HR-XML Consortium

3.3 Architecture

Copyright statement

Table of Contents

Terminology and Background

The sections below introduce key concepts associated with HR Interoperability standards. The intention here is to present concepts concisely using simple language within an HR systems context. Note that several standards organizations have developed far more complete ontologies or meta models relating to service oriented architecture.

Message

HR-XML standards are principally concerned with defining messages communicated between and among software components.

- Messages specify a behavior to be performed on, or a state to be associated with, an object. For this reason, messages tend to be named using a "verb-noun" construction. For example, ProcessPositionOpening (with "Process" being the verb and "Position Opening" being the noun).
- Messages can be used to invoke services and also can be a response from such invocation. In other words, a message may be a service input as well as a service output.
- "Canonical messages" are formally modeled messages intended to be reused broadly across an enterprise or among trading partners. These messages are "canonical" in that they define a single way to convey specific type of information or intention. Canonical messages are derived from a Canonical data model.
- Messages can be communicated using a variety of protocols, both standard and proprietary. Web services typically communicate messages using SOAP and HTTP. An Enterprise Service Bus (ESB) is an example of proprietary messaging technology used to carry messages among enterprise services. HR-XML standards are transport neutral so that they might be used under a variety of transport mechanisms.
- While a primary focus of the current HR-XML library is a message-oriented model and SOAP-style Web services, other software architecture approaches also might be supported by the HR-XML data model (for example, "Resource Representational state transfer" or so-called "RESTful" approaches).

Business Object Document

A Business Object Document (BOD) is a message constructed according to the methodologies of the Open Applications Group. BODs provide a high level of re-use. By making available a consistent messaging meta model across diverse business domains, BODs provide a faster learning curve for developers, business analysts, and integrators.

BODs encapsulate both behavior and structure for business interactions and facilitate data management among system actors.

Data Management

Data management refers to procedures and policies that trading partners use to communicate new and updated information between them so each can maintain an accurate account of the entities that are the subject of the collaborations. More specifically, data management is used as an umbrella term to refer to Create, Read, Update, and Delete (CRUD) operations performed on managed entities using the OAGIS business object document (BOD) architecture. HR-XML's data management guidelines are a subset of those drafted by the Open Applications Group.

Actors

- An actor is a person, an organization, or a system (a "system actor").
- Actors can be providers and/or consumers of services or otherwise have a role in triggering collaborations between service providers and consumers.
- The Actor Catalogue identifies and defines actors commonly involved in HR business processes covered by the HR-XML standards.
- HR-XML standards typically are concerned with enabling collaborations between and among arm-length trading partners. In other words, the standards are primarily designed around business-to-business communications. To a lesser extent, HR-XML standards are used within application-to-application integration scenarios where system actors are within the same organization.

Events

- A business event ontology is a reasonably complete collection of business events definitions that represent the points within business processes where collaborations between actors occur.
- Events trigger changes in the state of business objects and related communications between actors.
- Events relevant to HR collaborations can be broadly categorized as: 1. personal (for example, an employee change of address, marriage, divorce, disability, etc.); 2. employment and work-related (hire, termination, promotion, etc.); and 3. organizational (merger, acquisition; plant closing; bankruptcy; policy changes; vendor changes; etc.). Events also can be fine-grain, process-specific occurrences (for example, successful completion of employment screening may clear the way for a next step or stage of a hiring process).
- HR-XML standards commonly refer to both course-grain event categories (for example, personal information changes) as well as fine grain events (change of a home postal address).
- There can be relationships and priorities among different events. One event may trigger another. Likewise, the occurrence of a certain event may supersede another as a trigger for a collaboration.
- An "event driven architecture" is a style of application and system architecture characterized by a set of relatively independent actors working towards coordinated goals by communicating and responding in predictable ways on the basis of event triggers.

Scenarios

Scenarios are intended to provide guidance for implementers in understanding how to apply HR-XML standards.

A scenario is a generalized representation of a business collaboration. Scenarios identify and describe things such as:

- A business purpose or "use case" (or use cases).
- A collaboration among actors to fulfill such use cases.
- The actors involved in the collaboration.
- Events that trigger the collaboration(s) between and among Actors.
- Preconditions or requisites for the collaboration.
- Error and success cases (so called "Post Conditions").
- Variations in how the collaboration occurs or is carried out.

A scenario is realized through a collaboration between trading partners using appropriate "service compositions".

Service Compositions

A service composition is a collection of services (one or more services) organized in a way so as to fulfill one or more use cases.

In many collaborations, each trading partner will have its own service composition to fulfill a collaboration between or among the trading partners. For example, in an assessment procurement collaboration an Assessment supplier may host services that a customer or requester may call (ProcessAssessmentOrder, GetAssessmentReport). Likewise, a customer or requester may host services to support its part of the collaboration (for example, a NotifyAssessmentReport service that the supplier may call to push an assessment result back when one comes available).

The schemas also include a related WSDL for each service composition.

Profiles

A profile is a specification based on an HR-XML BOD schema that qualifies or constrains the complete schema to fit a particular business scenario, pattern of implementation, or trading partner implementation. Part or all of a profile can be tested in an automated manner. Examples of possible profiles are:

- **Candidate Profile.** HR-XML's Candidate Schema can be used to support such diverse use cases as Resume Parsing and Candidate submission from job boards and staffing agencies. Profiles could be developed to support one or more of these uses or a particular implementation of the schemas within those use case categories.
- **Screening Order Profile.** Three implementation patterns have been recognized in using HR-XML screening schemas. These so-called "Package Only", "a la Carte", and "Package Plus" patterns are explained in the related documentation. Again, profiles could be developed to support one or more of these broad use patterns or a particular implementer's pattern of use.

Code Lists

A "code" is a character string (letters, figures or symbols) that for brevity, language independence, or trading-partner neutrality may be used to represent or replace a definitive value or text of a property. Codes usually serve the purpose of "classifying" the associated entity. Note that this is separate from "identification," which is handled by an identifier type. Broadly speaking, HR-XML's version 3.X architecture recognizes three types of code lists:

- **External.** As the name implies, external code lists are not enumerated within the HR-XML library. The core component CodeType provides the meta data to optionally reference the particular code list (using the "listID" attribute) and the agency or company that controls that list (listAgencyID). External code lists typically are either well-known, official, "standard" code lists for the particular value domain that are controlled and maintained by a recognized authority or in other cases, these lists may be proprietary and controlled by a particular trading partner or business.
- **Open.** Open lists have values that are specified within the HR-XML library (see \OAGi-BPI-Platform\org_hr-xml\3_2_0\Developer\Common\CodeLists.xsd). However, these values are defined in a way so they are not enforced by HR-XML schemas. Other values may be specified. In other words, open lists can be extended simply by using alternate values than those enumerated within the HR-XML library. Essentially, open lists provide documentation of suggested values. Working groups or trading partners might set certain business rules regarding the use of open lists apart from what is enforced within the schema (see the section called “Profiles”.
- **Closed.** These are lists HR-XML develops and maintains. A closed list implies that the list will be implemented without restriction or extension. Note that most code lists within the HR-XML library are defined as open lists rather than closed lists.

Data Management

Overview

Data management refers to procedures and policies that trading partners (or internal actors) apply in communicating information between and among each other so that each maintains an accurate account of the entities being managed. Specifically, the data management approaches described in the sections that follow, aim to provide a common understanding of how create, read, update, delete (CRUD) operations are carried out.

This document provides broad guidance for data management using the "Business Object Document" message architecture developed by the Open Applications Group Inc. (OAGi). The guidelines set out in this document draw heavily upon OAGi data management guidelines and best practices. This document aims to introduce OAGi data management best practices and guidelines to the requirements and scenarios within the HR services industry. However, note that this document does not attempt to replicate the full set of OAGi data management guidelines currently under development and available to OAGi members.

About OAGi

The Open Applications Group Inc. (OAGi) was formed in late 1994 as the first post-EDI organization focusing on improving the state of application integration. The Open Application Group Integration Specification (OAGIS) is one of the most widely implemented XML business language standard for horizontal enterprise functions. OAGIS covers such topics as supply chain management (so-called "order-to-cash" processes), manufacturing, customer relationship management, logistics, and issue tracking and risk control integration intended to support Sarbanes-Oxley requirements in U.S. Markets. Like HR-XML's library, the OAGIS XML Standard is available completely royalty-free.

Conformance and Certification

This document sets out a variety of data management options and guidelines for implementing HR-XML specifications. It does not by itself set out requirements or rules to be used in determining whether or not a solution or trading partner implementation is "compliant."

The chapter does include a number of rules with respect to certain data management approaches. These rules are included for the following anticipated uses:

- Implementers may find the rules and related approaches useful to reference or include as best practices in developing their own data management policies, quality control procedures, or guidelines for trading partners.
- While this document does not constitute conformance criteria by itself, the rules and guidelines could be incorporated by reference within separate HR-XML implementation profiles that may be used for certification or conformance purposes.

Note that the rules are numbered. In the following documentation, certain sequence numbers are skipped, while others appear out of order. The intent is to keep the rules tied to related rules being explored by OAGi while presenting the rules so they are most meaningful to the HR-XML audience.

Business Object Documents

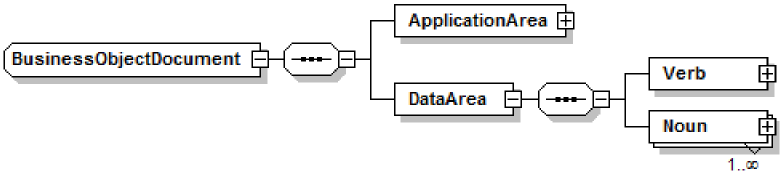
A Business Object Document (BOD) is a message constructed according to the methodologies of the Open Applications Group. BODs provide a high level of re-use. HR-XML's implementation of OAGIS BODs provides a faster learning curve for developers, business analysts, and integrators by making available a consistent messaging meta model across diverse business domains.

Basic BOD Architecture

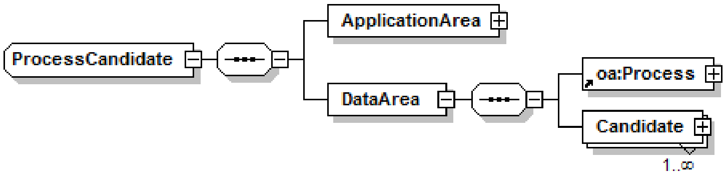
BODs encapsulate both behavior and structure for business interactions and facilitate data management between system actors. Simply stated a BOD pairs a verb with a noun. The verb identifies the action that the sender wants the receiver to perform on the noun. Verbs also have a role in the responses from a receiver back to the sender of the original message.

HR-XML draws upon a subset of the standard OAGIS verbs and includes a set of nouns designed around common HR application integration scenarios. HR-XML combines verb and noun according to the OAGIS BOD architecture. Verbs are described in depth in the section called "Subset of OAGIS Verbs Implemented". The nouns defined within the HR-XML library are described in Chapter 3, Noun Catalogue.

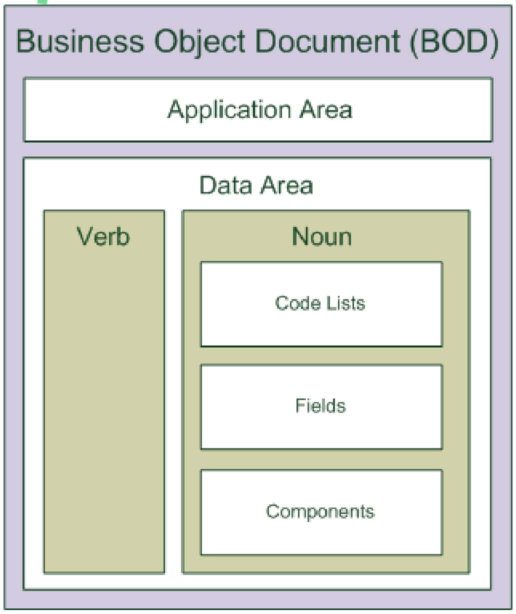
A general representation of the BOD architecture is depicted in the figure below.



In an actual BOD, the generic names (BusinessObjectDocument, Verb, Noun) are replaced by specific names. For example, the figure below shows a simplified representation of ProcessCandidate BOD, combining the OAGIS verb "oa:Process" with HR-XML's Candidate noun. Note that the use of the "oa:" prefix on the Process verb indicates the element is within the OAGIS namespace.



As shown in the diagram above, the BOD structure contains an application area and a data area. As shown in the diagram below, the verb and noun are part of the data area.



The major components of the BOD architecture break down as follows:

- The ApplicationArea is for transactional meta-data, such as identifiers associated with the sender, a creation time stamp, and a unique identifier for the BOD. Note that the ApplicationArea is covered in greater detail within the section called “BOD Components Described”.
- As explained above, the DataArea contains a verb (in other words an "operation") as well as a Noun (the business object of that operation).
- The verb can indicate either what operation to perform (for example, "Process") or a response to an operation (for example, "Acknowledge").
- As with other messaging schemes, the BOD architecture contemplates both requests and responses. In the case of a request, a BOD contains details of the operation. In the case of a response, it contains information fulfilling or pertaining to the request.
- The noun instance contains the data representing the business object that is being managed. A noun is composed from components (in this context, meaning elements that contain elements), fields (elements containing data), and code lists (sets of enumerations that comprise a "value domain" for a field-level element).

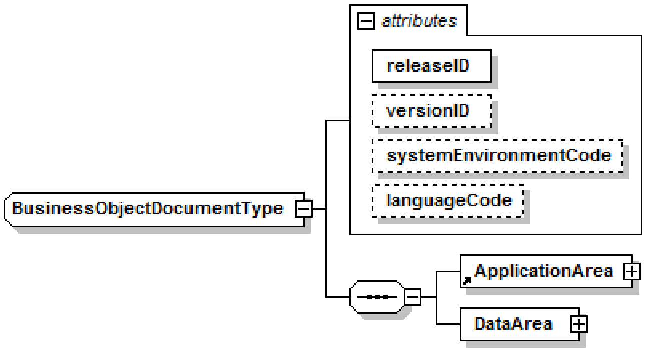
BOD Components Described

The introductory sections have already described the basic structure of a BOD. As described in the section called “Basic BOD Architecture”, a BOD has an ApplicationArea and a DataArea. The DataArea contains a Verb and a Noun. The sections below describe other BOD components not already examined in earlier sections.

Note that all the components described below are defined within the OAGIS namespace. The documentation that appears below, with just a few editorial customizations, has been drawn directly from the OAGIS documentation of these components.

BOD Attributes

Each BOD has the common set of attributes depicted in the diagram below. The use of the attributes is explained below. Note that a single attribute (releaseID) is mandatory and must be populated by implementers.



releaseID

Release ID is used to identify the release of the HR-XML library to which the BOD belongs. For the BODs from HR-XML 3.3 library, the value of this attribute will be "3.3". The releaseID is a required attribute.

versionID

Version ID is used to identify the version of the Business Object Document. Currently HR-XML does not currently plan to version BODs independently of the release. The versionID attribute is optional and generally would not be used in HR-XML implementations until such time as BODs are versioned independently of the release.

systemEnvironmentCode

The System Environment Code is used to identify whether this particular BOD is being sent as a result of a test or whether it represents production level integration. Often times as new systems are brought online testing must be performed in a production environment in order to ensure integration with existing systems. This attribute allows the integrator to flag these test messages as such. The environment attribute is an optional attribute of the BOD.

languageCode

The languageCode attributes indicates the language of the data being carried in the BOD message. It is possible to override the BOD level language for fields that may need to carry multi-lingual information. Examples of this are Comment and Description elements.

Application Area Described

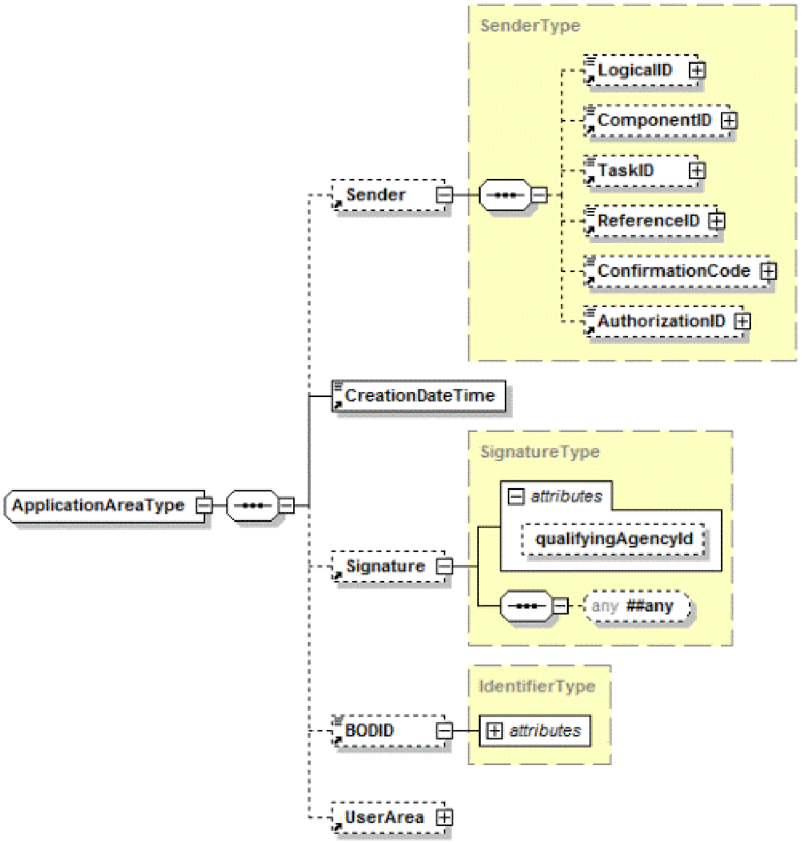
Each BOD definition and BOD message instance will contain a single ApplicationArea. A single element within a BOD instance is mandatory (CreationDateTime).

The ApplicationArea serves four main purposes:

1. To identify the sender of the message.
2. To identify when the document was created.
3. To provide authentication of the sender through the use of a digital signature, if applicable.
4. To uniquely identify a BOD instance. The BOD ID field is the Globally Unique Identifier for the BOD instance.

Note: The ApplicationArea carries information that an application may need to know in order to communicate in an integration of two or more business applications. The ApplicationArea is used at the applications layer of communication. Middleware and integration frameworks such as Web services provide the communication layer itself.

The components within the Application Area are illustrated and explained below.



Sender

The Sender identifies characteristics and control identifiers that relate to the application that created the Business Object Document. The sender area can indicate the logical location of the application and/or database server, the application, and the task that was processing to create the BOD.

The Sender area also provides the ability to create an audit trail to allow users to drill down from their Receiving business application to the information used to complete the business transaction being communicated in the BOD.

In today's business environments and advanced technology frameworks a single BOD may be routed to multiple destinations or receivers. For this reason, it is not feasible for the sending system to "know" all of the possible destinations of a BOD. For this reason, the Open Applications Group has made a conscious decision NOT to include a Receiver in the ApplicationArea. This is left to the middleware or infrastructure framework to ensure delivery to all locations that are interested in the content of the BOD.

LogicalID

The Logical Identifier element provides the logical location of the server and application from which the Business Object Document originated. It can be used to establish a logical to physical mapping, however its use is optional.

Each system or combination of systems should maintain an external central reference table containing 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 architecture used. This provides for a simple but effective directory access capability while maintaining application independence from the physical location of those resources on the network.

ComponentID

The Component ID provides a finer level of control than Logical Identifier and represents the business application that issued the Business Object document. Its use is optional.

TaskID

The Task ID describes the task that initiated the need for the Business Object Document to be created. Tasks are triggered by events - so a task is a generalization between an action and a triggering event.

ReferenceID

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 drill back from the BOD message into the sending application. The may be required in environments where an audit trail must be maintained for all transactions.

ConfirmationCode

The Confirmation Code request 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. The confirmation Business Object Document may indicate the successful processing of the original Business Object Document or return error conditions if the original Business Object Document was unsuccessful.

The confirmation request has the following valid values:

- **Never.** No confirmation Business Object Document requested

- **OnError.** Send back a confirmation Business Object Document only if an error has occurred
- **Always.** Always send a confirmation Business Object Document

AuthorizationID

The Task ID describes the task that initiated the need for the Business Object Document to be created. Tasks are triggered by events - so a task is a generalization between and action and a triggering event.

The Authorization Identifier describes the point of entry, such as the machine or device the user uses to perform the task that caused the creation of the Business Object Document.

The Authorization Identifier is used as a return routing mechanism for a subsequent BOD, or for diagnostic or auditing purposes. Valid Authorization Identifiers are implementation specific. The Authorization Identifier might be used for authentication in the business process.

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

CreationDateTime

CreationDateTime is the date time stamp that the given instance of the Business Object Document was created. This date must not be modified during the life of the Business Object Document.

Signature

If the BOD is to be signed the signature element is included, otherwise it is not.

Signature will support any digital signature that maybe used by an implementation of OAGIS. The qualifyingAgency identifies the agency that provided the format for the signature.

This element supports any digital signature specification that is available today and in the future. This 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.

This allows the user to carry a digital signature in the xml instance of a BOD. The choice of which digital signature to use is left up to the user and their integration needs.

For more information on the W3C's XML Signature specification refer to: http://www.w3.org/TR/xmldsig-core/.

BODID

The BODID provides a place to carry a Globally Unique Identifier (GUID) that will make each Business Object Document uniquely identifiable. This is a critical success factor to enable software developers to use the Globally Unique Identifier (GUID) to build the following services or capabilities:

- Legally binding transactions
- Transaction logging
- Exception handling
- Re-sending
- Reporting
- Confirmations
- Security

Subset of OAGIS Verbs Implemented

The Version 3.X library relies on a subset of the OAGIS verb catalogue. A glossary for this subset appears below.

Each OAGIS verb has associated semantics. Verbs also have associated structure depending on the verb type (see the section called “Verb Types and Response Patterns”). OAGIS verb names represent a coarse-grain action, response, or request in relation to a Noun. The structure available within each verb type enables communication of fine-grain actions on components within a Noun. Both the semantics and the structure of these verbs are explored in the sections that follow.

Verb Semantics

It is important to understand the semantics behind the OAGIS verbs. The semantics are intended to be precise. Definitions differ from common usage of the same terms as well as usage under other messaging frameworks.

Cancel. The Cancel verb is used when the sender of the BOD is not the owner of the data, but is sending a request for the document to be canceled. An example is the Cancel PO where the business implications must be calculated and a simple data processing term such as delete can not fully convey the business meaning and required processing associated with the meaning.

Change. The Change verb is used when the sender of the BOD is not the owner of the data, but is sending a request for the document to be changed.

Process. The Process verb is used to request processing of the associated noun by the receiving application or business party.

Notify. The Notify verb is used to inform the receiving party that an event has occurred or document has been created. For example, a supplier may have a proposed order that is sent to a trading partner. The noun will contain the data that has been proposed.

Get. The Get verb is to communicate to a business software component a request for an existing piece of information to be returned. The Get may be paired with most of the nouns defined in the OAGIS specification. The response to this request is the Show verb. The behavior of a BOD with a Get verb is quite predictable across most of the nouns. The Get is typically used to retrieve a single piece of information by using that information's primary retrieval field, or key field, but may support certain reporting needs by supporting retrieval of a range of documents matching a given filter. For further information, see the section called “Request Verb Type” and the section called “Get Verb”

Acknowledge. The Acknowledge verb is used to acknowledge the application receipt of a Process request. This function conveys the result of the original request. An example of this is Acknowledge PO, where a Process PO has been issued and the corresponding business application acknowledges the receipt of the PO and responds with an acceptance or a counter offer.

ConfirmBOD. The Confirm verb is used to respond to a request to confirm the receipt of information by the receiving system. The request for confirmation is set by the sending application in the ApplicationArea\Sender\Confirmation field of the original BOD. The Confirm conveys the result of the original request i.e. whether or not the message was understood and was successfully processed.

Respond. The Respond verb is used to communicate relative to another document. It may be used to communicate agreement, questions, answers to a question, or disagreement with the related document.

Show. The Show verb is used when sending the information about a specific instance of a business document or entity. The Show verb may be used to respond to a Get request or it can be used in a publish scenario, where it pushes information to other applications based on a business event. Although BODs based on this verb do not commonly cause updates to occur, there may be times when the component receiving the Show decides to use the information it receives to update. See the section called "Show Verb".

Sync. The Sync verb is used when the owner of the data is passing or publishing that information or change in information to other software components. This is to be used when the receiver of the SyncBOD does not own the data. This verb is commonly used when mass changes are necessary or when a publish and subscribe mechanism is used in the integration architecture.The purposes of this verb include application integrity and ease of data entry for the business user by enabling a single point of input.

Verb Types and Response Patterns

The structure of a verb is determined by its type. There are three types of OAGIS verbs. The three verb types are:

Action. These include verbs to create or otherwise act upon a business object. Action verbs include "Process," "Change," "Notify," etc. See the section called “Action Verb Type”.

Response. A response enables a receiver to reply or send relevant meta-data back to the sender of the original message instance. A response can be a response to a message with an Action verb or even a response to another response message. See the section called “Response Verb Type”.

Request. A request verb enables the communication of "read" data management instructions from a message sender to a receiver. The "Get" verb currently is the only request verb. Depending on the implementation, "read" instructions could encompass the simple retrieval of a discrete document or piece of data as well as complex queries or filters applied against a broad collection of objects. See the section called “Request Verb Type”

Verb Types and Responses

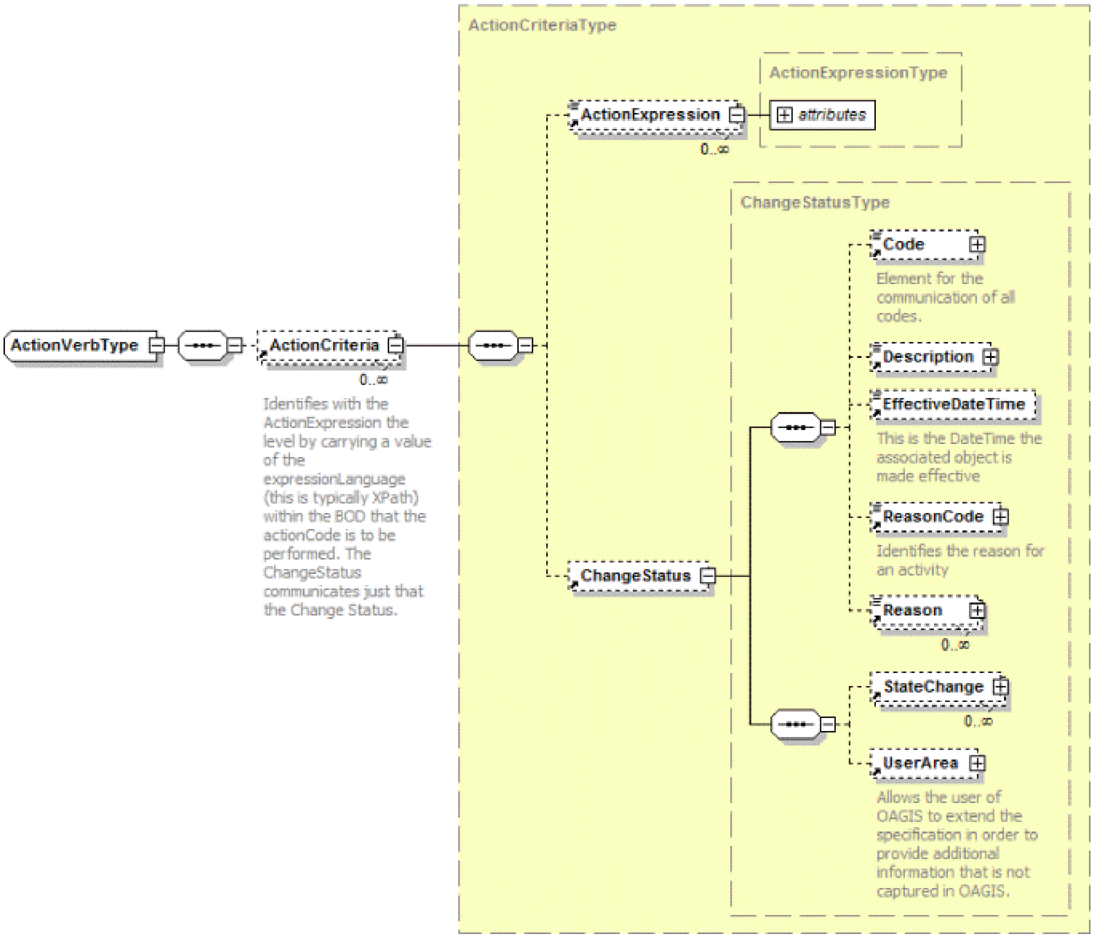
The table below shows which of the OAGIS verbs used within the HR-XML library are of which verb type. For each verb, the associated "noun-specific" and "noun-independent" responses (if any) also are indicated. As you can see, "ConfirmBOD" is the noun-independent "catch-all" response. It can be used as a simple confirmation receipt of a request as opposed to responding to specific business content within a request message. ConfirmBOD also is used to report exceptions in the processing of requests.

Verb	Verb Type	Noun-Specific Response	Noun-Independent Response
Cancel [noun]	Action	Acknowledge [noun]	ConfirmBOD
Change [noun]	Action	Acknowledge [noun]	ConfirmBOD
Process [noun]	Action	Acknowledge [noun]	ConfirmBOD
Notify [noun]	Action	Acknowledge [noun]	ConfirmBOD
Sync [noun]	Action	None	ConfirmBOD
Get [noun]	Request	Show [noun]	ConfirmBOD
Acknowledge [noun]	Response	None	ConfirmBOD
Respond [noun]	Response	None	ConfirmBOD
Show [noun]	Response	None	ConfirmBOD
ConfirmBOD	Response	None	None

Action Verb Type

Action verbs convey operations to be performed upon, or in relation to, a noun. Broadly speaking, action verbs encompass the kind of operations that "create," "update," and "delete" verbs convey under the so-called "CRUD" convention commonly used in referencing basic computer storage operations. The OAGi action verbs used within the HR-XML library are: Cancel, Change, Notify, Process, and Sync.

The diagram below shows the elements of the ActionVerbType definition.


ActionCriteriaType
ActionExpressionType
ActionExpression
attributes
0..∞
ChangeStatusType
Code
Element for the communication of all codes.
Description
EffectiveDateTime
This is the DateTime the associated object is made effective
ReasonCode
Identifies the reason for an activity
Reason
0..∞
StateChange
0..∞
UserArea
Allows the user of OAGIS to extend the specification in order to provide additional information that is not captured in OAGIS.
ActionVerbType
ActionCriteria
0..∞
Identifies with the ActionExpression the level by carrying a value of the expressionLanguage (this is typically XPath) within the BOD that the actionCode is to be performed. The ChangeStatus communicates just that the Change Status.
ChangeStatus

ActionVerbType Explained

The ActionVerbType depicted in the diagram above breaks down as follows:

- An action verb supports zero-to-many ActionCriteria. Each ActionCriteria supports zero-to-many ActionExpressions and zero-to-one ChangeStatus.
- The ActionExpression is the mechanism used to represent the data management instructions in a BOD instance. Specifically, this includes identification of the element(s) and the action to be taken on those elements. Identification of the element may consist of its location in the schema structure and possibly additional key value information if it is necessary to identify a specific element of interest in a BOD instance.

- ChangeStatus may be used to communicate state change information (e.g. the EffectiveDateTime and ReasonCode for the state change as well as the FromStateCode and ToStateCode).
- The ActionExpression has two attributes: actionCode and expressionLanguage. The actionCode attribute of ActionExpression specifies an action to be taken by the receiver of the BOD instance. The actionCode is restricted to a value domain. The actionCode's value domain includes: Add, Change, Delete, and Replace.

The noun or nouns within a BOD instance represent the entity or entities being managed. The ActionExpression, where used, sets the scope within the noun to which the instruction specified within the actionCode is applied. The following patterns apply:

- "Add" - the noun content represents the entities being added.
- "Change" - the noun content represents the change entities.
- "Delete" - the content of the noun represents or identifies the entities being deleted.
- "Replace" - the noun content represents the replacement entities.

Response Verb Type

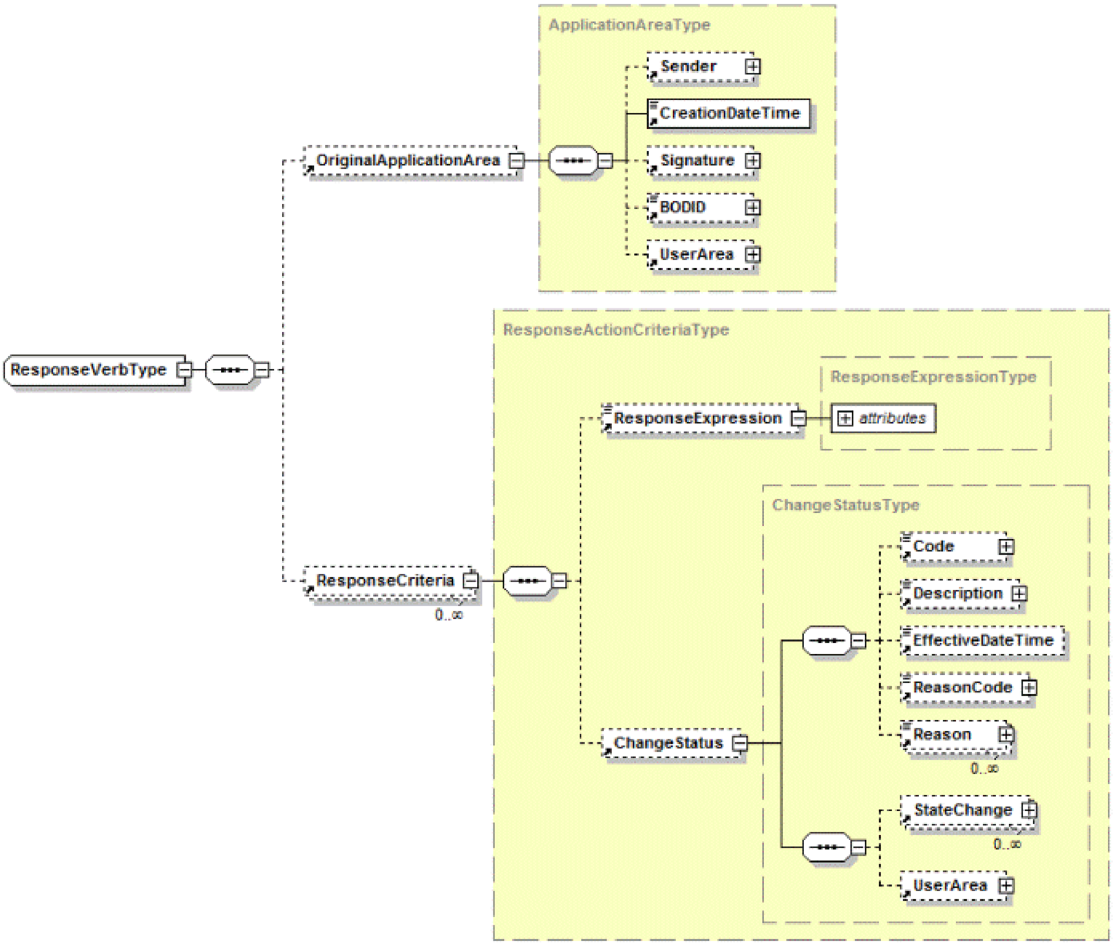
Response verbs are used by message receivers to convey meta-data to the sender of an original message instance. The OAGIS response verbs are Acknowledge, Confirm, Respond, and Show.

Response verbs are used in several different data management contexts. Consider that response verbs are used in message instances that respond to:

- **Action verb message instances.** For example, the Acknowledge verb would be used in a message instance responding to a message constructed with a Process verb. In this case, the response message conveys the functional result of the action specified in the original message instance.
- **Request verb messages instances.** For example, the Show verb would be used in a message instance responding to a message constructed with a Get verb. In this case, the response message conveys data for the entities matching the "read" instruction specified in the original request message instance.

As explained in the section called “Verb Types and Responses”, the Confirm verb is the noun-independent verb conveying message receipt and can be used in a response to a message using any other verb. Consider that a message instance using the Show verb might be responded to with a message instance using the Confirm response verb.

The diagram below shows the elements of the ResponseVerbType definition.



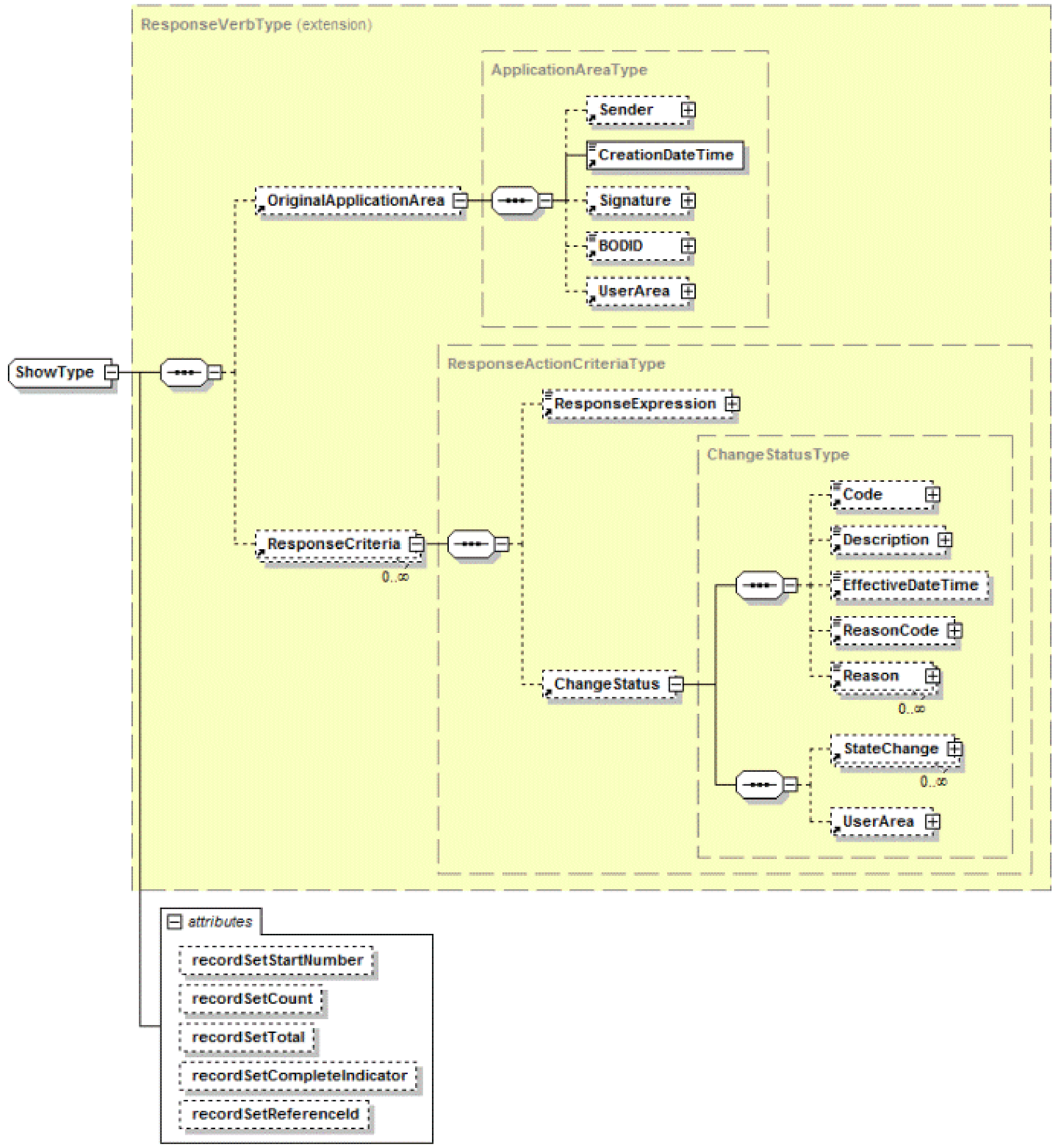
ResponseVerbType Explained

The ResponseVerbType depicted in the diagram above breaks down as follows:

- The ResponseExpression has two attributes: actionCode and expressionLanguage.
- The actionCode specifies an action that was taken by the receiver of the BOD instance.
- The ResponseExpression.actionCode is restricted to a value domain. The actionCode's value domain includes: Accepted, Modified, and Rejected.

Show Verb

The Show verb is based on the ResponseVerbType. It is used in message instances returning the results of "read" instructions sent in a message instance using the Get verb. The diagram below illustrates the Show verb's extension of the ResponseVerbType.



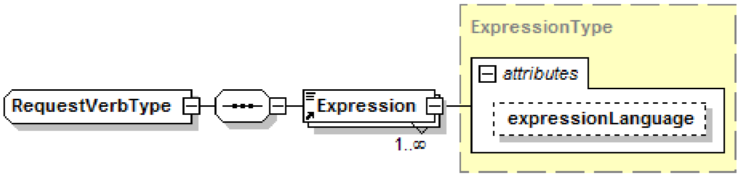
ShowType Explained

The extension includes several attributes whose values may be set as part of a Show response. The attributes are defined as follows:

- recordSetStartNumber. The record number identifying the first record returned in the Show response. The producer of the Show response generates this number. It is used by the requesting system to determine the start number of the subsequent Get request.
- recordSetCount. Number of records in the recordSet.
- recordSetTotal. Number of total records in a recordSet.
- recordSetCompleteIndicator. Indicates whether this is the last segment of the recordSet.
- recordSetReferenceID. Unique identifier of the RecordSet. It is generated by the producer of the Show response as a result of the original Get request.

Request Verb Type

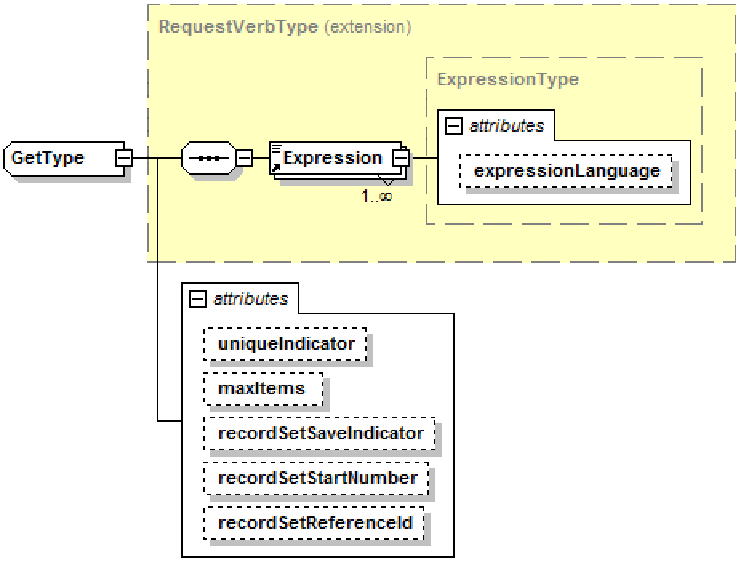
The request verbs are the OAGi language elements through which Read data management instructions are conveyed by message senders to receivers. As shown in section “Verb Types and Responses”, Get" currently is the only verb implementing RequestVerbType. The following diagram shows the elements of the RequestVerbType schema definition. The single element, Expression, has one attribute, expressionLanguage.



Get Verb

The Get verb is to communicate a request for an existing piece of information to be returned. The Get is typically used to retrieve a single piece of information by using an ID for the related element. However, Get may also support certain reporting needs by enabling the retrieval of a range of documents matching a given filter. An example might be a filter constructed to retrieve all ScreeningOrders or StaffingOrders or other documents with a given status.

The schema definition for the Get verb extends RequestVerbType as shown in the following diagram.



The extension includes several attributes whose values may be set as part of a Get request. The attributes are defined as follows:

- uniqueIndicator. Indicates whether duplicates should be filtered out.
- maxItems. Communicates the maximum number of records which should be returned in a segment from a recordSet.

- recordSetSaveIndicator. A true value indicates that receiver should save the record set.
- recordSetStartNumber. The record number identifying the first record that should be returned in the Show response. This attribute is specified on subsequent Get requests, not an initial Get request (see Initial and Subsequent Get Operations). The requesting system may determine this number from the prior Show response. For further information, see the section called “Show Verb”.
- recordSetReferenceID. Unique identifier of the RecordSet. It is generated by the producer of the Show response as a result of the original Get request.

Initial and Subsequent Get Operations

An initial Get request may be followed by related Get requests. As mentioned above, it is necessary to distinguish an initial request, from subsequent requests in that the recordSetStartNumber would only appear on the related, subsequent requests. These initial and subsequent Get requests are related in that they execute a single read operation using the same selection and filter criteria. The subsequent Get request(s) may be communicated when an initial Get request results in more records that can be returned in a single Show response. In other words, more records than allowed by the maxItems indicator.

Data Management Approaches

The previous sections described OAGIS data management constructs. The following sections examine approaches for using these constructs.

After the introduction of key concepts, the discussion is divided into two parts. The first part focuses on Create, Update, and Delete operations. The second part looks at Read data management operations.

Key Concepts

To properly represent and process messages, senders and receivers must have a common understanding of the set of elements managed by message instances. Accurately determining the set of managed elements is a key challenge for message senders and receivers. Without agreed upon approaches, message senders and receivers face ambiguity with regard to data management operations. The sections below introduce key concepts and explain potential sources of ambiguity.

Implementation Scope

Specifications developed by Consortia tend to encompass a broad range of requirements contributed from a diverse group of participants. As consequence, the elements that a Consortium specifies for a "noun" representing a business entity tends to be a superset of what is relevant to a given implementation. Trading partners typically implement only a subset of the complete noun definition. This is a subset of elements representing entities actually managed by the trading partner systems. Those elements are said to be within the implementation scope, whereas the remainder are "out of scope."

HR-XML developed Business Rules to assist trading partners in determining which entities are required or best practice for a particular use case. Note: Business rules are not currently available for all domains.

Set of Managed Elements

A requisite for the data management approaches outlined in this documentation is that the senders and receivers of message instances understand the "set of elements managed" by message instances. This section is intended to introduce this concept, which is relied on in subsequent sections.

A message instance manages a set of elements. This set of elements represents entities actually managed by trading partner systems (that is those that are in-scope as described in the section called “Implementation Scope”). The set of managed elements within a message instance may include all the elements within a noun definition or a subset of the entire definition. Central to the concept of the "set of elements managed" is understanding that this set includes a union of the elements within the message instance with value assignments (i.e., those elements present in the message instance containing data) as well as elements with no values assigned (or, under the methodology that will be introduced, elements with an explicit Null assignment). In other words, the concept recognizes that data management instructions can be conveyed by elements in a message instance with assigned values as well as elements without value assignments. One of several possible examples of the latter is distinguishing an element with no value assignment that represents an instruction to delete an entity that exists on the receiving system.

Without an agreement between sender and receiver on what constitutes the set of managed elements within a message instance, ambiguities will impede interoperability. Consider that senders and receivers may have difficulty distinguishing elements within the managed set that have no value assignment from elements that are merely "out of scope" (as discussed in the prior section). Consider also how choice of update approach -- such as an approach of sending changes only -- can affect interoperability if sender and receiver lack a shared understanding of the set of elements managed. If only changes are sent, how will the receiver of a message distinguish among elements with no value assignments representing delete instructions from elements without a value assignment representing entities that have remained unchanged and, thus are not part of the incremental changes reported? Some broad data management approaches and guidelines are introduced in the sections that follow.

Approaches for Create, Update, and Delete Operations

There are two basic data management approaches for Create, Update, and Delete operations:

- **Snap-Shot.** Under this approach, the sender conveys within the noun all the elements representing the current composition of the entity being managed. This can be thought of as sending a "snapshot" of the full entity as it exists at the moment. This also is referred to as the "full-refresh" approach. This approach is outlined in the section called “Snapshot Approach”
- **Incremental.** This involves sending a message containing only the particular elements representing entities that have been created, updated or deleted. This also is referred to as the "Delta" approach. For a detailed description, see the section called “Incremental Approach”.

While the snapshot approach is generally considered simpler to implement than the incremental approach, message instances based on the snapshot approach are larger in size and may require longer processing time than those based on the Incremental approach.

Note: The two approaches (snapshot and incremental) are not mutually exclusive within the context of a trading partner relationship as long as trading partners have a shared understanding of when and how to apply each approach. For example, "create" and "delete" operations involving relatively discrete elements tend not to pose the same degree of difficulty in calculating the "set of managed elements" as unconstrained "update" operations. In such cases, trading partners might decide to handle the simple creates and deletes using the incremental style, yet use the full-refresh for updates of existing entities.

General Rules

The following rules apply to both the Snapshot and the Incremental data management approaches described in separate sections.

For any ActionVerbType-based BOD instance the following rules apply:

> **[R14]** The set of elements being managed by a message instance must be well-defined and understood by message senders and receivers.

For a description of the underlying concept, refer back to the section called "Set of Managed Elements". The above rule also is revisited in the separate sections covering Snapshot and Incremental data management approaches.

IDs in Data Management

[R1] An entity to be represented and managed by an element in a noun instance, SHOULD be identified by a standard, primary ID or ID set (in the case of a composite key) for reference in data management operations.

The above rule applies to the design of the Noun as well as data management practices by message senders and receivers.

[R2] To preserve the utility of standard IDs in data management operations between trading partners, it is RECOMMENDED that trading partners avoid managing these same standard IDs within the message instances.

For example, Noun features such as AlternativeDocumentID SHOULD be used to associate a new ID with a noun instance versus migrating a primary ID, like DocumentID, to another ID value. This not only ensures IDs remain stable, but also avoids potential ambiguity in the way data management instructions are specified and interpreted.

[R3] A TaskID MAY be specified in the ApplicationArea (ApplicationArea/Sender/TaskID) as an annotation to tie a message instance back to a business task and related business event.

A business task is a generalization of business action and business event. Business actions correspond to commands and requests triggered by business events and event notifications.

Relating to Use of ActionCriteria and ActionExpression

The following rules apply to the use of ActionCriteria and ActionExpression elements within the ActionVerbType. These rules reflect constraints within the definition of ActionVerbType as well as within the OAGIS BOD architecture itself. The rules also clarify the use of ActionVerbType elements in data management.

[R4] The BOD architecture supports a many-to-many relationship between ActionCriteria and the Noun.

1. A Noun instance may be associated with multiple ActionCriteria instances.
2. An ActionCriteria instance may be associated with multiple Noun instances.

[R5] The flexibility of the schema supports a many-to-many relationship between the verb's ActionExpression and the Noun.

1. A Noun instance may be associated with multiple ActionExpression instances.
2. An ActionExpression instance may be associated with multiple Noun instances.

The next set of rules focus on the use of elements defined within ActionVerbType.

ActionExpression and ChangeStatus are associated through the ActionCriteria element. The cardinalities of these elements mandate that the set of ActionExpressions within the ActionCriteria may be associated with a single ChangeStatus. ChangeStatus can optionally contain such details as a Code (classifying the ChangeStatus), a Description, an EffectiveDateTime, a ReasonCode, zero or more Reason descriptions, and zero or more StateChange elements, each of which may contain a range of state change details.

[R6] If one-to-many ActionExpressions are associated with a ChangeStatus, then that association MUST be represented with exactly one ActionCriteria.

[R7] An actionCode MAY be specified in the actionCode attribute of ActionExpression.

[R8] Where "Add" is specified as the value of actionCode (ActionExpression@actionCode="Add"), the ActionExpression indicates the creation ("addition") of the entity represented by the element matched by the expression.

[R9] Where "Change" is specified as the value of actionCode (ActionExpression@actionCode="Change"), the ActionExpression indicates the modification of the entity represented by the element matched by the expression.

[R10] Where "Delete" is specified as the value of actionCode (ActionExpression@actionCode="Delete"), the ActionExpression indicates the removal ("deletion") of the entity represented by the element matched by the expression.

[**R11]** Where "Replace" is specified as the value of actionCode (ActionExpression@actionCode="Replace") the ActionExpression indicates the replacement of the entity represented by the element matched by the expression.

Note that the use of the "Replace" actionCode attribute is associated with the "Snapshot" data management approach. See the section called “Snapshot Approach”.

[R12] The expression of the ActionExpression must be constructed to match the element(s) of the noun instance representing the entities being managed.

Note: An ActionExpression can apply to a single noun instance or multiple nouns within a BOD and should be constructed appropriately. For example /ProcessCandidate/DataArea/Candidate matches all Candidate nouns within the ProcessCandidate BOD (assuming it contains multiple nouns) whereas /ProcessCandidate/DataArea/Candidate[DocumentIDGroup/DocumentID='C111222'] matches only the Candidate noun with a DocumentID equal to "C111222".

[R13]The expression of the ActionExpressions MUST be written in an XML expression language (XPath or XQuery). If expressionLanguage attribute is not specified, XPath is assumed to be the expression language as a default.

Snapshot Approach

The key characteristics of the snapshot approach are as follows:

- The message sender and receiver must have an understanding of the set of elements being managed. The "snapshot" communicated within a message instance will represent the set of managed elements in its entirety. This would include elements understood to be within the set of managed elements but that have no value assignment.
- The snapshot could be set at the level of a noun or it could be a major element within a noun. The snapshot should be an aggregate that can be referenced by a primary ID. This might be the noun itself (for example, a Candidate noun referenced by its DocumentID) or an aggregate therein (for example, CandidateProfile, which is repeatable within Candidate and referenced by CandidateProfile/ID).
- The sender of a snapshot may be either a system of record (SOR) publishing a snapshot of data or a non-SOR system that is requesting a receiving system to process a snapshot of data. In both cases, the sender may communicate the business task (ApplicationArea/Sender/TaskID) that caused the message to be created and communicated. ChangeStatus, available within the ActionVerbType, enables the reporting of meta-data such as ReasonCode.
- The receiver of a snapshot message must update its system with all the elements within the set of managed elements. This may result in:
    - The creation of entities that were represented in the message but that did not previously exist on the system;
    - The updating of existing entities with corresponding refreshed data from the message instance; and
    - The deletion of entities that have no value assignment within the message instance, but that existed on the receiving system.

Because the message instance includes all data within the scope of the snapshot, the message also may include elements representing entities that remain unchanged on the receiving system.

Guidelines

There are many considerations that go into a data management policy and its successful execution. The following are intended as baseline rules for using the snapshot approach within BOD instances.

Under the snapshot approach, the following rules apply for any BOD instance based on ActionVerbType:

[R14.1] The set of elements being managed must be well-defined and understood by message senders and receivers.

[R15] Any of the OAGIS action verbs implemented within the HR-XML library MAY be used.

[R16] The actionCode attribute of ActionExpression MUST be restricted to the set of values: {Replace}

Incremental Approach

The key characteristics of the incremental or "Delta" approach are as follows:

- A subset of a Noun is communicated in a message instance; note that the subset could be the Noun itself, or any element therein.
- The set of managed elements within a message instance must be well-defined and understood by the senders and receivers.
- Any subset of elements managed together should be aggregated as an element in the message and be identifiable through the use of standard IDs.
- Elements included in a message instance (with the exception of ID(s)) are limited to those that contain entities that have been created, updated or deleted.
- Detailed data management instructions for create, update, and delete operations are communicated in the message instance using a combination of ActionExpression, actionCode, and noun content.

The sender of an Incremental message may be either a system of record (SOR) publishing a create, update or delete operation or a non-SOR system that is requesting its targeted receivers to process a create, update, or delete operation.

The receiver of an Incremental message instance must update its system per the data management instructions (create, update, or delete operations on some set of elements) conveyed in the message instance through the ActionExpressions. This may result in the "creation", "update", or "deletion" of entities in the system as specified in the data management instructions.

Physical vs. Logical Delete

The receiving system may interpret a "deletion" as either a physical delete or logical delete and is dependent upon the receiving systems data retention policies.

Guidelines

Under the incremental approach, the following rules apply for any BOD instance based on ActionVerbType:

[R14.2] The set of elements being managed MUST be well-defined and understood by message senders and receivers.

[R17] Any of the OAGIS action verbs MAY be used.

For further information, see the section called “Verb Types and Responses”.

[R18] The actionCode attribute of ActionExpression MUST be restricted to the set of values: {Add, Change, Delete}

"Add" Operation Conditions

For any “Add” operation, the following rules apply:

[R19] The message instance must contain an ActionExpression with an actionCode of "Add".

[R20] The Expression of the ActionExpression must match the element of the noun instance that represents the created entity.

"Delete" Operation Conditions

For any "Delete" operation, the following rules apply:

[R21] The message instance MUST contain an ActionExpression with an actionCode of "Delete".

[R22] The Expression of the ActionExpression must match the element of the noun instance that represents the created entity.

For any "Delete" operation, where the entity being deleted is identifiable with ID(s) the following rules apply:

[R23] The message instance noun MUST only provide a reference to the entity via its ID(s).

"Update" Operation Conditions

Identification of the data entity deserves special note; cases exist where the data entity being managed does not have a designated identifier or key (i.e., surrogate key, etc.) that is common and available across the communicating systems. For example, consider a Person with many Addresses where the Address entity does not have an identifier that is shared across systems; it is the value of properties of the entity, itself, that provide for the unique identification. Because it is not possible to issue an update operation on an entity using a common ID, the sending system must issue two operations: one delete operation for the "old" entity and one create operation for the "new" entity. This approach realizes the "old/new value" of the "optimistic locking" data management transaction model for maintaining data integrity.

For any "Update" operation, the following rules apply:

[R24] The message instance must contain an ActionExpression with an actionCode of "Change".

[R25] The Expression of the ActionExpression must match the element of the noun instance that represents the updated entity.

[R26] The message instance noun must provide a reference to the entity via its ID(s) (if one exists) and only include the updated properties of the entity.

For any "Update" operation, where the entity is not identifiable with ID(s) the following rules apply:

[R27] The message instance must contain two ActionExpressions and two noun instances

The first ActionExpression with an actionCode of "Delete" must identify the element being deleted, as represented in the first noun instance

The second ActionExpression with an actionCode of "Add" must identify the element being created, as represented in the second noun instance

Note that add and update operations of the incremental approach apply R14 to represent element null assignments.

SubstitutionGroups

The use of Substitution Groups through xsi:type or xsd:substitutionGroups.is NOT RECOMMENDED by the HR-XML Consortium. Although the schema may validate, a trading partner may not be able to process it.

Appendix A: Sample Data Sets

The following examples further describe the concepts of BOD architecture and XML syntax.

Simple Request/Response Transaction

```
<ProcessCandidate
    releaseID="3.3"
    languageCode="en-US"
    xsi:schemaLocation="http://www.hr-xml.org/3  ../Developer/BODs/ProcessCandidate.xsd"
    xmlns="http://www.hr-xml.org/3"
    xmlns:oa="http://www.openapplications.org/oagis/9"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <oa:ApplicationArea>
        <oa:CreationDateTime>2013-10-17T10:09:02.01Z</oa:CreationDateTime>
        <oa:BODID>PC2013-11-17-052</oa:BODID>
    </oa:ApplicationArea>
    <DataArea>
        <oa:Process>
            <oa:ActionCriteria>
                <oa:ActionExpression
actionCode="Add">/ProcessCandidate/DataArea/Candidate</oa:ActionExpression>
                <oa:ChangeStatus>
                  <oa:ReasonCode>Candidate Applied</oa:ReasonCode>
                </oa:ChangeStatus>
            </oa:ActionCriteria>
        </oa:Process>
        <Candidate majorVersionID="1" minorVersionID="0">
            <DocumentID>123456</DocumentID>
            <CandidatePerson>
                <PersonName>
                    <FormattedName>Tom Smith</FormattedName>
                    <oa:GivenName>Tom</oa:GivenName>
                    <FamilyName>Smith</FamilyName>
                </PersonName>
            </CandidatePerson>
        </Candidate>
    </DataArea>
</ProcessCandidate>
```

Note the following:

- Components with the "oa:" prefix are from the Open Applications Group XML namespace.
- The application area contains a creation timestamp (oa:CreationDateTime) and a unique identifier for the BOD (oa:BODID).
- Within the DataArea, there is a "Process" verb and a Candidate noun instance.
- "Process" is an "action verb" type (see the section called “Verb Types and Response Patterns”). It has an "ActionExpression" that sets the scope of what is to be managed.
- An "ActionCode" of "Add" and the expression (in this case, using the default XPath expression language) comprise a "data management instruction" to be performed on the noun.
- The noun is an HR-XML "Candidate" instance. In this example, the instance has a DocumentID set to "123456" The DocumentID represents the primary identifier for the business object being managed. Within HR-XML nouns, alternative IDs are permitted as well.
- The noun instance contains a single component (the Candidate's name).

Two different approaches to handling updates are supported. One involves the communication of only "delta" or "incremental" change information whereas the other approach relies on a "snap shot" or full-refresh approach. The example above, by use of an "ActionCode" of "Add" and an "ActionExpression" is indicative of the incremental style of updates.

The BOD architecture anticipates requests as well as responses. So next in our example would be a response to the ProcessCandidate BOD. An example of a possible response is shown below.

```
<AcknowledgeCandidate
    releaseID="3.3"
    languageCode="en-US"
    xsi:schemaLocation="http://www.hr-xml.org/3  ../Developer/BODs/AcknowledgeCandidate.xsd"
    xmlns="http://www.hr-xml.org/3"
    xmlns:oa="http://www.openapplications.org/oagis/9"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <oa:ApplicationArea>
        <oa:CreationDateTime>2013-10-17T10:09:02.21Z</oa:CreationDateTime>
        <oa:BODID>PC2013-21-12-123</oa:BODID>
    </oa:ApplicationArea>
    <DataArea>
        <oa:Acknowledge>
            <oa:OriginalApplicationArea>
                <oa:CreationDateTime>2013-10-17T10:09:02</oa:CreationDateTime>
                <oa:BODID>PC2013-11-17-052</oa:BODID>
            </oa:OriginalApplicationArea>
             <oa:ResponseCriteria>
```

```
                <oa:ResponseExpression
actionCode="Accepted">/AcknowledgeCandidate/DataArea/Candidate</oa:ResponseExpression>
                <oa:ChangeStatus>
                    <oa:Code>Candidate Profile Created</oa:Code>
                </oa:ChangeStatus>
            </oa:ResponseCriteria>
        </oa:Acknowledge>
        <Candidate majorVersionID="1" minorVersionID="0">
            <DocumentID>123456</DocumentID>
        </Candidate>
    </DataArea>
</AcknowledgeCandidate>
```

Basically, this does little more than acknowledge that the entity communicated in the prior ProcessCandidate BOD represented by the DocumentID "123456" was created.

Let's assume that the sender of the original ProcessCandidate changes its mind the following day and wants to cancel the original request. Let's say that this isn't a simple delete, but something that would require some business calculation or approval by the recipient. Such a request might be communicated in a request like the one below.

```
<CancelCandidate
    releaseID="3.3"
    languageCode="en-US"
    xsi:schemaLocation="http://www.hr-xml.org/3  ../Developer/BODs/CancelCandidate.xsd"
    xmlns="http://www.hr-xml.org/3"
    xmlns:oa="http://www.openapplications.org/oagis/9"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <oa:ApplicationArea>
        <oa:CreationDateTime>2013-10-18T10:07:02.01Z</oa:CreationDateTime>
        <oa:BODID>PC2013-31-27-4123</oa:BODID>
    </oa:ApplicationArea>
    <DataArea>
        <oa:Cancel>
            <oa:ActionCriteria>
                <oa:ActionExpression
actionCode="Delete">/CancelCandidate/DataArea/Candidate</oa:ActionExpression>
                <oa:ChangeStatus>
                    <oa:ReasonCode>Requested Removal</oa:ReasonCode>
                </oa:ChangeStatus>
            </oa:ActionCriteria>
        </oa:Cancel>
        <Candidate majorVersionID="1" minorVersionID="0">
            <DocumentID>123456</DocumentID>
        </Candidate>
    </DataArea>
</CancelCandidate>
```

The above message instance simply requests that the receiving system cancel the entity represented by the DocumentID "123456". While not shown here, this request would be responded to by another AcknowledgeCandidate message instance that might accept, reject, or possibly modify the cancellation request.

Note that this simple example demonstrates one possible pattern of message exchange using BODs. Dozens of other examples are available within the library documentation. Also see the section called “Subset of OAGIS Verbs Implemented” and the section called “Data Management Approaches” for a complete description of OAGIS verbs and data management features.

Common Usage of BOD Expression

- XPath may be used to identify specific nodes within the schema.
- A String may be used for a search query. For example, PersonID = "12345"

Note: the following examples only include the XPATH expression. Complete xml examples would also include the noun data area.

```
<!-- In the following Notify BOD snippit, the status is modified and signature date is added.  -->
<NotifyStaffingAssignment>
    <DataArea>
        <oa:Notify>
            <oa:ActionCriteria>
                <oa:ActionExpression
actionCode="Modified">/NotifyStaffingAssignment/DataArea/StaffingAssignment/StaffingContractDetai
ls/ContractStatusCode</oa:ActionExpression>
            </oa:ActionCriteria>
            <oa:ActionCriteria>
                <oa:ActionExpression
actionCode="Add">/NotifyStaffingAssignment/DataArea/StaffingAssignment/StaffingContractDetails/Sig
natureDate</oa:ActionExpression>
            </oa:ActionCriteria>
        </oa:Notify>
    </DataArea>
</NotifyStaffingAssignment>
```

```
<!-- The following Process BOD snippet processes section 3 of the I9 document. -->
<ProcessEmploymentEligibilityI-9>
        <DataArea>
                <oa:Process>
                        <oa:ActionCriteria>
                                <oa:ActionExpression
actionCode="Add">/ProcessEmploymentEligibilityI-9/DataArea/EmploymentEligibilityI-9/I-
9Form/Section3</oa:ActionExpression>
                        </oa:ActionCriteria>
                </oa:Process>
        </DataArea>
</ProcessEmploymentEligibilityI-9>

<!-- The following Show BOD snippet specifies that the Get document was accepted and Show returned
based on the Document ID. -->
<ShowScreeningCatalogReport>
        <DataArea>
                <oa:Show>
                        <oa:ResponseCriteria>
                                <oa:ResponseExpression
actionCode="Accepted">/ShowScreeningCatalogReport/DataArea/ScreeningCatalogReport/DocumentID
</oa:ResponseExpression>
                        </oa:ResponseCriteria>
                </oa:Show>
        </DataArea>
</ShowScreeningCatalogReport>
```

Unique Identifiers

This next section provides additional examples for request/response BOD’s. Several methods may be used to reference a set of BOD's. For example, an implementer may use the BODID or the DocumentID.

DocumentID as Unique Identifier

In the following Get and Show BOD snippits, the DocumentID is used as the unique identifier between the two BOD's.

```
<GetScreeningCatalogOrder>
        <DataArea>
                <oa:Get>

        <oa:Expression>/GetScreeningCatalogOrder/DataArea/ScreeningCatalogOrder</oa:Expression>
                </oa:Get>
                <ScreeningCatalogOrder>
                        <DocumentID>12345076</DocumentID>
                </ScreeningCatalogOrder>
        </DataArea>
</GetScreeningCatalogOrder>

<!--The ShowScreeningCatalogReport uses the DocumentID to tie back to the DocumentID of the
GetScreeningCatalogOrder. -->
<ShowScreeningCatalogReport>
        <DataArea>
                <oa:Show>
                        <oa:ResponseCriteria>
                                <oa:ResponseExpression
actionCode="Accepted">/ShowScreeningCatalogReport/DataArea/ScreeningCatalogReport/DocumentID
</oa:ResponseExpression>
                        </oa:ResponseCriteria>
                </oa:Show>
                <ScreeningCatalogReport>
                        <DocumentID>12345076</DocumentID>
                </ScreeningCatalogReport>
        </DataArea>
</ShowScreeningCatalogReport>
```

BODID as Unique Identifier

In the following Sync and ConfirmBOD BOD snippits, the BODID is used as the unique identifier between the two BOD's.

```
<SyncScreeningCatalogReport >
        <oa:ApplicationArea>
                <oa:BODID>983765</oa:BODID>
        </oa:ApplicationArea>
        <DataArea>
```

```
                <oa:Sync>
                        <oa:ActionCriteria>
                                <oa:ActionExpression actionCode="Replace"></oa:ActionExpression>
                        </oa:ActionCriteria>
                </oa:Sync>
        </DataArea>
</SyncScreeningCatalogReport>

<!--ConfirmBOD uses the original BODID to tie back to the BODID of the SyncScreeningCatalogReport. -->
<oa:ConfirmBOD>
        <oa:ApplicationArea>
                <oa:BODID>123-983765</oa:BODID>
        </oa:ApplicationArea>
        <oa:DataArea>
                <oa:Confirm>
                        <oa:ResponseCriteria>
                                <oa:ResponseExpression
actionCode="Accepted">/oa:ConfirmBOD/oa:DataArea/oa:BOD/oa:OriginalApplicationArea/oa:BODID</
oa:ResponseExpression>
                        </oa:ResponseCriteria>
                </oa:Confirm>
                <oa:BOD>
                        <oa:OriginalApplicationArea>
                                <oa:BODID>983765</oa:BODID>
                        </oa:OriginalApplicationArea>
                </oa:BOD>
        </oa:DataArea>
</oa:ConfirmBOD>
```