INTERNATIONAL STANDARD

ISO 5054-1

First edition
2023-02

Specification for an enterprise canonical model —

Part 1: **Architecture**

Reference number
ISO 5054-1:2023(E)

ISO copyright office
CP 401 • Ch. de Blandonnet 8
CH-1214 Vernier, Geneva
Phone: +41 22 749 01 11
Email: copyright@iso.org
Website: www.iso.org

Published in Switzerland

Contents

Page

Foreword

ISO (the International Organization for Standardization) is a worldwide federation of national standards bodies (ISO member bodies). The work of preparing International Standards is normally carried out through ISO technical committees. Each member body interested in a subject for which a technical committee has been established has the right to be represented on that committee. International organizations, governmental and non-governmental, in liaison with ISO, also take part in the work. ISO collaborates closely with the International Electrotechnical Commission (IEC) on all matters of electrotechnical standardization.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular, the different approval criteria needed for the different types of ISO documents should be noted. This document was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2 (see www.iso.org/directives).

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO shall not be held responsible for identifying any or all such patent rights. Details of any patent rights identified during the development of the document will be in the Introduction and/or on the ISO list of patent declarations received (see www.iso.org/patents).

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation of the voluntary nature of standards, the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the World Trade Organization (WTO) principles in the Technical Barriers to Trade (TBT), see www.iso.org/iso/foreword.html.

This document was prepared by Technical Committee ISO/TC 154, *Processes, data elements and documents in commerce, industry and administration*.

A list of all parts in the ISO 5054 series can be found on the ISO website.

Any feedback or questions on this document should be directed to the user's national standards body. A complete listing of these bodies can be found at www.iso.org/members.html.

Introduction

This document focuses on the need for interoperability at the business process level and the benefits for companies of being able to mix and match different vendor solutions to address their requirements. When companies want to purchase business applications from multiple vendors, they have the difficult task of getting the business applications to work together without the benefit of controlling the integration of those business applications. Additionally, customers are struggling with the larger task of integrating all of their systems into a coherent information technology infrastructure to support their business.

By defining integration standards as an enterprise canonical model, application integration can be optimized for all, instead of having different costly point-to-point solutions between each business applications, businesses and software vendors benefit.

The definition of the enterprise canonical model is a “single source of truth” for a semantic language all enterprise applications can speak, for commerce worldwide, for enterprise applications such as enterprise resource planning (ERP), purchasing, order management, finance, customer management, quality data, and more. The long-term scope is to enable all current and future enterprise application integration in a syntax-neutral manner to keep pace with rapidly changing technology landscape and resource skillsets. Other applications of canonical model include business intelligence and reporting, to normalize and aggregate business data across applications to provide a consistent vocabulary to generate business insights (see Annex B).

The enterprise canonical model reflects many thousands of person hours contributed by the Open Applications Group[1)] (OAGi) member companies.

The contributors represent the people who are building, testing, and implementing their business applications in thousands of companies worldwide.

It is anticipated that future parts of this standard will:

— Provide specification of a physical data model that underlies the model-based approach to data exchange specification. The data model is an adoption of the ISO 15000-5 standard conceptual model, as is implemented in open-source application named Score.

— Discuss the content, including the actual nouns and messages (business object documents - BOD), of the standard. The document contains a discussion of the notion of business context, including its definition and use to specify precisely profiles (i.e. subsets of) nouns. A list of integration scenarios is provided, which are the basis for determining detailed business processes that are essential for the definition of business context. Each scenario is provided with links to specific BODs appearing in the scenarios. Additionally, list of all nouns, which make up the standard, is provided. Finally, differing delivery approaches for (including canonical, standalone, and database) and alternative formats (JSON Schema,[1] XSD, etc.) are described.

— Describe the platform aspect of the standard whereby the user can create his or her own noun or message (BOD). The purpose of the platform package is for its content to be reused by other organizations. The submission describes the library of core component and fields that may be used to create new messages.

— Describe the serialization of ‘profiles’ into XML schema[2][3] definition. Also, the submission describes mappings from the BIE profiles to XML schema.

— Describe the serialization of semantic ‘profiles’ into Javascript Serialized Object Notation (JSON) schema Draft 4, and Open API Specification 3.0 schema object representation. JSON syntax has been increasingly popular with business application and API developers ever since the introduction of mobile and cloud-based technologies due to its more compact payload size for specific instances. Combined with the Representational State Transfer (REST) architectural style, JSON syntax is

1) https://oagi.org/

currently what development resources expect to use for modern application integration and what students are learning in college and vocational schools.

Much of OAGi member research has suggested a hybrid approach for REST JSON, taking the 'best of' various existing and historical efforts to standardize metadata definitions for REST include OData,[4] JSON Schema and Open API Specification. The OAGi RESTful Web API Specification describes this hybrid of OData's URI Conventions,[20] JSON Schema Draft 4 (as published by the Internet Engineering Task Force - IETF) which influenced 'Swagger 2.0', vendor specific approaches such as OpenAPI,[5] and ultimately, the Open API Specification 3.0.

Mappings from the BIE profiles to JSON schema are provided, as well as best practices to incorporate in request/response (HTTP GET)[6][7] and synchronous one-way (HTTP POST, PUT, PATCH, DELETE) methods.

Specification for an enterprise canonical model —

Part 1:
Architecture

1 Scope

This document specifies the architecture of an enterprise canonical model. It defines the abstract model, conceptual model and physical model.

This document is applicable to organizations implementing an enterprise-wide interoperability capability. Some implementers can find it useful in more targeted applications (e.g. departmental interoperability, projects).

2 Normative references

There are no normative references in this document.

3 Terms and definitions

For the purposes of this document, the following terms and definitions apply.

ISO and IEC maintain terminology databases for use in standardization at the following addresses:

— ISO Online browsing platform: available at https://www.iso.org/obp

— IEC Electropedia: available at https://www.electropedia.org/

3.1
application area
structure that conveys information required by senders, receivers and a messaging infrastructure to effectively identify, annotate, transport, route and correlate BOD instances

EXAMPLE Message instance id, sender id and correlation id.

3.2
business object document
BOD
message that conforms to the BOD architecture

3.3
BOD architecture
specification that defines the common data structure and behaviour definition for all OAGIS messages

Note 1 to entry: OAGIS stands for Open Applications Group Integration Specification.

3.4
BODID
unique identifier (UUID) of the message instance

Note 1 to entry: For UUID, see Reference [8].

3.5
BOD instance
message instance of a *BOD* (3.2)

3.6
BOD definition
definition of a *BOD* (3.2) expressed in a specific format (JSON schema or XML schema definition)

Note 1 to entry: For JSON schema, see Reference [9].

3.7
component
data structure composed of substructures and *fields* (3.10)

Note 1 to entry: See the definition of "core component" in ISO 15000-5.

3.8
data area
component that carries the business-semantic payload being communicated by the *BOD* (3.2)

3.9
data type
set of distinct values, characterized by properties of those values, and by operations on those values

Note 1 to entry: A data type can be date, time, decimal, numeric, etc.

[SOURCE: ISO/IEC 11404:2007, 3.12, modified — The term has been changed from "datatype" to "data type"; Note 1 to entry has been added.]"

3.10
field
lowest level of data elements

Note 1 to entry: These elements contain the data values.

3.11
naming convention
set of rules to build a name

Note 1 to entry: Naming conventions are specified for example in ISO/IEC 11179-5[11] and ISO 15000-5[12].

3.12
noun
component (3.7) that specifies the business-specific data being communicated (e.g. purchase order, sales order, quote, route, and shipment)

Note 1 to entry: Nouns are contained within the data area of a *BOD* (3.2).

3.13
receiver
final receiving system of the *BOD instance* (3.5)

3.14
scenario
graphical (using UML sequence diagrams) and textual description of a business process

3.15
sender
application that created a *BOD instance* (3.5)

3.16
signature
digital signature for a *BOD instance* (3.5)

Note 1 to entry: See Reference [13].

3.17
verb
component (3.7) that specifies the behaviour associated with the *noun* (3.12)

4 Cybersecurity

Cybersecurity is critical to information-system design, implementation and operation. Cybersecurity is therefore applicable to systems that build upon the concepts described in this document. However, the concepts described in the ISO 5054 series are at an abstraction level that does not require security considerations. That said, the following brief statement about related work can serve as a preview of cybersecurity-related concepts.

Cybersecurity issues are solved by risk management frameworks. The purpose of risk management is to keep risk at an acceptable level. According to the NIST cybersecurity framework,[14] "understanding the business context, the resources that support critical functions, and the related cybersecurity risks enable an organization to focus and prioritize its efforts, consistent with its risk management strategy and business needs." A key part of risk management is risk assessment. OAGi is working with NIST to explore the development of a business process management framework to be integrated with an information technology (IT) risk assessment approach to support risk assessment automation and help organizations identify their related cybersecurity risks. To this purpose, a next-generation approach and a prototype tool is in development that implements this approach to conduct IT risk assessment continuously and automatically.

5 Message architectures

5.1 General

To achieve interoperability between disparate systems, companies and value chains, there should be a canonical message architecture that provides a common meaning and approach to interoperable business processes and communication.

Messages built upon such a messaging architecture are then used for defining system interactions that establish common processes to enable interoperability. These interactions or scenarios provide a step-by-step guide that is used to perform specific business tasks. Complex scenarios created by assembling basic scenarios with additional messaging steps can then be created to fulfil business functions.

5.2 Abstract message architecture

5.2.1 General

OAGIS (Open Applications Group Integration Specification) supports different realizations of an abstract message architecture (or meta model) shown in Figure 1.

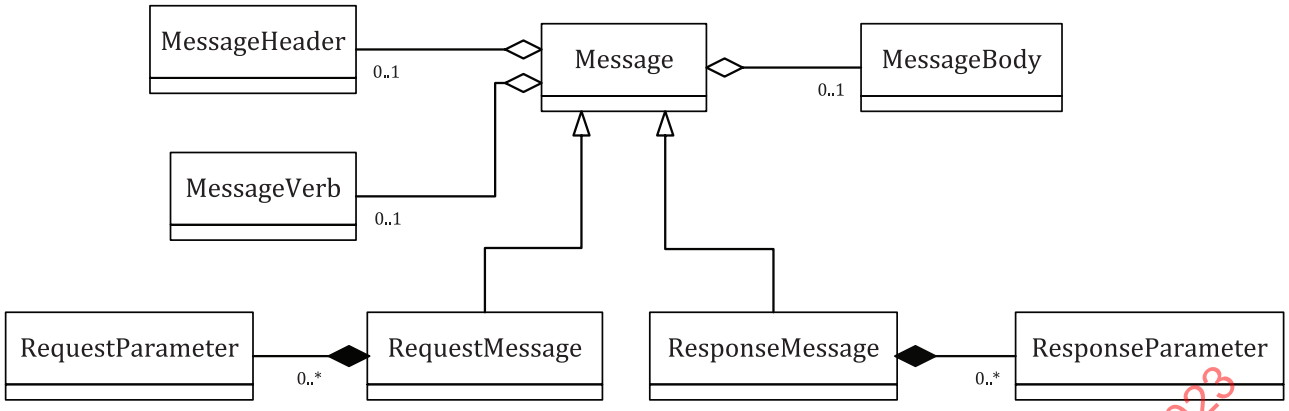

Figure 1 — Abstract message architecture

A `Message` is a definition (or specification) of conveyance of information from a sender to a receiver; it may include a `MessageHeader`, a `MessageVerb` and a `MessageBody`. The `MessageHeader` contains information about the message instance being sent, such as its creation datetime, its origin and destination, message and correlation identifiers, etc. The `MessageVerb` indicates the action or behaviour to be performed by the message receiver. The `MessageBody` contains the business data being managed by the message and may contain metadata related to the `MessageBody` instance, such as the number of records included. There are two types of Messages: `RequestMessage` and `ResponseMessage`. In addition to the message properties already described, a `RequestMessage` may have `RequestParameters`, such as selection and filter criteria.

Subclauses 5.2.2 and 5.2.3 show how OAGIS realizes this generalized message architecture.

5.2.2 Conceptual BOD message architecture

Figure 2 shows the first realization of the abstract message architecture. This realization is known as the BOD message architecture.

The business object document (BOD) realizes the message. The BOD `ApplicationArea` and `DataArea` realize the `MessageHeader` and `MessageBody`, respectively. Only the noun or component of the `DataArea` realizes the `MessageBody`. The verb of the BOD `DataArea` realizes the `MessageVerb`. The BOD request verbs also realize any `RequestParameters` of the `MessageRequest`. Similarly, BOD response verbs realize any `ResponseParameters` of the `MessageResponse`.

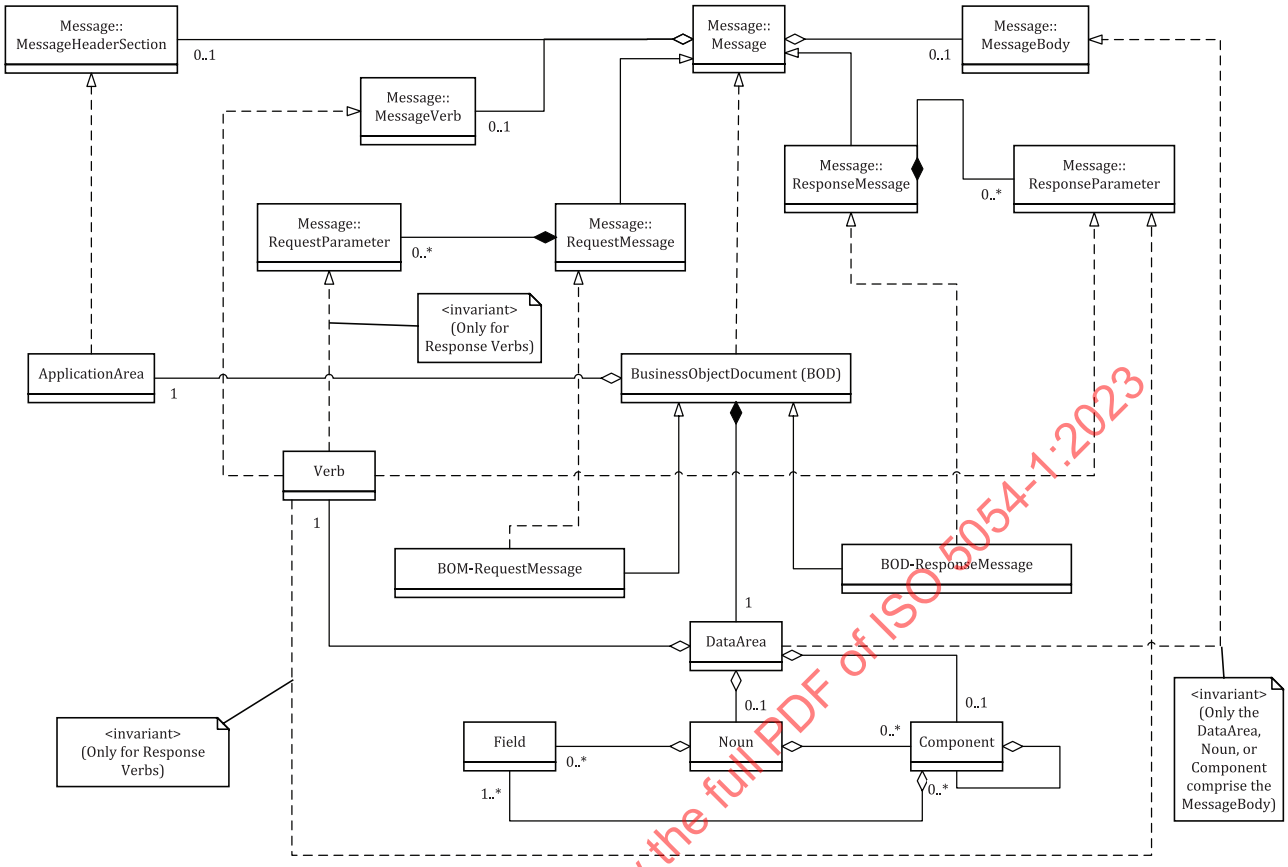

Figure 2 — BOD message architecture

5.2.3 Conceptual RESTful web message architecture

Figure 3 shows the second realization of the abstract message architecture. This realization is known as the RESTful web message architecture. A RESTful web API, as defined herein, is an application programming interface (API) that is offered by a service exposed on the World Wide Web and conforms to the REST (representational entity state transfer) architectural style; it uses the web's primary transfer protocol, the Hypertext Transfer Protocol (HTTP)[15] as the application communication protocol. Generally, an API comprises a set of named operations where each operation includes a request message and an optional response message. RESTful web API operations leverage the HTTP-Message to define their request and response messages (see Annex A). This HTTP-Message architecture[16] serves as the basis for the RESTful web message architecture.

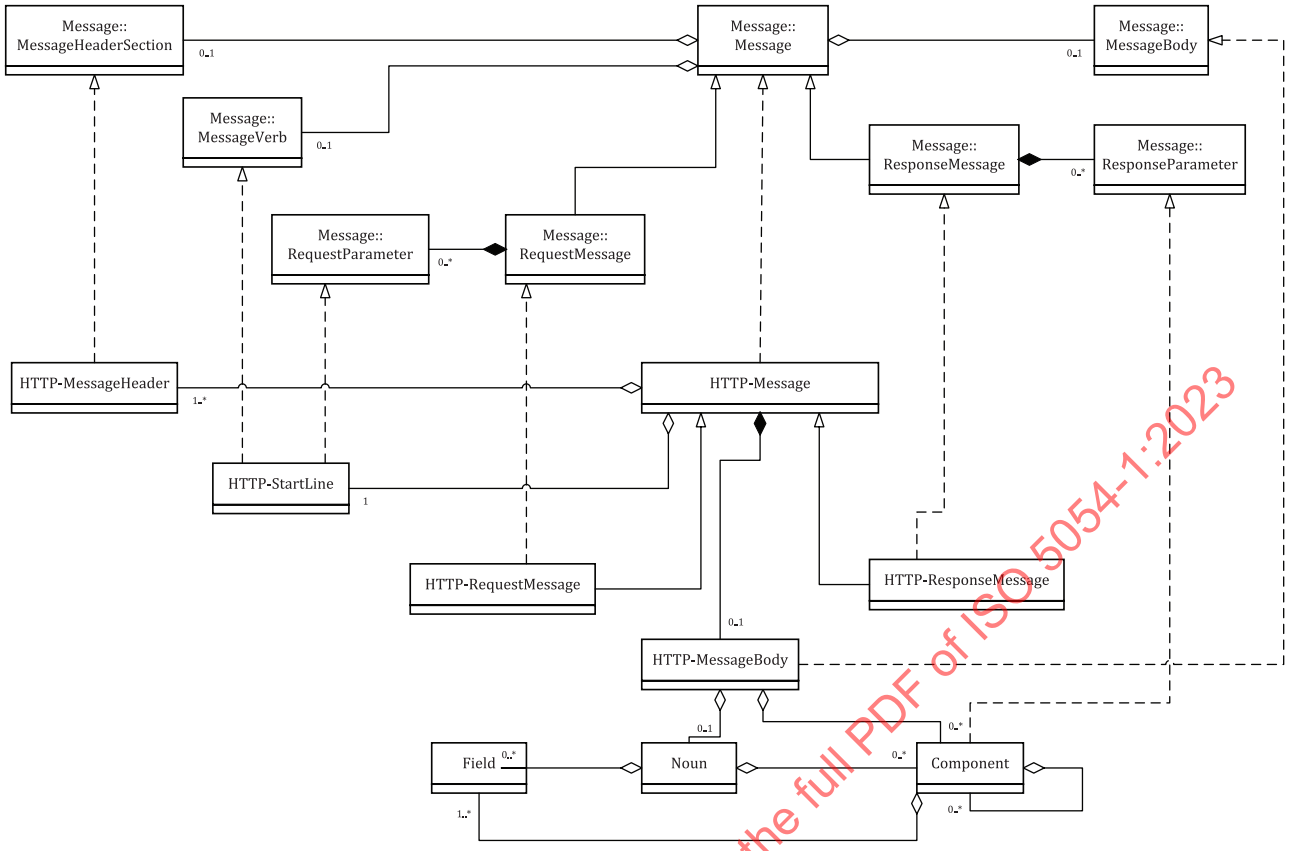

Figure 3 — RESTful web message architecture

The `HTTP-Message` realizes the message. The `HTTP-MessageHeader` and `HTTP-MessageBody` realize the `MessageHeader` and `MessageBody`, respectively. The OAGIS noun and/or component are included in the HTTP-`MessageBody`. The `HTTP-StartLine` represents both the `RequestLine` of the HTTP-Request-Message and the `StatusLine` of the `HTTP-ResponseMessage`. The `RequestLine` includes the HTTP version number, the HTTP method and URI. The `StatusLine` includes the HTTP version number, the response status code and the response status description. The `HTTP-StartLine` (i.e. the `RequestLine`) for the `HTTP-RequestMessage` realizes the `RequestParameters` of the `MessageRequest`. `ResponseParameters`, such as pagination results, are realized by components.

5.3 Physical business object documents (BODs) – message architecture

5.3.1 General

The BOD architecture defines a “master pattern” for all BOD-based messages. BODs themselves are the “specific business document masters” that define the range of possibilities that are agreed to be needed for a specific business message. The actual creation of the specific “instances” that are exchanged in scenarios is then governed by the needs of a specific interaction or exchange partner requirements.

A BOD that fulfils a “single step” in a scenario or process shall contain a wide range of possible business data that can be exchanged. By definition, BODs definitions are significantly larger than the “instance” messages that are exchanged.

The resulting instance exchange occurs between software applications that can exist within and across divisions and companies as well as between and across supply and value chains. The instance may be embedded within several transport protocols to complete a full exchange step.

A BOD shall also be able to communicate application and operational specifics such as any special conditions, errors, application requirements or status with the expected receiving system.

The BODs may be expressed in W3C XML schema (XSD)[12][17] or JSON schema, both of which represent the OAGIS enterprise canonical model. This data model was modelled using ISO 15000-5 which describes and specifies the core component solution as a methodology for developing a common set of semantic building blocks that represent the general types of business data and provides for the creation of new business vocabularies and restructuring of existing business vocabularies.

The BOD message architecture is independent of any communication mechanism. It can be used with simple transport protocols such as HTTP, FTP, and SMTP as well as more developed transport protocols such as RESTful web services, SOAP-based web services, AS1, AS2, AS3, AS4 ebMS and RNIF.

5.3.2 Business object document (BOD) details

5.3.2.1 General

The BOD message architecture may be depicted as shown in Figure 4.

The BOD architecture includes the following attributes as a part of every BOD.

— `ReleaseID` is used to identify the release that the BOD belongs. For the BODs from OAGIS 10.0 and later, the value of this attribute is “10.0”. The release ID is a required attribute of the BOD.

— `VersionID` is used to identify the version of the business object document. Each BOD has its own revision number to specifically identify the level of that BOD, not just the release ID of specification it is associated. The specific BOD version number is documented in each BOD documentation document. The outermost element name no longer includes the version number; it is instead now carried as an attribute of the BOD. The `versionID` attribute is an optional attribute.

— The `systemEnvironmentCode` is used to identify whether this BOD is being sent as a result of a “test” or as “production” level integration. Often as new systems are brought online, testing shall be performed in a production environment to ensure complete integration. This attribute allows the integrator to flag test messages as such. The `systemEnvironmentCode` attribute is an optional attribute.

— The `languageCode` attribute indicates the default language of the data being carried in the BOD message. It is possible to override this BOD level default for fields that will possibly need to carry multi-lingual information. Examples of this are notes and description.

It also performs two functions: the business data exchange and application interaction; a BOD consists of two separate sub-sections. The application-level communications (special conditions, errors, etc.) section is the application area.

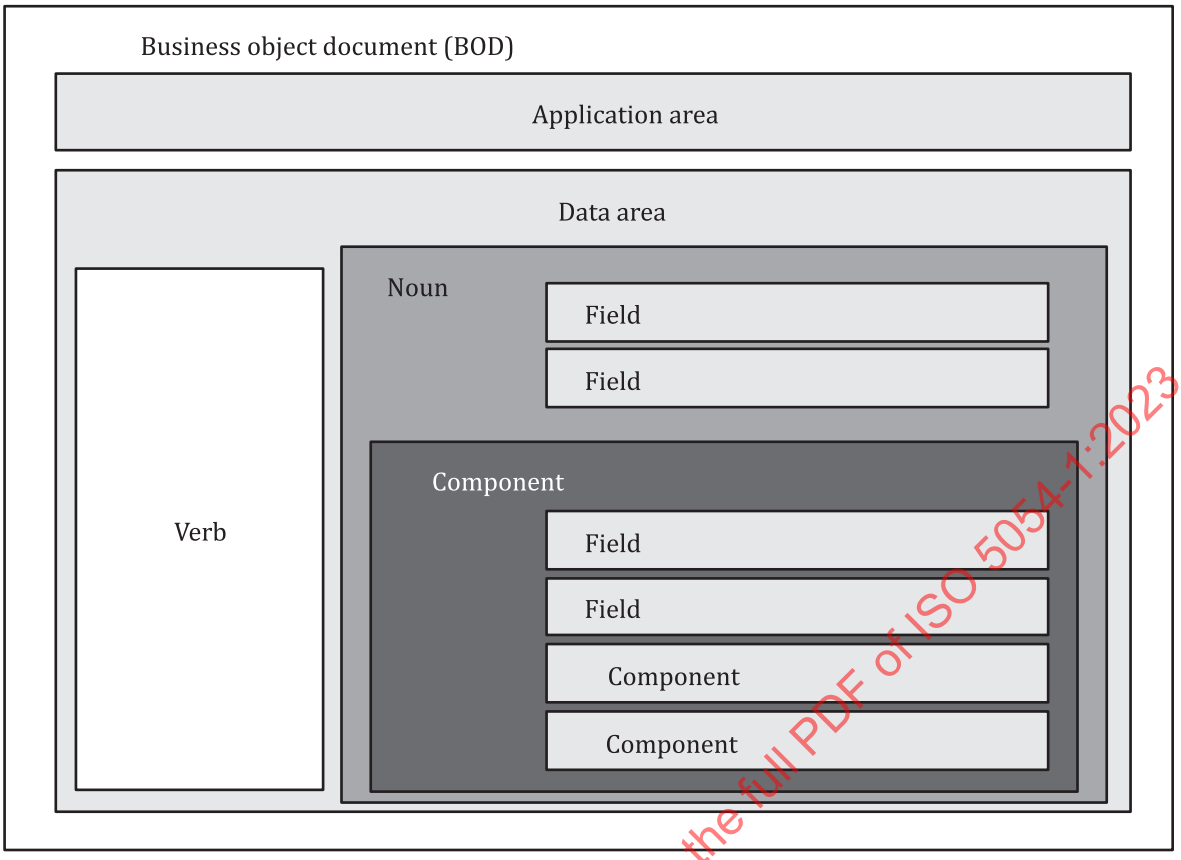

Figure 4 — BOD architecture

5.3.2.2 Application area

`ApplicationArea` includes the following (see Figure 5).

— `Sender` provides the identification of the application that created this instance of the BOD. It indicates the logical location of the application and/or database server, the application, and the task that was used to create the BOD. It also allows the sending application to request the receiving application to provide a receipt confirmation. It also provides the ability for the receiving business application to create an audit trail, to use the sender data for a more complete understanding of the information and process being communicated in this instance of the BOD.

  — `LogicalID` is the identification of the sending application from which the BOD originated. It can be used to establish a logical to physical mapping. Its use is optional. It is recommended that each system or combination of systems maintain an external central reference table containing the ID, the logical names or logical addresses of the application systems in the integration configuration. This enables the logical names to be mapped to the physical network addresses of the resources needed on the network.

  — `ComponentID` provides a finer level of control than logical identifier and represents the business component that issued the BOD. Its use is optional. A suggested naming convention is to use the application component names used in the scenario diagrams. Example components can be “INVENTORY”, or “PAYROLL”.

  — `TaskID` describes the business event that initiated the need for the BOD to be created. Its use is optional. Although the task may differ depending on the specific implementation, it is often important to enable a referential capability. Example tasks can be “RECEIPT” or “ADJUSTMENT”. Its use is optional.

- `ReferenceID` enables the sending application to indicate the instance identifier of the event or task that caused the BOD to be created. This allows tracking back from the BOD message into the sending application. This can be required in environments where an audit trail has to be maintained for all transactions. Its use is optional.

- `ConfirmationCodes` is a set of options controlled by the sender business application. It is a request to the receiving application to send back a confirmation BOD (`ConfirmBOD`) to the sender. The `ConfirmBOD` is a type of BOD message that is used to "confirm" BOD instance processing; it is business process-agnostic and capable of being used with any other defined BOD. A `ConfirmBOD` may indicate the successful processing of the original BOD or return error conditions if the original BOD processing was unsuccessful.

  The confirmation request has the following valid values:

  - `Never` – No confirmation `ConfirmBOD` requested;
  - `OnError` – Send back a `ConfirmBOD` only if an error has occurred;
  - `Always` – Always send a `ConfirmBOD` regardless.

- `AuthorizationID` identifies the authorization of the sender (user, machine, application or business), that causes the sending the BOD instance message. The receiver of the message uses this authorization to determine what levels of access the sender is allowed.

  In returning a BOD, the receiving application would pass the `AuthorizationID` back to the controller to allow the message to be routed back to the hand-held terminal.

- `Receiver`. In today's business environments with advanced technology frameworks and clouds, a single BOD may be routed to multiple destinations or receivers. In such an environment, it is not feasible for the sending system to "know" all of the possible destinations of a BOD but for audit trails and cloud tracking the `ApplicationArea` includes a receiver element that is repeatable for use in cloud or a publish and subscribe infrastructure if needed. Each occurrence of receiver permits identification of a receiving system using several ID methods.

  - `LogicalID` is the identification of the receiving application to which the BOD is being sent. It can be used to establish a logical to physical mapping. Its use is optional. It is recommended that each system or combination of systems maintain an external central reference table containing the ID, the logical names or logical addresses of the application systems in the integration configuration. This enables the logical names to be mapped to the physical network addresses of the resources needed on the network.

  - `ComponentID` provides a finer level of control than logical identifier and generally represents the business component that is to receive the business object document. Its use is optional.

  - `ID` provides an even finer level of control than logical or component identifier and represents a repeating ID that can be used to refine the business application that is to receive the BOD. Its use is optional.

    The Open Applications Group is a cross-industry consortium and does not construct or maintain any list of valid component names. However, a suggestion for naming is to use the application component names used in the scenario diagrams in section two of OAGIS. Example components can be "INVENTORY", or "PAYROLL".

- `CreationDateTime` is the date time stamp that the given instance of the BOD is created. Once assigned, this date shall not be modified during the life of the BOD. The DateTime type supports the ISO date and time format.[18]

— If the BOD is to be signed, the `Signature` is used. The choice of which digital signature is used is left up to the users and their integration needs. Signature supports any digital signature. The `qualifyingAgency` identifies the agency that provides the format for the signature.

  This element's adaptability is accomplished by not actually defining the content but by allowing the implementation to specify the digital signature to be used via an external XML schema namespace declaration. The signature element is defined to have any content from any other namespace.

  It allows the user to carry a digital signature in the instance of a BOD. For more information on the W3C's XML signature specification.[13]

— `BODID` provides a place to carry a universally unique identifier (UUID) that makes each BOD uniquely identifiable. Although optional, a UUID is needed to build any of the following services or capabilities:

  — legally-binding transactions;

  — transaction logging;

  — exception handling;

  — re-sending on condition;

  — reporting;

  — confirmations;

  — security;

  — cloud tracking.

— `Extension` is normally used only if the need arises to include more or different data in a component than is already there. `Extension` is a key feature of the OAGIS extension methodology.

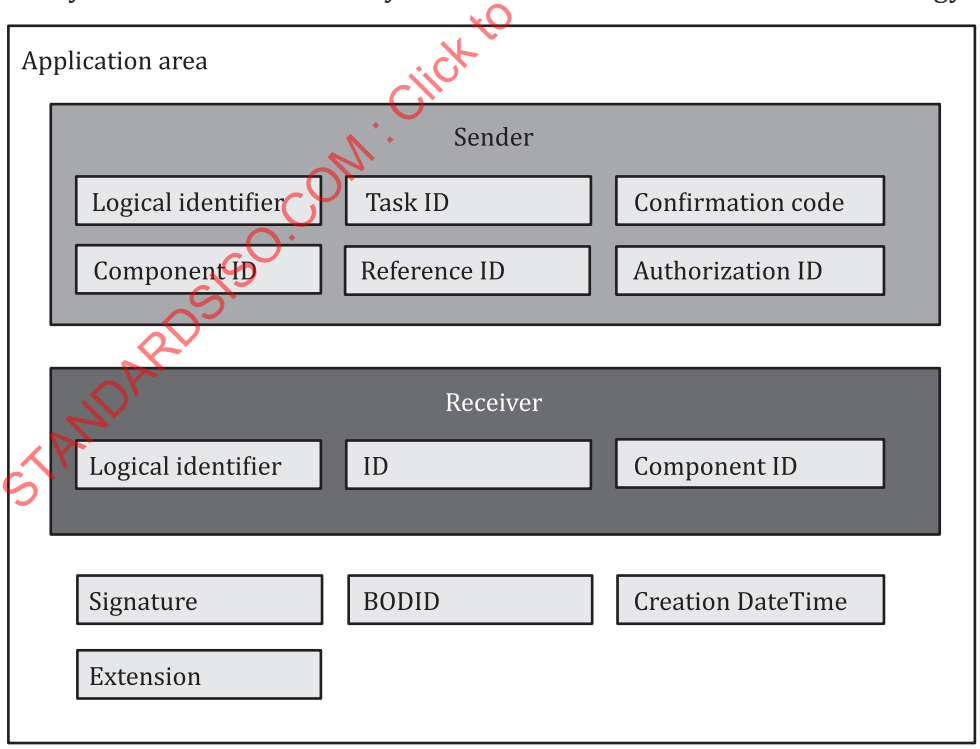

Figure 5 — Application Area

5.3.2.3 Data area

The actual business information is found in the data area (see Figure 4). The data area addresses two more unique requirements for business messaging through its own two sub-sections. It is typical to require a specific action be taken against the business data being exchanged.

— The desired action is defined by a using a verb. It defines the action or actions that are to be performed on the included business information. Table 1 provides a list of the different verbs and the combinations of request and response verbs used.

  Any additional information that is exclusively related to the action indicated by the verb is also carried within the verb element. For example, a process verb indicates that the requester is asking for a specific task to be performed by the receiving system and that the task is acknowledgeable and confirmable.

— The business information itself is contained in the other sub-section, the noun. The noun is composed of reusable groupings of business data and structures that fulfil a defined business "step" or action. The noun itself is made up of common, repeatable components which may be composed of components and fields. Likewise, the nouns themselves may be composed of fields.

  — Components comprise other, typically smaller, components which can consist of other smaller components in combination with fields. Each is a common repeatable business element that fulfils a portion of the noun's business function (i.e. purchase order header, line, address, contact, etc.). Most of the components are basic shared building blocks that are used by the BODs.

    Components obtain part of their unique meaning through contextual use (they are partially defined by what element or component they reside in). To foster reuse, this document discourages the use of additional descriptors unless a new, unique component is needed. The higher level "containing" component drives the contained component's value. So, a line component within an order component uses common component quantity to define the number of "things" that are being ordered on that line.

  — Fields are fundamental elements that are also used to create components and nouns.

Table 1 — Verbs pairs and examples

Request verb	Verb response	Example
`Process`	`Acknowledge`	Request: `ProcessPurchaseOrder` Response: `AcknowledgePurchaseOrder`
`Post` (synonym for process in financial scenarios)	`PostAcknowledge`	Request: `PostJournalEntry` Response: `PostAcknowledgeJournal`
`Change` (with options selected for add, change, delete, or replace)	`ChangeAcknowledge`	Request: `ChangePurchaseOrder` Response: `ChangeAcknowledgePurchaseOrder`
`Cancel`	`CancelAcknowledge`	Request: `CancelPurchaseOrder` Response: `CancelAcknowledgePurchaseOrder`
`Sync` (with options selected for add, change, delete, or replace)	`SyncResponse`	Request: `SyncCustomerAddress` Response: `SyncResponseCustomerAddress`
`Load` (synonym for sync in financial scenarios)	`LoadResponse`	Request: `LoadActualLedger` Response: `LoadResponseActualLedger`

Table 1 *(continued)*

Request verb	Verb response	Example
`Get`	`Show`	Request: `GetPurchaseOrder` Response: `ShowPurchaseOrder`
`Notify`	(No response required)	`NotifyRemittanceAdvice`

5.3.2.4 Extensions

The final piece of the architecture fulfils a business level standards requirement that is not seen in lower level, non-business standards such as those found in communication or transport. It addresses the requirement to adapt to differing needs while still providing an interoperable base. It is well understood that different industries have different requirements; this document is designed to address the horizontal needs of cross industry as well as the lines of business themselves. It shall offer a way to "customize the messages in a standard way." It shall be extensible using common methods and components.

Extensibility allows industries and value chains to plug in additional, non-standard information that is unique yet repeatable across their processes using standard components.

Extensibility is accomplished through a common methodology based on reusable components that can be "plugged" in "where needed" and that can be built onto. This document provides and maintains the base standard while offering this adaptability when required. In this regard, this document is unique to the category of business standards and meeting needs.

Annex A
(informative)

Physical RESTful web messages

A.1 General

RESTful web message architecture is based upon the HTTP-Message architecture described in the HTTP specification. There are, however, several API design-related concerns that are not addressed in the specification and left as an exercise to the adopter. Some of these concerns include: API versioning, URI design and format conventions, approach to managing resources (e.g. how to express filter and selection criteria in a read request message, how to express pagination criteria and results), resource representation conventions, adoption and use of additional headers and response codes. A quick web search typically reveals a multitude of approaches and no clear standard for addressing these RESTful web API design-related concerns. RESTful web message architecture addresses these concerns with additional details that both maintain conformity and extend the HTTP-Message architecture of the HTTP specification. Subclause 5.2.3 provides an overview of the major architectural elements of the RESTful web message architecture. The details of the full specification are available at OAGI[2)].

A.2 Message headers

The message headers of the RESTful web message are similar in purpose to the BOD ApplicationArea (5.3.2.2). This document has identified a set of headers that may serve as HTTP-MessageHeaders in its RESTful web message. The set of headers include: a subset the headers defined in the HTTP specification, certain headers that, while not being part of the HTTP specification, have been specified and proposed to the Internet Engineering Task Force (IETF), and a set of custom headers.

A.3 Message verbs

The message verbs of the RESTful web message are similar in purpose to the verb of the BOD DataArea (described above). This document has identified a set of verbs that may serve as methods in the HTTP-StartLine of the HTTP-RequestMessage (i.e. known as the request-line). The set of verbs include: the methods defined in the HTTP Specification as well as the Patch Method that while not part of the HTTP specification, has been specified and proposed to the Internet Engineering Task Force (IETF). In addition, to these methods, this document supports the concept of a controller resource. A controller resource represents a procedural concept that does not correspond to one of the standard or proposed HTTP methods.

A.4 Message body

The message body of the RESTful web message is similar in purpose to the noun or component of the BOD DataArea (5.3.2.3). HTTP-MessageBody of the RESTful web message may include the noun and/or components. Note that components may represent business data (e.g. address) or response metadata (e.g. PaginationResponse); it is in the latter case that a message instance may include both, a noun instance and a component instance or two component instances.

2) www.oagi.org

Annex B
(informative)

OAGIS availabilities

B.1 General

An example of what is possible with this architecture OAGIS is used as an example. There are three types of deliverables in OAGIS including the normative data model, the non-normative data model documentation, and the non-normative scenarios. This clause describes each of these deliverables and the various forms in which they are available.

OAGIS uses the architecture described in this document, standardized messages and example scenarios that are used as a basis for most integrations. By simply selecting a scenario that most closely matches an integration's needs, BODs or nouns are identified for accomplishing the desired business integration or processes, depending on whether message-oriented middleware (BODs) or REST (nouns and components) is leveraged for the integration.

OAGi provides tools to then 'profile' the selected BOD or noun to fit the business process and industry context of the integration. This involves selecting the fields required, add contextual semantics definitions to aid the developer, and generating the syntax required for the business application integration or application programming interface (API) required.

B.2 Data model

Data model is the core of the OAGIS standard. The data model is available in three types of packaging and in various syntactical representations, all of which conform to the OAGIS canonical model. The canonical model provides the terminology, semantics and structure common for all forms of deployments. These availabilities allow the OAGIS deployments to be flexible while maintaining the conformity.

At the time of publication, the OAGIS canonical model is the OAGIS enterprise in the XML schema representation. Going forward, OAGIS canonical model will be the OAGIS enterprise offered in the ISO 15000-5 conformant database and application. OAGIS enterprise and its various forms are described below.

B.3 OAGIS packages

B.3.1 General

OAGIS is packaged and available to the users in three different ways. Each of these packages is also available in differing forms as described in B.3.5.

B.3.2 OAGIS enterprise

OAGIS enterprise is a complete OAGIS package. It is inclusive of the other two packages, namely, the standalone and platform described in B.3.3 and B.3.4. OAGIS enterprise contains the whole OAGIS data model.

OAGIS enterprise modularize the specification with respect to the OAGIS message architecture. Entities in the data model are modularized into BODs, nouns and platform. The platform includes BODs that are used for data migration (independent of business semantics such as table), extension, reusable common components, fields, code list, agency identification list, business data types and primitive types.