[MS-WSPE]:

WebSocket Protocol Extensions

Intellectual Property Rights Notice for Open Specifications Documentation

- **Technical Documentation.** Microsoft publishes Open Specifications documentation for protocols, file formats, languages, standards as well as overviews of the interaction among each of these technologies.
- **Copyrights**. This documentation is covered by Microsoft copyrights. Regardless of any other terms that are contained in the terms of use for the Microsoft website that hosts this documentation, you may make copies of it in order to develop implementations of the technologies described in the Open Specifications and may distribute portions of it in your implementations using these technologies or your documentation as necessary to properly document the implementation. You may also distribute in your implementation, with or without modification, any schema, IDL's, or code samples that are included in the documentation. This permission also applies to any documents that are referenced in the Open Specifications.
- **No Trade Secrets**. Microsoft does not claim any trade secret rights in this documentation.
- **Patents**. Microsoft has patents that may cover your implementations of the technologies described in the Open Specifications. Neither this notice nor Microsoft's delivery of the documentation grants any licenses under those or any other Microsoft patents. However, a given Open Specification may be covered by Microsoft Open Specification Promise or the Community Promise. If you would prefer a written license, or if the technologies described in the Open Specifications are not covered by the Open Specifications Promise or Community Promise, as applicable, patent licenses are available by contacting iplg@microsoft.com.
- **Trademarks**. The names of companies and products contained in this documentation may be covered by trademarks or similar intellectual property rights. This notice does not grant any licenses under those rights. For a list of Microsoft trademarks, visit www.microsoft.com/trademarks.
- **Fictitious Names**. The example companies, organizations, products, domain names, e-mail addresses, logos, people, places, and events depicted in this documentation are fictitious. No association with any real company, organization, product, domain name, email address, logo, person, place, or event is intended or should be inferred.

Reservation of Rights. All other rights are reserved, and this notice does not grant any rights other than specifically described above, whether by implication, estoppel, or otherwise.

Tools. The Open Specifications do not require the use of Microsoft programming tools or programming environments in order for you to develop an implementation. If you have access to Microsoft programming tools and environments you are free to take advantage of them. Certain Open Specifications are intended for use in conjunction with publicly available standard specifications and network programming art, and assumes that the reader either is familiar with the aforementioned material or has immediate access to it.

Revision Summary

Date	Revision History	Revision Class	Comments
12/16/2011	1.0	New	Released new document.
3/30/2012	1.0	None	No changes to the meaning, language, or formatting of the technical content.
7/12/2012	2.0	Major	Significantly changed the technical content.
10/25/2012	2.0	None	No changes to the meaning, language, or formatting of the technical content.
1/31/2013	2.0	None	Updated the 11/14/2013 release information.
8/8/2013	3.0	Major	Significantly changed the technical content.
11/14/2013	3.0	None	No changes to the meaning, language, or formatting of the technical content.
2/13/2014	3.0	None	No changes to the meaning, language, or formatting of the technical content.
5/15/2014	3.0	None	No changes to the meaning, language, or formatting of the technical content.
6/30/2015	4.0	Major	Significantly changed the technical content.

Table of Contents

1 Introduction

The WebSocket Protocol is an Internet Engineering Task Force (IETF) standard protocol designed to allow asynchronous bidirectional communication between hosts over a network that might only provide connectivity via web proxies.

This document specifies extensions to the WebSocket Protocol.

Sections 1.8, 2, and 3 of this specification are normative and can contain the terms MAY, SHOULD, MUST, MUST NOT, and SHOULD NOT as defined in [RFC2119]. Sections 1.5 and 1.9 are also normative but do not contain those terms. All other sections and examples in this specification are informative.

1.1 Glossary

The following terms are specific to this document:

endpoint: A resource that can be addressed by an endpoint reference.

masking: The process of XOR'ing a specified mask to obfuscate the content of a data message.

sandbox: A security mechanism used to constrain the actions that a program can take. A sandbox restricts a program to a defined set of privileges and actions that reduce the likelihood that the program might damage the system hosting the program.

sandboxed: Deployed into a sandbox.

MAY, SHOULD, MUST, SHOULD NOT, MUST NOT: These terms (in all caps) are used as defined in [RFC2119]. All statements of optional behavior use either MAY, SHOULD, or SHOULD NOT.

1.2 References

Links to a document in the Microsoft Open Specifications library point to the correct section in the most recently published version of the referenced document. However, because individual documents in the library are not updated at the same time, the section numbers in the documents may not match. You can confirm the correct section numbering by checking the Errata.

1.2.1 Normative References

We conduct frequent surveys of the normative references to assure their continued availability. If you have any issue with finding a normative reference, please contact dochelp@microsoft.com. We will assist you in finding the relevant information.

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

[RFC2616] Fielding, R., Gettys, J., Mogul, J., et al., "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, June 1999, http://www.rfc-editor.org/rfc/rfc2616.txt

[RFC6455] Fette, I., and Melnikov, A., "The WebSocket Protocol", RFC 6455, December 2011, http://www.ietf.org/rfc/rfc6455.txt

1.2.2 Informative References

None.

1.3 Overview

The WebSocket Protocol [RFC6455] creates an asynchronous, bidirectional communication channel that works across existing network intermediaries such as web proxies and firewalls. A client uses HTTP [RFC2616] to communicate with a server and then both sides switch to using the WebSocket Protocol over the underlying protocol on which HTTP is layered, such as TCP or SSL over TCP. The goal is to first use HTTP to traverse network intermediaries and then use the established end-to-end underlying TCP/SSL channel for bidirectional application communication.

The WebSocket Protocol requires that all frames are **masked** by a random security key to avoid possible confusion with the HTTP protocol by intermediaries. Some intermediaries will continue to parse HTTP requests even if the beginning of the byte stream does not match the HTTP grammar. In the WebSocket Protocol, such intermediaries skip the frame header and interpret the application payload as an HTTP request. Such a deficiency allows an attacker to inject bad data as discussed in [RFC6455] section 10.3 by sending specially crafted HTTP requests as data through the WebSocket Protocol. Masking prevents such a deficiency.

However, masking can have a significant performance impact. If the WebSocket Protocol is used in a controlled environment, such as within an enterprise network where there are no intermediaries or where intermediaries recognize the WebSocket Protocol, masking might not be needed. Turning off masking in such cases thus has a positive impact on the performance.

If the WebSocket Protocol is used by a **sandboxed** application, such as running in a browser where the **sandbox** only allows the application to communicate over HTTP, the cache-poisoning attack can have serious consequences if a malicious application can bypass the restrictions imposed by the sandbox. However, non-sandboxed applications that can use TCP directly can already perform the same actions, and therefore, disabling masking does not introduce additional risk.

1.4 Relationship to Other Protocols

The WebSocket Protocol Extensions make no modifications to the protocol relationships defined in [RFC6455] sections 1.7 and 1.9.

1.5 Prerequisites/Preconditions

It is assumed by the implementation that the higher-layer protocol or application recognizes whether masking is required prior to using this extension.

1.6 Applicability Statement

These extensions are not applicable to sandboxed environments that only allow web access, such as applications running in a browser.

These extensions are not applicable to the wide-area Internet.

These extensions are only applicable to scenarios where an application is permitted to use TCP and where the intermediaries are well known (such as within a corporate intranet), and security is provided by another layer, such as SSL.

1.7 Versioning and Capability Negotiation

None.

1.8 Vendor-Extensible Fields

None.

1.9 Standards Assignments

None.

2 Messages

2.1 Transport

Message transport for the WebSocket Protocol Extensions is as defined in [RFC6455].

2.2 Message Syntax

Message syntax for the WebSocket Protocol Extensions is as defined in [RFC6455] section 5.

3 Protocol Details

3.1 Client Details

3.1.1 Abstract Data Model

This section describes a conceptual model of possible data organization that an implementation maintains to participate in this protocol. The described organization is provided to facilitate the explanation of how the protocol behaves. This document does not mandate that implementations adhere to this model as long as their external behavior is consistent with that described in this document.

set-mask-to-zero: When set to false, this flag indicates that the mask is to be derived from a strong source of entropy as defined in [RFC6455] section 5.3. When set to true, the mask is to be set to zero.

3.1.2 Timers

None.

3.1.3 Initialization

When a WebSocket connection is initialized, the higher-layer protocol or application specifies the required value for the **set-mask-to-zero** ADM element and the client MUST set **set-mask-to-zero** to the specified value. After the connection is initialized, the value of **set-mask-to-zero** never changes.

3.1.4 Higher-Layer Triggered Events

To send a frame, data MUST be framed according to the rules specified in [RFC6455] section 5. If the value of the **set-mask-to-zero** ADM element is false, the behavior is unchanged from that specified in [RFC6455] section 5.3. If the value of **set-mask-to-zero** is true, then the **Masking-key** field (as defined in [RFC6455] section 4.2) MUST be set to zero instead of being derived from a strong source of entropy.

3.1.5 Message Processing Events and Sequencing Rules

Message processing events and sequencing rules for the WebSocket Protocol Extensions are as defined in [RFC6455].

3.1.6 Timer Events

None.

3.1.7 Other Local Events

None.

3.2 Server Details

3.2.1 Abstract Data Model

This section describes a conceptual model of possible data organization that an implementation maintains to participate in this protocol. The described organization is provided to facilitate the explanation of how the protocol behaves. This document does not mandate that implementations

adhere to this model as long as their external behavior is consistent with that described in this document.

accept-unmasked-frame: When set to false, this flag indicates that unmasked frames are to cause the connection to be aborted. When set to true, the server is to accept frames regardless of whether they are masked.

3.2.2 Timers

None.

3.2.3 Initialization

When a WebSocket connection is initialized, the higher-layer protocol or application specifies the required value for the **accept-unmasked-frame** ADM element and the server MUST set **accept-unmasked-frame** to the specified value. After the connection is initialized, the value of **accept-unmasked-frame** never changes.

3.2.4 Higher-Layer Triggered Events

Higher-layer triggered events for the WebSocket Protocol Extensions are as defined in [RFC6455].

3.2.5 Message Processing Events and Sequencing Rules

The server MUST parse the received frame and extract the value of the **frame-masked** field ([RFC6455] sections 5.2) as specified in [RFC6455] section 45.3. If the value of the **accept-unmasked-frame** ADM element is false, then the behavior is unchanged from that specified in [RFC6455]. If the value of **accept-unmasked-frame** is true, the frame MUST be accepted by the server even if the **Mask bit** ([RFC6455] section 4.2) is set to zero.

3.2.6 Timer Events

None.

3.2.7 Other Local Events

None.

4 Protocol Examples

In the following example, an application is running in a controlled environment and performance is a crucial factor.

Because the application is running in a safe environment, the administrator disables masking as follows to improve performance:

1. The administrator changes the configuration of the client and server applications to disable masking.
2. When the server application initializes the WebSocket Protocol [RFC6455] for a given **endpoint**, the WebSocket Protocol server is instructed by the application to disable masking for the endpoint. The server sets the value of the **accept-unmasked-frame** ADM element to TRUE.
3. When the client application initializes the WebSocket Protocol for a given connection, the WebSocket Protocol client is instructed by the application to disable masking on the connection. The client sets the value of the **set-mask-to-zero** ADM element to TRUE.
4. When the client application sends data, the WebSocket Protocol client uses a masking-key of zero. Doing an XOR with zeroes results in the data being left unmodified.
5. When the WebSocket server receives such data, the message is delivered to the server application although the data was not masked.

5 Security

5.1 Security Considerations for Implementers

It is not appropriate to expose these extensions in sandboxed environments that only allow web access, such as applications running in a browser, since doing so could allow such applications to bypass the security restrictions of the sandbox.

It is appropriate to allow use of this extension only by higher-layer protocols and applications that are already permitted to use TCP.

5.2 Index of Security Parameters

Security parameter	Section
set-mask-to-zero	3.1.1
accept-unmasked-frame	3.2.1

6 Appendix A: Product Behavior

The information in this specification is applicable to the following Microsoft products or supplemental software. References to product versions include released service packs.

Note: Some of the information in this section is subject to change because it applies to an unreleased, preliminary version of the Windows Server operating system, and thus may differ from the final version of the server software when released. All behavior notes that pertain to the unreleased, preliminary version of the Windows Server operating system contain specific references to Windows Server 2016 Technical Preview as an aid to the reader.

- Windows 8 operating system
- Windows Server 2012 operating system
- Windows 8.1 operating system
- Windows Server 2012 R2 operating system
- Windows 10 operating system
- Windows Server 2016 Technical Preview operating system

Exceptions, if any, are noted below. If a service pack or Quick Fix Engineering (QFE) number appears with the product version, behavior changed in that service pack or QFE. The new behavior also applies to subsequent service packs of the product unless otherwise specified. If a product edition appears with the product version, behavior is different in that product edition.

Unless otherwise specified, any statement of optional behavior in this specification that is prescribed using the terms SHOULD or SHOULD NOT implies product behavior in accordance with the SHOULD or SHOULD NOT prescription. Unless otherwise specified, the term MAY implies that the product does not follow the prescription.

7 Change Tracking

This section identifies changes that were made to this document since the last release. Changes are classified as New, Major, Minor, Editorial, or No change.

The revision class **New** means that a new document is being released.

The revision class **Major** means that the technical content in the document was significantly revised. Major changes affect protocol interoperability or implementation. Examples of major changes are:

- A document revision that incorporates changes to interoperability requirements or functionality.
- The removal of a document from the documentation set.

The revision class **Minor** means that the meaning of the technical content was clarified. Minor changes do not affect protocol interoperability or implementation. Examples of minor changes are updates to clarify ambiguity at the sentence, paragraph, or table level.

The revision class **Editorial** means that the formatting in the technical content was changed. Editorial changes apply to grammatical, formatting, and style issues.

The revision class **No change** means that no new technical changes were introduced. Minor editorial and formatting changes may have been made, but the technical content of the document is identical to the last released version.

Major and minor changes can be described further using the following change types:

- New content added.
- Content updated.
- Content removed.
- New product behavior note added.
- Product behavior note updated.
- Product behavior note removed.
- New protocol syntax added.
- Protocol syntax updated.
- Protocol syntax removed.
- New content added due to protocol revision.
- Content updated due to protocol revision.
- Content removed due to protocol revision.
- New protocol syntax added due to protocol revision.
- Protocol syntax updated due to protocol revision.
- Protocol syntax removed due to protocol revision.
- Obsolete document removed.

Editorial changes are always classified with the change type **Editorially updated**.

Some important terms used in the change type descriptions are defined as follows:

- **Protocol syntax** refers to data elements (such as packets, structures, enumerations, and methods) as well as interfaces.
- **Protocol revision** refers to changes made to a protocol that affect the bits that are sent over the wire.

The changes made to this document are listed in the following table. For more information, please contact dochelp@microsoft.com.

Section	Tracking number (if applicable) and description	Major change (Y or N)	Change type
6 Appendix A: Product Behavior	Added Windows 10 to applicability list.	Y	Content update.

8 Index

T

V