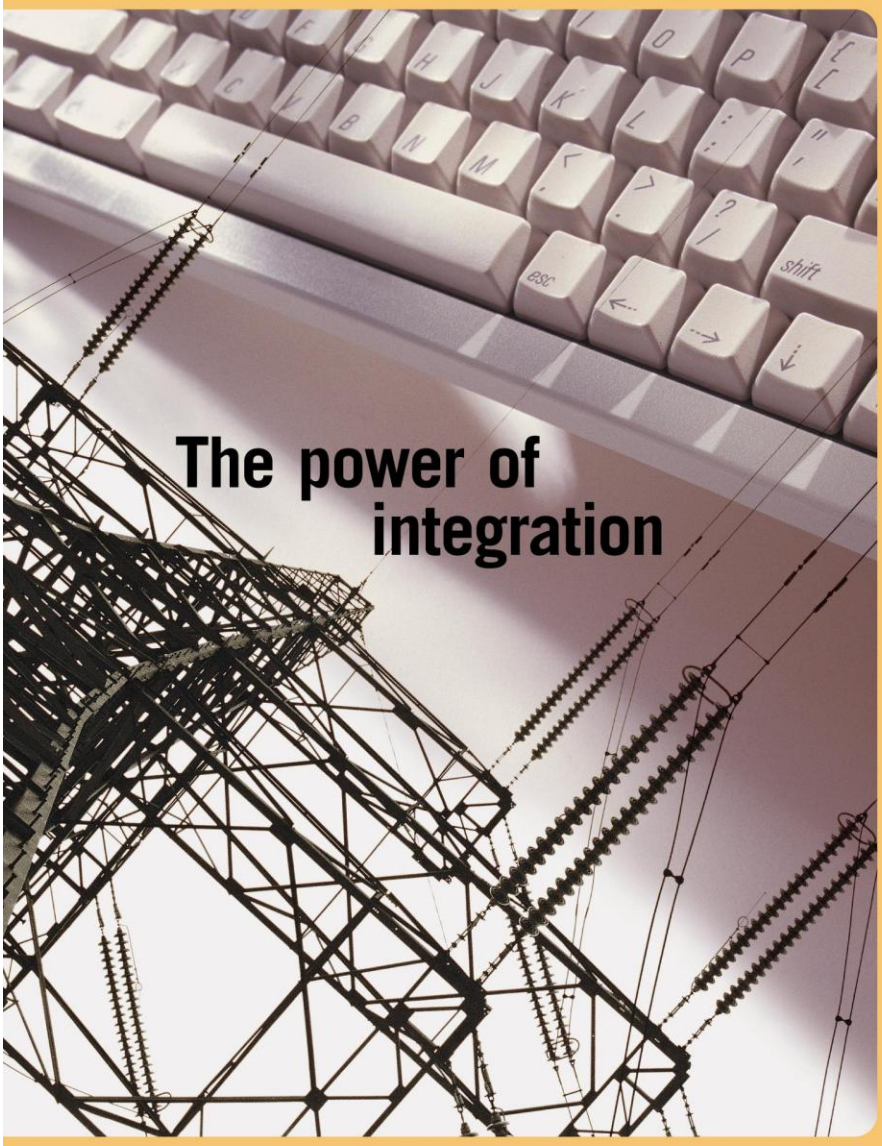
MISO DEMAND RESPONSE TOOL

Web Services Specification



May 29, 2012
Version: 1.1

UISOL
Utility Integration Solutions, Inc.

24 Benthill Ct
Lafayette, CA 94549
www.UISOL.com
Tel: 925-939-0449
Fax: 925-658-0023

Revisions

Date	Version	Description	Author(s)
3/8/10	1.0	Final Specification Posted	
5/29/12	1.1	Corrected Midwest ISO to MISO	Randi Terry

Contents

FIGURES

1 Introduction

This document describes the message structure for machine to machine interfaces for Market Participant applications that need to interact with the MISO's Demand Response Tool (DRT). The intended audience of this document is developers that will be integrating customer applications to DRT through the use of the interfaces described within this specification.

Where sections 1 and 2 of this document apply to all interfaces, sections 3 and beyond describe specific groupings of interfaces. The appendices provide XML Schemas, WSDLs and additional examples.

The interfaces and related interactions described by this document define the externally-visible (black box view) perspective of the services provided by this project. It is the intent of this specification and interface architecture to shield customers from the details of systems integration internal to DRT.

1.1 Purpose

The interfaces described by this document are intended to be used by Market Participants for machine to machine integration. This document is intended to provide all the details of the message structures and technical interoperability standards required for the machine to machine interface.

1.2 Scope

The scope of this document is to describe web services provided for integration by Market Participants from the perspective of external integration. This document has program level scope as related to web services that would be used by Market Participants for machine to machine interaction with Demand Response applications as detailed in an agreed list of interfaces to be managed by the project. The intent of this design is to leverage the integration layer (IL) to expose web services needed for external integration by Market Participants.

The following are specifically outside the scope of this document:

- Interactions with application User Interfaces (UI)
- Reports that might utilize the web-services framework
- Asynchronous web-services notifications or alerts
- Valid runtime data codes (zones, weather stations, organization names, etc)
- Business rules regarding read/write privileges for objects, organizations and users

1.3 Definitions, Acronyms, and Abbreviations

Term or Acronym	Definition
Alert	A message sent to a user to indicate a condition where action may be required, commonly communicated using e-mail
API	Application Programming Interface
AS	Ancillary Service market
BPM	Business Process Management, where Savvion is the BPM engine used within DRT for the management of potentially long running, multi-user workflows
CIM	Common Information Model
CSV	Comma separated value format
DR	Demand Response
DRT	Demand Response Tool
DRR	Demand Response Resource
Enrollment	The process by which a customer location is registered for a specific DR program
Event	An invocation of a DR program on a given day for a defined time period
HTTP	Hyper-Text Transfer Protocol, and IETF standard
HTTP/S	HTTP using secured sockets
IEC	International Electrotechnical Commission
IETF	Internet Engineering Task Force
ISO	International Organization for Standardization
LBA	Load Balancing Authority
Location	A place that can be metered and registered for participation in a DR program
Settlement	A process for generating the result of an event that provides the information required to initiate payment processing
SOAP	Simple object access protocol, based upon XML
W3C	Worldwide Web Consortium
WS	Web Services
WSDL	Web Services Definition Language
XML	Extensible Markup Language
XSD	XML Schema, used to define the structure of XML documents
XSL	XML Stylesheet Language, used to transform XML documents

1.4 References

Additional references to related standards are described in section 1.6

Artifact	Definition
XSDs	Specific message structure and element details.
Web Services User Guide	Use case based samples of web service requests.

1.5 Overview

This document focuses on the External Web Services (EWS) interface design and related interface definitions.

1.6 Program-level Standards

In general, the design described by this document will leverage web services and related security standards as defined by the World-Wide Web Consortium (W3C) and OASIS. Program-level standards include those related to security, as well as basic web service standards including:

- XML
- XML Schema
- XPath
- XSL
- SOAP
- Web Services
- WSDL

These are described in the companion security design document. W3C standards can be freely accessed from http://www.w3.org.

Another key program standard is the IEC Common Information Model (CIM), as defined by IEC 61970-301. This is used to define models used by the Market, which are exchanged using IEC 61970-501. It will also be leveraged by this design for the definition of messages used for interfaces. There is also a standard for message structures defined by IEC 61968-1. There standards can be purchased from the IEC web site at http://www.iec.ch. Materials related to IEC standards, including the CIM model it self can be freely obtained from the UCA International Users Group SharePoint at http://sharepoint.ucausers.group.org/CIM.

There are several key Internet Engineering Task Force references. These include:

- Internet Engineering Task Force RFC 2828: Internet Security Glossary
- Internet Engineering Task Force RFC 2119: Key words to indicate RFC requirement levels
- Internet Engineering Task Force RFC 2246: Transport Layer Security (TLS)
- Internet Engineering Task Force RFC 3275: XML Digital Signature and Processing

IETF documents can be freely obtained from http://www.ietf.org.

OASIS standards can be freely obtained from http://www.oasis-open.org.

The definition of timestamps is specified by ISO-8601, with the exception that timestamps of 24:00:00 are not used for compatibility reasons. This is partly a consequence of the XML Schema definition for 'dateTime', where hour 24 is not explicitly allowed. There are some implementations of timestamps within software products that do not correctly handle timestamps of 24:00:00.

ISO standards can be purchased online from a variety of sources including ANSI, at http://www.ansi.org. Descriptions of ISO-8601 can be freely obtained from other sources including Wikipedia.

1.7 Services Organization

The services described by this document are defined using a combination of Web Services Definition Language (WSDL) and XML Schema. The WSDLs are organized as follows:

- One or more WSDLs, defining operations related to synchronous request/reply web service messages

In both of the above cases, one or more XML Schemas (XSD) is used to define the structure of message payloads.

Throughout this document there are diagrams representing XSD structures. Understanding the diagrams will assist in implementing any services based on these XSDs. Below is a link which elaborates the details and structures of the XSD diagrams. http://www.diversitycampus.net/Projects/TDWG-SDD/Minutes/SchemaDocu/SchemaDesignElements.html

Example WSDL and XSD are provided in the appendices. It is anticipated that these would be key design artifacts for developers.

1.8 Common Message Structure

Unless otherwise specified, all messages use a common message envelope, where a predefined structure is used for requests and another structure is used for responses. This structure is based upon the IEC 61968-1 standard. Messages are constructed with several sections, including:

- Header: required for all messages, using a common structure for all service interfaces
- Request: optional, defining parameters needed to qualify request messages
- Reply: Used for response messages to indicate success, failure and error details
- Payload: optional, used to convey message information as a consequence of the 'verb' and 'noun' in the message Header. The payload structure provides options for payload compression.

1.8.1 Header Structure

Common to both the request and response messages is a header structure. The header has several required fields that must be populated, these include:

- Verb, to identify a specific action to be taken. There are an enumerated set of valid verbs, where commonly used values include:
  - ‘get’, ‘create’, ‘change’, ‘action’, ‘cancel’, ‘close’ and ‘reply’.
  - Implementations treats verbs ‘update’ and ‘updated’ as synonyms to ‘change’ and ‘changed’.
- Noun: to identify the subject of the action and/or the type of the payload (e.g. intervaldata, dailydata) if a payload is provided.
- Revision: To indicate the revision of the message definition.
- Source: identifying the source of the message which should be the ID of the Asset Owner (AO) or MISO (for reply messages).
- UserID: Optional and will be overwritten by the UserID associated with the certificate used for the data exchange.
- MessageID: If populated on a request, it will be returned on the reply.
- Comment: It is never used for any processing-related logic.

The following diagram describes the header structure used for request and response messages.

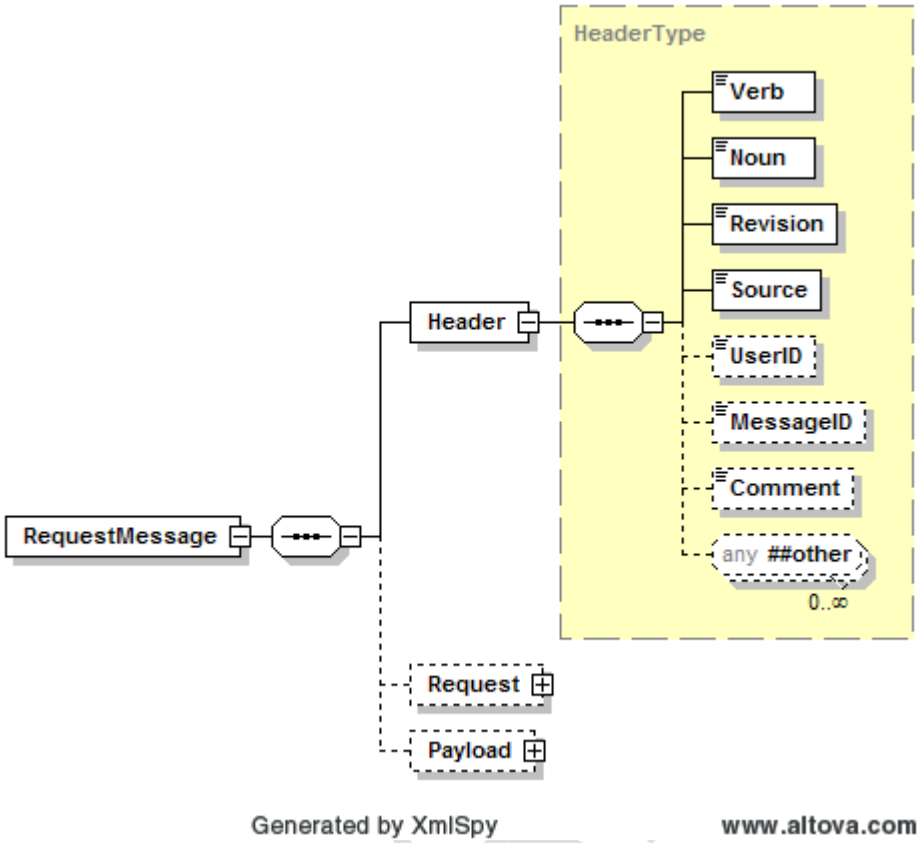


Figure 1 - Message Header Structure

There are several optional fields that may be populated. In the above diagram, the optional items are represented using dashed borders. If the MessageID is populated on a request, it will be returned on the reply. The Comment field is never used for any processing-related logic. The UserID may be used to indicate the person responsible for initiating a transaction, and will be logged as appropriate, but verification is the responsibility of the Demand Response Tool (DRT).
In order to identify the Market Participant in a uniform manner, the registered 'short name' of the Assset Owner should be supplied as the value of Header/Source. DRT will verify that this value is consistent with the AO as identified by the certificate.

1.8.2 Request Message Structure

The following diagram describes the structure of a request message that would be used in conjunction with a WSDL operation.

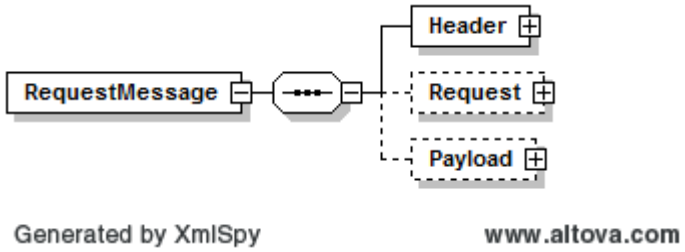


Figure 2 - Request Message Structure

The RequestMessage can also optionally contain a package with parameters relevant to the request, called Request. It is likely that different or variant Request packages may be defined to be used in conjunction with messages for a specific web service operation. In those cases, the corresponding WSDL and XSD would identify the optional parameters. The description of the interface (in subsequent sections of this document) would identify the usage of those parameters. The following is the RequestType used in the definition of a Request package that defines some common parameters used for requests. These parameters are most commonly used in conjunction with 'get' requests as qualifiers.

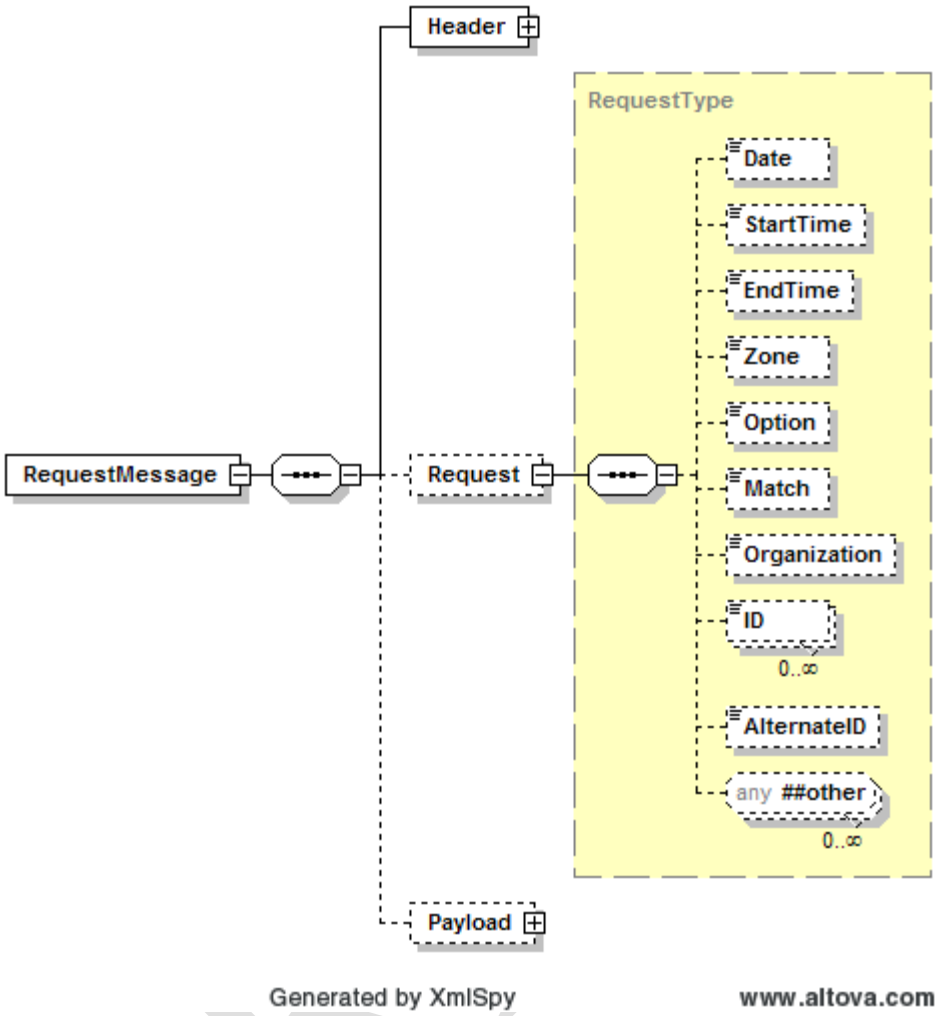


Figure 3 - Message Request Parameters

1.8.3 Payload Structure

There are some requests where a Payload must be provided, as would be the case for a message with a verb of 'create' or 'change'. Payloads are typically XML documents that conform to a defined XML schema. There may be cases where a large payload must be compressed, in the event that it would become very large and otherwise consume significant network bandwidth. In order to accommodate a variety of payload format options the following payload structure is used.

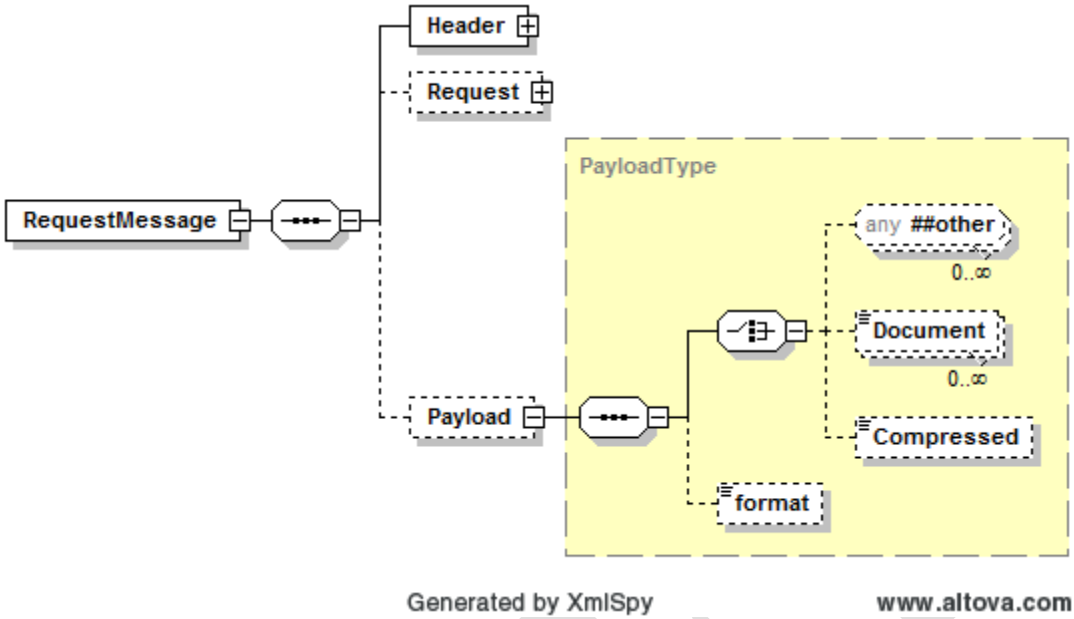


Figure 4 - Message Payload Container Structure

In the previous diagram, any type of XML document may be included, using the XML 'any' structure. While this provides options for loose-coupling, specific complex types defined by XML schemas (XSDs) can be used as well. The WSDL in the appendix provides an example of this case.

Payloads can also be supplied as XML encoded strings using the 'Document' tag, although this method is less preferred than used of the XML 'any'.

There are also some cases where a zipped, base64 encoded string is necessary, and would be passed using the 'Compressed' tag. The Gnu Zip compression will be used in order to provide compatibility with Java and Microsoft .Net implementations. A Java example is provided in Appendix C. Specific examples of the usage of payload compression would be where:

- An XML payload, conforming to a recognized XML schema exceeds a predefined size (e.g. 1MB). This would be very common for large Market Participant sets of meter data
- A payload has a non-XML format, such as PDF, Excel spreadsheet, CSV file or binary image
- A payload is XML, but has no XML schema and exceeds a predefined size, as would be the case of a dynamic query that would return an XML result set

When a payload is compressed and base64 encoded, it is stored within the Payload/Compressed message element as a string.

The format tag can be used to identify specific data formats, such as XML, RDF, PDF, DOC, CSV, etc. This is especially useful if the payload is compressed. The use of this tag is optional, and would typically only be used when the payload is stored using the Payload/Compressed message element.

1.8.4 Response Structure

The following diagram describes the structure of a response message that would be used in conjunction with a WSDL operation, as a response to the request message.

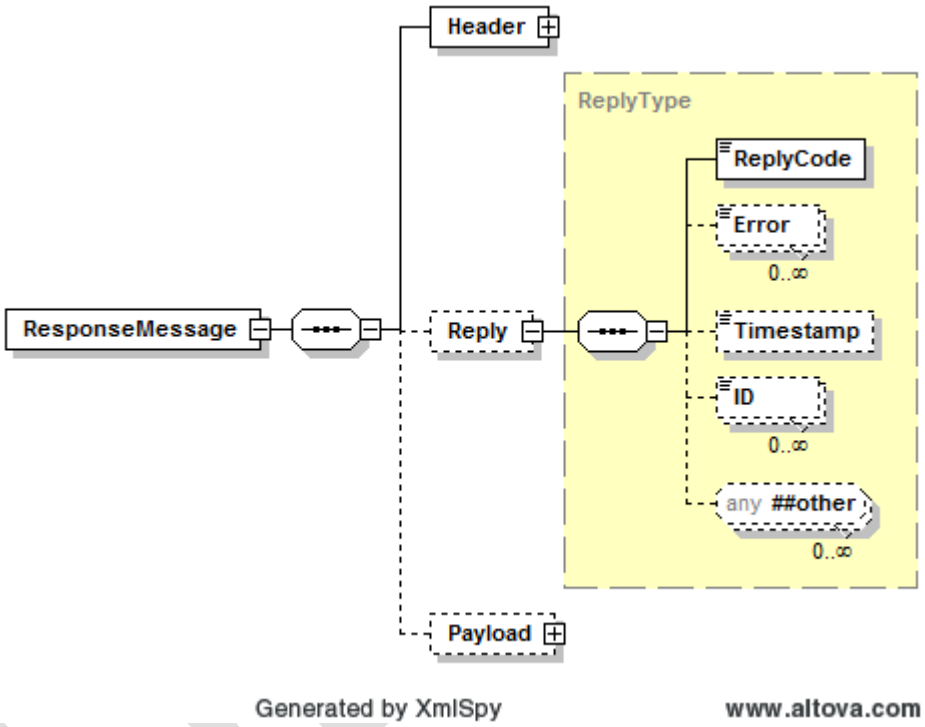


Figure 5 - Response Message Structure

The ReplyCode will be set to OK if the request was successful, otherwise it will be set to either ERROR or FATAL, and one or more Error elements will be provided to describe the error(s). A ReplyCode of FATAL indicates that an internal error has occurred. There may also be more specific error information provided within the payload, as in the case of Enrollment within an Enrollment container.

If the MessageID was set in the Header for the RequestMessage, the value will be returned in the Header of the ResponseMessage.

1.9 Modeling and Conventions

There are several conventions that are used for definitions, data items and information models.

1.9.1 Use of the IEC CIM

Where possible the IEC CIM should be leveraged. DRT and related systems use the CIM as a key standard. Examples of leveraging the CIM for external web services include:

- The use of data structures defined by the IEC CIM where appropriate in payload definitions
- CIM naming conventions are used wherever possible, e.g. ClassName, propertyName
- The properties 'startTime' and 'endTime' are typically used to identify time intervals, as they are also used within many CIM classes. Instead of using combinations of start date, start hour and potentially an interval number (e.g. to represent 15 minute intervals), absolute times should be specified.

The following class diagram describes the structure of CIM Curves, IrregularIntervalSchedules and RegularIntervalSchedules. These classes are CIM building blocks, where there are many other classes in the CIM that inherit from these three basic types of curves and schedules (e.g. PriceCurve).

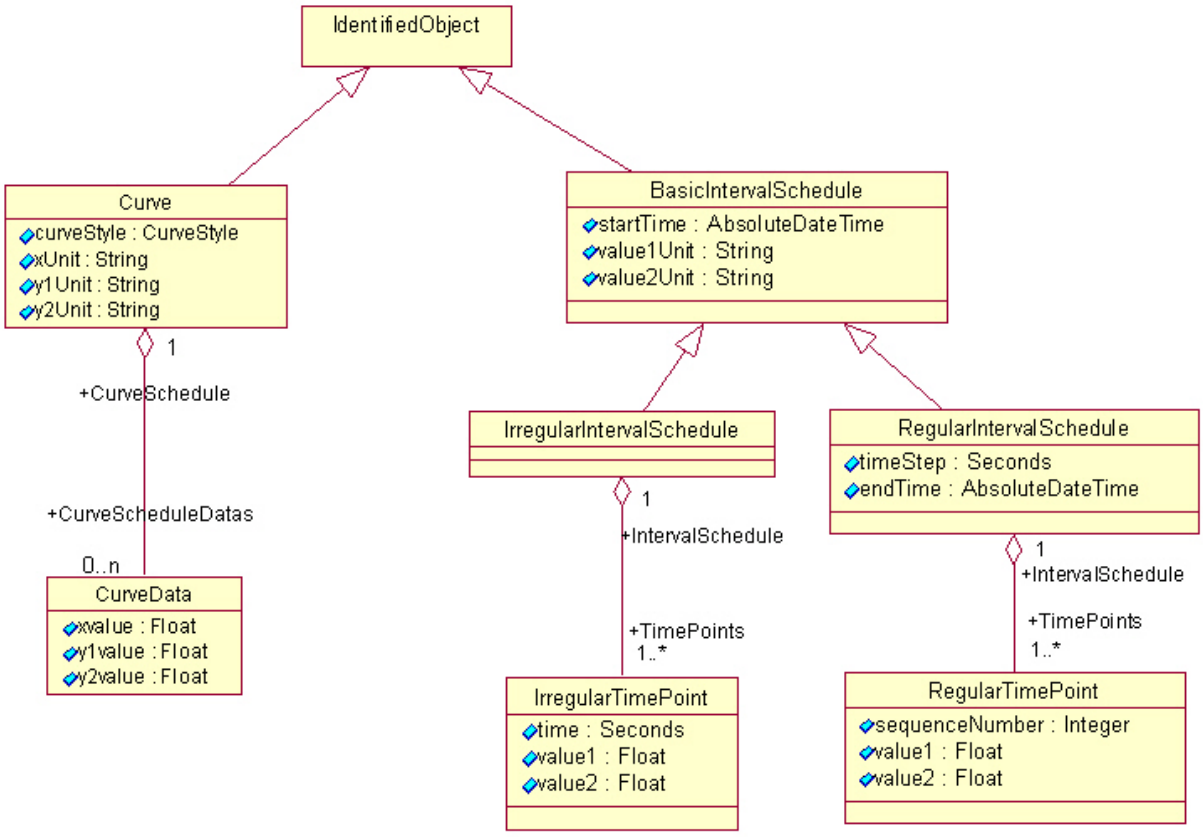


Figure 6 – Example class diagram

1.9.2 Representation of Time

The ISO 8601 standard is used to define the representations of time values that are conveyed through interfaces. This avoids issues related to time zones and daylight savings time changes.

- For timestamps in messages published by DRT would use Eastern Standard Time, using the following format: 2007-03-27T14:00:00-05:00.
- Timestamps in messages sent to DRT by Market Participants could use any ISO 8601 compliant timestamp.

1.9.3 Other Conventions

The following are other conventions that must be followed by this specification:

- Within XML definitions, tags should be namespace qualified. For example, a tag of <tag> shall be prefixed by a specific namespace reference, e.g. <ns:tag>. This will help to eliminate ambiguity. *(Note that many examples in this document are not namespace qualified for brevity and to aid legibility)*
- Units for power quantities are in kilowatts (KW).
- Units for capacity quantities are in kilowatts (KW).
- Units for energy quantities are in kilowatt-hours (KWh).
- Trading dates are specified using YYYY-MM-DD, which indicates the operating day

1.10 Versioning

It is important to recognize that new versions of interfaces may be provided over time, largely as a consequence of:

- Staging of initial implementation
- New requirements
- Upgrades to vendor products

Wherever possible, interfaces will be evolved through augmentation, where a newer version of an interface is compatible with a previous version of an interface. However, this will not always be possible. New versions of interfaces will be manifested by:

- Changes to WSDLs
- Changes to XML Schemas
- Changes to software implementations

2 DRR Objects

This section describes the data objects used in web-services payloads by DRT. Use of the fields is typically dependent on the type of request. This is summarized on the right (1=provided/mandatory, 0=optional, 0|1..n=set, blank=ignored), and defined further in later sections of this document. All fields are subject to business rules regarding organization (and user) read/write privileges for each object.

2.1 Daily Data

A Daily Data object is used to describe an entire operating day of measurement data for a given location of a specific measurement type. Currently this is used for 24 intervals of measurement data, typically as part of settlements.

See the Web Services User Guide for usage examples.

2.1.1 Daily Data Fields

The following is a description of all the fields in a Daily Data object.

Field	Description	Example		get	create
Summary info				1	1
Location id	Unique id of location	12837		1	1
Enrollment id	Unique id of Enrollment	3526		0	0
LBA Name	LBA short name	ALTE		1	0
Unique ID	Unique ID	28329		1	0
Measurements				1	1
Datestamp	Operating Date	2010-01-19T00:00:00.000-05:00		1	1
Type	Enumerated type of submission	DailyLoad		1	1
UOM	Enumerated type of data	KW		1	1
Intervals	Number of intervals provided (24)	19		1	1
Hour Value measurement/Hour	Hour number for of the measurement	Always 1 thru 24 for DRT		1	1
Hour Value measurement/values	Measurement value for the corresponding hour	273.2 Awalys 24 Values for DRT		1	1
Values	Comma separated string of #interval values	24 CSV if not provided by Hours Value element		1	1

Field	Description	Example		get	create
		273.2,284.5,,,			
Latest	Latest submission flag	True		1	
Submitted by	User name of submitter	Mday		1	
Submitted date	Date-time of submission	2010-02-16T11:15:22:329-05:00		1	
Submitted error	Error detected	No records found			

2.1.2 Daily Data Diagram

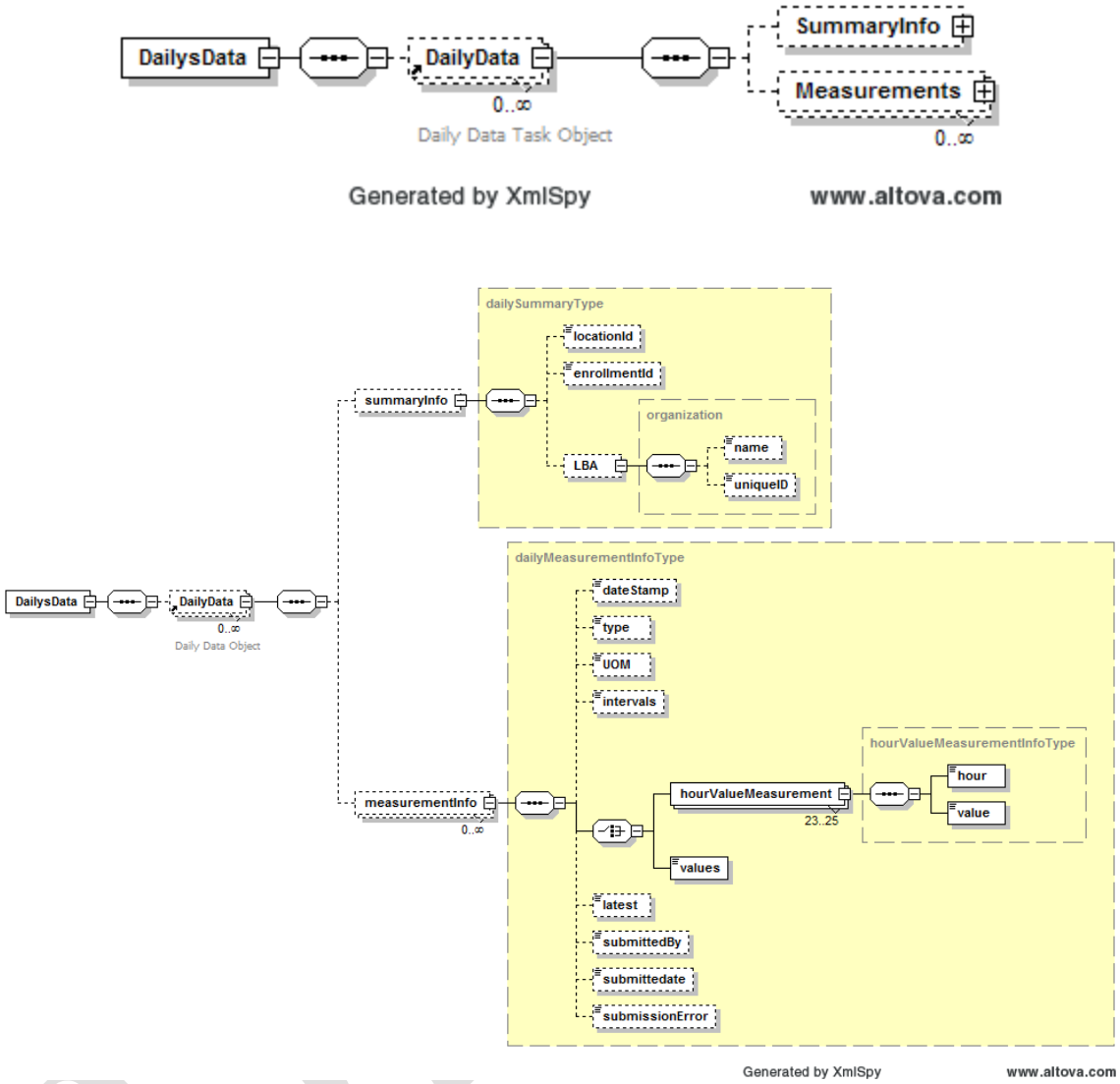


Figure 7: Daily Data

2.1.3 Daily Data Example

```
<?xml version="1.0" encoding="UTF-8"?>
<!--Sample XML file generated by XMLSpy v2008 rel. 2 sp2 (http://www.altova.com)-->
<DailysData xmlns="http://www.uisol.com/schema/drbiznet/2009/model"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.uisol.com/schema/drbiznet/2009/model LRSDailyData.xsd">
	<DailyData>
		<summaryInfo>
			<locationId>201</locationId>
			<enrollmentId>200</enrollmentId>
			<LBA>
				<name>ALTE</name>
				<uniqueID>EDC13223456</uniqueID>
			</LBA>
		</summaryInfo>
		<measurementInfo>
```

```
                    <dateStamp>2009-10-18T00:00:00.000+05:00</dateStamp>
                    <type>HourlyLoad</type>
                    <UOM>KW</UOM>
                    <intervals>24</intervals>

        <values>12.3,13.5,14.6,15.7,16.7,17.8,18.9,19.0,20.1,21.2,22.3,23.4,12.3,13.5,14.6,15.7,16.7,17.8,18.9,19.0,2
0.1,21.2,22.3,23.4</values>
                    <latest>true</latest>
                    <submittedBy>bsmith</submittedBy>
                    <submittedate>2009-10-18T13:30:50.000+05:00</submittedate>
                </measurementInfo>
                <measurementInfo>
                    <dateStamp>2009-10-18T00:00:00.000+05:00</dateStamp>
                    <type>HourlyGen</type>
                    <UOM>KW</UOM>
                    <intervals>24</intervals>

        <values>12.3,13.5,14.6,15.7,16.7,17.8,18.9,19.0,20.1,21.2,22.3,23.4,12.3,13.5,14.6,15.7,16.7,17.8,18.9,19.0,2
0.1,21.2,22.3,23.4</values>
                    <latest>true</latest>
                    <submittedBy>manderson</submittedBy>
                    <submittedate>2009-10-18T13:30:50.000+05:00</submittedate>
                </measurementInfo>
                <measurementInfo>
                    <dateStamp>2009-10-18T00:00:00.000+05:00</dateStamp>
                    <type>HourlyLoad</type>
                    <UOM>KW</UOM>
                    <intervals>24</intervals>

        <values>12.3,13.5,14.6,15.7,16.7,17.8,18.9,19.0,20.1,21.2,22.3,23.4,12.3,13.5,14.6,15.7,16.7,17.8,18.9,19.0,2
0.1,21.2,22.3,23.4</values>
                    <latest>true</latest>
                    <submittedBy>bsmith</submittedBy>
                    <submittedate>2009-10-18T13:30:50.000+05:00</submittedate>
                </measurementInfo>
                <measurementInfo>
                    <dateStamp>2009-10-18T00:00:00.000+05:00</dateStamp>
                    <type>HourlyGen</type>
                    <UOM>KW</UOM>
                    <intervals>24</intervals>

        <values>12.3,13.5,14.6,15.7,16.7,17.8,18.9,19.0,20.1,21.2,22.3,23.4,12.3,13.5,14.6,15.7,16.7,17.8,18.9,19.0,2
0.1,21.2,22.3,23.4</values>
                    <latest>true</latest>
                    <submittedBy>manderson</submittedBy>
                    <submittedate>2009-10-18T13:30:50.000+05:00</submittedate>
                </measurementInfo>
        </DailyData>
</DailysData>
```

2.2 Interval Data

An Interval Data object is used to describe a random interval of measurement data for a given location of a specific measurement type. Currently this is used for event compliance submissions.

See the Web Services User Guide for usage examples.

2.2.1 Interval Data Fields

The following is a description of all the fields in an Interval Data object.

Field	Description	Example		get	create
Summary info				1	1
Location id	Unique id of location	12837		1	1
Enrollment id	Unique id of Enrollment	3526		0	0
LBA name	LBA short name	ALTE		1	0
Unique ID	Unique ID	28329		1	0
Measurements				1..n	1..n
Timestamp	Operating time	2010-02-16T11:15:00:000-05:00		1	1
Type	Enumerated type of submission	MinuteLoad		1	1
UOM	Enumerated type of data	KW		1	1
Value	Value for interval	273.4		1	1
Latest	Latest submission flag (1= latest, otherwise 0)	true		1	
Submitted by	User name of submitter	mday		1	
Submitted date	Date-time of submission	2010-02-16T11:15:22:329-05:00		1	
Submitted error	Error detected	No records found			

2.2.2 Interval Data Diagram

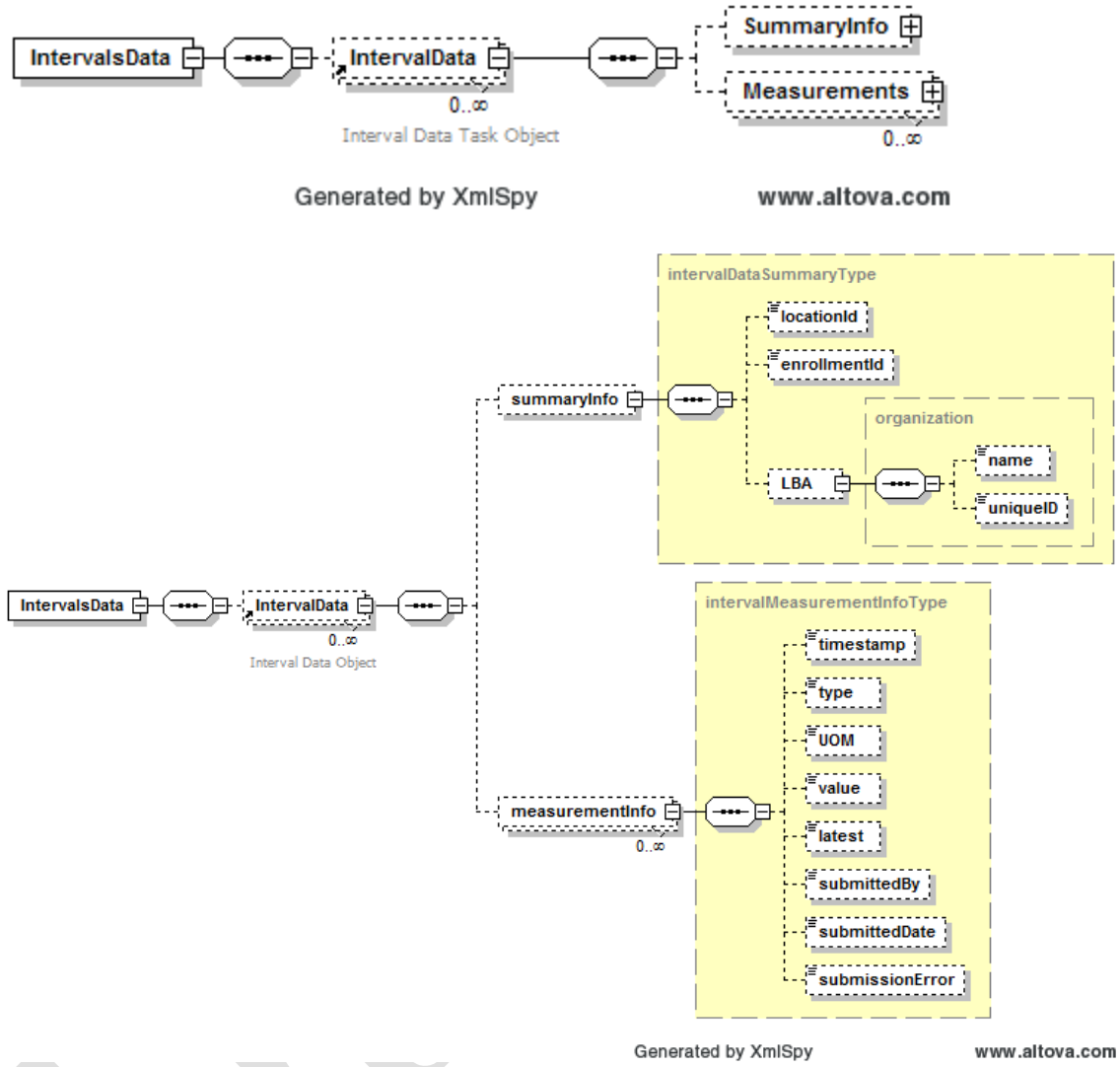


Figure 8: Interval Data

2.2.3 Interval Data Example

```
<?xml version="1.0" encoding="UTF-8"?>
<!--Sample XML file generated by XMLSpy v2009 rel. 2 sp2 (http://www.altova.com)-->
<IntervalsData xmlns="http://www.uisol.com/schema/drbiznet/2009/model"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.uisol.com/schema/drbiznet/2009/model LRSIntervalData.xsd">
        <IntervalData>
                <summaryInfo>
                        <locationId>201</locationId>
                        <enrollmentId>2002</enrollmentId>
                        <LBA>
                                <name>ALTE</name>
                                <uniqueID>13223456</uniqueID>
                        </LBA>
                </summaryInfo>
```

```
        <measurementInfo>
                <timestamp>2009-10-10T12:00:00.000+05:00</timestamp>
                <type>5MinuteLoad</type>
                <UOM>KW</UOM>
                <value>12.3</value>
                <latest>true</latest>
                <submittedBy>manderson</submittedBy>
                <submittedDate>2009-10-18T13:30:50.000+05:00</submittedDate>
        </measurementInfo>
        <measurementInfo>
                <timestamp>>2009-10-10T12:05:00.000+05:00</timestamp>
                <type>5MinuteLoad</type>
                <UOM>KW</UOM>
                <value>12.6</value>
                <latest>true</latest>
                <submittedBy>manderson</submittedBy>
                <submittedDate>2009-10-18T13:30:50.000+05:00</submittedDate>
        </measurementInfo>
        <measurementInfo>
                <timestamp>>2009-10-10T12:10:00.000+05:00</timestamp>
                <type>5MinuteLoad</type>
                <UOM>KW</UOM>
                <value>12.7</value>
                <latest>true</latest>
                <submittedBy>manderson</submittedBy>
                <submittedDate>2009-10-18T13:30:50.000+05:00</submittedDate>
        </measurementInfo>
        <measurementInfo>
                <timestamp>>2009-10-10T12:15:00.000+05:00</timestamp>
                <type>5MinuteLoad</type>
                <UOM>KW</UOM>
                <value>12.9</value>
                <latest>true</latest>
                <submittedBy>manderson</submittedBy>
                <submittedDate>2009-10-18T13:30:50.000+05:00</submittedDate>
        </measurementInfo>
        <measurementInfo>
                <timestamp>>2009-10-10T12:20:00.000+05:00</timestamp>
                <type>5MinuteLoad</type>
                <UOM>KW</UOM>
                <value>13.2</value>
                <latest>true</latest>
                <submittedBy>manderson</submittedBy>
                <submittedDate>2009-10-18T13:30:50.000+05:00</submittedDate>
        </measurementInfo>
</IntervalData>
<IntervalData>
        <summaryInfo>
                <locationId>203</locationId>
                <enrollmentId>2004</enrollmentId>
                <LBA>
                        <name>ALTE</name>
                        <uniqueID>13223456</uniqueID>
                </LBA>
        </summaryInfo>
        <measurementInfo>
                <timestamp>>>2009-10-10T12:00:00.000+05:00</timestamp>
                <type>5MinuteLoad</type>
                <UOM>KW</UOM>
                <value>13.1</value>
                <latest>true</latest>
                <submittedBy>bsmith</submittedBy>
                <submittedDate>2009-10-18T13:30:50.000+05:00</submittedDate>
        </measurementInfo>
        <measurementInfo>
                <timestamp>>>2009-10-10T12:05:00.000+05:00</timestamp>
                <type>5MinuteLoad</type>
                <UOM>KW</UOM>
                <value>12.3</value>
                <latest>true</latest>
```

```
                <submittedBy>manderson</submittedBy>
                <submittedDate>2009-10-18T13:30:50.000+05:00</submittedDate>
            </measurementInfo>
            <measurementInfo>
                <timestamp>>2009-10-10T12:10:00.000+05:00</timestamp>
                <type>5MinuteLoad</type>
                <UOM>KW</UOM>
                <value>12.9</value>
                <latest>true</latest>
                <submittedBy>manderson</submittedBy>
                <submittedDate>2009-10-18T13:30:50.000+05:00</submittedDate>
            </measurementInfo>
            <measurementInfo>
                <timestamp>>2009-10-10T12:15:00.000+05:00</timestamp>
                <type>5MinuteLoad</type>
                <UOM>KW</UOM>
                <value>13.2</value>
                <latest>true</latest>
                <submittedBy>manderson</submittedBy>
                <submittedDate>2009-10-18T13:30:50.000+05:00</submittedDate>
            </measurementInfo>
            <measurementInfo>
                <timestamp>>2009-10-10T12:20:00.000+05:00</timestamp>
                <type>5MinuteLoad</type>
                <UOM>KW</UOM>
                <value>13.3</value>
                <latest>true</latest>
                <submittedBy>manderson</submittedBy>
                <submittedDate>2009-10-18T13:30:50.000+05:00</submittedDate>
            </measurementInfo>
        </IntervalData>
</IntervalsData>
```

3 DRR Information Service

The purpose of the LRSInformation Service is to support interfaces required for retrieving DRR related information. These are read-only interfaces, as opposed to transactional interfaces.

3.1 Interfaces Provided

Specific interfaces using specific combinations of verbs and nouns (i.e. payload types) are defined to permit a Market participant to programmatically access DRT information. The verb to be used for requests would in all cases be 'get'. The noun would identify the type of information being requested. Each request could use a message 'Request' package to specify one or more parameters that would qualify the request. The processing sequence is shown in the following sequence diagram.

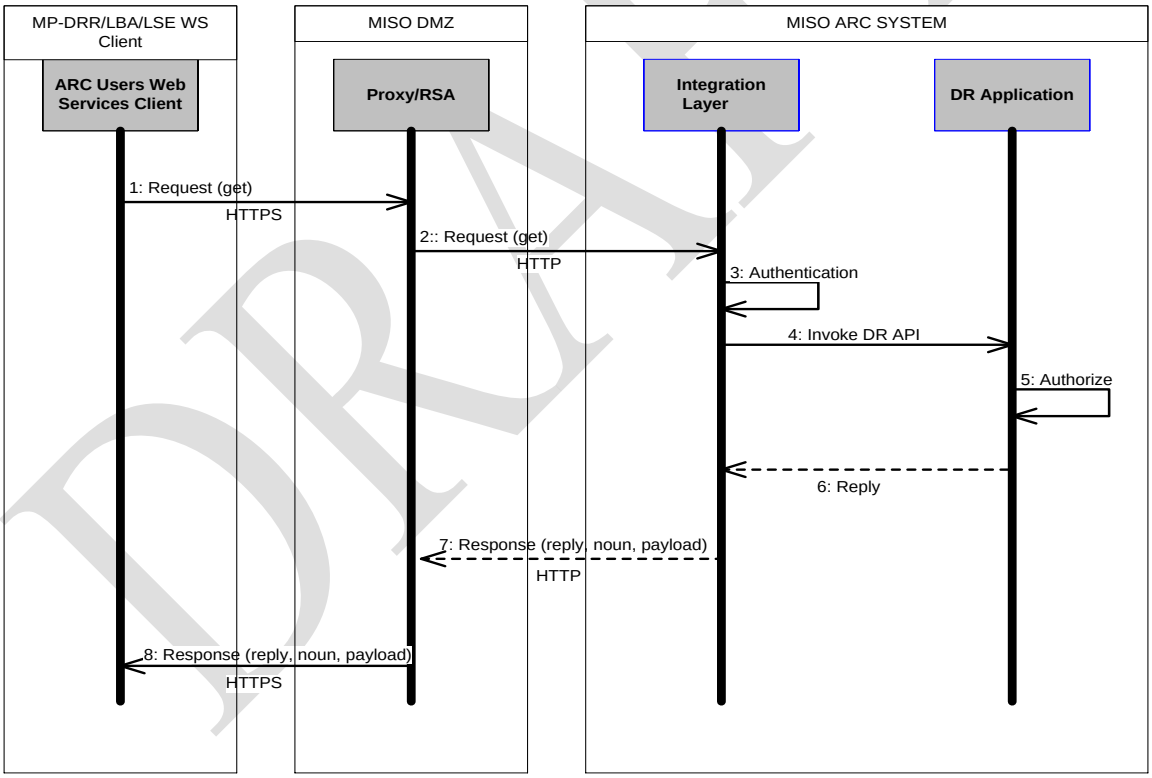


Figure 9 - DRT Information Request Sequence Diagram

3.2 Interfaces Required

The messages for DRT information requests would use the following message fields:

Message Element	Value
Header/Verb	Get
Header/Noun	*Name of payload type*
Header/Source	*DR participant ID*
Header/UserID	*Optional: ID of user*
Header /MessageID	*Optional: TransactionID*
Request/?	*Optional: Other request parameters may be specified as needed*
Payload	*Message payload data with type defined by Noun*

The response message would contain the following fields:

Message Element	Value
Header/Verb	Reply
Header/Noun	*Defined payload type name*
Header/Source	DRT
Header /MessageID	Echo back MessageID if MessageID is provided in request other wise The TransactionID will be return
Reply/ReplyCode	*Reply code, success=OK, error=ERROR or FATAL*
Reply/Error	*May be any number of error messages*
Payload	*Defined payload type*

In the cases of payloads that would otherwise exceed 1 megabyte, the payloads would be zipped, base64 encoded and stored within the 'Payload/Compressed' tag.

3.3 Message Specifications

DRT Information Payload definitions are dataset specific and are described in this section.

3.3.1 Measurement Data

Measurement Data related information use cases include -

Description	Verb	Noun	Input	Output
Gets submitted daily data for a location and date range	get	dailydata	Request/ID= location id Request/StartTime= start date Request/EndTime= end date	Payload= set of daily data objects
Gets submitted daily data for a Enrollment and date range	get	dailydata	Request/AlternateID=Enrollment id Request/StartTime= starttime Request/EndTime= endtime	Payload= set of daily data objects
Gets submitted interval data for a location and date range	get	intervaldata	Request/ID= location id Request/StartTime= start date Request/EndTime= end date	Payload= set of interval data objects
Gets submitted interval data for a Enrollment and date range	get	intervaldata	Request/AlternateID=Enrollment id Request/StartTime= starttime Request/EndTime= endtime	Payload= set of interval data objects

3.3.1.1 'get' dailydata or intervaldata

The following parameters are specified in the RequestMessage:

Message Element	Value
Header/Verb	get
Header/Noun	dailydata or intervaldata
Header/Source	*Asset Owner ID*

Header/UserID	*Optional: ID of user*
Header /MessageID	*Optional: TransactionID*
Request/ID	*Optional: Location id*
Request/Alternate ID	*Optional: Enrollment id*
Request/Start time	*Start date-time of reads*
Request/End time	*End date-time of reads*

The corresponding response messages would use the following message fields:

Message Element	**Value**
Header/Verb	reply
Header/Noun	dailydata or intervaldata
Header/Source	DRT
Header /MessageID	Echo back MessageID if MessageID is provided in request other wise The TransactionID will be return
Reply/ReplyCode	*Reply code, success=OK, error=ERROR or FATAL*
Reply/Error	*Error message, if error encountered*
Payload	*DailyData(s) or IntervalData(s)*

The payload objects will contain all available fields, with the exception that the Enrollment Id will only be populated if it was part of the search criteria.

4 DRR Transaction Service

The purpose of the DRT Transaction Service is to support required interfaces. This section describes the use of web services by Market Participants that involve the creation, submission, change, and cancellation of Enrollments, Location, Settlement, etc.

4.1 Interfaces Provided

Specific interfaces using specific combinations of verbs and nouns (i.e. payload types) are defined to permit a Market Participant to programmatically interact with DRT Transactions. The verb to be used for requests would be one of {create, change, action, cancel}. The noun would identify the type of transaction being requested. Each request could use a message 'Request' package to specify one or more parameters that would qualify the request.
The processing sequence is shown in the following sequence diagram.

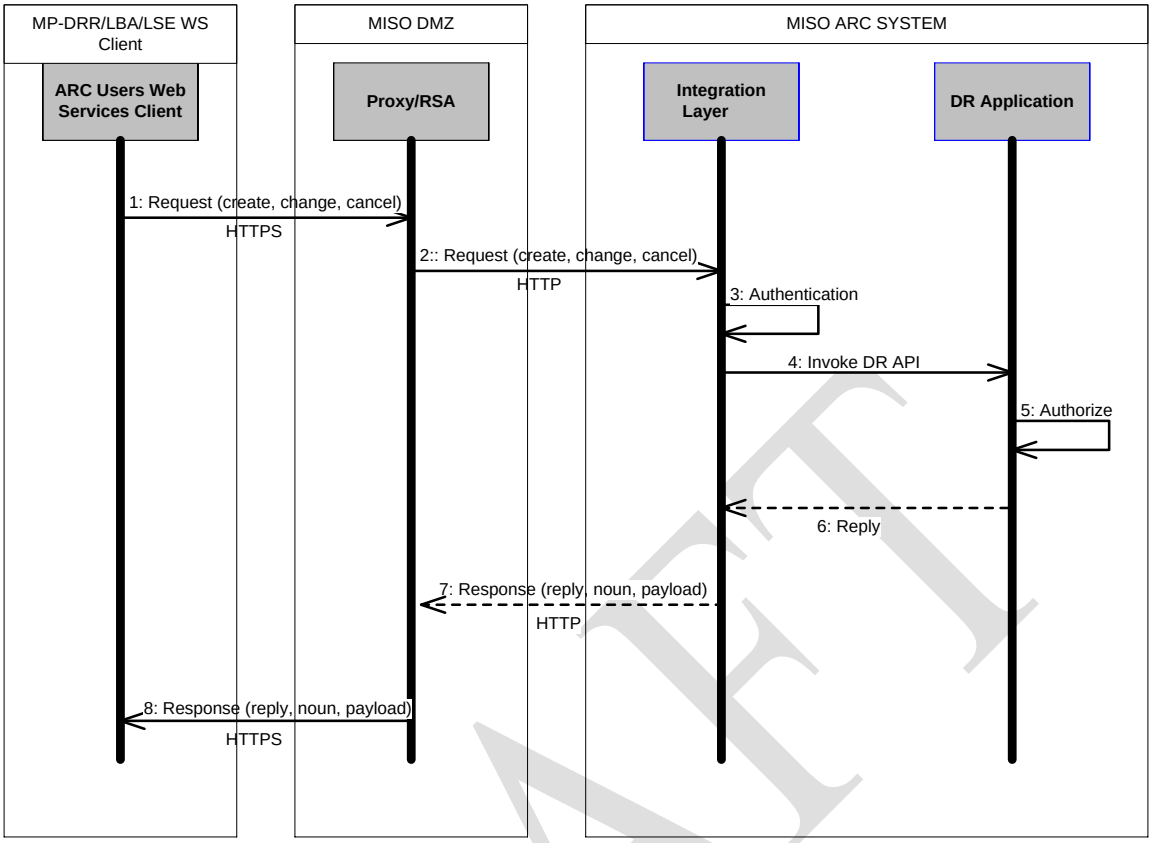


Figure 10 - DRT Transaction Request Sequence Diagram

4.2 Interfaces Required

The messages for DRT information requests would use the following message fields for creation or update (change) of a record:

Message Element	Value
Header/Verb	create/change
Header/Noun	*Name of payload type*
Header/Source	*Asset Owner ID*
Header/UserID	*Optional: ID of user*
Header /MessageID	*Optional: TransactionID*
Request/?	*Optional: Other request parameters may be specified as needed*
Payload	*Message payload data with type defined by Noun*

The response message would contain the following fields:

Message Element	Value
Header/Verb	reply
Header/Noun	*Name of payload type*
Header/Source	DRT
Header /MessageID	Echo back MessageID if MessageID is provided in request other wise The TransactionID will be return
Reply/ReplyCode	*success=OK, pending=PENDING, error=ERROR or FATAL*
Reply/Error	*May be any number of error message if the ReplyCode is ERROR*
Reply/Timestamp	*The time the submission was received by* DRT
Reply/ID	*Process id (if PENDING)*
Payload	*Message payload data with type defined by Noun.*

In the cases of payloads that would otherwise exceed 1 megabyte, the payloads would be zipped, base64 encoded and stored within the 'Payload/Compressed' tag.

The messages for DRT information requests would use the following message fields for cancellation (delete) of a record:

Message Element	Value
Header/Verb	cancel
Header/Noun	*Name of payload type*
Header/Source	*Asset Owner ID*
Header/UserID	*Optional: ID of user*
Header /MessageID	*Optional: TransactionID*
Request/ID	*Required: ID of the record to be deleted*

The response message would contain the following fields:

Message Element	Value
Header/Verb	reply
Header/Noun	*Name of payload type*
Header/Source	DRT
Header /MessageID	Echo back MessageID if MessageID is provided in request other wise The TransactionID will be return
Reply/ReplyCode	*success=OK, pending=PENDING, error=ERROR or FATAL*
Reply/Error	*May be any number of error message if the ReplyCode is ERROR*
Reply/ID	*Process id (if PENDING)*
Reply/Timestamp	*The time the submission was received by* DRT

4.3 Message Specifications

DRT Transaction verb, noun and payload definitions are dataset specific and are described in this section.

4.3.1 Measurement Data

Measurement Data related transactional use cases include –

Description	Verb	Noun	Input	Output
Upload daily measurement data	create	dailydata	Payload=Daily data	Reply/ID= process id
Upload interval measurement data	create	intervaldata	Payload=Interval data	Reply/ID= process id

4.3.1.1 'create' Daily Data

The create DailyData interface provides the means for a Market participant to upload (i.e., 'submit') DailyData Information, typically for settlement. The following parameters are specified in the RequestMessage:

Message Element	Value
Header/Verb	create
Header/Noun	dailydata
Header/Source	*Asset Owner ID*
Header/UserID	*Optional: ID of user*
Header /MessageID	*Optional: TransactionID*
Payload	*DailyData*

The response message would contain the following fields:

Message Element	Value
Header/Verb	reply
Header/Noun	dailydata
Header/Source	DRT
Header /MessageID	Echo back MessageID if MessageID is provided in request otherwise the TransactionID will be return
Reply/ReplyCode	*success=OK, pending=PENDING, error=ERROR or FATAL*
Reply/Error	*Error message, if error encountered*
Reply/ID	*Process id (if PENDING)*

4.3.1.2 'create' Interval Data

The create IntervalData interface provides the means for a DR participant to create (i.e., 'submit') IntervalData Information, typically for compliance. The following parameters are specified in the RequestMessage:

Message Element	Value
Header/Verb	change
Header/Noun	intervaldata
Header/Source	*Asset Owner ID*
Header/UserID	*Optional: ID of user*

Header /MessageID	*Optional: TransactionID*
Payload	*IntervalData*

The response message would contain the following fields:

Message Element	Value
Header/Verb	reply
Header/Noun	intervaldata
Header/Source	DRT
Header /MessageID	Echo back MessageID if MessageID is provided in request otherwise the TransactionID will be returned
Reply/ReplyCode	*success=OK, pending=PENDING, error=ERROR or FATAL*
Reply/Error	*Error message, if error encountered*
Reply/ID	*Process id (if PENDING)*

5 Appendix A: WSDL for DR Requests

This WSDL uses a set of operations for servicing all DR requests, related to information requests and alert acknowledgements.

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions xmlns:ns="http://www.uisol.com/wsdl/2009/LRSAbstract"
xmlns:ns2="http://www.uisol.com/schema/drbiznet/2009/message" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.uisol.com/wsdl/2009/LRSAbstract" attributeFormDefault="unqualified"
elementFormDefault="qualified">
        <wsdl:types>
                <xsd:schema targetNamespace="http://www.uisol.com/schema/drbiznet/2009/message">
                        <xsd:include schemaLocation="Message.xsd"/>
                </xsd:schema>
        </wsdl:types>
        <wsdl:message name="RequestMessage">
                <wsdl:part name="Message" element="ns2:RequestMessage"/>
        </wsdl:message>
        <wsdl:message name="ResponseMessage">
                <wsdl:part name="Message" element="ns2:ResponseMessage"/>
        </wsdl:message>
        <wsdl:message name="FaultMessage">
                <wsdl:part name="part1" element="ns2:FaultMessage"/>
        </wsdl:message>
        <wsdl:portType name="Operations">
                <wsdl:operation name="LRSTransactions">
                        <wsdl:input message="ns:RequestMessage"/>
                        <wsdl:output message="ns:ResponseMessage"/>
                        <wsdl:fault name="fault1" message="ns:FaultMessage"/>
```

```
        </wsdl:operation>
        <wsdl:operation name="LRSInformation">
                <wsdl:input message="ns:RequestMessage"/>
                <wsdl:output message="ns:ResponseMessage"/>
                <wsdl:fault name="fault1" message="ns:FaultMessage"/>
        </wsdl:operation>
        <wsdl:operation name="Alerts">
                <wsdl:input message="ns:RequestMessage"/>
                <wsdl:output message="ns:ResponseMessage"/>
                <wsdl:fault name="fault1" message="ns:FaultMessage"/>
        </wsdl:operation>
</wsdl:portType>
<wsdl:binding name="MISOSOAP" type="ns:Operations">
        <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
        <wsdl:operation name="LRSTransactions">
                <soap:operation soapAction="http://www.uisol.com/LRSTransactions"/>
                <wsdl:input>
                        <soap:body use="literal"/>
                </wsdl:input>
                <wsdl:output>
                        <soap:body use="literal"/>
                </wsdl:output>
                <wsdl:fault name="fault1">
                        <soap:body use="literal"/>
                </wsdl:fault>
        </wsdl:operation>
        <wsdl:operation name="LRSInformation">
                <soap:operation soapAction="http://www.uisol.com/LRSInformation"/>
                <wsdl:input>
                        <soap:body use="literal"/>
                </wsdl:input>
```

```
                <wsdl:output>
                        <soap:body use="literal"/>
                </wsdl:output>
                <wsdl:fault name="fault1">
                        <soap:body use="literal"/>
                </wsdl:fault>
        </wsdl:operation>
        <wsdl:operation name="Alerts">
                <soap:operation soapAction="http://www.uisol.com/Alerts"/>
                <wsdl:input>
                        <soap:body use="literal"/>
                </wsdl:input>
                <wsdl:output>
                        <soap:body use="literal"/>
                </wsdl:output>
                <wsdl:fault name="fault1">
                        <soap:body use="literal"/>
                </wsdl:fault>
        </wsdl:operation>
    </wsdl:binding>
    <wsdl:service name="uisol">
        <wsdl:port name="UISOLSOAP" binding="ns:UISOLSOAP">
                <soap:address location="http://www.uisol.com/"/>
        </wsdl:port>
    </wsdl:service>
</wsdl:definitions>
```

6 Appendix B: XML Schemas

The following XML schema is used to define Message structures for request and response messages for the WSDL defined in Appendix B. These are provided here for reference purposes, but versions should be downloaded from the Market web site for development use.

6.1 message.xsd

```
<?xml version="1.0" encoding="UTF-8"?>

<xsd:schema xmlns="http://www.uisol.com/schema/drbiznet/2009/message"
       xmlns:xsd="http://www.w3.org/2001/XMLSchema"
       targetNamespace="http://www.uisol.com/schema/drbiznet/2009/message"
       version="0.0.1"
       elementFormDefault="qualified"
       attributeFormDefault="unqualified">
      <xsd:complexType name="RequestType">
            <xsd:sequence>
                  <xsd:element name="Date" type="xsd:dateTime" minOccurs="0"/>
                  <xsd:element name="StartTime" type="xsd:dateTime" minOccurs="0"/>
                  <xsd:element name="EndTime" type="xsd:dateTime" minOccurs="0"/>
                  <xsd:element name="Zone" type="xsd:string" minOccurs="0"/>
                  <xsd:element name="Option" type="xsd:string" minOccurs="0"/>
                  <xsd:element name="Match" type="xsd:string" minOccurs="0"/>
                  <xsd:element name="Organization" type="xsd:string" minOccurs="0"/>
                  <xsd:element name="ID" type="xsd:string" minOccurs="0" maxOccurs="unbounded"/>
                  <xsd:element name="AlternateID" type="xsd:string" minOccurs="0"/>
                  <xsd:any namespace="##other" processContents="strict" minOccurs="0"
maxOccurs="unbounded"/>
            </xsd:sequence>
      </xsd:complexType>
      <xsd:complexType name="ReplyType">
            <xsd:sequence>
                  <xsd:element name="ReplyCode" type="xsd:string"/>
                  <xsd:element name="Error" type="xsd:string" minOccurs="0" maxOccurs="unbounded"/>
                  <xsd:element name="Timestamp" type="xsd:dateTime" minOccurs="0"/>
```

```
                  <xsd:element name="ID" type="xsd:string" minOccurs="0" maxOccurs="unbounded"/>
                  <xsd:any namespace="##other" processContents="strict" minOccurs="0"
maxOccurs="unbounded"/>
            </xsd:sequence>
      </xsd:complexType>
      <xsd:complexType name="PayloadType">
            <xsd:sequence>
                  <xsd:choice>
                        <xsd:any namespace="##other" processContents="skip" minOccurs="0"
maxOccurs="unbounded"/>
                        <xsd:element name="Document" type="xsd:string" minOccurs="0"
maxOccurs="unbounded"/>
                        <xsd:element name="Compressed" type="xsd:string" minOccurs="0"/>
                  </xsd:choice>
                  <xsd:element name="format" type="xsd:string" minOccurs="0"/>
            </xsd:sequence>
      </xsd:complexType>
      <xsd:complexType name="HeaderType">
            <xsd:sequence>
                  <xsd:element name="Verb" default="get">
                        <xsd:simpleType>
                              <xsd:restriction base="xsd:string">
                                    <xsd:enumeration value="cancel"/>
                                    <xsd:enumeration value="canceled"/>
                                    <xsd:enumeration value="change"/>
                                    <xsd:enumeration value="changed"/>
                                    <xsd:enumeration value="create"/>
                                    <xsd:enumeration value="created"/>
                                    <xsd:enumeration value="close"/>
                                    <xsd:enumeration value="closed"/>
                                    <xsd:enumeration value="delete"/>
                                    <xsd:enumeration value="deleted"/>
                                    <xsd:enumeration value="get"/>
                                    <xsd:enumeration value="reply"/>
                                    <xsd:enumeration value="submit"/>
                                    <xsd:enumeration value="action"/>
                                    <xsd:enumeration value="update"/>
                                    <xsd:enumeration value="updated"/>
```

```
                        </xsd:restriction>
                  </xsd:simpleType>
            </xsd:element>
            <xsd:element name="Noun" type="xsd:string"/>
            <xsd:element name="Revision" type="xsd:string" default="001"/>
            <xsd:element name="Source" type="xsd:string"/>
            <xsd:element name="UserID" type="xsd:string" minOccurs="0"/>
            <xsd:element name="MessageID" type="xsd:string" minOccurs="0"/>
            <xsd:element name="Comment" type="xsd:string" minOccurs="0"/>
            <xsd:any namespace="##other" processContents="strict" minOccurs="0"
maxOccurs="unbounded"/>
      </xsd:sequence>
</xsd:complexType>
<xsd:element name="Message">
      <xsd:complexType>
            <xsd:sequence>
                  <xsd:element name="Header" type="HeaderType"/>
                  <xsd:choice>
                        <xsd:element name="Request" type="RequestType" minOccurs="0"/>
                        <xsd:element name="Reply" type="ReplyType" minOccurs="0"/>
                  </xsd:choice>
                  <xsd:element name="Payload" type="PayloadType" minOccurs="0"/>
            </xsd:sequence>
      </xsd:complexType>
</xsd:element>
<xsd:complexType name="RequestMessageType">
      <xsd:sequence>
            <xsd:element name="Header" type="HeaderType"/>
            <xsd:element name="Request" type="RequestType" minOccurs="0"/>
            <xsd:element name="Payload" type="PayloadType" minOccurs="0"/>
      </xsd:sequence>
</xsd:complexType>
<xsd:element name="InformationRequest" type="RequestMessageType"/>
<xsd:element name="TransactionRequest" type="RequestMessageType"/>
<xsd:element name="AlertRequest" type="RequestMessageType"/>
<xsd:element name="ResponseMessage">
      <xsd:complexType>
            <xsd:sequence>
```

```
                        <xsd:element name="Header" type="HeaderType"/>
                        <xsd:element name="Reply" type="ReplyType" minOccurs="0"/>
                        <xsd:element name="Payload" type="PayloadType" minOccurs="0"/>
                  </xsd:sequence>
            </xsd:complexType>
      </xsd:element>
      <xsd:element name="FaultMessage">
            <xsd:complexType>
                  <xsd:sequence>
                        <xsd:element name="Reply" type="ReplyType" minOccurs="0"/>
                  </xsd:sequence>
            </xsd:complexType>
      </xsd:element>
</xsd:schema>
```

6.2 LRSDailyData.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XMLSpy v2008 rel. 2 sp2 (http://www.altova.com) by Anu (EDS) -->
<xs:schema xmlns="http://www.uisol.com/schema/drbiznet/2009/model"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.uisol.com/schema/drbiznet/2009/model" elementFormDefault="qualified"
attributeFormDefault="unqualified">
      <xs:include schemaLocation="LRSCommonTypes.xsd"/>
      <xs:element name="DailysData">
            <xs:complexType>
                  <xs:sequence>
                        <xs:element ref="DailyData" minOccurs="0" maxOccurs="unbounded"/>
                  </xs:sequence>
            </xs:complexType>
      </xs:element>
      <xs:complexType name="dailySummaryType">
            <xs:sequence>
                  <xs:element name="locationId" type="objectId" minOccurs="0"/>
                  <xs:element name="enrollmentId" type="objectId" minOccurs="0"/>
                  <xs:element name="LBA" type="organization" minOccurs="0"/>
            </xs:sequence>
```

```
      </xs:complexType>
      <xs:complexType name="dailyMeasurementInfoType">
            <xs:sequence>
                  <xs:element name="dateStamp" type="xs:dateTime" minOccurs="0"/>
                  <xs:element name="type" type="xs:string" minOccurs="0"/>
                  <xs:element name="UOM" type="xs:string" minOccurs="0"/>
                  <xs:element name="intervals" type="xs:integer" minOccurs="0"/>
                  <xs:choice>
                        <xs:element name="hourValueMeasurement" type="hourValueMeasurementInfoType"
minOccurs="23" maxOccurs="25"/>
                        <xs:element name="values" type="listOfValues" nillable="false"/>
                  </xs:choice>
                  <xs:element name="latest" type="xs:boolean" minOccurs="0"/>
                  <xs:element name="submittedBy" type="xs:string" minOccurs="0"/>
                  <xs:element name="submittedate" type="xs:dateTime" minOccurs="0"/>
                  <xs:element name="submissionError" type="xs:string" minOccurs="0"/>
            </xs:sequence>
      </xs:complexType>
      <xs:element name="DailyData">
            <xs:annotation>
                  <xs:documentation>Daily Data Object</xs:documentation>
            </xs:annotation>
            <xs:complexType>
                  <xs:sequence>
                        <xs:element name="summaryInfo" type="dailySummaryType" minOccurs="0"/>
                        <xs:element name="measurementInfo" type="dailyMeasurementInfoType" minOccurs="0"
maxOccurs="unbounded"/>
                  </xs:sequence>
            </xs:complexType>
      </xs:element>
      <xs:complexType name="hourValueMeasurementInfoType">
            <xs:sequence>
                  <xs:element name="hour" type="xs:integer" nillable="false"/>
                  <xs:element name="value" type="xs:double"/>
            </xs:sequence>
      </xs:complexType>
</xs:schema>
```

6.3 LRSIntervalData.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XMLSpy v2007 rel. 3 sp1 (http://www.altova.com) by Caro (EMBRACE) -->
<xs:schema xmlns="http://www.uisol.com/schema/drbiznet/2009/model"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.uisol.com/schema/drbiznet/2009/model" elementFormDefault="qualified"
attributeFormDefault="unqualified">
      <xs:include schemaLocation="LRSCommonTypes.xsd"/>
      <xs:element name="IntervalsData">
            <xs:complexType>
                  <xs:sequence>
                        <xs:element ref="IntervalData" minOccurs="0" maxOccurs="unbounded"/>
                  </xs:sequence>
            </xs:complexType>
      </xs:element>
      <xs:complexType name="intervalDataSummaryType">
            <xs:sequence>
                  <xs:element name="locationId" type="objectId" minOccurs="0"/>
                  <xs:element name="enrollmentId" type="objectId" minOccurs="0"/>
                  <xs:element name="LBA" type="organization" minOccurs="0"/>
            </xs:sequence>
      </xs:complexType>
      <xs:complexType name="intervalMeasurementInfoType">
            <xs:sequence>
                  <xs:element name="timestamp" type="xs:dateTime" minOccurs="0"/>
                  <xs:element name="type" type="xs:string" minOccurs="0"/>
                  <xs:element name="UOM" type="xs:string" minOccurs="0"/>
                  <xs:element name="value" type="xs:double" minOccurs="0"/>
                  <xs:element name="latest" type="xs:boolean" minOccurs="0"/>
                  <xs:element name="submittedBy" type="xs:string" minOccurs="0"/>
                  <xs:element name="submittedDate" type="xs:dateTime" minOccurs="0"/>
                  <xs:element name="submissionError" type="xs:string" minOccurs="0"/>
            </xs:sequence>
      </xs:complexType>
      <xs:element name="IntervalData">
```

```
            <xs:annotation>
                  <xs:documentation>Interval Data Object</xs:documentation>
            </xs:annotation>
            <xs:complexType>
                  <xs:sequence>
                        <xs:element name="summaryInfo" type="intervalDataSummaryType" minOccurs="0"/>
                        <xs:element name="measurementInfo" type="intervalMeasurementInfoType" minOccurs="0"
maxOccurs="unbounded"/>
                  </xs:sequence>
            </xs:complexType>
      </xs:element>
</xs:schema>
```

7 Appendix C: Payload Compression Example

The section provides an example of the code required to compress and encode a payload, where the payload is then passed as the contents of the Payload/Compressed element. The following is a Java example that leverages commonly used classes for compression and base64 encoding.

```
package BusinessService.TS.WebServiceClient.CompressionClient;
import java.util.*;
import java.io.*;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.InputStream;
import java.io.IOException;
import java.util.zip.*;
import org.apache.soap.encoding.soapenc.Base64;

public class CompressionClientCompressandEncode{
      protected byte[] input = null;
      protected String base64GzipInput = "";
      public byte[] getinput() {    return input;     }
```

```
	public void setinput(byte[] val) {  input = val;	}
	public String getbase64GzipInput() {	return base64GzipInput; }
	public void setbase64GzipInput(String val) {	base64GzipInput = val;  }
	public CompressionClientCompressandEncode() {	}
	public void invoke() throws Exception {
		In  : byte[] input
		Out : String base64GzipInput

		ByteArrayOutputStream bas = new ByteArrayOutputStream();
		GZIPOutputStream bis = new GZIPOutputStream(bas);
		bis.write(input);

		bis.close();
		base64GzipInput = Base64.encode(bas.toByteArray());
	}
}
```