PRODML Generic Data Access (GDA) Web Service Specification

PRODML™:	Enabling Production Operations, Optimization, Reporting and Data Management for the Exploration and Production (E&P) Domain.
Version:	2.1
Abstract	This document describes the PRODML Generic Data Access (GDA) Web Service.
Prepared by:	Energistics and the PRODML SIG
Date created:	22-Nov-2011

energistics™
The Energy Standards Resource Centre

Document Information	
DOCUMENT VERSION:	2.1
DATE:	22-Nov-2011
Technical	Color: R: 210 G:124, B50
Language	US English

This document was produced by
Energistics and the PRODML SIG

Amendment History			
Version	**Date**	**Comment**	**By**
V2.1	22-Nov-2011	• DeleteData operation added. • Split PRODML 2.0 API document into individual documents for each service interface/WSDL. • Replaced pseudo-code descriptions with UML diagrams. • Revamped error handling to specify the use of SOAP faults instead of return codes. • Enhanced descriptions of parameters.	Mark Hollier Michael Knewtson Gary Masters Bill McKenzie
V2.0	09-Jun-2009	Final Revisions for Publication	Stefan Avramtchev Robin Getty John Gotsell Mark Hollier Bill McKenzie

Table of Contents

1 Overview

1.1 Document Aims

This document describes the core components of PRODML Generic Data Access (GDA) Web service. It specifies the contracts for the interfaces and the data schemas used in these contracts.

1.2 Terminology and Basic Concepts

- A *PRODML Server* is an individual instance of an implementation of (at least) a subset of the interfaces which are defined by the PRODML standard.
- The term *API* as used in this document refers, depending on the context, either to a Web Service interface by which a client node accesses a server node.
- The term *PRODML GDA API* or simply *GDA API* as used in this document means the API defined in this document for Generic Data Access. The PRODML GDA API is a Web service definition, supporting access to a range of data-objects. GDA is used as shorthand for the Generic Data Access Service.
- A *data-object* is a logical organization and grouping of the data items associated with the major components and operations involved in some application domain or scope. A data-object is not a formal programming object (with methods, properties, events, etc.). It represents the actual data to be processed by a function as opposed to the other parameters which qualify the function to be applied. The data-object is required to be a member of a particular element substitution group that is referenced in the WSDL. This service definition does not pre-define the list of supported data-objects but all PRODML or WITSML defined data-object are targeted as candidates for use in a particular implementation of the service.
- *Parameters* are the values supplied to a Web service method when it is called. *Return Parameters* are the values returned by a Web service upon return.
- A Service *call* means a request-response message exchange sequence between the client and the service. Thus *calling* should be interpreted as initiating such a message exchange.
- A *field* in an object refers to a subset of data in a data instance, corresponding to an element or attribute in the schema of that data instance.
- The term *currently* as used in this document means 'at the time of publication of this document'.

1.3 Style Guidelines

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY" and "OPTIONAL" in this document are to be interpreted as described in RFC 2119. (http://www.ietf.org/rfc/rfc2119.txt). To distinguish between server and client requirements, these key words SHALL be prefaced with SERVER and/or CLIENT.

1.4 Typographical Conventions

This section describes common typographical use conventions used throughout this document.

Convention	Description and Example
Key Names and Combinations	Capital letters are used for the names of keys and key sequences, such as ENTER and CTRL+R. A plus sign (+) between key names indicates a combination of keys or menu item selections. For example, ALT + F1 means to hold down the ALT key while pressing the F1 key.
Parameter Names	Italic font is used when referring to methods, parameter names, variables, or the names of XML elements and attributes. For example: The operation accepts the parameter *Uid*.
Parameter Values	A special font marks parameter values or the value of XML elements and attributes. For example: Specify the value of `UidOnly` for the *SubsetType* parameter.

XML Examples	Blocks of XML examples are set apart from the body and marked accordingly. For example:
	`<MissingParameterFault>`
	`  <Parameter>DateTimeCriterion/Path</Parameter>`
	`</MissingParameterFault>`

2 Introduction

This document describes PRODML GenericDataAccess (GDA) Web Service Interface ('API'). This document is presented in an 'API' style which is intended to be familiar to application programmers. It describes and documents the behavior associated with the normative WSDL and XSD definitions of the GenericDataAccess (GDA) Web Service interface. All pseudo-code programming language examples given in this document are simply augmenting and non-normative.

The internal representation of data, data structures, data-object any other concepts *within the client and server systems* is considered to be out of scope and, therefore, unspecified.

2.1 Versioning

Release 1.1 of PRODML was the final version in which data schemas and service descriptions were released as a single package. Subsequent releases of the service definitions follow a schedule and version numbering sequence that is independent of the supported data-schemas. This specification contains the PRODML Generic Data Access (GDA) Web Service and is part of v2.1 of the PRODML Services Specification.

2.2 WSDL and XSD

The signatures of the Web service interfaces are defined in WSDL. The data involved in these web service interfaces are described using XSD Schema syntax. The WSDL file is the normative definition of the data structure but this document is the normative definition of the behavior related to the service.

In order to use the GDA with a specific data-object, either the WSDL must be modified to add one or more “import” statements that incorporate a path to the appropriate file or the code generation toolset must be configured to utilize specific data-object schemas. However, in order to allow the GDA specification to be agnostic toward which data-object schemas can be used, the published WSDL does not import any explicit data-object schemas. Rather, it references an abstract “element substitution group”. Any schema which is a member of this group is a candidate to be accessed by the GDA. The PRODML v1.2+ and WITSML v1.4.1+ plural root of the data-object schemas are all members of this element substitution group.

2.3 Faults and Exceptions

For functional faults, a SERVER MUST generate and return the appropriate SOAP Fault.

For any nonfunctional and infrastructural failures and in general for any condition which causes the processing of a service request to abruptly terminate, the SERVER MUST generate and return a SOAP Fault. The textual component of the SOAP Fault (the SOAP String) should be a message which is informative to the operator as to the cause of the condition and/or possible remedial action. The code component of the SOAP Fault (the SOAP Code) is unspecified.

On receipt of a SOAP Fault, the CLIENT SHOULD throw a SOAP exception whose message includes the message from the SOAP fault.

3 Generic Data Access (GDA) Service

3.1 Description

The GDA service operates on the principle that the client requests or submits data by issuing a call to the service. The call has parameters providing contextual data, as well as parameters typifying, identifying and constraining the data and/or the data retrieval process. The service returns all of the requested data along with diagnostics information about the outcome of the data access operation.

A particular GDA server may not be able to provide the full range of objects that it is possible to ask for. In order for the client to find out what can be provided, the GDA service allows its capabilities (such as data-object availability, identification methods, and other service capabilities) to be queried.

The service operations and parameters are described below. There are method and class definitions given, that are kept clean of language and platform specifics in order to avoid assumptions of any programming environment.

3.2 Service Operations

The operation of the Generic Data Access interface are summarized in this section. The complete definition of the parameters are to be found further in this section.

Table 1 Generic Data Access Service Operations

Operation	Message	Element	XSD Type Name	Description
GetCapabilities – retrieves information about the particular service capabilities.	GetCapabilities	n/a		Request for the server capabilities.
	GetCapabilitiesResponse	getCapabilitiesResult	GenericDataAccessCapabilites	Information describing the capabilities of the service, like different supported kinds, operations, identification mechanisms, etc.
GetData – retrieves data-objects from the service.	GetData	query	GetDataQuery	Request for the retrieval of different data-object kinds.
	GetDataResponse	getDataResult	GetDataResult	The response of the service containing the results of the operation.
PutData – uploads data-objects to the service	PutData	abstractDataObject	Any member of the abstract substitution group	List of data-objects to upload to the service.
		options	NameValuePair[]	List of options passed to the server.
	PutDataResponse	putDataResult	PutDataResult	Server response containing results of the operation.
DeleteData – permanently deletes data-objects from the server.	DeleteData	Uid	string[]	List of data-object identifiers for deletion.
	DeleteDataResponse	deleteDataResult	DeleteDataResult	Server response containing the results of the operation.

3.2.1 GetCapabilities

Provides detailed information about a PRODML GDA Service implementation capabilities.

A CLIENT MAY call this method to retrieve detailed information about the service implementation, supported data selection and identification mechanisms, and more. However, a SERVER MUST implement this method and return information about its capabilities.

Request Message
The GetCapabilities request message does not contain any parameters.

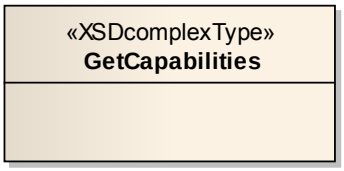

Figure 1 GetCapabilities Request Message

Response Message
The *GetCapabilitiesResponse* response message contains information about the supported data-objects and other details of the service, as described below.

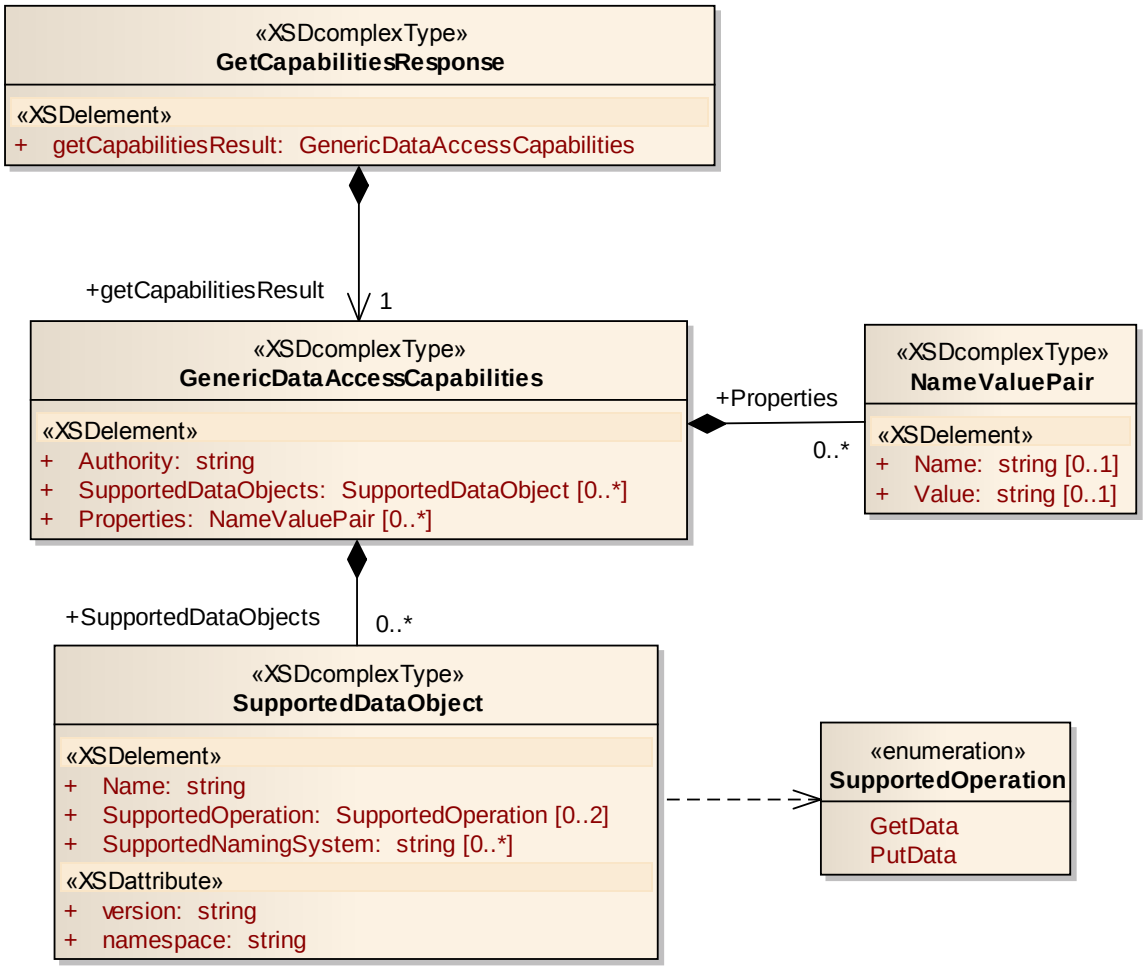

Figure 2 GetCapabilites Response Message

Authority
The *Authority* parameter specifies naming system authority for the primary uids used by the server.

SupportedDataObjects
SupportedDataObjects provides an array of information about supported data-object schemas.

Name
Specifies the kind of substitution group member that is supported by the GDA server. A SERVER MUST specify the element name of a member of the abstractDataObject element substitution group (e.g. `productVolumes`, `wellTests`, `timeSeriesDatas`, etc.).

SupportedOperation
An enumeration which indicates the operations supported on a given data-object for a GDA server. A SERVER MUST return either `PutData` or `GetData` or both for all supported data-objects.

SupportedNamingSystem
Specifies the naming systems supported on a given data-object for a GDA server (e.g. `api`, `witsml`, etc.) Refer to the *PRODML Identifier Specification* for more information on naming systems.

Version
A string specifying the version of the schema containing the supported data-object. In the case of PRODML/WITSML data-objects, this MUST exactly match the version attribute found in the top-level plural object (e.g. '1.2.0.0(PRODML)', '1.3.1.1', etc.). A SERVER MUST specify the versions for all supported data-objects.

Namespace
A string specifying the XML schema namespace of the supported data-object (e.g. 'http://www.prodml.org/schemas/1series', 'http://www.witsml.org/schemas/1series', etc.). A SERVER MUST specify the namespaces for all supported data-objects.

Properties
Properties is an array of name-value pairs used to provide additional information about the server implementation. A SERVER MAY return one or more of the following properties:

Property	Description	Example
`Name`	The name of the server implementation.	`Server #1`
`Description`	A description of the server implementation.	`PRODML GDA 2.0 for Production Accounting`
`Version`	The version (build) of the server implementation.	`1.0.0.4092`
`Vendor`	The vendor (producer) of the software	`Acme Energy`

GenericDataAccessCapabilities Example

```
<GenericDataAccessCapabilities
        xmlns="http://www.prodml.org/api/210/genericDataAccess">
  <Authority>www.bigoil.com</Authority>
  <SupportedDataObjects>
    <SupportedDataObject version="1.2.0.0(PRODML)"
          namespace="http://www.prodml.org/schemas/1series">
      <Name>productionOperations</Name>
      <SupportedOperation>Get</SupportedOperation>
      <SupportedOperation>Put</SupportedOperation>
    </SupportedDataObject>
    <SupportedDataObject version="1.2.0.0(PRODML)"
          namespace="http://www.prodml.org/schemas/1series">
      <Name>productVolumes</Name>
      <SupportedOperation>Get</SupportedOperation>
      <SupportedOperation>Put</SupportedOperation>
    </SupportedDataObject>
    <SupportedDataObject version="1.3.1.0"
          namespace=" http://www.witsml.org/schemas/131">
      <Name>wells</Name>
      <SupportedOperation>Get</SupportedOperation>
      <SupportedNamingSystem>api</SupportedNamingSystem>
      <SupportedNamingSystem>witsml</SupportedNamingSystem>
    </SupportedDataObject>
  </SupportedDataObjects>
  <Properties>
    <NameValuePair>
      <Name>Name</Name>
      <Value>Server #1</Value>
    </NameValuePair>
    <NameValuePair>
      <Name>Version</Name>
      <Value>1.0.0.4092</Value>
    </NameValuePair>
    <NameValuePair>
      <Name>Vendor</Name>
      <Value>Acme Energy</Value>
    </NameValuePair>
  </Properties>
</GenericDataAccessCapabilities>
```

3.2.2 GetData

A service call which returns one or more instances of the same kind of data-object from the server.

Request Message

Requests to the GDA are made by the use of a *query* parameter that defines, in abstract terms, a common set of criteria that can be applied to most data-objects in the PRODML and WITSML domains. In this section of the document we discuss the structure of the query parameter and how its data fields should be used in the context of various kinds of data.

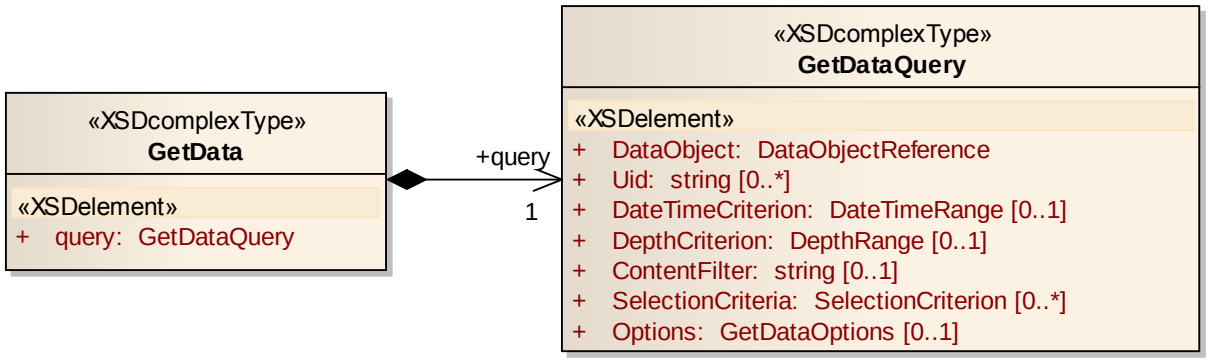

Figure 3 GetData Request Message

query

The *query* parameter encapsulates a single set of query information. A SERVER MUST treat the query items in this structure as a logical AND (all must be true in order for the result to be included) to come up with a list of returned data-objects.

DataObject

The *DataObject* parameter specifies the kind of substitution group member that is to be returned by the GDA server. The *DataObject* parameter is defined as a DataObjectReference structure, as shown in the following diagram.

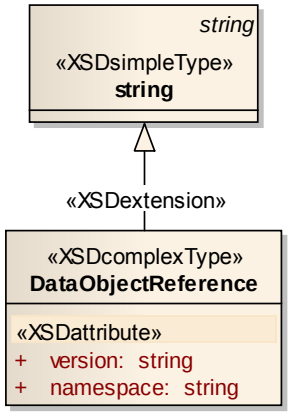

Figure 4 DataObjectReference Request Parameter

The CLIENT MUST specify an element name of a member of the abstractDataObject element substitution group (e.g., `productVolume`**`s`**, `wellTest`**`s`**, `timeSeriesData`**`s`**, etc.) supported by the server. If not, then the SERVER MUST return a *MissingParameterFault* SOAP fault.

In addition, the CLIENT MAY also supply the version for the requested data-object. If the client requests a specific version, then the SERVER MUST return an object up-to the number of digits specified. If a specific version is not requested, then the SERVER MUST return the lowest supported version. For example, if the client requests version '1.2', then the SERVER MAY return an object version of '1.2', '1.2.1', or '1.2.1.4' but the SERVER MUST NOT return an object version of '1.3'.

The SERVER SHALL assume that a request for a plural WITSML or PRODML element substitution group member is a request for singular data-objects in the specified namespace and version supported by the server.

version

The requested version of the data-object which is to be returned.

namespace

The namespace of a data-object which is to be returned.

For example, the following specifies a WITSML v1.3.1 well data-object.

```
<DataObject namespace="http://www.witsml.org/schemas/131"
            version="1.3.1.0">wells</DataObject>
```

Uid

The *Uid* parameter is an identifier that identifies a data-object to be included in the returned results. Multiple Uid fields may be specified. The Uid value is considered to be a PRODML URI and is recommended to be formatted as such. Refer to the *PRODML Identifiers Specification* for more information. The Uid parameter is considered the primary way of constraining a returned data set to a subset of the data-objects that are held in the server.

It is important to understand that the identifiers in this input parameter are not necessarily the same as the identifiers to be returned, although they may be. For example, in requesting the `wellLogs` data-object from a GDA server, a Uid may be for a well, a wellbore, or even a wellLog. If a wellbore identifier is passed, it can be assumed that all logs for that wellbore are requested (or a filtered list, based on other parameters discussed below). If a URI for a specific wellLog is passed, then that individual log will be returned.

For example, the following specifies a identifier for a well with a specified API number.

```
<Uid>prodml://api/well/4200150001</Uid>
```

DateTimeCriterion

The *DateTimeCriterion* parameter is defined as a DateTimeRange structure, as shown in the following diagram.

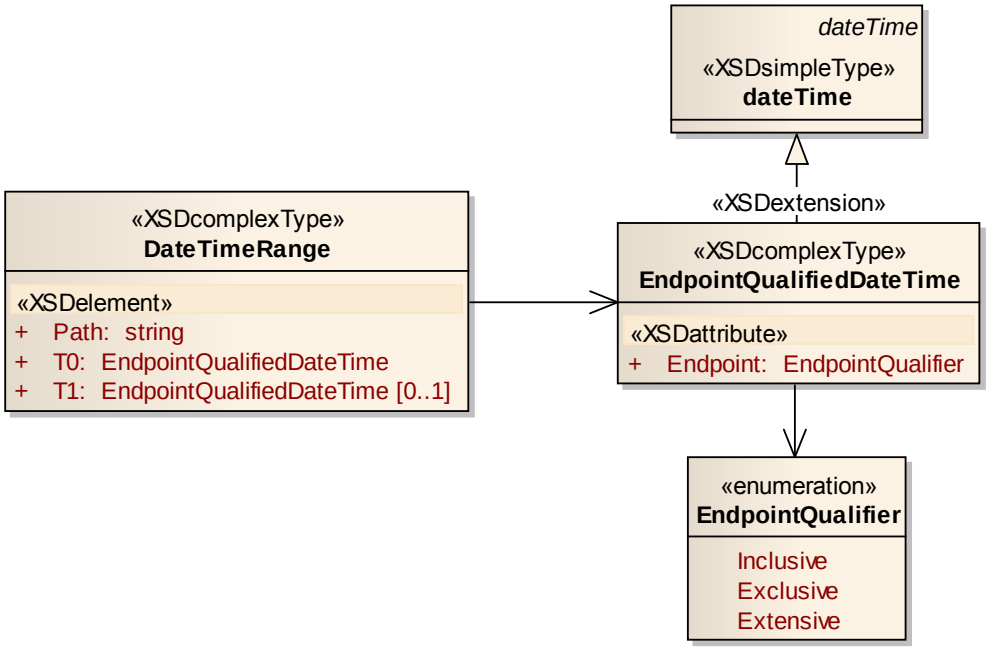

Figure 5 GetData DateTimeCriterion Request Parameter

The *DateTimeCriterion* parameter specifies the start and end dates/times. This query parameter is meant to be interpreted as a 'where clause' on the objects being requested. It includes a *Path* parameter, which should refer to the name of the date/time element or attribute of interest in the object being requested, using a path syntax form (see Appendix B – Element Path Syntax). The root of the path SHALL be the element substitution group member.

The semantics of the where clause depend on the recurrence of the element specified in the path. If the element specified is non-recurring with respect to the overall data-object, then the criteria is used to determine which data-objects are selected. If the element specified is part of a recurring section of the data-object, then it is interpreted to mean which part of the recurring section is returned.

The CLIENT MUST specify the *Path* parameter if *DateTimeCriterion* is used. The CLIENT MUST specify the *Path* parameter if *DateTimeCriterion* is used. If not, then the SERVER MUST return a *MissingParameterFault* SOAP fault with *Parameter* set to `DateTimeCriterion/Path`.

```
<MissingParameterFault>
    <Parameter>DateTimeCriterion/Path</Parameter>
</MissingParameterFault>
```

In addition, the CLIENT MUST specify a start date/time in the *T0* parameter. If not, then the SERVER MUST return a *MissingParameterFault* SOAP fault.

Optionally, the CLIENT MAY also specify the end date/time in the *T1* parameter.

Optionally, the CLIENT MAY specify one of the following values for *EndPoint* in the T0 and T1 parameters:

- `Inclusive` – The interval returned includes all data between and including the boundary points.
- `Exclusive` – The interval returned includes all data between, but not including the boundary points.
- `Extensive` - The interval returned includes all data between and including the first value outside of the boundary, if there is not a value at the boundary.

If *EndPoint* is not specified, then the SERVER SHALL assume the default value of "Inclusive".

If an implementation does not support the *DateTimeCriterion* parameter, then the SERVER MUST return a *UnsupportedCriterionFault* SOAP fault with the *CriterionName* field set to `DateTimeCriterion`.

```
<UnsupportedCriterionFault xmlns="http://www.prodml.org/api/210/genericDataAccess">
    <CriterionName>DateTimeCriterion</CriterionName>
</UnsupportedCriterionFault>
```

DateTimeCriterion Example

For example, to request the WITSML well data-objects which were modified in 2009, the following would be used:

```
<DateTimeCriterion>
    <Path>dTimLastChange</Path>
    <T0 Endpoint="Inclusive">2009-01-01T00:00:00Z</T0>
    <T1 Endpoint="Exclusive">2010-01-01T00:00:00Z</T1>
</DateTimeCriterion>
```

DepthCriterion

The *DepthCriterion* parameter is defined as a DepthRange structure, as shown in the following diagram.

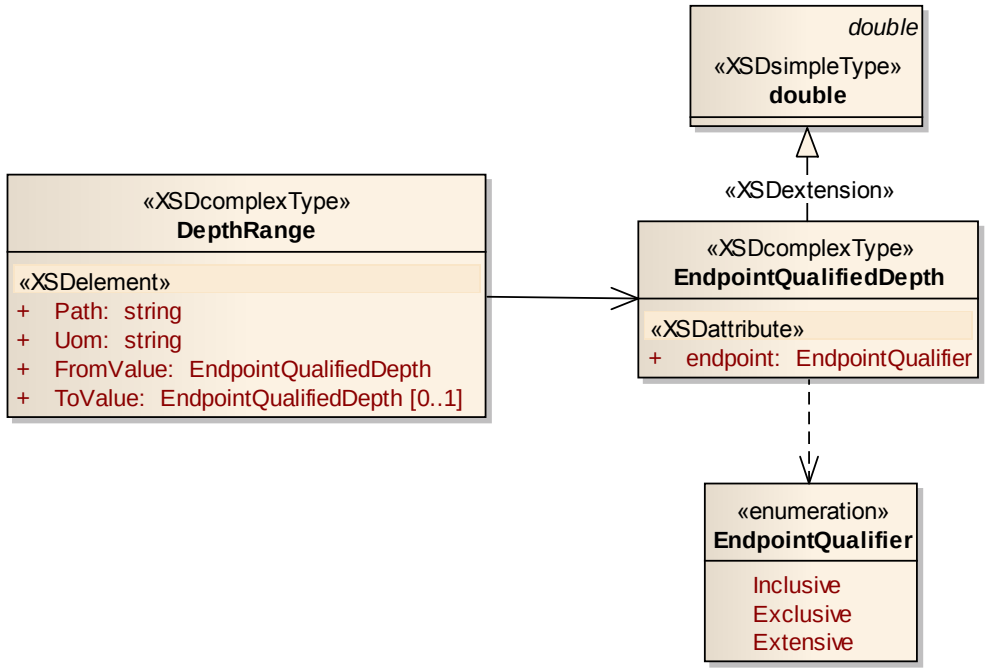

Figure 6 DepthCriterion Parameter

The *DepthCriterion* parameter specifies the start and end depths. This is meant to be interpreted as a 'where clause' on the whole of the object being selected. It is not meant to specify the range of depth desired in a growing data-object. It includes a *Path* parameter, which should refer to the name of the depth element of interest in the object being requested, using a using a path syntax form (see Appendix B – Element Path Syntax. The root of the path SHALL be the element substitution group member.

The semantics of the where clause depend on the recurrence of the element specified in the path. If the element specified is non-recurring with respect to the overall data-object, then the criteria is used to determine which data-objects are selected. If the element specified is part of a recurring section of the data-object, then it is interpreted to mean which part of the recurring section is returned.

The CLIENT MUST specify the *Path* parameter if *DepthCriterion* is used. If not, then the SERVER MUST return a *MissingParameterFault* SOAP fault with *Parameter* set to `DepthCriterion/Path`.

```
<MissingParameterFault>
    <Parameter>DepthCriterion/Path</Parameter>
</MissingParameterFault>
```

In addition, the CLIENT MUST specify a start depth in the *T0* parameter. The CLIENT MUST specify the *Path* parameter if *DepthCriterion* is used. If not, then the SERVER MUST return a *MissingParameterFault* SOAP fault with *Parameter* set to `DepthCriterion/FromValue`.

```
<MissingParameterFault>
    <Parameter>DepthCriterion/FromValue</Parameter>
</MissingParameterFault>
```

Optionally, the CLIENT MAY also specify the end depth in the *T1* parameter.

Optionally, the CLIENT MAY specify one of the following values for *EndPoint* in the T0 and T1 parameters:

- `Inclusive` – The interval returned includes all data between and including the boundary points.
- `Exclusive` – The interval returned includes all data between, but not including the boundary points.
- `Extensive` - The interval returned includes all data between and including the first value outside of the boundary, if there is not a value at the boundary.

If *EndPoint* is not specified, then the SERVER SHALL assume the default value of "Inclusive".

If an implementation does not support the *DepthCriterion* parameter, then the SERVER MUST return a *UnsupportedCriterionFault* SOAP fault with the *CriterionName* field set to `DepthCriterion`.

```
<UnsupportedCriterionFault xmlns="http://www.prodml.org/api/210/genericDataAccess">
    <CriterionName>DepthCriterion</CriterionName>
</UnsupportedCriterionFault>
```

DepthCriterion Example

For example, the WITSML wellbore contains several depth-related elements, such as `mdCurrent`, `tvdCurrent`, `mdKickoff`, etc. To request wellbores which have a current measured depth between 10000 and 12000 feet (inclusive), the following would be used:

```
<DepthCriterion>
    <Path>mdKickoff</Path>
    <Uom>ft</Uom>
    <FromValue>10000</FromValue>
    <ToValue>12000<ToValue>
</DepthCriterion>
```

ContentFilter

The *ContentFilter* parameter defines a selection criterion as an XPath statement into the returned data-objects. In the case of a pure XML database, it could possibly be applied as a query into the datastore. In most cases, however, backing databases will be SQL, in which case the ContentFilter would be applied post-query, but before objects are returned to the caller. Note that this also means that objects must be serialized to XML before the XPath can be applied. The XPath statement should be interpreted as an 'assertion' on the candidate object. If the XPath evaluates to any nodes at all, then it is included in the result set. If it does not, it is discarded. For the purposes of this filter, the root node of the XPath statement is considered to be the element substitution group member (e.g., `productVolumes`, `wellTests`, `timeSeriesDatas`, etc.).

For example,

/*:wells/*:well[*ContentFilter*]

SelectionCriteria

An array of path/operator/value triadic parameters used to send server-defined options for the request.

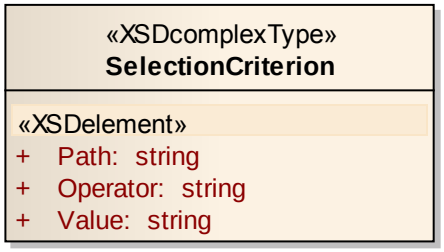

Figure 7 SelectionCriterion

Each path/operator/value triadic may be applied to non-recurring elements. The SERVER SHALL apply this criterion BEFORE the XPath based ContentFilter.

The path must be constructed according to the element path rules in appendix B with the additional constraint that the a recurring element cannot exist in the path.

The operators are limited to the following well-known concepts: “eq”, “ne”, “gt”, “lt”, “ge”, “le”,

For example, the following may be used to specify monthly production when requesting a productVolume data-objects.

```
<ServerDefinedCriterion>
    <Path>periodKind</Path>
    <Operator>eq</Operator>
    <Value>monthly<Value>
</ServerDefinedCriterion>
```

If an implementation does not support the *ServerDefinedCriterion* parameter, then the SERVER MUST return a *UnsupportedCriterionFault* SOAP fault with the *CriterionName* field set to `ServerDefinedCriterion`.

```
<UnsupportedCriterionFault xmlns="http://www.prodml.org/api/210/genericDataAccess">
    <CriterionName>ServerDefinedCriterion</CriterionName>
</UnsupportedCriterionFault>
```

Options

The *Options* parameter is used to specify all of the options for *GetData* and is defined as a GetDataOptions structure, as shown in the following diagram.

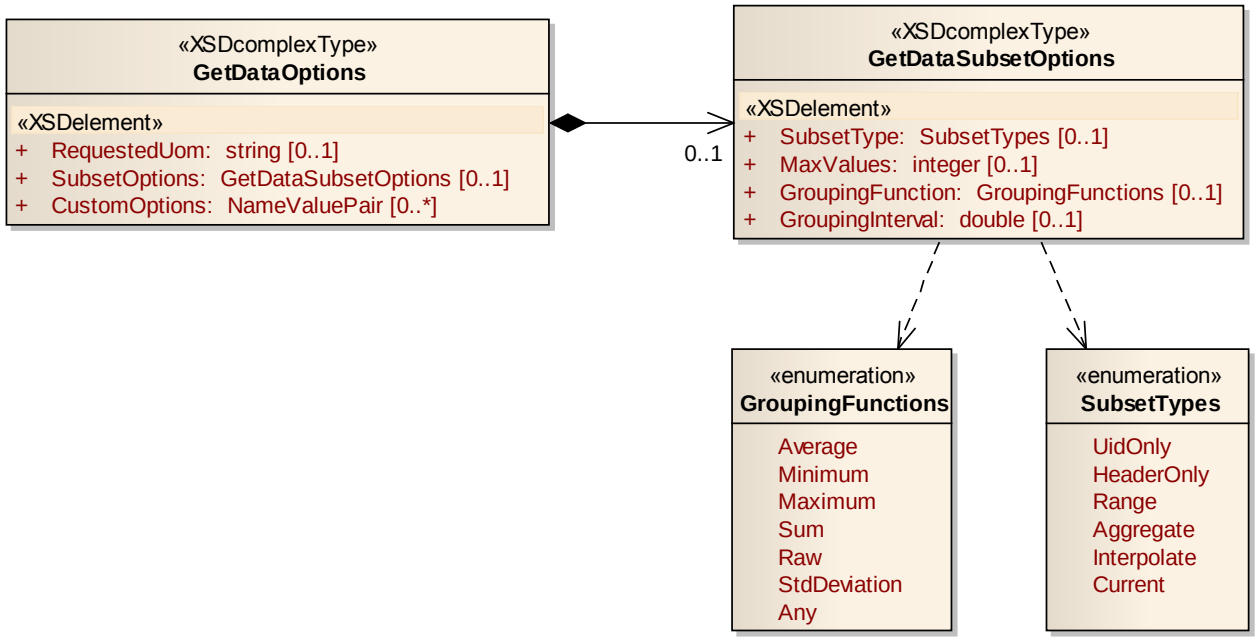

Figure 8 GetData Options Request Parameter

RequestedUom

The requested units of measure field. This value is intended primarily for data-objects for which only one kind of data is being returned (e.g. the `timeSeries` data-object). It is of no practical value for most complex data-objects.

If an implementation does not support the *RequestedUom* parameter, then the SERVER MUST return a *UnsupportedOptionFault* SOAP fault with the *OptionName* field set to `RequestedUom`.

```
<UnsupportedOptionFault xmlns="http://www.prodml.org/api/210/genericDataAccess">
    <OptionName>RequestedUom</OptionName>
</UnsupportedOptionFault>
```

SubsetOptions

The *SubsetOptions* parameter is used to specify what portion or subset of a growing object should be returned. Growing data-objects are those that have a sub-component that grows over time, such as a time series from a data historian, or a well log. The *SubsetOptions* parameter has the following elements:

SubsetType

The *SubsetType* parameter is an enumeration that allows various parts of the requested objects to be returned. If the SubsetType parameter is not specified, then the SERVER SHALL return all data fields available for the the request data-object(s).

SubsetType	Description
`UidOnly`	Indicates that the return parameter of the query should only contain the URIs or ID elements of the selected data-objects. In cases of compound identity all Uid components would be returned. For example a query against *wellLogs* would return the *uidWell*, *uidWellbore*, and *uid* attributes.
`HeaderOnly`	Indicates that the return parameter of the query should only contain the identifying elements and the top-level non-growing parts of a growing data-object. For instance, the WITSML log header contains a recurring *logCurveInfo* element and a non-recurring *logData* element.
`Aggregate`	Indicates that the growing values be aggregated based on the GroupingFunction parameter. Used in combination with the GroupingInterval value to request values that are calculated based on 'buckets'.
`Interpolate`	Indicates that the growing values be interpolated based on the GroupingInterval parameter. Used in combination with the GroupingInterval value to request values that are calculated based on 'buckets'.
`Current`	Indicates that only the latest values in each column of a systematically growing data-object be returned.
`Range`	Indicates a data portion of a growing data-object. For instance, for a wellLog it would indicate that only a specific depth range would be returned. For a process control historian, it would indicate that a range of time samples be included.

If an implementation does not support the *SubsetType* parameter, then the SERVER MUST return a *UnsupportedOptionFault* SOAP fault with the *OptionName* field set to `SubsetOptions/SubsetType`.

```
<UnsupportedOptionFault xmlns="http://www.prodml.org/api/210/genericDataAccess">
    <OptionName>SubsetOptions/SubsetType</OptionName>
</UnsupportedOptionFault>
```

MaxValues

The MaxValues parameter allows the client to specify the maximum number of data points to return.

If an implementation does not support the *MaxValues* parameter, then the SERVER MUST return a *UnsupportedOptionFault* SOAP fault with the *OptionName* field set to `SubsetOptions/MaxValues`.

```
<UnsupportedOptionFault xmlns="http://www.prodml.org/api/210/genericDataAccess">
    <OptionName>SubsetOptions/MaxValues</OptionName>
</UnsupportedOptionFault>
```

GroupingFunction
An enumeration of available or requested methods for combining or grouping values when requesting data from a 'growing' or time-series object in specified intervals, or as an aggregate.

GroupingFunction	Description
Average	
Minimum	
Maximum	
Sum	
Raw	
StdDeviation	
Any	If there are multiple values within the interval, then any one value will be selected.

If an implementation does not support the *GroupingFunction* parameter, then the SERVER MUST return a *UnsupportedOptionFault* SOAP fault with the *OptionName* field set to `SubsetOptions/GroupingFunction`.

```
<UnsupportedOptionFault xmlns="http://www.prodml.org/api/210/genericDataAccess">
    <OptionName>SubsetOptions/GroupingFunction</OptionName>
</UnsupportedOptionFault>
```

GroupingInterval
Specifies the interval, or 'bucket' size used when requesting aggregated or interpolated data from a growing data-object.

```
<Options>
  <SubsetOptions>
      <SubsetType>Interpolate</SubsetType>
      <GroupingInterval uom="s">3</GroupingInterval>
  </SubsetOptions>
</Options>
```

If an implementation does not support the *GroupingInterval* parameter, then the SERVER MUST return a *UnsupportedOptionFault* SOAP fault with the *OptionName* field set to `SubsetOptions/GroupingInterval`.

```
<UnsupportedOptionFault xmlns="http://www.prodml.org/api/210/genericDataAccess">
    <OptionName>SubsetOptions/GroupingInterval</OptionName>
</UnsupportedOptionFault>
```

CustomOptions
An array of name-value pair parameters used to send generic or server-specific options for the request. No interoperability is guaranteed for these data and the server and client must agree on the meaning of the name/value pairs.

If an implementation does not support the *CustomOptions* parameter, then the SERVER MUST return a *UnsupportedOptionFault* SOAP fault with the *OptionName* field set to `SubsetOptions/CustomOptions`.

```
<UnsupportedOptionFault xmlns="http://www.prodml.org/api/210/genericDataAccess">
    <OptionName>CustomOptions</OptionName>
</UnsupportedOptionFault>
```

Response Message

The `GetDataResponse` response message is returned by the *GetData* operation of the GDA service.

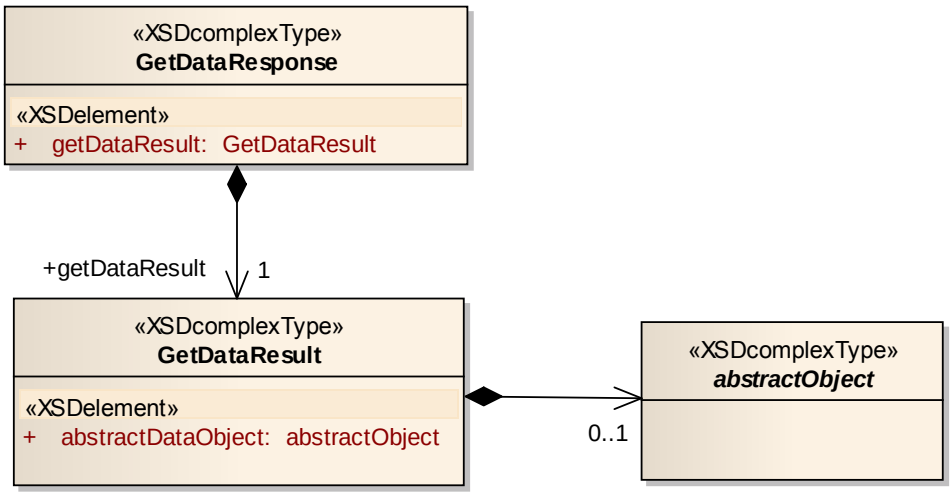

Figure 9 GetData Response Message

GetDataResult

On success, the *getDataResult* parameter carries the requested data-object(s).

abstractDataObject

The *abstractDataObject* parameter is an element substitution group for the returned member that contains the data-objects. In actual usage, the content will be content of the referenced member schema. Only one substitution group member can be returned to the client. Note however that for PRODML and WITSML data-objects, the member will actually be the plural root that encompasses the singular data-objects. Thus, while only one plural member may be returned, multiple data-objects may be returned.

For example, the following shows a response containing a WITSML well data-object.

```
<GetDataResponse xmlns="http://www.prodml.org/api/210/genericDataAccess">
  <getDataResult>
    <abstractDataObject xmlns:q1="http://www.witsml.org/schemas/131"
                        xsi:type="q1:obj_wells"
                        xmlns="http://www.energistics.org/schemas/abstract">
        <q1:well uid="26725">
            <q1:name>TITAN #1</q1:name>
            <q1:field>BLIZZARD</q1:field>
            <q1:county>MARTIN</q1:county>
            <q1:block>TARTARUS</q1:block>
            <q1:operator>BIG OIL</q1:operator>
            <q1:numAPI>42001100000001</q1:numAPI>
      </q1:well>
    </abstractDataObject>
  </getDataResult></GetDataResponse>
```

3.2.3 PutData

A service call which passes one or more data-objects to the server.

Request Message

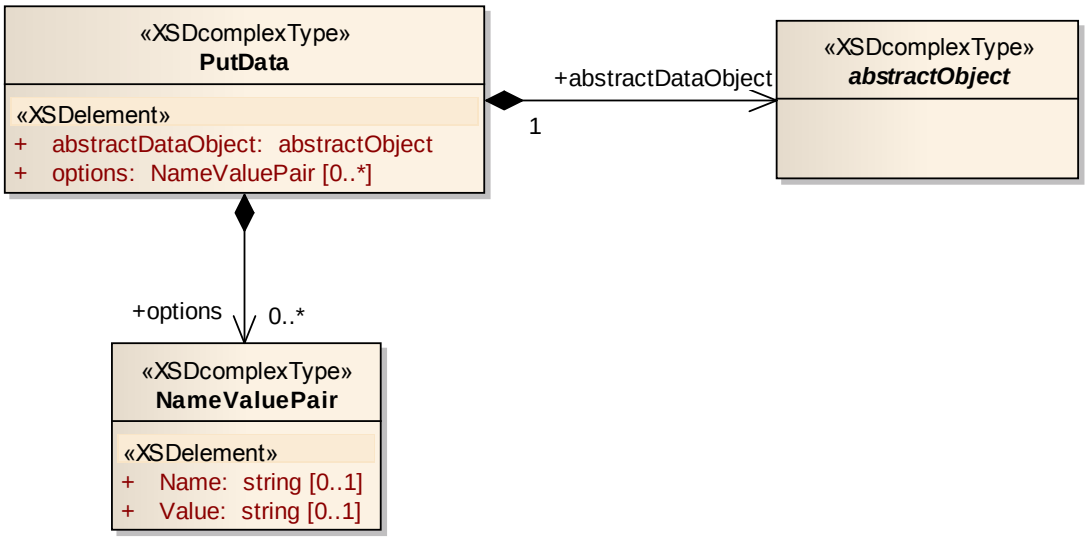

Figure 10 PutData Request Message

abstractDataObject
The element substitution group for supported data-objects. In actual usage, the the abstractDataObject content will be content of the referenced member schema. Only one substitution group member can be uploaded to the server (e.g., `productVolumes`, `wellTests`, `timeSeriesDatas`, etc.). Note however that for PRODML and WITSML data-objects, the member will actually be the plural root that encompasses the singular data-objects. Thus, while only one plural member may be uploaded, multiple data-objects may be uploaded.

NOTE: It is possible for a member schema to be constructed such that the member is the data-object. Future versions of this specification may allow multiple members if future versions of the WITSML and/or PRODML schemas make the singular data-object the member.

options
Options as an array of NameValuePair objects to be used by the server. This is an open collection of name-value pairs. There are no standard names for the options and any use is server specific.

Response Message

The *PutDataResponse* response message is returned by the PutData operation of the GDA service.

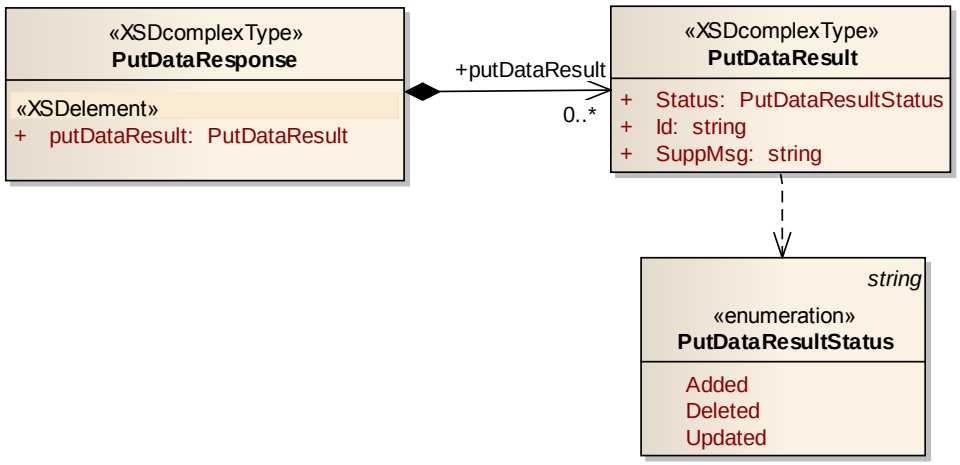

Figure 11 PutData Response Message

PutDataResult

The *PutDataResult* parameter contains information regarding the resultsof the operation.

For example, the following response may be returned for 2 wells that have been added to a back-end database by the PutData operation.

```
<PutDataResponse xmlns=" http://www.prodml.org/api/210/genericDataAccess">
    <putDataResult>
        <Status>Added</Status>
        <Id>prodml://gda.bigoil.com/well/W-1</Id>
    </putDataResult>
    <putDataResult>
        <Status>Added</Status>
        <Id>prodml://gda.bigoil.com/well/W-2</Id>
    </putDataResult>
</PutDataResponse>
```

3.2.4 DeleteData

A service call which deletes one or more data-objects from the server.

Request Message

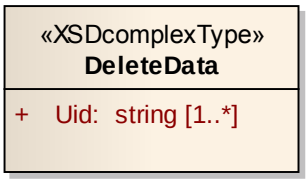

Figure 12 DeleteData Request Message

Uid

The *Uid* parameter is an identifier that identifies a data-object to be permanently deleted from the server. Multiple Uid fields may be specified.

The Uid value is considered to be a PRODML URI and is recommended to be formatted as such. Refer to the *PRODML Identifiers Specification* for more information.

For example, the following specifies a identifier for a well with a specified API number.

```
<Uid> prodml://gda.bigoil.com/well/W-1</Uid>
```

Response Message

The *DeleteDataResponse* response message is returned by the DeleteData operation of the GDA service.

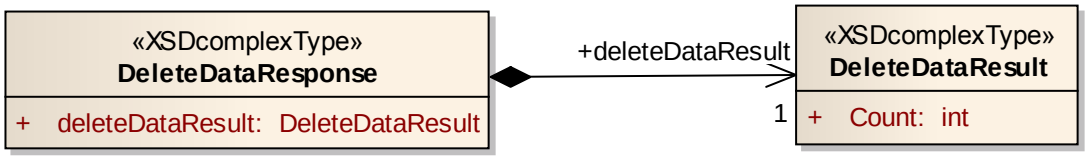

Figure 13 DeleteData Response Message

DeleteDataResult

The *DeleteDataResult* parameter contains information regarding the resultsof the operation.

For example, the following response may be returned for 2 wells that have been deleted by the DeleteData operation.

```
<DeleteDataResponse xmlns=" http://www.prodml.org/api/210/genericDataAccess">
    <deleteDataResult>
        <Count>2</Count>
    </putDataResult>
</DeleteDataResponse>
```

Notes

Cascading deletes are not supported. Only bottom level objects should be deleted and all child objects should be deleted before the parent is deleted. For example, a child wellbore should be deleted before the parent well is deleted. That is, a server is not required to support referential integrity.

4 Security

PRODML recognizes that security implementations will vary according to the policies, procedures and requirements of both the target organizations environment and the nature of the data and interactions (sensitivity, importance etc.) in a given PRODML exchange.

Security is generally composed of three main areas – User Authentication, User Authorization and Transport Security.

4.1 User Authentication

User Authentication is the ability of a PRODML server to accept a client's assertion of a 'users' identity. This user may be the identity of an individual person or of a 'generic ID' which could represent an application login.

The Basic Authentication scheme was originally defined by RFC 1945 (*Hypertext Transfer Protocol – HTTP/1.0*) although further information regarding security issues may be found in RFC 2616 (*Hypertext Transfer Protocol – HTTP/1.1*) and RFC 2617 (*HTTP Authentication: Basic and Digest Access Authentication*).

A SERVER MAY implement Basic Authentication to confirm an identity assertion. This method allows a user-id and password pair to be used to identify a user and allow the server to authenticate the user's credentials.

If this or a similar authentication method is implemented, the SERVER MUST NOT cache authentication credentials in any way as this would break the concept of stateless operation. Therefore, the CLIENT MUST send Authentication data with each PRODML message if this type of security scheme is implemented.

4.2 User Authorization

User Authorization is the ability of a PRODML server to restrict access to functions or data items based on a user's identity. PRODML does not specify in any way the model used by a server to implement authorization.

4.3 Transport Security

Secure Sockets Layer (SSL) for HTTP (HTTPS) can be used to encrypt the HTTP traffic. This ensures that the data cannot be intercepted and viewed by a third party, and also that that data cannot be changed whilst in transit without the recipient being aware that the data stream has been interfered with.

5 WSDL and XSD

The WSDL and the related XML schemas are discussed in this section. These contracts are functionally complete. They can be used to implement services and client applications. The only changes that apply are the endpoint address configurations in the WSDL file.

The WSDL defines the service, bindings, operations, and message formats. The XSD defines the data-objects used with this service.

5.1 Namespaces

The namespace for the GDA API is:

- http://www.prodml.org/api/210/genericDataAccess

Data-object and extensions will be defined in separate namespaces. Each data-object schema MUST refer to whatever schemas they make use of. Extension namespaces MUST NOT use Energistics namespaces and SHOULD use a proprietary namespace in order to insure that name clashes will not occur with anything defined by Energistics.

Appendix A. Requirements for Custom Data Objects

In this section are given example definitions of custom transportable data type that would extend the set of GDA transportable data-objects.

Although PRODML and WITSML will provide a set of useful data-object schemas, the need for new data-object schemas to be retrieved by different the GDA Service implementations will emerge. The following is an example of custom transportable schema designed to transport data-objects of type *myType:*

```
<xsd:schema
  xmlns:abs="http://www.energistics.org/schemas/abstract"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  targetNamespace="http://www.my.org/schemas/”
  elementFormDefault="qualified"
  attributeFormDefault="unqualified"
  version="1">

<xsd:import
    namespace="http://www.energistics.org/schemas/abstract"
schemaLocation="http://w3.energistics.org/schema/abstract_v1.0/xsd_schemas/sub_abstractSubstitutionGroup.xsd"/>

  <xsd:element
       name="myObject"
       type="myType"
       substitutionGroup="abs:abstractDataObject">
  </xsd:element>

  <xsd:complexType name="myType">
    <xsd:complexContent>
      <xsd:extension base="abs:abstractObject">
        <xsd:sequence>
           <!—More stuff -->
        </xsd:sequence>
        <xsd:attribute name=”version” type=”xsd:string” use=”required”/>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
</xsd:schema>
```

Now this element can be used for the abstractDataObject parameter in GetData and PutData.

Appendix B. Element Path Syntax

The following discussion refers to allowable syntax which is used in query parameters such as DateTimeCriterion and DepthCriterion to specify the element that is the target of the filter. Although the syntax is similar to XPath it is not a true XPath query in any real sense.

1. The context node for the purposes of this path is considered to be the singular root of the document in question. This root element itself MUST NOT appear in the path.
2. The only allowable syntax is composed of '/'and element names. Element names must be separated by a '/'. Relative path syntax (i.e. '//', '..', etc.) is not supported. The first element in the path MUST NOT begin with a '/'.

For example, the following paths are valid:

Data Object	Path	Description
`wellTest`	`testDate`	Return all well tests within the given data range.
`wbGeometry`	`wbGeometrySection/tvdBottom`	Return a list of wbGeometry elements, with only the sections within the specified range.
	`Maximum`	

The following paths are invalid: