapping

No.	HSDA DAIS DA	OPC DA	IEC 61970-301 Future IEC 61970-552-4	Comment
1	Node.id	Implemented as an OPC custom property - name=NodeID - type=VT_ARRAY of 4*VT_I4	rdf:ID	In HSDA the Node.id identifies objects defined by the classes in IEC 61970-301. The Node.id is a ResourceID as defined in IEC 61970-402
2	Node.label	Item name	Naming.localName	
3	Node.pathname	Partially qualified Item name	Naming.pathName	
4	Node.type_id	Implemented as an OPC custom property - name=NodeType - type=VT_BSTR Contains the CIM class name, i.e. same as HSDA Type.label	-	Used in HSDA to retrieve meta data about the Node Type
5	Node.parent	Provided by the method IOPCBrowseServerAddressSpace:: ChangeBrowsePosition	Refer to IEC 61970-301 and IEC 61970-402 for a description of parent.child relations	
6	Item.value	Item value (PropertyID = 2)	Any CIM class attribute value	
7	Item.quality	Item quality (PropertyID = 3)	-	
8	Item.time_stamp	Item Timestamp (PropertyID = 4)	-	
10	Item.pathname	Fully qualified Item name	Corresponds to a concatenation of the Naming.pathName and a Property.label for a class attribute	
11	Item.id.node_id	Provided by the method IOPCBrowseServerAddressSpace:: ChangeBrowsePosition	Refer to IEC 61970-301 for the classes and their properties	Used in HSDA to retrieve the Node that contains the Item

Table 1 *(continued)*

No.	HSDA DAIS DA	OPC DA	IEC 61970-301 Future IEC 61970-552-4	Comment
12	Item.id.property_id	Provided by the method IOPCItemProperties::QueryAvailableProperties	Refer to IEC 61970-301 for the classes and their properties	Used in HSDA to retrieve meta data about the Item, i.e. the Property that describes the Item
13	Type.id	-	-	Used in HSDA to retrieve the Type information about Nodes. The Node.id is a ResourceID as defined in IEC61970-402
14	Type.label	Refer to the row Node.type_id in this table	Class name	
15	Property.id	PropertyID. The data type is DWORD (32 bit unsigned integer) while the HSDA Property.id is a ResourceID (a 128 bit quatity). See Note.	-	Used in HSDA to retrieve the Property information about Items
16	Property.label	The property name described in OPC DA specification	The CIM property name consisting of the class name and the class attribute name concatenated with a period (".") in-between, e.g. AnalogValue.value.	
17	Property.data_type	The OPC property data type described in OPC DA specification	Class attribute data type	
NOTE The OPC DA PropertyID data type is 32 bit long while the HSDA Property.id data type is 128 bit long. Hence to support OPC DA the allowed Property.id range is 0 to 65 534. The OPC DA specification has reserved the PropertyID numbers 0 to 4 999. The numbers 5 000 and up is for vendor specific use. Hence the allowed range for PropertyID when using OPC DA is 5 000 to 65 534. Standard assignments of numbers to HSDA Property.id and OPC DA PropertyID are outside the scope of this part of IEC 61970.				

Annex A
(informative)

Details on SCADA use case

This annex provides additional detail on the SCADA use case. In a SCADA, the HSDA interface can be used for several different purposes:

- Presentation of data from a process (collected or calculated) in various graphical displays, for example, process graphics diagrams, faceplates, forms, etc.
- Presentation and/or entry of parameter values from displays as listed above.
- An application or subsystem that reads input data from another subsystem or application, for example, a State Estimator that typically reads data from SCADA.
- Commands sent to actuators in the process.

All cases are divided into a configuration phase and a data transfer phase. The actor in the configuration phase is a maintenance engineer and in the data transfers phase, an operator as shown in Figure A.1.

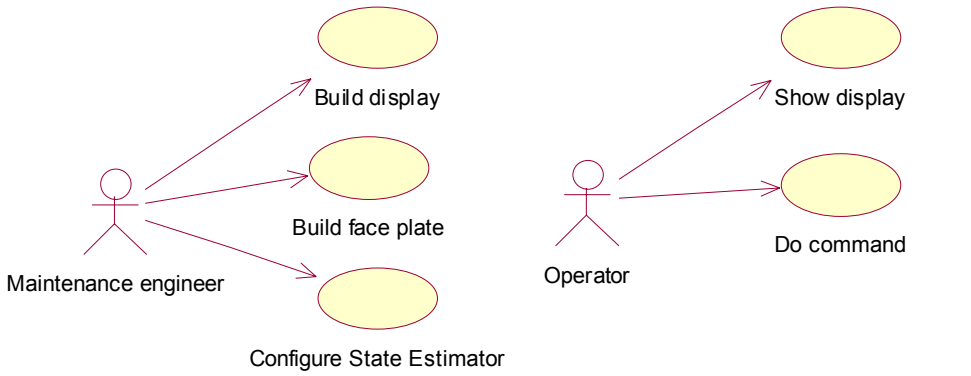

Figure A.1 – Example use cases

The actors in Figure A.1 interact with a GUI subsystem. However, for the HSDA interface, the interaction is between a data user (client) and a data provider (server). Hence, the use cases of interest at the interface are:

- Configuration of data transfer.
- Initiation of data transfer.
- Data transfer using either a read operation or subscription.
- Commands that are a combination of read and write operations.

To relate the use cases in Figure A.1 to the process graphics diagrams,

- the configuration of data transfer happens when a diagram is drawn. The HSDA browse APIs are used for discovery of the data to be presented in the diagrams.
- the initiation of data transfer happens when the diagram is called up for presentation.
- the data transfer happens for all data in the diagram after the initiation is complete and continues with changing data by subscription as long as the diagram is shown.

A typical example of an application reading data is a State Estimator reading data from SCADA. The sequence of actions is similar to the process graphics case:

- Configuration occurs when the State Estimator data is defined. Again, the browse APIs can be used to discover the SCADA input data. In an integrated Data Engineering environment, the SCADA data can be known in advance, hence, the browse API may not be used.
- Initiation of data transfer occurs at start-up of the State Estimator.
- Data transfer can be configured in different ways. For a tracking State Estimator, breaker changes are sent by subscription at change. The complete set of analog measurements is transferred to the State Estimator before start of estimation. Analogs can be transferred cyclically by subscription (the server takes initiative) or by cyclic reads from the State Estimator.

A command is made as a sequence of reads and writes on a Command object. The command object is usually shown to an operator in a specific dialogue called a faceplate (a form display having graphics similar to a faceplate). The command sequence is managed by the faceplate dialogue depending on the actions made (e.g., buttons pushed) by the operator. An example on/off Command object managed by a DAIS Data Access server is shown in Figure A.2.

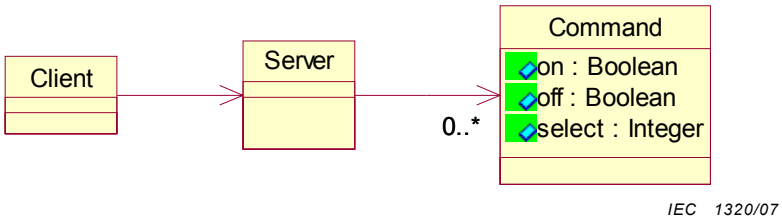

Figure A.2 – Example on/off command object

A simple sequence diagram with the basic command operations using the Command object properties is shown in Figure A.3.

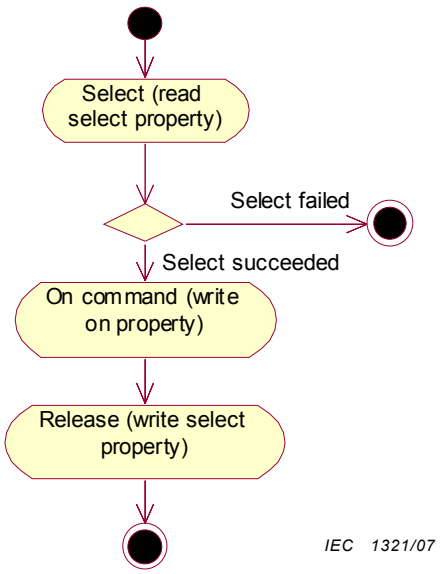

Figure A.3 – Command sequence

In the select step, the client will either get a zero or non-zero number. A non-zero number means that the select succeeded. No other client can make a new select as long as the client is holding the non-zero number. Release is made by returning the non-zero number to the server with a write.

INTERNATIONAL
ELECTROTECHNICAL
COMMISSION

3, rue de Varembé
P.O. Box 131
CH-1211 Geneva 20
Switzerland

Tel: + 41 22 919 02 11
Fax: + 41 22 919 03 00
info@iec.ch
www.iec.ch