111176 **N95-17591**

31996
P-8

STANDARDIZING THE INFORMATION ARCHITECTURE FOR SPACECRAFT OPERATIONS

C. R. Easton
McDonnell Douglas Aerospace
Space Station Division, MS 17-4
5301 Bolsa Avenue
Huntington Beach, CA 92647

ABSTRACT

This paper presents an information architecture developed for the Space Station Freedom as a model from which to derive an information architecture standard for advanced spacecraft. The information architecture provides a way of making information available across a program, and among programs, assuming that the information will be in a variety of local formats, structures and representations. It provides a format that can be expanded to define all of the physical and logical elements that make up a program, add definitions as required, and import definitions from prior programs to a new program. It allows a spacecraft and its control center to work in different representations and formats, with the potential for supporting existing spacecraft from new control centers. It supports a common view of data and control of all spacecraft, regardless of their own internal view of their data and control characteristics, and of their communications standards, protocols and formats. This information architecture is central to standardizing spacecraft operations, in that it provides a basis for information transfer and translation, such that diverse spacecraft can be monitored and controlled in a common way.

ACKNOWLEDGMENT

The Space Station Freedom Program (SSFP) funded the development of the information system standard, "Data and Object Standards" (DAOS), SSP 30552, which forms the basis of the standard recommended for spacecraft control. The SSFP adopted only the naming portion of the standard. The remainder was under consideration when SSFP transitioned to the International Space Station Alpha (ISSA) Program.

The concepts presented in this paper are not part of ISSA. Funding constraints and the newness of the technology led to a more conventional approach for initial operations. The concepts in this paper may be appropriate for inclusion in a program growth phase to simplify information management and reduce operating costs.

Several people contributed to the development of the SSFP DAOS standard. The author would like to especially acknowledge Virgil Enos (formerly with McDonnell Douglas Aerospace in Houston, Texas) and Lee Neitzel (with CTA in Houston, Texas) for their leadership in developing and prototyping the SSFP standard which forms the basis for this recommended spacecraft standard.

The concepts from DAOS have been incorporated into the Instrument Society of America, Fieldbus Application Layer Specification as a draft international standard.

INTRODUCTION

The operation and control of spacecraft requires data exchange between the spacecraft and its control center. Ground control centers must also communicate with payload specialists and technical support teams in off-site locations. With an ever increasing number of spacecraft, and with limited resources, multi-program control centers are likely to become common. It will be essential to standardize many aspects of the spacecraft and their control centers. An integrated information architecture standard will support this goal by providing a common view and understanding of data regardless of source.

The IEEE Standard Dictionary of Electrical and Electronic Terms defines data as, "any representation such as characters or analog quantities to which meaning may be assigned." An information architecture provides a formal mechanism for assigning meaning, as well as defining data in its various representations.

The purpose of an information architecture is to provide a standardized way of identifying, formatting, transporting and storing program data. Ideally, an information architecture would be in place at the beginning of a program, and would encompass all program data in a compatible format. But the reality of spacecraft programs is that such an approach does not happen. Designers develop data in various formats in many different repositories. A program approved information architecture is not available at the start if the program. Even if it were, it might not be compatible with all of the Computer Aided Design (CAD) and Computer Aided Software Engineering (CASE) tools to be used. In addition, all of the program team would have to be trained in the use of the architecture from the beginning of the program. Finally, there is a need for new, multi-program control centers to be backward compatible with existing spacecraft designs and data.

From the above, it becomes clear that an information architecture must take into account the practical realities of spacecraft programs. It must permit local representations of data in local, user defined data bases and spreadsheets. It must support data exchange among local, user defined representations. It must support data collection, integration, and validation throughout the design, development, test and evaluation (DDT&E) phases of the program. It must support the promotion of DDT&E data to the operations phase of the program, its integration with operations data and the eventual post-mission evaluation of the program. It must support the operation of multiple, dissimilar spacecraft from the same control center, including the retrofit for operation of existing spacecraft missions.

DESIRED FEATURES

A standardized information architecture would provide for a basic set features which standardize information formats, access to and exchange of data. Among these features are the following:

A common way of naming data - Data names control the access to, transport of and utilization of the data. No single naming standard will suffice for all of these purposes. The standard must define a common way of naming data which allows for various short form names for various purposes. Ideally, all of the short form names would be related to the standard by definitive rules and program specific data. All such names need to be defined in a program dictionary. The common way of naming data provides the user with a way of locating the data in the dictionary.

A common way of accessing data - Data will tend to reside in user defined repositories. Access may be locally controlled, and access procedures are also user defined. To make data readily available, the information architecture needs to include a directory indicating where data are located and how to access the data. Ideally, the architecture should support access in the requester's local representation. The dictionary/directory would then also support conversion of local syntax, format and storage between local representations.

A common way of transporting data between different local representations - Data accessible over a network may be imported into other data bases on demand or automatically imported via linkages. When the local representations differ, the import routines must be customized to make the necessary conversions. With many different local representations, conversion can become unmanageable. The information architecture should define a common transport representation which allows each local representation to map to and from a single, common representation for data exchange.

A common way of viewing information - The architecture must support user oriented views of information. These views are normally organized around the spacecraft design, subsystem function and mission operations. The same data may be important to all such views in differing contexts.

A common way of understanding information - Human users and computers need to understand the data. The understanding usually comes from defined relationships. A human operator sees a number displayed next to an icon labeled, "Pump 1 Inlet Temperature", and understands the data. Behind the display, the computer "understands" that a particular data item is the inlet temperature attribute of Pump 1. Thus, humans and computers have different needs for data and ways of understanding data. Both must be supported by the information architecture.

A common way of relating information - The same data may be used in a variety of contexts. For example, system architects are interested in device connectivity to assure failure tolerance. Hardware designers are interested in the same information for wiring harness design. Software developers need connectivity to relate I/O ports to commands and data. Test personnel need to verify connectivity and I/O function. The information architecture must support these various relations of information to context.

TOP LEVEL REQUIREMENTS

The information architecture must meet a set of top level requirements in order to be able to support a wide variety of spacecraft applications. Some of these requirements are stated below as mission goals.

Be robust enough to describe complex spacecraft and spacecraft constellations - There is a trend toward designing simple spacecraft for limited missions and using multiple spacecraft for more complex missions. While this trend may make it seem that the information architecture need only deal with simple spacecraft, the architecture should not preclude more complex spacecraft which may be developed in the future.

Support a common view of all spacecraft and payloads - The goal is to provide an operator control interface such that all spacecraft can be viewed in the same way. Note that this does not mean that all spacecraft views are identical. Rather, it means that the logical approach to accessing and working with spacecraft capabilities and functions is the same for all spacecraft. Spacecraft will not be designed with the all of the same capabilities and functions. Those which are the same may be implemented differently. As a result, the information architecture will take on a portion of the responsibility for providing the common view of the spacecraft and its operation.

Be transaction oriented to support remote operation and access - Spacecraft control is not limited to working with data local to the control center. It involves message exchange with the spacecraft, with payload specialists at various locations, and consultation with spacecraft designers and other specialists. It may involve access to remote data bases. The transaction orientation separates the action of the operator (or software) to access data from the process of accessing the data.

Provide global definitions of information and relationships - There are two aspects to this requirement. First, a program will often develop differing definitions of data to serve the design, test and operations phases of the program. Second, data and definitions will usually vary from one program to another. The global definitions serve to integrate data throughout the phases of a program and to make common data definitions available to new programs. Thus, once "Control Moment Gyro", "Greenwich Mean Time" and "CCSDS Packet" have been defined, those definitions can integrate data across a program. The definitions are portable from one program to another.

Support a variety of local representations and formats - People develop local representations to meet local needs. Some of the data in local repositories need to be made available to outside access. Most of the data can be readily converted and transferred. But then the system has multiple copies of the same data, with the attendant configuration management problems. The information architecture needs to support the exchange of data among local representations. This will not solve the configuration management problems, but will make them more tractable.

Provide for definitions to be transferred with data - Many information exchanges will be made with the definitions of the information already known. As systems become more open, there will be an increasing need to transfer the definitions of the data with the data. This will be especially important in the sharing of payload and spacecraft data with outside investigators. One major limitation on the ability of investigators to access such data is that the definitions are not available and may be permanently lost. The information architecture should support standardized definitions and the ability to store and transfer the definitions with the data.

Be well grounded in proven standards - Grounding in existing standards is desirable for two reasons. First, it is far more efficient to use or modify an existing standard than it is to develop a new standard. Second, the development and consensus building that have gone into forming an existing standard will make it easier to form a consensus on an extension to a new application of the standard.

Have the potential to support growth and technology upgrade - There are three reasons for supporting growth and technology upgrade. First, individual programs experience growth and upgrade. Upgrade may come from on orbit refurbishment, or from new generations of the same satellite. Second, a control center may be tasked to host controls for an entirely new spacecraft or constellation. Third, both the types of satellite technologies used and the technologies for hosting the information, itself, will change over time. Since the information architecture is to grow from one program to another, it must support the technology upgrade and growth.

OVERVIEW OF THE STANDARD

The Space Station Freedom Program developed an information architecture standard having the features and meeting the requirements noted above. Subsequently, the Instrument Society of America incorporated this standard into its Field Bus standard. It is being considered as a draft international standard for field buses. The same standard is also under review by the Spacecraft Control Working Group of the AIAA Space-Based Observing Systems Committee on Standards as its information architecture standard.

SSFP used the terminology, "Data and Object Standards" (DAOS) to describe the information standard. It includes standards for an integrated Data Dictionary/Directory, Object Definition, Data Modeling, and Message/Data Structure Definition. This paper will continue to use the term "DAOS" to refer to the several individual standards which make up the information architecture standard.

Dictionary/Directory Standard

DAOS uses the term "Encyclopedia" to mean an integrated Dictionary and Directory. The encyclopedia provides for a single source for definitions, contextual references and access information.

The dictionary part of the encyclopedia is based on the Information Resource Dictionary System (IRDS), (ANSI X3.138 and FIPS PUB 156). It defines classes of objects (or families of spacecraft devices) as determined by the common characteristics of real devices.

The directory portion of the encyclopedia is based on the ISO/ IEC Directory Standard (ISO 9594). ISO/IEC provides rules for naming objects and protocols for querying remote directories and receiving replies. In the DAOS encyclopedia standard, the directory provides information used to locate object instances. This includes object names, descriptions, and object specific attributes (such as location).

The encyclopedia is fully integrated, in that all information is organized about objects or devices, their classes and their relationships. This way of organizing information intermixes the dictionary and directory within object descriptions.

The encyclopedia is comprised of four layers, as shown in Figure 1 and recommended by IRDS. The top layer defines the schema for the second layer, and is not to be altered by the users.

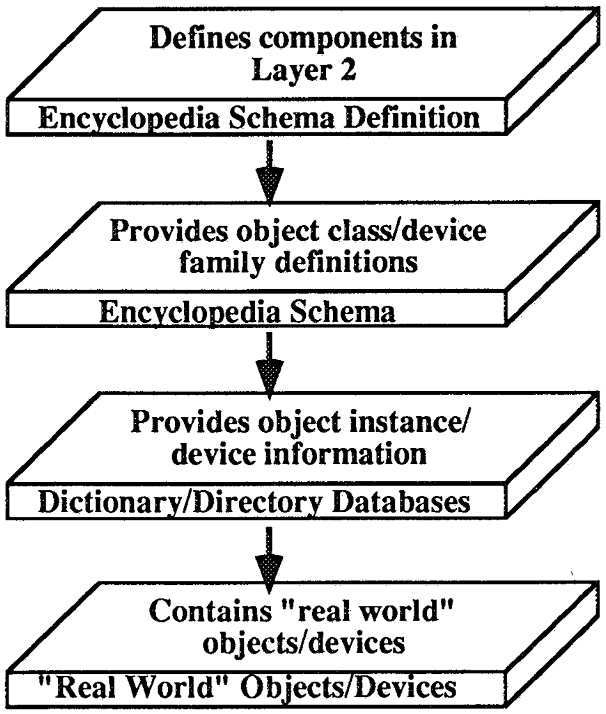
Defines components in Layer 2
Encyclopedia Schema Definition
Provides object class/device family definitions
Encyclopedia Schema
Provides object instance/ device information
Dictionary/Directory Databases
Contains "real world" objects/devices
"Real World" Objects/Devices

Figure 1 Encyclopedia layering

This top layer specifies such how objects are to be defined, how relationships are to be specified, what data modeling is assumed, etc. Thus, the top layer defines all of the tools to be used for defining object classes/device families.

The top layer can be expanded as needs for new definitions arise. Existing definitions will not normally be modified. If modifications are required, they will normally be included by creating new definitions. The definitions are not to be modified or extended by the users.

The second layer provides the object class/device family definitions. While there are differences between "object class" and "device family", there are no differences which are important to this paper. An encyclopedia may be developed using either or both.

The directory part of Layer 2 contains rules for naming data, syntax rules for storing and transporting data, and attribute or data types for common definitions of data.

Object classes will be discussed below. An object instance or device exhibits the characteristics specified for the object class or device family to which it belongs. The information about the object instances or devices is carried in layer three of the encyclopedia.

Layer four holds the actual objects or devices. As such, it is not directly a part of the encyclopedia, but is a part of the model for the encyclopedia. Layer 3 of the encyclopedia does contain the description of these real world devices.

From the above, it can be seen that each layer of the encyclopedia contains the information necessary to understand the successive layer.

Object/Device Model

The standard is Object Oriented, in that all information is categorized as either exchanged between objects or describing an object. Objects exchange actions, responses and other message types. They are defined by attributes, functions, events and behavior. The object class structure standardizes information definitions.

An object is anything that is accessible and of interest to a user. It may be a representation of a physical device, a software function, a message, or other. The generic model allows an object to be tailored to include just those portions necessary to describe it.

The object model for DAOS is shown in Figure 2. Beginning with the upper left hand side, an object communicates with other objects via messages. (Note that devices are not necessarily constrained to communicate by messages.)

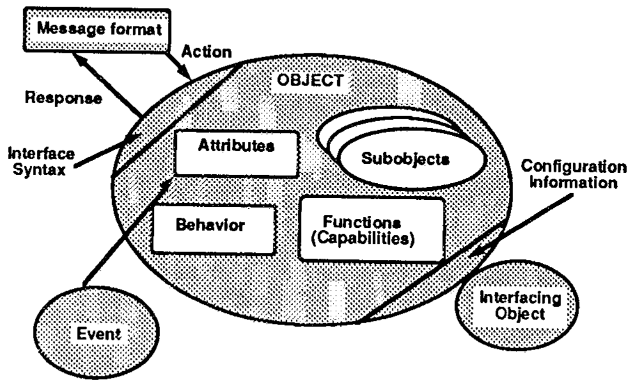

Figure 2 Generic object model

An action is a specific message type which acts as a command. Commands may result in responses such as the ability of the object to perform the indicated action. Other messages may be defined to provide more general information.

Messages require an interface syntax to define the structure and content of the message.

Attributes are data about an object. For a pump, the attributes might include its pressure, speed and temperature. Attributes might also specify the working fluid, capacity, manufacturer and serial number. In general, attributes may be variable data about a device, such as its present state, status, use, location, etc. They may also be invariant data about the device design or construction, etc. Some of the invariant data is the same for all devices of a class, and is carried as object class data.

Objects will usually perform one or more functions. The functions are described as though the "real world" object were performing them.

Objects may exhibit one or more behaviors. Behaviors may describe the way an object performs its functions, such as a telescope slewing in such a way as to avoid pointing at the earth or sun. They may describe the response to detected failures. They may describe the characteristics of action processing and the conditions for action responses.

Objects may contain sub-objects. A sub-object is an object which is wholly contained within or dedicated to another object. A coolant loop contains a circulation pump. The pump contains a bearing. The bearing has a temperature sensor. Each is an object, and has its own class, function attributes, etc.

Objects may also interface with other objects. For example, software is configured with an operating system. It may be configured with certain application software. The same drive motor may be configured with or without a brake.

An object may detect its own events, such as failures and off-nominal conditions. But it is more common for an object to be monitored by another object to prevent a possibly failed device from providing incorrect data about itself, resulting in an inappropriate failure response.

Each object must have an object class which defines the template for the object. The object class identifies each attribute, describes the functions and behaviors, defines the syntax and format of messages, and identifies sub-object and interfacing object classes.

Messages, attributes, actions, functions and behaviors all have class definitions or types. The type definitions allow for complex data types to contain other data types. Thus a quaternion is defined to contain a three component vector and a scalar, each of which may defined by units, valid ranges, precision, etc.

Data Model

The standard uses entity-relationship data modeling. An entity is anything someone wants to know something about. An entity may represent an object or a spacecraft device.

Attributes describe an entity. In this context, attributes include everything which describes an entity in isolation from other entities. The features of an object, other than its relationships, are attributes in this modeling.

Relationships describe information about an object as it is associated with other objects. Relationships include information exchanged, or messages, as well as many other types of relationships. Some of the useful relationships are shown in the figures that follow.

Figure 3 shows a "contains" relationship. In the figure, an assembly can contain one or more subassemblies. Hardware trees, indentured parts lists, bills of materials and logistics data bases will use this relationship.

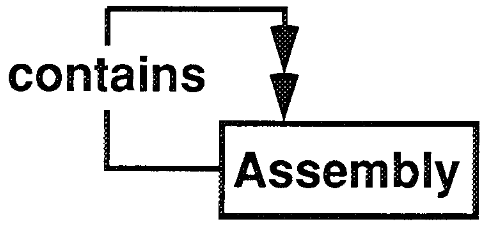

Figure 3 "Contains" relationship

Figure 4 shows the "connects to" relationship. This relationship is used to describe such things as wiring harnesses and assembly sequences.

Figure 5 shows the "communicates with" relationship. This relationship can be used to associate instrumentation with pin-outs on control devices. It can also show logical processor hierarchies and logical interfaces among software.

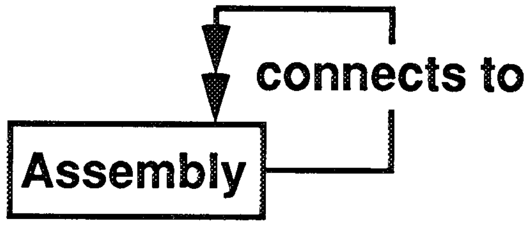

Figure 4 "Connects to" relationship

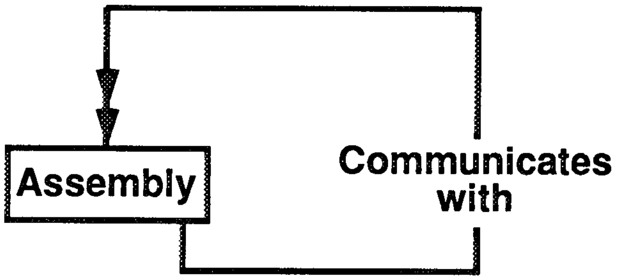

Figure 5 "Communicates with" relationship

The similar "provides value for" relationship connects the source of a data value with the process that uses the data value. A temperature sensor may provide the value for the bearing temperature attribute of a pump device. A square root library routine may provide an input to a computation.

Figure 6 shows the logical decomposition of a system. Requirements will also follow a logical decomposition, and may be allocated to the decomposition products of a system. In the figure, the functional decomposition is carried out to a point that allows functions to be allocated to physical objects such as assemblies, components, devices, software objects, etc.

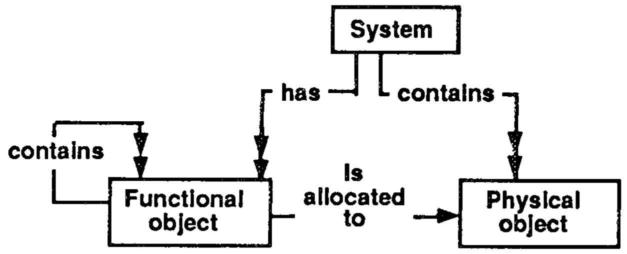

Figure 6 Decomposition relationships

The above examples show just a few of the relationships defined for spacecraft. The encyclopedia allows additional relationships to be defined as needed. The relationships may be defined on line by the users, as can the object classes, data types, and object instances.

Procedures for configuration management have been defined for a single program, but have yet to be defined for multiple programs.

Data Naming

The standard uses attribute based naming. A name is comprised of a verb (if the name is that of an action), an administrative name expression to identify the "owner", and a technical name expression to identify what is named.

No single construct of names appears to meet all of a program's needs. Descriptive name forms with logical, hierarchical structure are preferred for browsing through a dictionary or directory. The logical structure allows a user to locate items in the encyclopedia without prior knowledge of the actual name.

Descriptive names, by their nature, are long and syntactically precise. Software developers and data bases will not usually devote memory to support full descriptive names. Various aliases are not only required, but become the "official" names for a program by virtue of their usage.

The alias names should be constructed to meet two conditions. First, they need to be able to be related to a descriptive name form. The descriptive name form does not need to be actually stored in an encyclopedia if it is derivable from an alias name, the encyclopedia information and a rules set. The encyclopedia can include logic to permit user browsing without having the descriptive name form actually present.

The second condition for an alias name form is that it should be meaningful to the system users. An object instance name should include all of the parts of the attribute name, but may encode these parts and allow portions to be understood from the context. If the name is not meaningful to the users, it will not be used.

The exact construction of names must also take into account the fact that some names are inherited from one program to another. This is true of object class names, and the associated attribute, action, function, and behavior names. These names must not contain an administrative name expression that limits them to a single program. Program specific administrative name expressions are proper for object instances, but may be understood from the context of the object.

The SSFP construct for names needs a significant amount of work to be adapted to the more general usage of spacecraft control across multiple programs.

Data Format, Syntax and Semantics

The standard provides a means for defining the format and syntax of messages, data stores and data structures. Figure 7 illustrates the means for defining messages, using entity-relationship modeling. A message "contains" one or more fields. Each field contains a single data item. The data item is defined by its data type, as was previously described in the data model.

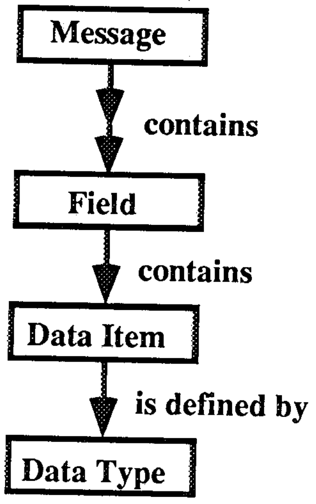

Figure 7 Message definition

Similarly, a data base contains files, which contain records, which contain fields, each of which contains a data item which is defined by a data type.

Data structures are data items which contain sub-elements which are meaningful data items. A 16 bit integer might be constructed such that individual bits represent the state of individual switches in a power control box. Each bit is a defined data item, and the integer may also be defined as a field in a message, or in processing to determine whether the measured switch state matches the currently commanded switch state. In this case, the integer would be defined as a data item with it data type definition, and the type definition would include that each bit is a state variable. In general, data structures are defined by their data types. The data type definitions include parsing rules for the structure, similar to those of a message. The data types for the individual data items in the structure are also defined by the data structure data type.

The information standard provides data type definitions for all of the program data. The data type definitions are standardized across programs. These type definitions provide the semantics (meaning) of the data. Data types may be added as needed.

DATA TRANSFER

Because there is a global set of data type definitions, it becomes possible to automate data transfers among different local representations. Each local representation needs to map its data to a common transfer syntax and format. Automatic code generators can then be used to for export and import conversions. The export conversion puts local data into the transfer syntax and format. The import conversion transforms the data from the transfer syntax and format to the destination local syntax and format. Thus, each local representation need map only to the common transfer representation to make data globally available.

CONCLUSIONS

Use of an information architecture standard will help reduce the cost of developing and operating spacecraft by providing a common view of all information. This will allow reuse of displays and controls and facilitate adapting control centers to the control of multiple spacecraft.

The proposed information architecture allows different spacecraft with different views of their data to interface with a control center using either a common view of the data for all spacecraft, or separate views specialized to each spacecraft.

The proposed information architecture standard also supports exchange of data between different local representations. It does this by defining a mapping between each individual local representation and a common data transfer representation. Only the common data transfer representation needs to conform to the standard.

REFERENCES

(April 2, 1991). *Flight Software Data and Object Standards*. (Report SSP 30552). Space Station Freedom Program Office

American National Standard for Information Systems - Information Resource Dictionary System (IRDS). (ANSI X3.138). New York, NY: American National Standards Institute, Inc,

Information Resource Dictionary System (IRDS). (FIPS PUB 156). Gaithersburg, MD: National Computer Systems Laboratory, National Institute of Standards and Technology

Information Processing Systems - Open System Interconnection - The Directory. (ISO/IEC 9594, Parts 1-8, (CCITT X.500 0 X,521)). Geneva, Switzerland: International Organization for Standardization

(April 30, 1992). *Fieldbus Application Layer Specification*. Instrument Society of America Draft International Standard