

Third-Party Application Integration

ObServer v20 XML

Introduction

ObServer is the heart of North's solutions for building automation and management, including the ObSys 2.0 software suite and the Commander II product. It allows you to collect, integrate, control and manage information from building services across the enterprise in real-time.

ObServer operates on Windows platforms and can be licensed in a variety of ways to suit the solution, ranging from a simple embedded controller to a unified building management server.

With this integration guide, you will learn how to access values from the user database module using the XML service.

This document is for North users and system integrators interested in adding ObServer to their applications.

User Database

The user database provides a method of accessing values from the connected building systems. The database is engineered to collect values from and write values to the systems periodically.

This benefits third-party applications as all the values within the user database are available immediately, removing the latency experienced when communicating with the building systems directly.

XML Service

The ObServer XML service uses a simple XML model for presenting data from the user database in machine readable format (M2M).

This data is accessed across a TCP/IP network using one of the following binding methods:

- HTTP – a client sends XML requests via HTTP to the server and awaits a response;
- SOAP – a sender sends a SOAP request and awaits a response.

The XML service allows information from the user database to be requested and an entry's current value to be adjusted.

A Web Services Description Language (WSDL) and XML Schema Definition (XSD) are also provided to describe the ObServer XML service and allow rapid deployment using tools for Web Services.

The XML service is provided by the North ObServer WebView module v1.2 build 20/02/09 or later.

Contents

Introduction.....	1
User Database.....	2
XML Service Architecture	3
Object Model	4
HTTP Binding	11
SOAP Binding.....	12
Examples.....	13
Appendix A: WSDL Document.....	16
Appendix B: XML Schema.....	18
Appendix C: Document History.....	21

User Database

The user database is a multi-functional element of ObServer. The database can be used to:

- Store and provide fast access to selected values within the North extensible object model
- Monitor and notify for out-of-range alarm events
- Log values.

The database works alongside other modules and can be configured to hold up to 960 entries. This central database provides data to the XML service in addition to other modules such as the web server, BACnet device, Zip display, GSM gateway, etc.

Within the user database, up to 200 entries can be enabled to periodically log their value. Each of these logs can contain up to 2000 recorded values.

Structure

The user database module contains 60 pages. Each page contains a label, security access level and 16 configurable objects.

Each of the 16 objects within the page can be configured to store a value using the following parameters:

- Label – a brief label describing the object (20 characters);
- Type – specifies the type of value stored by the object: text string; integer; floating point number; on-off; no-yes; on-off times; date & time value; or enumerated value;
- Adjustable – enables adjustment of the value by a user;
- Access Level – controls user access when adjusting the value;
- Value High/Low limit – the typical operating range of the value. These limits are used when a user adjusts the value, and to generate an alarm message when the value exceeds this range;
- Units – used to define what the values is measured in (8 characters);
- Remote Action – an option to read a remote object, or write the value provided by a user to one of the connected building services;
- Remote Object – the object reference to read or write;
- Remote Rate – the frequency in which the Remote Object should be read;
- Remote Fails – the number of consecutive failed attempts at reading/writing the remote object (used for troubleshooting);
- Alarm Priority – enables the sending of an alarm message if the value is outside its high/low limits or Remote Fails exceeds its threshold;
- Alarm State – the current alarm state of the value;
- Data Log – enables logging of the value at the rate specified.

Example object configuration (shown in ObView):

Label	UPS Load Power
Type	Float
Adjustable	No
Access Security	0
Type ENum Alternatives	
Type Float Dps/Time periods	1
Value High Limit	100.0000
Value Low Limit	0.0000
Current Value	39
Units	%W
Value Alarm State	Ok
Value Last Updated	22/01/09 14:29:03
Remote Action	Read
Remote Object	IP.RAK.UD.P.L
Remote Rate	1 min
Remote Fails	0
Alarm Enable/Priority	0
Data Log Enable/Rate	1 min

XML Service Architecture

Client/Server

The ObServer XML service is based on a client/server network model. Third-party applications, acting as the client, make requests over the network to the XML service, acting as the server, which then responds with the data.

The XML service is part of the ObServer WebView server, which uses the default port 80.

Request/Response

Requests to the server are based on two types of request/response types:

Read

Return the object from the user database. The read request references the object at the specified URI. The read response is an XML document containing the data requested.

Write

Update the object value within the user database. The write request contains an XML document referencing the object to update and the value to send. The write response is an XML document containing the updated data.

Object Model

The user database consists of 60 pages each containing 16 objects. Pages are referenced Px, where x is the page number; and objects are referenced Oy, where y is the object number.

All information from the database is represented within a corresponding structure of page and object elements. Each object can then contain the elements label, units, log and a value.

There is a choice from eight types of element used to store the value, depending on the object's type attribute:

- text – character string;
- bool – boolean value;
- num – integer value;
- float – floating point value;
- times – list of on and off time periods;
- datetime – absolute date and time stamp;
- date – date only;
- enum – enumerated value.

XML

The ObServer XML service uses a simple XML model for presenting data from the user database in machine readable format (M2M).

The following rules apply to request and response documents:

- Documents must be well formed XML;
- Use UTF-8 character encoding, without a byte order mark;
- Documents should include the XML namespace: "http://www.northbt.com/ns/schema/observer/v10";
- Text element values containing the illegal characters '<' or '&' must be encoded as CDATA.

Errors

Errors are indicated using the result attribute within the XML document response. Possible errors are:

- ok – success, no error;
- object – unknown object;
- action – invalid action;
- value – invalid object value or type written;
- error – general error, such as a malformed xml document.

Security

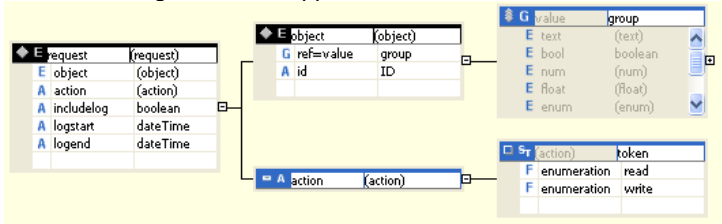
Security, if enabled, uses the HTTP basic authentication method. As part of the user database, access to individual pages and objects can be controlled by specifying the access level required.

The XML service will present objects from the user database based on the privileges available to the client. If a client has insufficient privileges to read an object, then it will not be discoverable. If a client has insufficient privileges to write to an object, then the writable attribute will be set to false.

Component Model

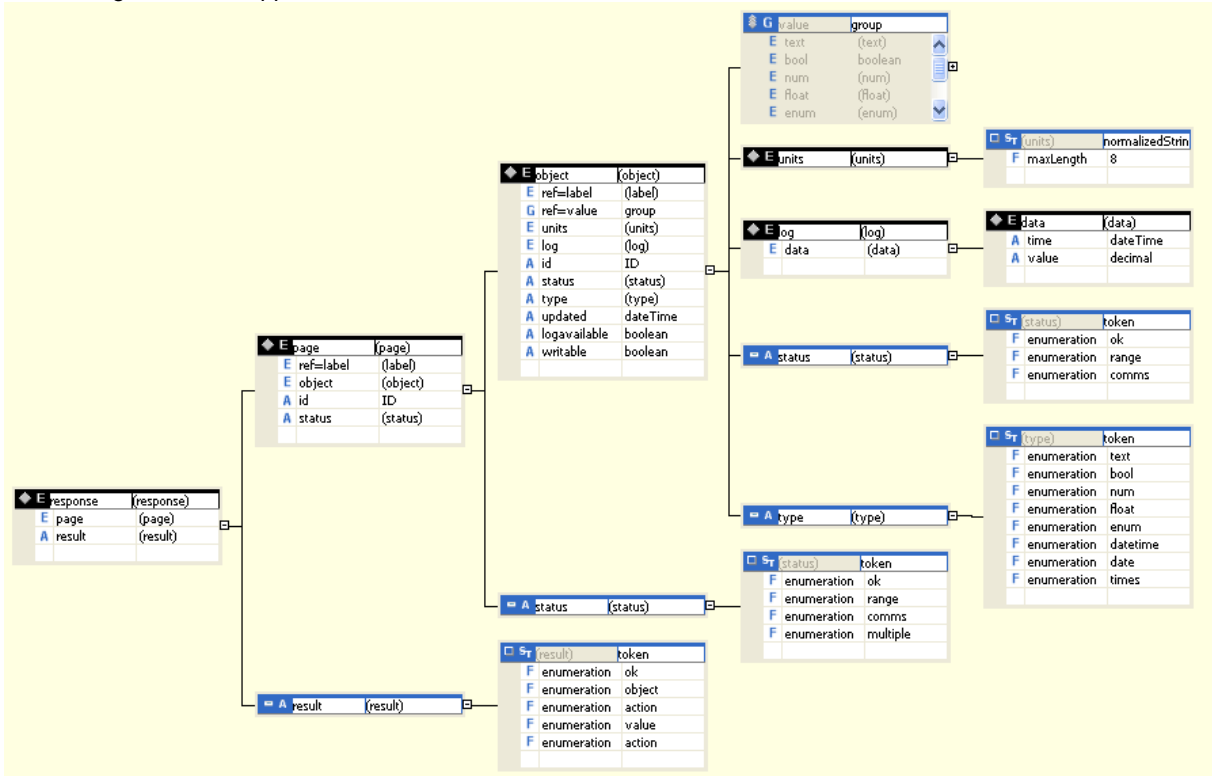
Request

The component model of a write request document from the client is summarised in the following illustration. Elements are listed along with their supported attributes.



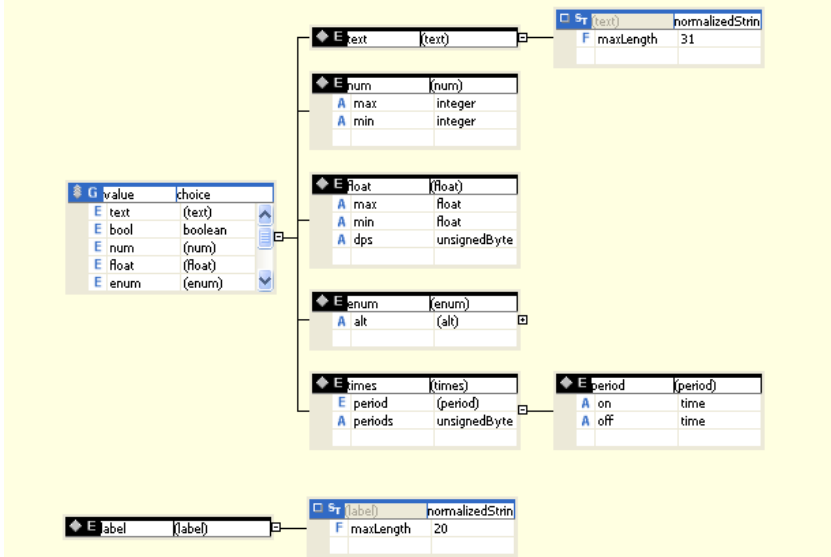
Response

The component model of a response document from the server is summarised in the following illustration. Elements are listed along with their supported attributes.



Referenced Elements

These elements are referenced from both the request and response documents.



Element Dictionary

Element <request>

The request element is the root element of a request XML document and must be present.

Attributes

Attribute	Use	Required	Value
action	Indicates the type of the request	Yes	write read
includeLog	Indicates the read response should include log data when available	No	boolean: true false. Default value false
expandLog	Indicates the read response log data should be expanded to include an entry for every sample period [note 1]	No	boolean: true false. Default value false
logstart	When includeLog is true, indicates the start time of log data [note 2]	No	dateTime: yyyy-mm-ddT hh:mm:ss
logend	When includeLog is true, indicates the end time of log data [note 2]	No	dateTime: yyyy-mm-ddT hh:mm:ss

Contains

Element	Min Occurrences	Max Occurrences
object	1	100

Notes

1. Data is logged using exception logging, where the value is recorded only on a change. This optimises the storage required. The expand log option inserts these repeated values, based on the log sample rate.
2. When requesting log data, the data returned will be one reading before the start time and one reading after the end time specified. This is to aid in the plotting of the log data.

May be included in elements

None.

XML

```
<request action="write">
  <object ...>
  </object>
</request>
```

Element <response>

The response element is the root element of a response XML document and must be present.

Attributes

Attribute	Use	Required	Value
result	Indicates the validity of the response	Yes	ok: success object: invalid object id action: invalid request action value: invalid object value or type error: general error

Contains

Element	Min Occurrences	Max Occurrences
page	0	60

May be included in elements

None.

XML

```
<response result="ok">
  <page ...>
  </page>
</response>
```

Element <page>

The page element represents a user database page.

Attributes

Attribute	Use	Required	Value
id	Contains the page identifier	Yes	P#
status	Indicates the overall status of all objects within the page	Yes	ok: object values healthy range: object value out-of-range comms: communications failure multiple: both range and comms

Contains

Element	Min Occurrences	Max Occurrences
label	1	1
object	1	16

May be included in elements

[response](#)

XML

```
<page id="P1" status="ok">
  <label>Page Label</label>
  <object ...>
  </object>
</page>
```

Element <object>

The object element represents a user database object.

Attributes

Attribute	Use	Required	Value
id	Contains the page object identifier	Yes	P#O# (see note 1)
type	Indicates the type of object	Yes (response only)	text bool num float enum datetime date times
status	Indicates the status of the object	Yes (response only)	ok: object value healthy range: object value out-of-range comms: communications failure
updated	Date and time value last updated	No	dateTime: yyyy-mm-ddT hh:mm:ss
logavailable	Indicates that log data is available	No	boolean: true false Default value false
writable	Indicates that the object value can be modified	No	boolean: true false. Default value false

Contains

Element	Min Occurrences	Max Occurrences
label	1 (response only)	1
units	0	1
log	0	Undefined

A choice of one of the following elements will also be present, and matches the type attribute:

[text](#) | [bool](#) | [num](#) | [float](#) | [enum](#) | [datetime](#) | [date](#) | [times](#)

Notes

1. If part of a read request message, the object id can have the format:
P#O# read the page object specified;
P# read the value of all objects within the page specified;
* read all objects.

May be included in elements

[page](#) and [request](#)

XML

```
<object id="P101" type="float" status="ok" updated="2009-01-22T12:15:00"
logavailable="true" writable="false">
  <label>Flow</label>
  <float ...></float>
  <units>m3/hr</units>
</object>
```

Element <label>

The label element contains a descriptive text string for the parent element.

Attributes

No attributes supported.

Contains

The element contains a string with a maximum length of 20 characters. The string may be encoded using “<![CDATA[...]]>”, particularly if the string contains the characters ‘<’ or ‘&’.

May be included in elements

[page](#)

[object](#)

XML

```
<label>String</label>
```

Element <units>

The units element contains a text string representing the object value's units of measurement. This element is optional.

Attributes

No attributes supported.

Contains

The element contains a string with a maximum length of 8 characters. The string may be encoded using “<![CDATA[...]]>”, particularly if the string contains the characters ‘<’ or ‘&’.

May be included in elements

[object](#)

XML

```
<units>°C</units>
```

Element <text>

The text element contains the value of the user database object which has been configured as a type text.

Attributes

No attributes supported.

Contains

The element contains a string with a maximum length of 31 characters. The string may be encoded using “<![CDATA[...]]>”, particularly if the string contains the characters ‘<’ or ‘&’.

May be included in elements

[object](#)

XML

```
<object id="P102" type="text" status="ok" updated="2009-01-22T12:15:00">
  <label>Message</label>
  <text>Hello World!</text>
</object>
```

Element <bool>

The bool element contains the value of the user database object which has been configured as a type NoYes or OffOn. This value is interpreted as a boolean value where ‘no’ or ‘off’ are the false state, and ‘yes’ or ‘on’ are the true state.

Attributes

No attributes supported.

Contains

The element contains the text true or false.

May be included in elements

[object](#)

XML

```
<object id="P103" type="bool" status="ok" updated="2009-01-22T12:15:00">
  <label>Door Locked</label>
  <bool>true</bool>
</object>
```

Element <num>

The num element contains the value of the user database object which has been configured as a type number.

Attributes

Attribute	Use	Required	Value
min	Indicates the minimum value range to use when adjusting the value	No	integer
max	Indicates the maximum value range to use when adjusting the value	No	integer

Contains

The element contains a signed integer.

May be included in elements

[object](#)

XML

```
<object id="P104" type="num" status="ok" updated="2009-01-22T12:15:00">
  <label>Hours Run</label>
  <num min="0" max="60000">123</num>
  <units>Hrs</units>
</object>
```

Element <float>

The float element contains the value of the user database object which has been configured as a type float.

Attributes

Attribute	Use	Required	Value
min	Indicates the minimum value range to use when adjusting the value	No	float
max	Indicates the maximum value range to use when adjusting the value	No	float
dps	Recommended number of decimal places to use when displaying or adjusting the value	No	unsignedByte

Contains

The element contains a floating point number (32-bit).

May be included in elements

[object](#)

XML

```
<object id="P101" type="float" status="ok" updated="2009-01-22T12:15:00"
logavailable="true">
  <label>Flow</label>
  <float min="0" max="30" dps="1">25.3</float>
  <units>m3/hr</units>
</object>
```

Element <enum>

The enum element contains the value of the user database object which has been configured as a type enumerated.

Attributes

Attribute	Use	Required	Value
alt	List of enumerated states available	No	List of tokens, separated by a space.

Contains

The element contains the enumerated state token.

May be included in elements

[object](#)

XML

```
<object id="P105" type="enum" status="ok" updated="2009-01-22T12:15:00">
  <label>Fire Alarm State</label>
  <enum alt="ok flt isolate fire">fire</enum>
</object>
```

Element <datetime>

The datetime element contains the value of the user database object which has been configured as a type date and time.

Attributes

No attributes supported.

Contains

The element contains the date and time in the format *yyyy-mm-ddThh:mm:ss*.

May be included in elements

[object](#)

XML

```
<object id="P106" type="datetime" status="ok" updated="2009-01-22T12:15:00">
  <label>Last reset</label>
  <datetime>2009-01-22T12:15:00</datetime>
</object>
```

Element <date>

The date element contains the value of the user database object which has been configured as a type date.

Attributes

No attributes supported.

Contains

The element contains the date in the format *yyyy-mm-dd*.

May be included in elements

[object](#)

XML

```
<object id="P107" type="date" status="ok" updated="2009-01-22T12:15:00">
  <label>Today</label>
  <date>2009-01-22</date>
</object>
```

Element <times>

The times element contains the value of the user database object which has been configured as a type times. This value holds a time profile with several on and off times.

Attributes

Attribute	Use	Required	Value
periods	Indicates the maximum number of on-off time periods.	No	Unsigned byte

Contains

Element	Min Occurrences	Max Occurrences
period	0	See periods attribute

May be included in elements

[object](#)

XML

```
<object id="P108" type="times" status="ok" updated="2009-01-22T12:15:00">
  <label>Heating</label>
  <times periods="2">
    <period on="08:00:00" off="10:00:00" />
    <period on="19:30:00" off="23:00:00" />
  </times>
</object>
```

Element <period>

The period element contains part of a time profile value with a single on and off time.

Attributes

Attribute	Use	Required	Value
on	On time	Yes	time: hh:mm:00
off	Off time	Yes	time: hh:mm:00

Contains

Empty.

May be included in elements

[times](#)

XML

```
<times periods="2">
  <period on="08:00:00" off="10:00:00" />
  <period on="19:30:00" off="23:00:00" />
</times>
```

Element <log>

The log element contains a list of logged values collected periodically for the object.

Attributes

No attributes supported.

Contains

Element	Min Occurrences	Max Occurrences
data	0	undefined

May be included in elements

[object](#)

XML

```
<log>
  <data ... />
  <data ... />
</log>
```

Element <data>

The data element contains a single log entry.

Attributes

Attribute	Use	Required	Value
time	Time collected	Yes	dateTime: yyyy-mm-ddT hh:mm:ss
value	Object value	Yes	decimal

Contains

Empty.

May be included in elements

[log](#)

XML

```
<log>
  <data time="2009-01-22T12:15:00" value="12.3"/>
  <data time="2009-01-22T12:30:00" value="15"/>
  <data time="2009-01-22T12:45:00" value="18.1"/>
  ...
</log>
```

HTTP Binding

The HTTP binding specifies a simple mapping of request-response messages to the HTTP 1.1 specification.

Requests

A read request can be sent in one of two ways: either a simple HTTP GET message is sent with the data referenced by the specified URI; or an HTTP POST message is sent containing an XML document referencing the data. Both methods return an XML document containing the data requested.

The write request is formed of an HTTP POST message containing an XML document referencing the data to update and the value to send. This returns an XML document containing the updated data.

URIs

The XML service binds its addressing to a URI (uniform resource identifier), the standard way to identify a resource on the Internet.

The URI of the XML service has the format: `http://servername/data/`

For a read request using the HTTP GET method, the URI is used to reference the target object:

<code>http://servername/data/</code>	read/discover all objects
<code>http://servername/data/P5.xml</code>	read all objects within page 5
<code>http://servername/data/P2010.xml</code>	read page 2 object 10 data
<code>http://servername/data/P103L.xml</code>	read page 1 object 3 data, including log data

HTTP Headers

The XML service processes a request, and then returns the XML document response with an HTTP status code of 200 OK. If the URI is not of the format `http://server/data/`, or authentication is required, then an alternative HTTP status code may be used.

The Content-Type header must be included with the MIME type of "application/xml".

Example

```
GET /data/P107.xml HTTP/1.1

HTTP/1.1 200 OK
Server: WebView/1.2.1 ObServer/2.0
Content-Type: application/xml

<?xml version="1.0" encoding="utf-8"?>
<response ...>

</response>
```

SOAP Binding

The SOAP binding specifies a mapping of request and response messages to the SOAP 1.2 specification, with HTTP binding.

Message Exchange Patterns

The XML service supports both SOAP Request-Response and SOAP Response message exchange patterns:

The SOAP Request-Response message exchange pattern binds to the HTTP POST message. A SOAP read or write request message is sent referencing the object within the body; the SOAP response includes the object data requested.

The SOAP Response message exchange pattern binds to the HTTP GET message with the URI referencing the object. The SOAP response includes the object data requested. This is regarded as a safe and idempotent method.

URIs

The XML service binds its addressing to a URI (uniform resource identifier), the standard way to identify a resource on the Internet.

The URI of the XML SOAP service is: `http://servername/data/soap/`

For a SOAP Response message exchange pattern, the URI is used to reference the target object:

<code>http://servername/data/soap/</code>	read/discover all objects
<code>http://servername/data/soap/P5</code>	read all objects within page 5
<code>http://servername/data/soap/P2010</code>	read page 2 object 10 data
<code>http://servername/data/soap/P103L</code>	read page 1 object 3 data, including log data

HTTP Headers

The XML service returns the SOAP document response with an HTTP status code of 200 OK. If the URI is not of the format `http://server/data/soap/`, or authentication is required, then an alternative HTTP status code may be used.

The Content-Type header must be included with the MIME type of "application/soap+xml" and the encoding type "utf-8".

Error Handling

This specification does not use SOAP Fault codes. Any failure is indicated within the response element attribute `result`.

Example

```
POST /data/soap/ HTTP/1.1
Host: webview.northbt.com
Content-Type: application/soap+xml; charset="utf-8"
Content-Length: nnnn

<?xml version="1.0" encoding="utf-8"?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Body>
    <request xmlns="http://www.northbt.com/ns/schema/observer/v10" action="read">
      <object ...>

      </object ...>
    </request>
  </env:Body>
</env:Envelope>

HTTP/1.1 200 OK
Server: WebView/1.2.1 ObServer/2.0
Content-Type: application/soap+xml; charset="utf-8"

<?xml version="1.0" encoding="utf-8"?>
<env:Envelope xmlns:env="http://w3.org/2003/05/soap-envelope">
  <env:Body>
    <response ...>

    </response>
  </env:Body>
</env:Envelope>
```

Examples

Example 1: Discovery of Objects Available

To request all the user database objects available, the client sends a read request without a page or object reference. The simplest method for this is to send an HTTP GET request to the server with the URI `http://server/data/`

The server will respond with an XML document listing all user database pages/objects configured:

```
<?xml version="1.0" encoding="utf-8"?>
<response xmlns="http://www.northbt.com/ns/schema/observer/v10" result="ok">
  <page id="P1" status="ok">
    <label>Page 1 Label</label>
    <object id="P101" type="float" status="ok" updated="2009-01-22T16:14:00"
logavailable="true">
      <label>Flow</label>
      <float min="0" max="30" dps="1">12.3</float>
      <units>m3/hr</units>
    </object>
    <object id="P102" type="text" status="ok" updated="2009-01-22T16:14:00" writable="true">
      <label>Message</label>
      <text>Hello World!</text>
    </object>
    <object id="P103" type="bool" status="ok" updated="2009-01-22T16:14:00">
      <label>Door locked</label>
      <bool>true</bool>
    </object>
    <object id="P104" type="num" status="ok" updated="2009-01-22T16:14:00">
      <label>Hours Run</label>
      <num min="0" max="60000">123</num>
      <units>Hrs</units>
    </object>
    <object id="P105" type="enum" status="ok" updated="2009-01-22T12:15:00">
      <label>Fire Alarm State</label>
      <enum alt="ok flt isolate fire">fire</enum>
    </object>
    <object id="P106" type="datetime" status="ok" updated="2009-01-22T12:15:00">
      <label>Last reset</label>
      <datetime>2009-01-22T12:15:00</datetime>
    </object>
    <object id="P107" type="date" status="ok" updated="2009-01-22T12:15:00">
      <label>Today</label>
      <date>2009-01-22</date>
    </object>
    <object id="P108" type="times" status="ok" updated="2009-01-22T12:15:00">
      <label>Heating</label>
      <times periods="2">
        <period on="08:00:00" off="10:00:00" />
        <period on="19:30:00" off="23:00:00" />
      </times>
    </object>
  </page>
  <page id="P2" status="comms">
    <label>Page 2 Label</label>
    <object id="P201" type="float" status="ok" updated="2009-01-20T11:11:30" writable="true">
      <label>Setpoint</label>
      <float dps="1">25.1</float>
      <units>°C</units>
    </object>
    <object id="P202" type="bool" status="comms" updated="2009-01-22T16:14:00">
      <label>Occupied</label>
      <bool>false</bool>
    </object>
  </page>
</response>
```

Example 2: Reading an Object's Value

This example requests the value of page 1 object 7 from the user database. Again we send an HTTP GET request to the server with the URI `http://server/data/P107`, although we could use a POST request if preferred.

The server will respond with the xml document:

```
<?xml version="1.0" encoding="utf-8"?>
<response xmlns="http://www.northbt.com/ns/schema/observer/v10" result="ok">
  <page id="P1" status="ok">
    <label>Page 1 Label</label>
    <object id="P107" type="date" status="ok" updated="2009-01-26T00:00:00">
      <label>Today</label>
      <date>2009-01-26</date>
    </object>
  </page>
</response>
```

Example 3: Reading an Object's Log Data

If an object has been enabled to periodically log its value, indicated by the attribute `logavailable`, then this log data may be requested. The entire log data can be requested using the URI, e.g. `http://server/data/P101L`

This example sends an HTTP POST to request the log data from page 1 object 1 using the XML service URI `http://server/data/`. Using this method we can include the optional attributes `logstart` and `logend`.

Note how the returned log data includes one reading before `logstart` and one after `logend`. This is to assist when plotting the logged data.

The client sends the xml document:

```
<?xml version="1.0" encoding="utf-8"?>
<request xmlns="http://www.northbt.com/ns/schema/observer/v10" action="read"
  includelog="true" logstart="2009-01-20T10:00:00" logend="2009-01-20T19:00:00">
  <object id="P101"/>
</request>
```

The server responds with the document containing the updated objects:

```
<?xml version="1.0" encoding="utf-8"?>
<response xmlns="http://www.northbt.com/ns/schema/observer/v10" result="ok">
  <page id="P1" status="ok">
    <label>Page 1 Label</label>
    <object id="P101" type="float" status="ok" updated="2009-01-20T19:45:00"
  logavailable="true">
      <label>Flow</label>
      <float min="0" max="30" dps="1">12.3</float>
      <units>m3/hr</units>
      <log>
        <data time="2009-01-20T09:55:36" value="12.3"/>
        <data time="2009-01-20T10:29:12" value="15.1"/>
        <data time="2009-01-20T11:00:00" value="16"/>
        <data time="2009-01-20T11:30:05" value="13.9"/>
        <data time="2009-01-20T12:00:01" value="24.5"/>
        <data time="2009-01-20T12:30:00" value="18.1"/>
        <data time="2009-01-20T13:00:16" value="16.6"/>
        <data time="2009-01-20T13:30:00" value="10.2"/>
        <data time="2009-01-20T14:00:01" value="12.8"/>
        <data time="2009-01-20T14:30:00" value="13.1"/>
        <data time="2009-01-20T15:00:00" value="18"/>
        <data time="2009-01-20T15:30:00" value="20.6"/>
        <data time="2009-01-20T16:00:00" value="22.2"/>
        <data time="2009-01-20T16:29:56" value="22.1"/>
        <data time="2009-01-20T17:00:00" value="23.5"/>
        <data time="2009-01-20T17:30:00" value="24.8"/>
        <data time="2009-01-20T18:00:00" value="21.7"/>
        <data time="2009-01-20T18:30:00" value="25.2"/>
        <data time="2009-01-20T19:01:08" value="28"/>
      </log>
    </object>
  </page>
</response>
```

Example 4: Writing an Object's Value

If an object is enabled with write permissions, indicated by the attribute `writable`, then the value of this object may be adjusted.

This example sends an HTTP POST write request to adjust the value of page 2 object 1, at the XML service URI `http://server/data/`. Note how the value element (i.e. `float`) matches the object type obtained from a previous read request.

The client sends the xml document:

```
<?xml version="1.0" encoding="utf-8"?>
<request xmlns="http://www.northbt.com/ns/schema/observer/v10" action="write">
  <object id="P201">
    <float>30</float>
  </object>
</request>
```

The server responds with the document containing the updated objects:

```
<?xml version="1.0" encoding="utf-8"?>
<response xmlns="http://www.northbt.com/ns/schema/observer/v10" result="ok">
<page id="P2" status="comms">
  <label>Page 2 Label</label>
  <object id="P201" type="float" status="ok" updated="2009-01-20T14:10:06" writable="true">
    <label>Setpoint</label>
    <float dps="1">30</float>
    <units>°C</units>
  </object>
</page>
</response>
```

Example 5: Error Responses

This example illustrates the error response after sending a write request to an unknown object.

As the object does not exist within the user database, the server will send the following response with the `result` attribute indicating the object is in error:

```
<?xml version="1.0" encoding="utf-8"?>
<response xmlns="http://www.northbt.com/ns/schema/observer/v10" result="object">
</response>
```

This example illustrates the error response after sending a write request with an incorrect value type.

The request sent the object value within a `float` element but the object type is a datetime type, so the server will send the following response with the `result` attribute indicating the value is in error:

```
<?xml version="1.0" encoding="utf-8"?>
<response xmlns="http://www.northbt.com/ns/schema/observer/v10" result="value">
</response>
```

Example 6: SOAP Request

This example repeats [example 2](#) from above, but within a SOAP envelope.

A request for the value of page 1 object 7 from the user database is made using an HTTP GET request to the server at the URI `http://server/data/soap/P107`.

The server will respond with the SOAP message:

```
<?xml version="1.0" encoding="utf-8"?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Body>
    <response xmlns="http://www.northbt.com/ns/schema/observer/v10" result="ok">
      <page id="P1" status="ok">
        <label>Page 1 Label</label>
        <object id="P107" type="date" status="ok" updated="2009-01-26T00:00:00">
          <label>Today</label>
          <date>2009-01-26</date>
        </object>
      </page>
    </response>
  </env:Body>
</env:Envelope>
```

Appendix A: WSDL Document

The XML service is described in the following WSDL 2.0 document. The latest version is available at <http://www.northbt.com/ns/wsd/observer>

In the service element of the WSDL document replace 'localhost' with your own WebView server hostname.

```
<?xml version="1.0" encoding="utf-8" ?>
<wsdl:description xmlns:w3="http://www.w3.org/ns/wsd/"
  targetNamespace="http://www.northbt.com/ns/wsd/observer/v10"
  xmlns:tns="http://www.northbt.com/ns/wsd/observer/v10"
  xmlns:obns="http://www.northbt.com/ns/schema/observer/v10"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:w3="http://www.w3.org/ns/wsd-extensions"
  xmlns:wsoap="http://www.w3.org/ns/wsd/soap"
  xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
  xmlns:whttp="http://www.w3.org/ns/wsd/http">

  <wsdl:types>
    <xs:import namespace="http://www.northbt.com/ns/schema/observer/v10"
      schemaLocation="http://www.northbt.com/ns/schema/observer/v10.xsd"/>
  </wsdl:types>

  <wsdl:interface name="ObServerInterface">
    <wsdl:operation name="opSimpleRead"
      pattern="http://www.w3.org/ns/wsd/in-out"
      style="http://www.w3.org/ns/wsd/style/iri"
      wsdl:safe="true">
      <wsdl:input messageLabel="In"
        element="#none"/>
      <wsdl:output messageLabel="Out"
        element="obns:response"/>
    </wsdl:operation>

    <wsdl:operation name="opObjectQuery"
      pattern="http://www.w3.org/ns/wsd/in-out">
      <wsdl:input messageLabel="In"
        element="obns:request"/>
      <wsdl:output messageLabel="Out"
        element="obns:response"/>
    </wsdl:operation>
  </wsdl:interface>

  <wsdl:binding name="ObServerSOAPBinding"
    interface="tns:ObServerInterface"
    type="http://www.w3.org/ns/wsd/soap"
    wsoap:protocol="http://www.w3.org/2003/05/soap/bindings/HTTP/">
    <wsdl:operation ref="tns:opSimpleRead"
      wsoap:mep="http://www.w3.org/2003/05/soap/mep/soap-response"/>
    <wsdl:operation ref="tns:opObjectQuery"
      wsoap:mep="http://www.w3.org/2003/05/soap/mep/request-response"/>
  </wsdl:binding>

  <wsdl:binding name="ObServerXMLBinding"
    interface="tns:ObServerInterface"
    type="http://www.w3.org/ns/wsd/http">
    <wsdl:operation ref="tns:opSimpleRead"
      whttp:method="GET"
      whttp:location="{id}"/>
    <wsdl:operation ref="tns:opObjectQuery"
      whttp:method="PUT"/>
  </wsdl:binding>

  <wsdl:service name="ObServerSOAPService"
    interface="tns:ObServerInterface">
    <wsdl:endpoint name="ObServerSOAPEndpoint"
      binding="tns:ObServerSOAPBinding"
      address="http://localhost/data/soap/">
  </wsdl:service>
</wsdl:description>
```

```
        whttp:authenticationScheme="basic"
        whttp:authenticationRealm="WebView Secure Area"/>
</wsdl:service>

<wsdl:service name="ObServerXMLService"
  interface="tns:ObServerInterface">
  <wsdl:endpoint name="ObServerXMLEndpoint"
    binding="tns:ObServerHTTPBinding"
    address="http://localhost/data/"
    whttp:authenticationScheme="basic"
    whttp:authenticationRealm="WebView Secure Area"/>
  </wsdl:service>
</wsdl:description>
```

Appendix B: XML Schema

The XML service uses the following XML schema. The latest version of the schema is available at <http://www.northbt.com/ns/schema/observer>

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema xmlns="http://www.northbt.com/ns/schema/observer/v10" elementFormDefault="qualified"
targetNamespace="http://www.northbt.com/ns/schema/observer/v10"
xmlns:xs="http://www.w3.org/2001/XMLSchema">

  <xs:group name="value">
    <xs:choice>
      <xs:element name="text">
        <xs:simpleType>
          <xs:restriction base="xs:normalizedString">
            <xs:maxLength value="31" />
          </xs:restriction>
        </xs:simpleType>
      </xs:element>
      <xs:element name="bool" type="xs:boolean" />
      <xs:element name="num">
        <xs:complexType>
          <xs:simpleContent>
            <xs:extension base="xs:integer">
              <xs:attribute name="max" type="xs:integer" default="0" />
              <xs:attribute name="min" type="xs:integer" default="0" />
            </xs:extension>
          </xs:simpleContent>
        </xs:complexType>
      </xs:element>
      <xs:element name="float">
        <xs:complexType>
          <xs:simpleContent>
            <xs:extension base="xs:float">
              <xs:attribute name="max" type="xs:float" default="0" />
              <xs:attribute name="min" type="xs:float" default="0" />
              <xs:attribute name="dps" type="xs:unsignedByte" default="0" />
            </xs:extension>
          </xs:simpleContent>
        </xs:complexType>
      </xs:element>
      <xs:element name="enum">
        <xs:complexType>
          <xs:simpleContent>
            <xs:extension base="xs:token">
              <xs:attribute name="alt">
                <xs:simpleType>
                  <xs:list itemType="xs:token" />
                </xs:simpleType>
              </xs:attribute>
            </xs:extension>
          </xs:simpleContent>
        </xs:complexType>
      </xs:element>
      <xs:element name="datetime" type="xs:dateTime" />
      <xs:element name="date" type="xs:date" />
      <xs:element name="times">
        <xs:complexType>
          <xs:sequence minOccurs="0">
            <xs:element name="period">
              <xs:complexType>
                <xs:attribute name="on" type="xs:time" use="required" />
                <xs:attribute name="off" type="xs:time" use="required" />
              </xs:complexType>
            </xs:element>
          </xs:sequence>
          <xs:attribute name="periods" type="xs:unsignedByte" />
        </xs:complexType>
      </xs:element>
    </xs:choice>
  </xs:group>

```

```

    </xs:choice>
  </xs:group>

  <xs:element name="label">
    <xs:simpleType>
      <xs:restriction base="xs:normalizedString">
        <xs:maxLength value="20" />
      </xs:restriction>
    </xs:simpleType>
  </xs:element>

  <xs:element name="response">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="page" minOccurs="0" maxOccurs="60">
          <xs:complexType>
            <xs:sequence>
              <xs:element ref="label" minOccurs="1" maxOccurs="1" />
              <xs:element name="object" minOccurs="1" maxOccurs="16">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element ref="label" minOccurs="1" maxOccurs="1" />
                    <xs:group ref="value" minOccurs="1" maxOccurs="1" />
                    <xs:element name="units" minOccurs="0" maxOccurs="1">
                      <xs:simpleType>
                        <xs:restriction base="xs:normalizedString">
                          <xs:maxLength value="8" />
                        </xs:restriction>
                      </xs:simpleType>
                    </xs:element>
                    <xs:element name="log" minOccurs="0" maxOccurs="1">
                      <xs:complexType>
                        <xs:sequence>
                          <xs:element name="data" minOccurs="0" maxOccurs="unbounded">
                            <xs:complexType>
                              <xs:attribute name="time" type="xs:dateTime" use="required" />
                              <xs:attribute name="value" type="xs:decimal" use="required" />
                            </xs:complexType>
                          </xs:element>
                        </xs:sequence>
                      </xs:complexType>
                    </xs:element>
                  </xs:sequence>
                </xs:complexType>
              </xs:element>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
        <xs:attribute name="id" type="xs:ID" use="required" />
        <xs:attribute name="status" use="required">
          <xs:simpleType>
            <xs:restriction base="xs:token">
              <xs:enumeration value="ok" />
              <xs:enumeration value="range" />
              <xs:enumeration value="comms" />
            </xs:restriction>
          </xs:simpleType>
        </xs:attribute>
        <xs:attribute name="type">
          <xs:simpleType>
            <xs:restriction base="xs:token">
              <xs:enumeration value="text" />
              <xs:enumeration value="bool" />
              <xs:enumeration value="num" />
              <xs:enumeration value="float" />
              <xs:enumeration value="enum" />
              <xs:enumeration value="datetime" />
              <xs:enumeration value="date" />
              <xs:enumeration value="times" />
            </xs:restriction>
          </xs:simpleType>
        </xs:attribute>
        <xs:attribute name="updated" type="xs:dateTime" />
        <xs:attribute name="logavailable" type="xs:boolean" default="false" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>

```

```

        <xs:attribute name="writable" type="xs:boolean" default="false" />
    </xs:complexType>
</xs:element>
</xs:sequence>
<xs:attribute name="id" type="xs:ID" use="required" />
<xs:attribute name="status" use="required">
    <xs:simpleType>
        <xs:restriction base="xs:token">
            <xs:enumeration value="ok" />
            <xs:enumeration value="range" />
            <xs:enumeration value="comms" />
            <xs:enumeration value="multiple" />
        </xs:restriction>
    </xs:simpleType>
</xs:attribute>
</xs:complexType>
</xs:element>
</xs:sequence>
<xs:attribute name="result" use="required">
    <xs:simpleType>
        <xs:restriction base="xs:token">
            <xs:enumeration value="ok" />
            <xs:enumeration value="object" />
            <xs:enumeration value="action" />
            <xs:enumeration value="value" />
            <xs:enumeration value="error" />
        </xs:restriction>
    </xs:simpleType>
</xs:attribute>
</xs:complexType>
</xs:element>

<xs:element name="request">
    <xs:complexType>
        <xs:sequence>
            <xs:element name="object" minOccurs="1" maxOccurs="100">
                <xs:complexType>
                    <xs:sequence>
                        <xs:group ref="value" minOccurs="1" maxOccurs="1" />
                    </xs:sequence>
                    <xs:attribute name="id" type="xs:ID" use="required" />
                </xs:complexType>
            </xs:element>
        </xs:sequence>
        <xs:attribute name="action" use="required">
            <xs:simpleType>
                <xs:restriction base="xs:token">
                    <xs:enumeration value="read" />
                    <xs:enumeration value="write" />
                </xs:restriction>
            </xs:simpleType>
        </xs:attribute>
        <xs:attribute name="includeLog" type="xs:boolean" default="false" />
        <xs:attribute name="expandLog" type="xs:boolean" default="false" />
        <xs:attribute name="logstart" type="xs:dateTime" />
        <xs:attribute name="logend" type="xs:dateTime" />
    </xs:complexType>
</xs:element>
</xs:schema>

```

Appendix C: Document History

Date	Details
20/02/2009	Initial release of XML service and documentation.
16/05/2009	Added the attribute 'expandlog' to requests for log data.

The latest version of this document is available from <http://www.northbt.com/library>.