Bindings for OBIX: WebSocket Bindings Version 1.0

Committee Specification Draft 01

19 December 2013

Specification URIs

This version:

http://docs.oasis-open.org/obix/obix-websocket/v1.0/csd01/obix-websocket-v1.0-csd01.pdf (Authoritative)
http://docs.oasis-open.org/obix/obix-websocket/v1.0/csd01/obix-websocket-v1.0-csd01.html
http://docs.oasis-open.org/obix/obix-websocket/v1.0/csd01/obix-websocket-v1.0-csd01.doc

Previous version:

N/A

Latest version:

http://docs.oasis-open.org/obix/obix-websocket/v1.0/obix-websocket-v1.0.pdf (Authoritative)
http://docs.oasis-open.org/obix/obix-websocket/v1.0/obix-websocket-v1.0.html
http://docs.oasis-open.org/obix/obix-websocket/v1.0/obix-websocket-v1.0.doc

Technical Committee:

OASIS Open Building Information Exchange (oBIX) TC

Chair:

Toby Considine (toby.considine@unc.edu), University of North Carolina at Chapel Hill

Editor:

Matthias Hub (matthias.hub@de.ibm.com), IBM

Related work:

This specification is related to:

- *OBIX Version 1.1*. Edited by Craig Gemmill. Latest version. http://docs.oasis-open.org/obix/obix/v1.1/obix-v1.1.html.
- *Bindings for OBIX: REST Bindings Version 1.0*. Edited by Craig Gemmill and Markus Jung. Latest version. http://docs.oasis-open.org/obix/obix-rest/v1.0/obix-rest-v1.0.html.
- *Bindings for OBIX: SOAP Bindings Version 1.0.* Edited by Markus Jung. Latest version.http://docs.oasis-open.org/obix/obix-soap/v1.0/obix-soap-v1.0.html.
- *Encodings for OBIX: Common Encodings Version 1.0.* Edited by Marcus Jung. Latest version. http://docs.oasis-open.org/obix/obix-encodings/v1.0/obix-encodings-v1.0.html.

Abstract:

This document specifies WebSocket binding for OBIX.

Status:

This document was last revised or approved by the OASIS Open Building Information Exchange (oBIX) TC on the above date. The level of approval is also listed above. Check the “Latest version” location noted above for possible later revisions of this document.

Technical Committee members should send comments on this specification to the Technical Committee’s email list. Others should send comments to the Technical Committee by using the

"Send A Comment" button on the Technical Committee's web page at http://www.oasis-open.org/committees/obix/.

For information on whether any patents have been disclosed that may be essential to implementing this specification, and any offers of patent licensing terms, please refer to the Intellectual Property Rights section of the Technical Committee web page (http://www.oasis-open.org/committees/obix/ipr.php).

Citation format:

When referencing this specification the following citation format should be used:

[OBIX-WebSocket]

Bindings for OBIX: WebSocket Bindings Version 1.0. Edited by Matthias Hub. 19 December 2013. OASIS Committee Specification Draft 01. http://docs.oasis-open.org/obix/obix-websocket/v1.0/csd01/obix-websocket-v1.0-csd01.html. Latest version: http://docs.oasis-open.org/obix/obix-websocket/v1.0/obix-websocket-v1.0.html.

Notices

All capitalized terms in the following text have the meanings assigned to them in the OASIS Intellectual Property Rights Policy (the "OASIS IPR Policy"). The full Policy may be found at the OASIS website.

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published, and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this section are included on all such copies and derivative works. However, this document itself may not be modified in any way, including by removing the copyright notice or references to OASIS, except as needed for the purpose of developing any document or deliverable produced by an OASIS Technical Committee (in which case the rules applicable to copyrights, as set forth in the OASIS IPR Policy, must be followed) or as required to translate it into languages other than English.

The limited permissions granted above are perpetual and will not be revoked by OASIS or its successors or assigns.

This document and the information contained herein is provided on an "AS IS" basis and OASIS DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY OWNERSHIP RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

OASIS requests that any OASIS Party or any other party that believes it has patent claims that would necessarily be infringed by implementations of this OASIS Committee Specification or OASIS Standard, to notify OASIS TC Administrator and provide an indication of its willingness to grant patent licenses to such patent claims in a manner consistent with the IPR Mode of the OASIS Technical Committee that produced this specification.

OASIS invites any party to contact the OASIS TC Administrator if it is aware of a claim of ownership of any patent claims that would necessarily be infringed by implementations of this specification by a patent holder that is not willing to provide a license to such patent claims in a manner consistent with the IPR Mode of the OASIS Technical Committee that produced this specification. OASIS may include such claims on its website, but disclaims any obligation to do so.

OASIS takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; neither does it represent that it has made any effort to identify any such rights. Information on OASIS' procedures with respect to rights in any document or deliverable produced by an OASIS Technical Committee can be found on the OASIS website. Copies of claims of rights made available for publication and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this OASIS Committee Specification or OASIS Standard, can be obtained from the OASIS TC Administrator. OASIS makes no representation that any information or list of intellectual property rights will at any time be complete, or that any claims in such list are, in fact, Essential Claims.

The name "OASIS" is a trademark of OASIS, the owner and developer of this specification, and should be used only to refer to the organization and its official outputs. OASIS welcomes reference to, and implementation and use of, specifications, while reserving the right to enforce its marks against misleading uses. Please see http://www.oasis-open.org/policies-guidelines/trademark for above guidance.

Table of Contents

Table of Tables

1 Introduction

All text is normative unless otherwise labeled.

1.1 Terminology

The key words “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “MAY”, and “OPTIONAL” in this document are to be interpreted as described in **[RFC2119]**.

1.2 Normative References

[RFC2119] Bradner, S., “Key words for use in RFCs to Indicate Requirement Levels”, BCP 14, RFC 2119, March 1997. http://www.ietf.org/rfc/rfc2119.txt.

[OBIX] *OBIX Version 1.1*. Edited by Craig Gemmill. Latest version. http://docs.oasis-open.org/obix/obix/v1.1/obix-v1.1.html.

[OBIX Encodings] *Encodings for OBIX: Common Encodings Version 1.0.* Edited by Marcus Jung. Latest version. http://docs.oasis-open.org/obix/obix-encodings/v1.0/obix-encodings-v1.0.html.

[OBIX REST] *Bindings for OBIX: REST Bindings Version 1.0*. Edited by Craig Gemmill and Markus Jung. Latest version. http://docs.oasis-open.org/obix/obix-rest/v1.0/obix-rest-v1.0.html.

[RFC3986] Berners-Lee, T., Fielding, R., Masinter, L., “Uniform Resource Identifier (URI): Generic Syntax”, IETF RFC 3986, January 2005. http://www.ietf.org/rfc/rfc3986.txt.

[RFC6455] Fette, I Melnikoverners, A, *“The WebSocket Protocol”*, IETF RFC 6455, December 2011. http://www.ietf.org/rfc/rfc6455.txt.

[SOA-RM] *Reference Model for Service Oriented Architecture 1.0*, October 2006. OASIS Standard. http://docs.oasis-open.org/soa-rm/v1.0/soa-rm.pdf

1.3 Non-Normative References

–

2 WebSocket Binding

The WebSocket binding specifies a simple mapping of OBIX requests to WebSocket. After connecting to endpoint URL and switching to the WebSocket protocol, OBIX messages can be exchanged continuously.

2.1 Lobby

The WebSocket binding SHOULD be announced in the Lobby (see section 5.4.3 in **[OBIX]**) as follows:

```
<uri name="ws" displayName="WebSocket Binding" val="https://www.oasis-
open.org/committees/document.php?document_id=51536&wg_abbrev=obix"/>
```

2.2 Requests

The following table describes the mapping of OBIX request and its WebSocket equivalent. As WebSocket is a message-based protocol it cannot be mapped directly, but as OBIX messages contain naming the messages can be send also using this kind of protocol. For more details regarding the request flow see the sections below.

OBIX Request	WebSocket	Target
Read	After connect use obix:Read messages to read objects and the WatchService functionality to subscribe to objects and receive continuous updates of their state (which is using messages of type obix:Update)	Lobby (single point of WebSocket connection)
Write	Send an obix:Write message containing an obj	Any object with an href and writable=true, sent within an open WebSocket connection context
Invoke	Send an obix:Invoke message containing op element holding input parameters as children, expecting obix:Response message with corresponding request ID as response.	Any op object with an href (especially Watch), sent within an open WebSocket connection context
Delete	If an object has an delete operation defined this operation is used	Any object with delete operation

Table 2-1. OBIX Request Mapping

2.2.1 Connect request

The connect URL is the name or IP of the OBIX server prefixed by the WebSocket protocol, i.e. either “ws” or “wss” for a secure connection using TLS. If the server supports multiple encodings a client MAY request the encoding with the “encoding” parameter on connect (e.g. “wss://myhome/?encoding=json”), if not specified the server uses its default encoding (it is recommend to support XML encoding as default). The response send to client upon successful connection MUST be the Lobby object.

2.2.2 Request, Response and Update messages

To ensure that a request and response in the asynchronous message exchange of WebSocket is bound together, the concept of a request with a defined request ID (denoted as attribute `rid`) is introduced. A response to a request contains that specific request ID so that the client can match the request and response. If the server sends a message without the request and response context, it uses the `obix:Update` type to denote this case.

Following are the contract definitions of Read, Write, Invoke, Response and Update:

```
<obj href="obix:Read">
</obj>


<obj href="obix:Write">
</obj>

<obj href="obix:Invoke">
</obj>

<obj href="obix:Response">
</obj>

<obj href="obix:Update">
</obj>
```

For `obix:Read`, `obix:Write`, `obix:Invoke` and `obix:Response` there is a facet `rid` defined as `xs:int`, which MUST be included (e.g. the attribute can have the value rid=”1” to denote the request ID 1). The `obix:Request`, `obix:Response` and `obix:Update` objects MUST contain an `obj` or `list`. Here an example for a response object:

```
<obj is=”obix:Response” rid=”1”>
    <obj href="/device/BrightnessSensor" name="BrightnessSensor" location="Outside"
         is="example:Brightness" displayName="Brightness Outside">
            <real name="value" val="45.5" unit="obix:units/lux" />
    </obj>
</obj>
```

2.2.3 Watches

As WebSocket follows a message exchange pattern the REST-style messages of OBIX needs to be wrapped. For that, extensive use is made of the “Watch” concept. After a successful connection to the OBIX server, the client can add a “Watch” to subscribe to object changes. This is done using the make operation on the WatchService object. As long as the WebSocket connection is open, the server MAY push unsolicited updates via `obix:Update` messages to the client, as defined in section 12.2 in **[OBIX]**. This ensures that the client has a consistent state with the server.

2.2.4 Example Request Flow

The request and response flow below shows an example of WebSocket exchanges in the XML encoding style:

Client	Server

Client initiates action on its own timing Connect to WebSocket server: wss://myhome/	→

←	*Server sends message in response to connection from Client*
	Returns the Lobby:
	`<obj is="obix:Lobby">`
	`  <ref name="about" is="obix:About"/>`
	`  <op name="batch" in="obix:BatchIn" out="obix:BatchOut"/>`
	`  <ref name="watchService" is="obix:WatchService"/>`
	`  <ref name="device" href="/device/" is="example:Device"></ref>`
	`</obj>`

Table 2-2. Exchange 1: Client initiates connection with server for subsequent data exchange

Client	**Server**
Client sends message on its own timing Call WatchService.make operation: `<obj is="obix:Invoke" rid="1" href="/watchService/make" />`	→

←	*Server sends message in response to "watch service" message from Client*
	Returns the Watch (the lease time is not used):
	`<obj is="obix:Response" rid="1">`
	`  <obj is="obix:Watch" href="/watch/1">`
	`  <reltime name="lease" val="PT0S" />`
	`  </obj>`
	`</obj>`

Table 2-3. Exchange 2: Client sets up a watch service on the server

Client	**Server**

Client sends message on its own timing
Call Watch.add operation to add /device/:
`<obj is="obix:Invoke" rid="2" href="/watch/1/add">`
`  <obj is="obix:WatchIn">`
`  <list names="hrefs">`
`  <uri val="/device/" />`
`  </list>`
`  </obj>`
`</obj>`
	→

←	*Server sends message in response to "add device" message from Client*
	List devices:
	`<obj is="obix:Response" rid="2">`
	`  <list name="device" of="obj">`
	`  <obj href="/device/bathTemp" name="BathTemperature"`

	<pre>location="Bathroom" is="example:Temperature"
	displayName="Temperature Bathroom">
	<abstime name="Timestamp"
	val="2013-07-24T10:01:15.883+02:00">
	</abstime>
	<real name="ActualValue" val="28.2"
	unit="obix:units/celsius"
	displayName="ActualValue">
	</real>
	<bool name="Warm" val="true"
	displayName="Warm"></bool>
	</obj>
	<obj href="/device/bathLight" name="BathLight"
	location="Bathroom" is="example:Switch"
	displayName="Light Bathroom">
	<abstime name="Timestamp"
	val="2013-07-14T22:25:31.331+02:00">
	</abstime>
	<bool name="Status" val="false"
	displayName="Status" writeable="true">
	</bool>
	</obj>
	</list>
	</obj></pre>

99 *Table 2-4. Exchange 3: Client adds default devices to established watch service*

100

Client	**Server**

Client sends message on its own timing
Call Watch.remove operation to remove /device/:
<pre><obj is="obix:Invoke" rid="3" href="/watch/1/remove">
<obj is="obix:WatchIn">
<list names="hrefs">
<uri val="/device/" />
</list>
</obj>
</obj></pre>
	→

X	*Server does not send out any message upon reception of "watch remove" message from Client* Removed successfully, no response

101 *Table 2-5. Exchange 4: Client removes established default devices from an established watch service*

102

Client	**Server**

Client sends message on its own timing
Watch.add /device/bathTemp:
<pre><obj is="obix:Request" rid="4" href="/watch/1/add">
<obj is="obix:WatchIn">
<list names="hrefs"></pre>
	→

<table>
<tr><td colspan="2">

```
                    <uri val="/device/bathTemp" />
                </list>
            </obj>
        </obj>
```

</td><td></td></tr>
<tr><td>←</td><td>

Server sends message in response to “add device” message from Client

Send bathTemp information within the WatchOut object:

```
<obj is="obix:Response" rid="4">
    <obj is="obix:WatchOut" href="/watch/1">
            <list names="values">
                    <obj href="/device/bathTemp"
                         name="BathTemperature"
                         location="Bathroom"
                         is="example:Temperature"
                         displayName="Temperature Bathroom">
                       <abstime name="Timestamp"
                         val="2013-07-24T10:01:15.883+02:00">
                       </abstime>
                       <real name="ActualValue" val="28.2"
                           unit="obix:units/celsius"
                           displayName="ActualValue"></real>
                       <bool name="Warm" val="true"
                             displayName="Warm"></bool>
                    </obj>
            </list>
    </obj>
</obj>
```

</td></tr>
<tr><td colspan="2">

Client sends message on its own timing after having received the “device information” message from Server

Watch.pollChanges

```
<obj is="obix:Invoke" rid="5" href="/watch/1/pollChange">
</obj>
```

</td><td>→</td></tr>
<tr><td>←</td><td>

Server sends message in response to “watch poll changes” message from Client

Send empty response as the state is current

```
<obj is="obix:Response" rid="5">
</obj>
```

</td></tr>
<tr><td colspan="2">

Client sends message on its own timing

To keep the WebSocket session open send an empty WebSocket frame like e.g. “”

</td><td>→</td></tr>
<tr><td>X</td><td>

Server does not send out any message upon reception of empty WebSocket messages from Client

No response, just the session is kept open

</td></tr>
</table>

103 *Table 2-6. Exchange 5: Client adds first device with ability to watch for changes, but that device has no changes that*
104 *occur*

105

Client	**Server**

Client sends message on its own timing

Watch.add /device/kitchenTemp:

```
<obj is="obix:Request" rid="6" href="/watch/1/add">
    <obj is="obix:WatchIn">
            <list names="hrefs">
                    <uri val="/device/kitchenTemp" />
            </list>
    </obj>
</obj>
```

→

Server sends message in response to "add device" message from Client

Send kitchenTemp containing the current object:

```
<obj is="obix:Response" rid="6">
    <obj is="obix:WatchOut" href="/watch/1">
            <list names="values">
                    <obj href="/device/kitchenTemp"
                         name="KitchenTemperature"
                         location="Kitchen"
                         is="example:Temperature"
                         displayName="Temperature Kitchen">
                       <abstime name="Timestamp"
                        val="2013-07-24T10:01:15.883+02:00">
                       </abstime>
                       <real name="ActualValue" val="26.1"
                        unit="obix:units/celsius"
                        displayName="ActualValue"></real>
                       <bool name="Warm" val="true"
                        displayName="Warm"></bool>
                    </obj>
            </list>
    </obj>
</obj>
```

←

A period of two minutes has elapsed during this time slot, in the mean time only the empty frames are sent to keep the WebSocket connection open

Server sends message after 2 minutes from previous message

Send unsolicited update as an update from the temperature sensor was received:

```
<obj is="obix:Update">
    <obj is="obix:WatchOut" href="/watch/1">
            <list names="values">
                    <obj href="/device/kitchenTemp"
                         name="KitchenTemperature"
                         location="Kitchen"
                         is="example:Temperature"
                         displayName="Temperature Kitchen">
                       <abstime name="Timestamp"
                        val="2013-07-24T10:03:15.883+02:00">
                       </abstime>
                       <real name="ActualValue" val="26.2"
                        unit="obix:units/celsius"
                        displayName="ActualValue"></real>
                       <bool name="Warm" val="true"
                        displayName="Warm"></bool>
                    </obj>
```

←

	</list> </obj> </obj>

Table 2-7. Exchange 6: Client adds second device with ability to watch for changes, and that device has changes that occur

Client	Server
Client sends message on its own timing Update bathLight <obj is="obix:Request" rid="7"> <obj href="/device/bathLight" name="BathLight" location="Bathroom" is="example:Switch" displayName="Light Bathroom"> <bool name="Status" val="true" displayName="Status" writeable="true"></bool> </obj> </obj>	→
X	*Server does not send out any message upon reception of "update" messages from Client* No direct response as not watched

Table 2-8. Exchange 7: Client attempts to update a device that has not been setup for watching

Client	Server
Client sends message on its own timing Disconnect from wss://myhome/	→
	Server disconnects from Client

Table 2-9. Exchange 8: Client removes connection from Server

2.3 Security

Existing standards SHOULD be used when applicable for OBIX WebSocket implementations including:

- RFC 4346/2246 – The TLS Protocol (Transport Layer Security)

2.4 Localization

Servers SHOULD localize appropriate data based on the desired locale of the client agent. Localization SHOULD include the `display` and `displayName` attributes. The desired locale of the client SHOULD be determined through authentication. A suggested algorithm is to check if the authenticated user has a preferred locale configured in the server's user database.

Localization MAY include auto-conversion of units. For example if the authenticated user has configured a preferred unit system such as English versus Metric, then the server might attempt to convert values with an associated `unit` facet to the desired unit system.

3 Conformance

An implementation is conformant with this specification if it satisfies all of the MUST and REQUIRED level requirements defined herein for the functions implemented. Normative text within this specification takes precedence over normative outlines, which in turn take precedence over examples.

An implementation is a conforming OBIX Server supporting WebSocket if it meets the conditions described in Section 3.1. An implementation is a conforming OBIX Client supporting WebSocket if it meets the conditions described in Section 3.2. An implementation is a conforming OBIX Server supporting WebSocket and a conforming OBIX Client supporting WebSocket if it meets the conditions of both Sections 3.1 and 3.2.

3.1 Conditions for conforming OBIX Server supporting WebSocket

1. An OBIX server supporting WebSocket MUST conform to an OBIX server as defined in **[OBIX]**.
2. An OBIX server supporting WebSocket MUST accept WebSocket connections and MUST return the Lobby object on successful connection.
3. An OBIX server supporting WebSocket MUST support the make operation of the obix:WatchService object.
4. An OBIX server supporting WebSocket MUST support the obix:Request, obix:Response and obix:Update contracts and return the request id “rid” within the obix:Response object.

3.2 Conditions for conforming OBIX Client supporting WebSocket

1. An OBIX client supporting WebSocket must conform to an OBIX client as defined in **[OBIX]**.
2. A conformant OBIX client supporting WebSocket must support WebSocket connections and the request flow as stated in Section 2.1.
3. A conformant implementation MUST generate request IDs for each obix:Request message

Appendix A. Acknowledgments

The following individuals have participated in the creation of this specification and are gratefully acknowledged:

Participants:

Gareth Johnson, Tridium Inc.
Craig Gemmill, Tridium Inc.
Ludo Bertsch, CABA
Markus Jung, University of Vienna

Appendix B. Revision History

Revision	Date	Editor	Changes Made
WD01	1-Aug-2013	Matthias Hub	Initial submission
WD02	8-Aug-2013	Toby Considine	Moved to standard template, added some normative references
WD03	13-Aug-2013	Matthias Hub	Incorporated review comments by Gareth Johnson
WD04	15-Oct-2013	Matthias Hub	Incorporated review comments from TC: removed separate watch concept – instead re-using standard watch concept, added definition of Watch properties
WD05	18-Oct-2013	Matthias Hub	Incorporated Craig Gemmill input to Watches
WD06	29-Oct-2013	Ludo Bertsch	Improved example in Section 2.1.5
WD07	30-Oct-2013	Matthias Hub	Updated Terminology section Added bufferHandling property to the Watch Changed request / response flow style in the example Updated conformance section for different naming and to refer to the core spec
WD08	18-Nov-2013	Matthias Hub	Introduced obix:Read, obix:Write and obix:Invoke as message type similar to the SOAP binding Clarified FIFO / LIFO means that messages are dropped Adapted request / response flow style
WD09	25-Nov-2013	Matthias Hub	Added definition of obix:Read, obix:Write and obix:Invoke Updated the example flow to use obix:Read, obix:Write and obix:Invoke Using "example" prefix instead of "gateway"
WD10	16-Dec-2013	Matthias Hub	Added Lobby definition section Removed duplicate Watches definition as they are moved into core Fixed spelling (OBIX-85) Updated table titles (OBIX-86)