Business Object Document (BOD) Message Architecture
For OAGIS® Release 10.+

an OAGi White Paper

Document #20130909V1.0

Open standards that open markets™

Open Applications Group, Incorporated – OAGi®
A consortium dedicated to simplifying business across all value and information chains
www.OAGi.org

Copyright 2013 OAGi
Contents

1. Architecture Overview ............................................................................................................. 3

1.0 OAGIS BODs Architecture ................................................................................................... 7

1.0.1 BOD Contents Overview .................................................................................................. 8

Application Area ....................................................................................................................... 8
Data Area ................................................................................................................................... 8
Verb ........................................................................................................................................... 9
Noun ........................................................................................................................................... 9
Components ............................................................................................................................... 9
Fields ......................................................................................................................................... 9

1.1.1 BOD Attributes ................................................................................................................ 11
release ID .................................................................................................................................. 12
version ID ................................................................................................................................. 12
system Environment Code ....................................................................................................... 12
language Code ......................................................................................................................... 12

1.1.2 Application Area .............................................................................................................. 13
Sender ...................................................................................................................................... 14
Receiver ................................................................................................................................. 17
CreationDateTime .................................................................................................................... 19
Signature ................................................................................................................................. 19
BODID ....................................................................................................................................... 19
Extension .................................................................................................................................. 20

1.1.3 Data Area ......................................................................................................................... 20
1.1.4 Verb .................................................................................................................................. 21
1.1.5 Noun .................................................................................................................................. 22
1.1.5 Verb/Noun Combinations - BODs.................................................................................................................. 23

1.1.6 Components .................................................................................................................................................. 23

1.1.7 Fields .......................................................................................................................................................... 24

2.0 OAGIS Extensions .......................................................................................................................................... 26

3.0 OAGIS Constraints ......................................................................................................................................... 27

4.0 OAGIS Error Handling .................................................................................................................................. 28

5.0 OAGIS Summation .......................................................................................................................................... 29

Architecture Overview

In order to achieve interoperability between disparate systems, companies, and value chains, there must be a horizontal 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 in defined system interactions that use 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 fulfill any needed business function.

The Open Applications Group Incorporated (OAGi) creates and maintains the Open Application Group Integration Specification (OAGIS) which consists of this architecture, 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, the messages or Business Object Document (BODs) are identified that will accomplish the desired business integration or processes.
This document describes the architecture of the OAGIS specification and the derived BOD messages. Together they provide both an interoperable horizontal messaging architecture as well as a full function vertical architecture to meet most all integration needs.

The BODs themselves are expressed as schemas that are refined subsets from the OAGIS library – both from the platform as well as the OAGIS standard.

The BOD Architecture defines a “master pattern” for all OAGIS 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 will fulfill a “single step” in a scenario or process must contain a wide range of possible business data that could be exchanged. So by definition, BODs are much, much larger than the actual “instance” messages that are actually exchanged.

The resulting instance exchange occurs between software applications that could 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 must also be able to communicate application and operational specifics such as any special conditions, errors, application requirements or status with the expected receiving system.
To perform both functions of business data exchange and of application interaction, a BOD consists of two separate sub-sections. The application level communications (special conditions, errors, etc.) section is called the BOD’s Application Area. The actual business information is found in the other area called the Data Area.

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 or a functional directive. It defines the action or actions that are desired for the included business information. The business information itself is contained in the other sub-section called a 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 other components and fields.

The final piece of the OAGIS Architecture fulfills 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; OAGIS is designed to address the horizontal needs of cross industry as well as address the lines of business themselves. So OAGIS must offer a way to “customize the messages in a standard way.” It must be extensible using common methods and components.

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

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

Finally, to provide the greatest range of value to implementations, the OAGIS BOD Message Architecture is independent of the communication or transport mechanism used to exchange the actual messages or BOD Instances. OAGIS can be used with any transport protocol from
simple FTP, HTTP and SMTP to the more complex transport protocols such as SOAP, ebXML. It can even be used with vendor specific transport and routing, within other Enterprise Application Integration (EAI) or Enterprise System Bus (ESB) systems.

To review then, a BOD is an XML Schema (.xsd) “master pattern.” This BOD pattern is used to create a BOD Instance or the XML (.xml) messages that are actually exchanged. The meaning of each is based the business context and grouping.
1.0 OAGIS BODs Architecture

Graphically a BOD consists of multiple parts:

The basic parts that make up a BOD are:

1. An envelope containing the entire message (the BOD)
2. A section for application specific data (the Application Area)
3. A section for business data (the Data Area)
4. A section for the business action being requested (the Verb)
5. A section for the business data being acted upon (the Noun)
6. Groupings of different elements that contain sub-sets of business information that are repeatable and reusable (the Components)
7. Groupings of basic business data or information that are repeatable and reusable (the fields)
1.0.1 BOD Contents Overview

The first inner layer within the BOD envelope consists of both the Application Area that conveys message specific information and the Data Area that contains the actual business information:

![Diagram of BOD structure with Application Area and Data Area]

**Application Area** - communicates information that can be used by the infrastructure and the involved applications to handle and process the actual exchanged instance such as code revision levels and environmental specifics for instance use (re: Test or Production), a unique message ID or digital signature information.

**Data Area** - carries the requested business action and specific payload or data being communicated by this BOD instance.

The next inner layers inside the Data Area contain the actual business data for this instance within the Verb and Noun:

![Diagram of Verb and Noun within Data Area]

**Figure 3**

**Figure 4**
Verb - the Verb identifies a desired business action to be performed on the specific Noun or Nouns contained within.

Noun - Nouns contain grouped business specific data that is being communicated (i.e. PurchaseOrder, SalesOrder, Quote, Route, Shipment, etc.).

Within the Noun and the Verb is the actual business information that is to be used or acted upon. It consists of very structured groupings or Components, some of which are repeatable and most of which are optional.

Figure 5

Components - Components are building blocks or reusable sub-sections that are grouped for a business reason. They are comprised of other, typically smaller components which can consist of other smaller components in combination with fields. Each is a common repeatable business element that fulfills a portion of the Noun's business function (i.e. Purchase Order Header, Line, Address, Contact, etc.). Most of the Components are even more basic shared building blocks that are used by all BODs.

Within OAGIS, 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, OAGIS discourages the use of additional descriptors unless a new, unique component is needed. Within OAGIS, 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 - Fields are the lowest level elements defined in OAGIS. Fields are fundamental elements that are also used to create Components.

Note: To even further expand the OAGIS interoperability value, many Components and Fields in OAGIS are derived from other standards such as UN/CEFACT and ISO.
1.1 Business Object Document (BOD) Details

Note: Many of the graphics within this document are illustrations courtesy of XML Spy. They are shown here as a way for the reader to visually see the constructs are being defined and not as a specific endorsement of any application. However, OAGI does recommend using an XML Integrated Development Environment (IDE) to simplify working with complex XML Schema languages like OAGIS and we do use XMLSpy for development, documentation and for training and services.

The term Business Object Document or BOD refers to both the “master pattern” or schema (.xsd) as well as the subset actually exchanged or the BOD Instance (.xml). This section will explain the master pattern or the schema view but also applies to any instance that is created from the schema. Note again, that since OAGIS desires to minimize the actual instance sizes, most elements within a BOD are optional.

Note also that the OAGIS Architecture relies on relationship specific needs and methods to enforce content.

![Diagram of BOD structure]

Figure 6

The general structure for all BODs is:

![Diagram of BOD structure with BusinessObjectDocument, ApplicationArea, Verb, Noun, and DataArea]

Figure 7

For a given BOD, the generic names (BusinessObjectDocument, Verb, Noun) are refined by using specific names (ProcessPurchaseOrder, Process, and PurchaseOrder) for each resulting in something similar to:
The child elements of a BusinessObjectDocument then are (figure 9):

- **ApplicationArea**
- **DataArea**

The Application Area and Data Area separate the application-specific information from the information that is specific to each type of BOD. Each is discussed in more detail in the following sections.

### 1.1.1 BOD Attributes

In addition to the Application and DataArea elements, each BOD contains four attributes to permit a more detailed view of the BOD, the BOD's `releaseID`, `versionID`, `systemEnvironmentCode`, and `languageCode`. Note that the names for attributes use a naming convention called “lowerCamelCase.”
release ID

release ID is used to identify the release of OAGIS that the BOD belongs. For the BODs from OAGIS 9.+ the value of this attribute will be “9.0”. The release ID is a required attribute of the BOD.

version ID

version ID 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 OAGIS. The specific BOD version number is documented in each chapter of OAGIS. 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.

system Environment Code

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

language Code

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 may need to carry multi-lingual information. Examples of this are Notes and Description.
Note that XML supports only one encoding for an XML message - as such the languages carried within a BOD are limited to the set that the XML encoding can support.

1.1.2 Application Area

![Diagram of BOD and Application Area](image)

Figure 11

The ApplicationArea is meant to convey information that a middleware, intermediate application or final application may need to know in order to perform a business function effectively. It is a combination of and sometimes a duplication of some of the information needed or used at the transport layer. This is done to enable an instance to retain information regardless of the transport used and regardless of the number of differing transports used for a specific instance to reach a desired destination. OAGIS is agnostic to any specific transport or set of transport protocols. It needs to remain so as it cannot know how many and what types of “hops” are required to complete the entire message path (FTP, SOAP, Web Service, Cloud, etc.).

Note that since some transports do carry details and information about the transported message, this seems like a duplication of effort but OAGIS still needs to assure that the message can satisfy its business requirement: an instance should contain sufficient “inherent knowledge” to fulfill its expected business function. To assist, OAGi provides guides and recommendations for these multiple “hop” cases on its website [www.oagi.org](http://www.oagi.org). They will guide the use of those standards when combined with OAGIS.

As the following graphic illustrates, each BOD contains one and only one ApplicationArea by which to identify the instance. XML Spy represents this cardinality as a solid line around the Element (see Application Area in figure below)
Figure 12

The ApplicationArea serves four main purposes - to:

- Uniquely identify this BOD instance. The BODID uses a Globally Unique Identifier (GUID) for every BOD instance. Note that it is optional but it is strongly recommended.
- Identify the sender of the message.
- Identify when the document was created (CreationDateTime).
- Provide the ability for authentication of the sender through the use of a digital signature.

The ApplicationArea elements include:

**Sender**

The Sender permits the identification of the application that created this instance of the BOD. The sender area can indicate the logical location of the application and/or database server, the application, and the task that was used to create the BOD. The Sender element also allows the sending application to request the receiving application to provide a receipt confirmation.
The Sender component 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.

**Logical ID**

The Logical Identifier element 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.

**Note:** The technical implementation of this Domain Naming Service is not dictated by this specification.

This logical to physical mapping may be done at execution time by the application itself or by a middleware transport mechanism, depending on the integration specifics used.

It would provide for a simple but effective directory access capability while maintaining application independence from the physical location of those resources on the network.

**Component ID**
The Component ID provides a finer level of control than Logical Identifier and represents the business component that issued the BOD. Its use is optional.

The Open Applications Group has not constructed a list of valid Component names. A suggestion for naming is to use the application component names used in the scenario diagrams in section two of OAGIS. Example Components may be "INVENTORY", or "PAYROLL".

**Task ID**
The Task ID describes the business event that initiated the need for the Business Object Document 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 may be "RECEIPT" or "ADJUSTMENT". Its use is optional.

**Reference ID**
Reference ID 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 may be required in environments where an audit trail must be maintained for all transactions. Its use is optional.
Confirmation Code
The Confirmation Code is an option controlled by the Sender business application. It is a request to the receiving application to send back a confirmation BOD to the sender. A Confirmation BOD (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:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Never</td>
<td>No confirmation ConfirmBOD requested</td>
</tr>
<tr>
<td>OnError</td>
<td>Send back a ConfirmBOD only if an error has occurred</td>
</tr>
<tr>
<td>Always</td>
<td>Always send a ConfirmBOD regardless</td>
</tr>
</tbody>
</table>

Table 1

Authorization ID
The AuthorizationID (Identifier) identifies the authorization of the sender (user, machine, application or business), that caused the sending the BOD instance message. The receiver of the message will use 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.
**Logical ID**

The Logical Identifier element is the identification of the receiving application to which the BOD 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.

**Component ID**

The Component ID 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**

The 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 Business Object document. Its use is optional.

NOTE: 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 may be “INVENTORY”, or “PAYROLL”.

**CreationDateTime**

CreationDateTime is the date time stamp that the given instance of the BOD was created. Once assigned, this date must not be modified during the life of the BOD.

The OAGIS DateTime type supports the ISO DateTime format.

**Signature**

If the BOD is to be signed, the signature element is used. The choice of which digital signature is used is left up to the user and their integration needs.

Signature will support any digital signature that may be used. The qualifyingAgency identifies the agency that provided the format for the signature.

Note that 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 permits the user to carry a digital signature in the instance of a BOD.

For more information on the W3C’s XML Signature specification refer to: [http://www.w3.org/TR/xmldsig-core/](http://www.w3.org/TR/xmldsig-core/).

**BODID**

The BODID provides a place to carry a Globally Unique Identifier (ID) or a (GUID) that will make each Business Object Document uniquely identifiable. Although optional, a (GUID) will be needed to build any the following services or capabilities:

1. Legally binding transactions
2. Transaction logging
3. Exception handling
4. Re-sending on condition
5. Reporting
6. Confirmations
7. Security
8. Cloud tracking

For an example for obtaining a GUID for a VisualBasic generator, see [http://www.vbccelerator.com/codelib/tlb/guid.htm](http://www.vbccelerator.com/codelib/tlb/guid.htm). Your system should indicate how to obtain a GUID from it.

**Extension**

Extension is normally used only if the need arises to include more or different data in an Element than is already there. Extension is a key component of the OAGIS Extension Methodology that is covered later.

### 1.1.3 Data Area

![Figure 15](image)

The Data Area (DataArea) of the Business Object Document contains the data values for the business transaction. For example, to send a Purchase Order or multiple Orders to a business partner, the Data Area will contain Verb (the action) and the Noun(s) (the object(s) to be worked on – in this case all the Purchase Orders) on which the action is to be performed.

The DataArea contains a single Verb and one or more occurrences of a Noun. This is shown in the following example where the repeating PurchaseOrder element indicates that one or more elements of the type "**PurchaseOrder**" can be included and are to be "**Process**"ed according to the single Verb. Typically a single instance of "Purchase Order" is included but some systems can and do accept multiple instances of "PurchaseOrder" in one BOD.
1.1.4 Verb

The Verb identifies the action that the Sender application wants the Receiver application to perform on the Noun. OAGIS defines a standard list of Verbs and Nouns that are needed in most supply chain and manufacturing integration scenarios. Those Verbs are included and documented with the OAGIS package (see OAGIS Documentation).

The **Verb** is the action to be applied to the object (the **Noun**). Examples of Verbs include Cancel, Change, Process, and Synchronize.

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.
1.1.5 Noun

Figure 18

Noun contains the object or “document” that is to be acted upon. It contains all the information needed for this business interaction. Noun examples include PurchaseOrder, RequestForQuote, and Invoice. Nouns are defined within OAGIS by Working Groups who specialize in specific business functional areas. They form the basis for a message “grouping” (a series of actions usually enabled for a business function – Process, Update, Show, …).

Nouns are composed of Components and Fields that represent all the best practice and common business needs for accomplishing a “family” of business functions. When combined with Verbs, Nouns typically enable a business step/task associated with the business function.

Nouns provided within OAGIS represent the majority of general business function but not all. By design, an OAGIS Noun will include all the common business information needed for an agreed to business step/task. Since OAGIS is a cross industry standard, it will never contain all the elements needed by everyone, every time.

To address this challenge, Nouns from OAGIS are extensible by reusing as much of OAGIS content as possible and by adding additional content via components and fields. Almost anything can be added to an existing Noun where required. This additional content is defined external to OAGIS. This extensions methodology will be discussed briefly later in this document.
1.1.5 Verb/Noun Combinations - BODs

When combined together to form different BODs, the number of actions against a Noun (document) offer a wide range of business functionality.

An example would be use of the Noun PurchaseOrder. Some possible combinations could include; ProcessPurchaseOrder, UpdatePurchaseOrder and ShowPurchaseOrder. Each BOD combination indicates what the sender or requestor wants to happen with the Noun (document) called PurchaseOrder when the receiver receives it.

In addition, these combinations are intended to specify the contents that should be provided to perform the intended transaction.

For example; in a ProcessPurchaseOrder transaction, the information about a buyer and the things being bought would need to be provided, whereas in CancelPurchaseOrder only the order identifier would need to be provided.

1.1.6 Components
Components are the basic building blocks of a Noun. Components consist of other Components and Fields. Examples of Components include: PurchaseOrderHeader, PurchaseOrderLine, Item, Party, and Address.

Components are business logical groupings of other components and fields that represent identifiable groups of business information. They are typically very similar to the sections and sub-sections of printed forms.

The groupings can contain multiple levels (parent/child relationships) and are the heart of OAGIS. By defining these repeatable groupings or elements, OAGIS enables interoperability at two levels. First, business analysts will easily recognize these groupings and their relationship to business application data and their hierarchical structure. Second they provide a representation that is machine process-able to enable greater use.

An example for defining the language used to describe an item to be ordered would be:

Order – Line – Item – Description – Language

with each element or component containing the all the components to the right.

OAGIS structured this way; each component “contains” the components or fields to the right.

An example of reuse is the Line component shown above being used within an Invoice. The Component Line obtains its full context by occurring within other components to enable - reuse.

Components are extensible within OAGIS meaning that additional content (components and fields can be added to an existing Component or they can be defined using the OAGIS Methodology and added where needed) (see Extensions section).

1.1.7 Fields

Fields are the lowest level elements used in OAGIS.
They stem from the approval of the [W3C XML Schema Recommendation (May, 2001)](http://www.w3.org/TR/xmlschema-2) which allows the expression of dates and quantities in lower-level elements. This recommendation offered the creation of multiple XML “types” based on:

- Built-in datatypes, based on ISO standards, e.g. for dates, times, and decimal numbers.
- Simple User-definable types - user-constrained versions of the standard types
- Complex User-definable types which are user-defined structures built up from other simple and complex type

Fields therefore may be based on:

- an OAGIS-defined type drawn from other standards (ISO, UN/CEFACT,…)
- an OAGIS-defined type that meets specific needs
- a user-defined type
2.0 OAGIS Extensions

While it is important to clearly define messages to be passed between business partners and between business applications, it is neither possible nor practical for any standards body to identify all of the possible information that may be needed in every given situation. It is also not possible to completely identify the unique characteristics and relationships of that data such that it may provide full value for a given implementation.

In other words, there are always going to be extensions needed such that a company or value chain can communicate its unique needs. These may be in the form of additional fields or in the form of additional values for a given field(s) or they may be in the form of additional components. For this reason OAGIS is designed to be extended in the following ways:

Extension –
Extension provide an optional element within many OAGIS defined components that may be used by an implementer to carry any necessary additional information.

For example, it may be necessary to carry some internal customer information (we will call that XYZ) in the Header of a ProcessPurchaseOrder BOD in order to meet the unique customer or internal integration demands. This can easily be accomplished by defining the field XYZ in a XML Schema file and referencing this file via a namespace in the XML instance of the BOD; the extended field is included in the UserArea of the ProcessPurchaseOrder’s Header. As long as this additional XML Schema file is reference-able, the extension can be validated.

Overlay Extensions –
It is possible to extend, the content of any OAGIS Noun or Component. Doing so appends new element content to an existing Noun or Component.

This provides users with the ability to have their extensions appear within OAGIS defined components. In order to add elements in this way a user must extend the OAGIS types using their own namespace. This is accomplished by creating the necessary files similar to the OAGIS resource files and inserting them into the schemas where needed. By doing this it is possible for users to enforce additional restrictions and/or add additional elements to OAGIS defined Nouns or Components. It is also possible for users to provide additional constraints XSL or their own applications, which may then be applied, to OAGIS.

For more on how to extend OAGIS, please see the OAGIS XML Schema Extensions white paper on the OAGi web site. This OAGIS Extensions paper details this methodology much further.

Please note that while it is possible to carry externally defined elements within the UserArea, OAGI recommends using Overlay extensions to add additional information to OAGIS.
3.0 OAGIS Constraints

While XML Schema provides a very powerful mechanism for validating types and structures, it does not provide a good mechanism for applying rules and constraints that may differ from implementation to implementation. For this reason the designers elected to separate the type and structure validation from rule and constraint validation.

Their design allocated that the validation of the structure and type system should be provided by XML Schema. That leaves the validation of the rules and constraints to be provided by XSL or other methods such as Schematron or database query. Using XSL generally involves the use of XPath as it is a part of XSL.

Currently, XSL validation is based on leveraging XPath but since there are many different XSL processors available and no simple XPath processors, the specifics are not yet standardized. As more implementers choose a methodology for this, OAGIS will track and offer services and support based on the current “best practices.”

Also available today are database technologies that enable the storage and query of XML instances using XQuery. These offer yet another approach to validation of content.

Using this "split" approach to using standards and still addressing constraints allows OAGI to define the required fields that our constituency agrees are required for a given interaction while still offering a given implementation the ability to apply their own requirements to specific fields and components by simply populating an XSL and applying the additional constraints.

There is considerable information on this topic available on User Groups and through the OAGi web site.
4.0 OAGIS Error Handling

OAGIS facilitates error handling at the application layer through the ConfirmBOD.

If an error occurs in the processing of a BOD in the receiving application and the Sender requested a Confirmation based on either OnError, Never or Always conditions, the receiving application must provide a ConfirmBOD that references the original BODs BODId. This ConfirmBOD can then indicate that there was an error in the original BOD or its processing and even transmit the error messages from the receiving system if that is valuable.

Once the original sending system receives the ConfirmBOD it will have some indication that an error occurred and what that error might be. Currently OAGIS has not yet specified the action that follows; instead that action is left to the integrators because of the infinite number of variables.

While it is possible for the sending system to resend the BOD or to attempt to correct any missing information through advanced error correction mechanisms, it is important to consider all the ramifications of such a process and how best to work this in. OAGIS provides the message platform to exchange very detailed information. What is actually processed and how that impacts business is up to the implementation.

Note that the OAGIS ConfirmBOD is in addition to any communication layer error handling that may be provided by the infrastructure framework, web service, or middleware.
5.0 OAGIS Summation

The OAGIS Architecture is designed to provide the best of industry business knowledge within the boundaries of the practical and best practices. By definition then, it will continue to evolve over time. It is driven by the Architecture Council within OAGi and as such reflects the best knowledge and practices of its membership.

This document represents but a small part of that extensive library of information available. Also available are several other documents that can assist with understanding and implementing OAGIS. Figure 22 represents their relationship to using OAGIS.

While OAGIS Architecture reflects the most current approach to business improvement and optimization but it will continue to change as new and different technologies enable, simplify and address the ever evolving world of business technology.

The very best way to be part of that evolution as well as part of the vast knowledge base that is applied to OAGIS is to become a member of OAGi. For more information on doing just that, please start at www.oagi.org/dnn2/HowtoJoin.aspx and become part of the solution that improves business.

Respectfully,
OAGi Staff and Membership