Standards Track Work Product

Specification for Transfer of OpenC2 Messages via HTTPS Version 1.1

Committee Specification 01

30 November 2021

This stage:

https://docs.oasis-open.org/openc2/open-impl-https/v1.1/cs01/open-impl-https-v1.1-cs01.md (Authoritative)
https://docs.oasis-open.org/openc2/open-impl-https/v1.1/cs01/open-impl-https-v1.1-cs01.html
https://docs.oasis-open.org/openc2/open-impl-https/v1.1/cs01/open-impl-https-v1.1-cs01.pdf

Previous stage:

https://docs.oasis-open.org/openc2/open-impl-https/v1.1/csd01/open-impl-https-v1.1-csd01.md (Authoritative)
https://docs.oasis-open.org/openc2/open-impl-https/v1.1/csd01/open-impl-https-v1.1-csd01.html
https://docs.oasis-open.org/openc2/open-impl-https/v1.1/csd01/open-impl-https-v1.1-csd01.pdf

Latest stage:

https://docs.oasis-open.org/openc2/open-impl-https/v1.1/open-impl-https-v1.1.md (Authoritative)
https://docs.oasis-open.org/openc2/open-impl-https/v1.1/open-impl-https-v1.1.html
https://docs.oasis-open.org/openc2/open-impl-https/v1.1/open-impl-https-v1.1.pdf

Technical Committee:

OASIS Open Command and Control (OpenC2) TC

Chair:

Duncan Sparrell (duncan@sfractal.com), sFractal Consulting LLC

Editor:

David Lemire (david.lemire@hii-tsd.com), National Security Agency

Related work:

This specification is related to:

Open Command and Control (OpenC2) Language Specification Version 1.0. Edited by Jason Romano and Duncan Sparrell. Latest stage: https://docs.oasis-open.org/openc2/oc2ls/v1.0/oc2ls-v1.0.html.

Open Command and Control (OpenC2) Profile for Stateless Packet Filtering Version 1.0. Edited by Joe Brule, Duncan Sparrell and Alex Everett. Latest stage: https://docs.oasis-open.org/openc2/oc2slpf/v1.0/oc2slpf-v1.0.html.

Abstract:

Open Command and Control (OpenC2) is a concise and extensible language to enable the command and control of cyber defense components, subsystems and/or systems in a manner that is agnostic of the underlying products, technologies, transport mechanisms or other aspects of the implementation. HTTP over TLS is a widely deployed transfer protocol that provides an authenticated, ordered, lossless delivery of uniquely-identified messages. This document specifies the use of HTTP over TLS as a transfer mechanism for OpenC2 Messages. A Testing conformance target is provided to support interoperability testing without security mechanisms.

Status:

This document was last revised or approved by the OASIS Open Command and Control (OpenC2) TC on the above date. The level of approval is also listed above. Check the "Latest stage" location noted above for possible later revisions of this document. Any other numbered Versions and other technical work produced by the Technical Committee (TC) are listed at https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=openc2#technical.

TC members should send comments on this specification to the TC's email list. Others should send comments to the TC's public comment list, after subscribing to it by following the instructions at the "Send A Comment" button on the TC's web page at https://www.oasis-open.org/committees/openc2/.

This specification is provided under the Non-Assertion Mode of the OASIS IPR Policy, the mode chosen when the Technical Committee was established. 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 TC's web page (https://www.oasis-open.org/committees/openc2/ipr.php).

Note that any machine-readable content (Computer Language Definitions) declared Normative for this Work Product is provided in separate plain text files. In the event of a discrepancy between any such plain text file and display content in the Work Product's prose narrative document(s), the content in the separate plain text file prevails.

Key words:

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] and [RFC8174] when, and only when, they appear in all capitals, as shown here.

Citation format:

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

[OpenC2-HTTPS-v1.1]

Specification for Transfer of OpenC2 Messages via HTTPS Version 1.1. Edited by David Lemire. 30 November 2021. OASIS Committee Specification 01. https://docs.oasis-open.org/openc2/open-impl-https/v1.1/cs01/open-impl-https-v1.1-cs01.html. Latest stage: https://docs.oasis-open.org/openc2/open-impl-https/v1.1/open-impl-https-v1.1.html.

Notices

Table of Contents

1 Introduction
- 1.1 Changes from Earlier Versions
- 1.2 Glossary
  - 1.2.1 Definition of Terms
  - 1.2.2 Acronyms and Abbreviations
  - 1.2.3 Document Conventions
    - 1.2.3.1 Naming Conventions
    - 1.2.3.2 Font Colors and Style

2 Operating Model

3 Protocol Mappings
- 3.1 Layering Overview
- 3.2 General Requirements
  - 3.2.1 HTTP Usage
  - 3.2.2 URI Scheme
  - 3.2.3 TLS Usage
  - 3.2.4 Authentication
- 3.3 OpenC2 Message Format
  - 3.3.1 Content Type and Serialization
  - 3.3.2 OpenC2 Message Structure
  - 3.3.3 Message Identification
- 3.4 OpenC2 Consumer as HTTP/TLS Server

4 Conformance
- 4.1 Conformance Targets
- 4.2 Conformance Requirements
  - 4.2.1 Testing Target Conformance Requirements
  - 4.2.2 Operations Target Conformance Requirements

Appendix A. References
- A.1 Normative References
- A.2 Informative References

Appendix B. Safety, Security and Privacy Considerations

Appendix C. Acknowledgements
- C.1 Special Thanks
- C.2 Participants

Appendix D. Revision History

Appendix E. Examples
- E.1 HTTP Request / Response Examples: Consumer as HTTP Server
  - E.1.1 Producer HTTP POST with OpenC2 Command
  - E.1.2 Consumer HTTP Response with OpenC2 Response

Appendix F. Notices

1 Introduction

The content in this section is non-normative, except where it is marked normative.

OpenC2 is a suite of specifications that enables command and control of cyber defense systems and components. OpenC2 typically uses a request-response paradigm where a *Command* is encoded by a *Producer* (managing application) and transferred to a *Consumer* (managed device or virtualized function) using a secure transfer protocol, and the Consumer can respond with status and any requested information.

This document specifies the use of Hypertext Transfer Protocol (HTTP) over Transport Layer Security (TLS) as a transfer mechanism for OpenC2 Messages; this HTTP/TLS layering is typically referred to as HTTPS [RFC2818]. As described in [RFC3205], HTTP has become a common "substrate" for information transfer for other application-level protocols. The broad availability of HTTP makes it a useful option for OpenC2 message transport in support of prototyping, interoperability testing, and for operational use in environments where appropriate security protections can be provided.

Similarly, TLS is a mature and widely-used protocol for securing information transfers in TCP/IP network environments. This specification provides guidance to the OpenC2 implementation community when utilizing HTTPS for OpenC2 message transport. It includes guidance for selection of TLS versions and options suitable for use with OpenC2. In addition, a Testing conformance target is defined to support interoperability testing without security mechanisms.

This OpenC2 over HTTPS transfer specification is suitable for operational environments where:

- Connectivity between OpenC2 Producers and OpenC2 Consumers is:
  - Highly available, with infrequent network outages
  - Of sufficient bandwidth that no appreciable message delays or dropped packets are experienced
- In-band negotiation of a connection initiated by either Producer or Consumer is possible without requiring an out-of-band signaling network.
- The overhead of HTTPS is acceptable.

An additional application for this transfer specification is interoperability test environments.

1.1 Changes from Earlier Versions

Changes since v1.0, CS01:

- Defined a standard Uniform Resource Identifier (URI) scheme and added a corresponding conformance requirement.
- Specified the use of the atomic OpenC2 message structure, updated content-type accordingly, and adjusted examples to match
- Testing and Operations conformance targets and requirements have been defined to support both secure (HTTPS) and non-secure (HTTP) message transfers with a single specification
- Added new section with requirements for message correlation
- Restructured to use the updated OASIS work products outline
- Other minor changes and corrections have been incorporated based on plug fest and interoperability testing experiences

1.2 Glossary

1.2.1 Definition of Terms

This section is normative.

- **Action**: The task or activity to be performed (e.g., 'deny').
- **Actuator**: The function performed by the Consumer that executes the Command (e.g., 'Stateless Packet Filtering').
- **Argument**: A property of a Command that provides additional information on how to perform the Command, such as date/time, periodicity, duration, etc.
- **Command**: A Message defined by an Action-Target pair that is sent from a Producer and received by a Consumer.
- **Consumer**: A managed device / application that receives Commands. Note that a single device / application can have both Consumer and Producer capabilities.
- **Message**: A content- and transport-independent set of elements conveyed between Consumers and Producers.
- **Producer**: A manager application that sends Commands.
- **Response**: A Message from a Consumer to a Producer acknowledging a Command or returning the requested resources or status to a previously received Command.

- **Specifier**: A property or field that identifies a Target or Actuator to some level of precision.
- **Target**: The object of the Action, i.e., the Action is performed on the Target (e.g., IP Address).

1.2.2 Acronyms and Abbreviations

This section is non-normative.

Term	Expansion
0-RTT	Zero Round Trip Time
API	Application Programming Interface
HTTP	Hypertext Transfer Protocol
HTTPS	HTTP over TLS
IETF	Internet Engineering Task Force
IPR	Intellectual Property Rights
JSON	JavaScript Object Notation
RFC	Request For Comment
TC	Technical Committee
TCP	Transmission Control Protocol
TLS	Transport Layer Security

1.2.3 Document Conventions

1.2.3.1 Naming Conventions

- [RFC2119]/[RFC8174] key words (see Section 1.2) are in all uppercase.
- All property names and literals are in lowercase, except when referencing canonical names defined in another standard (e.g., literal values from an IANA registry).
- Words in property names are separated with an underscore (_), while words in string enumerations and type names are separated with a hyphen (-).
- The term "hyphen" used here refers to the ASCII hyphen or minus character, which in Unicode is "hyphen-minus", U+002D.

1.2.3.2 Font Colors and Style

The following color, font and font style conventions are used in this document:

- A fixed width font is used for all type names, property names, and literals.
- All examples in this document are expressed in JSON. They are in fixed width font, with straight quotes, black text and a light shaded background, and 2-space indentation. JSON examples in this document are representations of JSON Objects. They should not be interpreted as string literals. The ordering of object keys is insignificant. Whitespace before or after JSON structural characters in the examples are insignificant [RFC8259].
- Parts of the example may be omitted for conciseness and clarity. These omitted parts are denoted with the ellipses ("...").

Example:

```
HTTP/1.1 200 OK
Date: Wed, 19 Dec 2018 22:15:00 GMT
Content-type: application/openc2+json;version=1.0
{
  "headers": {
    "request_id": "0e3d8fa8-0bae-4055-a341-9c97b4f328f7",
    "created": 1545257700000,
    "from": "...",
    "to": [
```



```
      "..."
    ]
  },
  "body": {
    "openc2": {
      "request": {
        "action": "deny",
        "target": {
          "file": {
            "hashes": {
              "sha256": "22fe72a34f006ea67d26bb7004e2b6941b5c3953d43ae7ec24d41b1a928a6973"
            }
          }
        }
      }
    }
  }
}
```


2 Operating Model

This section is non-normative.

This section describes the operating model used when transferring OpenC2 Commands and Responses using HTTPS.

> NOTE: This specification provides for both HTTP and HTTPS message transfer. For convenience HTTPS is used as the general term for the transfer protocol throughout this specification. For non-secure operations using the Testing coformance target "HTTPS" should be read as equivalent to "HTTP".

Each endpoint of an OpenC2-over-HTTPS interaction has both an OpenC2 role and an HTTP function. OpenC2 Consumers will be HTTP listeners (i.e., servers) so that they can accept connections and receive unsolicited Commands from OpenC2 Producers. OpenC2 Producers act as HTTP clients and transmit Commands to Consumers.

Figure 2 illustrates the Producer / Consumer interactions. A Producer that needs to send OpenC2 Commands initiates a TCP connection to the Consumer. Once the TCP connection is created, a TLS session is initiated to authenticate the endpoints and provide connection confidentiality. The Producer can then issue OpenC2 Commands by sending HTTP requests using the POST method through the TLS connection, with Consumer OpenC2 Responses returned in the HTTP response.

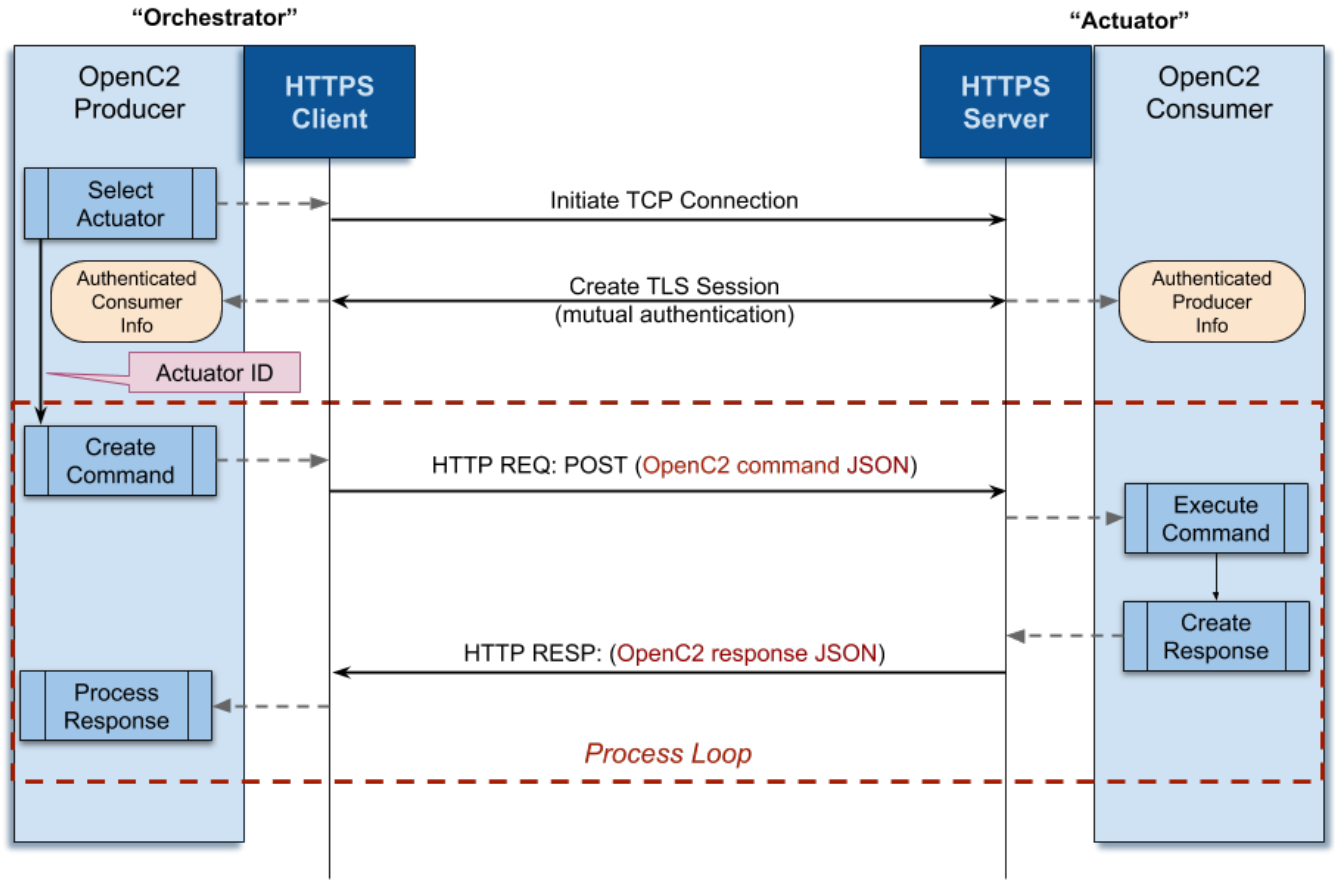

Figure 2 -- OpenC2 Producer / Consumer Interactions

3 Protocol Mappings

The section defines the requirements for using HTTP and TLS with OpenC2, including general requirements and protocol mappings for the operating configuration described in Section 2.

3.1 Layering Overview

When using HTTPS for OpenC2 Message transfer, the layering model is:

Layer	Description
OpenC2 Content	The *OpenC2 Language Specification* defines the overall OpenC2 language, and the *Actuator Profile(s)* implemented by any particular endpoint scopes the OpenC2 actions, targets, arguments, and specifiers that apply when commanding that type of Actuator.
Serialization	Serialization converts internal representations of OpenC2 content into a form that can be transmitted and received. The OpenC2 default serialization is JSON.
Message	The message layer provides a content- and transport-independent mechanism for conveying requests and responses. A Message consists of serialized content plus a set of metadata items such as content type and version, sender, timestamp, and correlation id. This layer maps the transport-independent definition of each message element to its transport-specific on-the-wire representation.
HTTP	The HTTP layer is responsible for conveying request and response Messages, as described in this specification.
TLS	The TLS layer is responsible for authentication of connection endpoints and confidentiality and integrity of transferred Messages.
Lower Layer Transport	The lower protocol layers are responsible for end-to-end delivery of Messages. TCP/IP is the most common suite of lower layer protocols used with HTTPS.

3.2 General Requirements

This section defines general requirements for using HTTPS to transfer OpenC2 messages, including HTTP method, message serialization, and TLS requirements.

3.2.1 HTTP Usage

OpenC2 Consumers MUST be HTTP listeners to implement the operating model described in Section 2. OpenC2 Consumers acting as HTTP listeners SHOULD listen on:

- port 80, the registered port for HTTP, when used for testing.
- port 443, the registered port for HTTPS, when used for operations.

OpenC2 endpoints MUST implement all HTTP functionality required by this specification in accordance with HTTP/1.1 ([RFC7230], *et. al.*). As described in Table 3-1, the only HTTP request method utilized is POST.

Utilized?	HTTP Methods
Yes	POST
No	GET, HEAD, PUT, DELETE, CONNECT, OPTIONS, TRACE

Table 3-1: HTTP Method Use

Each HTTP message body MUST contain only a single OpenC2 Command or Response message. This does not preclude a Producer and Consumer exchanging multiple OpenC2 Command and Response Messages over time during a single HTTPS session. Depending on the set-up, a server and client can have multiple connections, but a sequence of OpenC2 interactions can spread over multiple connections. In some cases the connection may drop, but the session remains open (in an idle state).

All HTTP request and response messages containing OpenC2 payloads SHOULD include the "Cache-control:" header with a value of "no-cache".

3.2.2 URI Scheme

When transferring OpenC2 Command messages over HTTPS, the Uniform Resource Identifier (URI) structure in Table 3-2 MUST be employed:

Scheme	Address	Path
`https://`	`[Consumer Address]`	`/.well-known/openc2`

Table 3-2: OpenC2 HTTPS URI Structure

This path format conforms to the the IETF's "`/.well-known/`" path prefix for well-known locations, as defined in [RFC8615]

OpenC2 Producers sending Command messages MUST POST those messages to the URI defined in Table 3-2.

OpenC2 Consumers acting as HTTP listeners MUST accept Command messages POSTed to the URI defined in Table 3-2.

3.2.3 TLS Usage

HTTPS, the transmission of HTTP over TLS, is specified in Section 2 of [RFC2818]. OpenC2 endpoints MUST accept TLS version 1.2 [RFC5246] connections or higher for confidentiality, identification, and authentication when sending OpenC2 Messages over HTTPS, and SHOULD accept TLS Version 1.3 [RFC8446] or higher connections.

OpenC2 endpoints MUST NOT support any version of TLS prior to v1.2 and MUST NOT support any version of Secure Sockets Layer (SSL).

The implementation and use of TLS SHOULD align with the best currently available security guidance, such as that provided in [RFC7525]/BCP 195.

The TLS session MUST use non-NULL ciphersuites for authentication, integrity, and confidentiality. Sessions MAY be renegotiated within these constraints.

OpenC2 endpoints supporting TLS v1.2 MUST NOT use any of the deprecated ciphersuites identified in Appendix A of [RFC7540].

OpenC2 endpoints supporting TLS 1.3 MUST NOT implement zero round trip time resumption (0-RTT).

3.2.4 Authentication

Each participant in an OpenC2 communication MUST authenticate the other participant.

3.3 OpenC2 Message Format

This section describes how OpenC2 messages are represented in HTTP requests.

3.3.1 Content Type and Serialization

While the OpenC2 language is agnostic of serialization, when transferring OpenC2 Messages over HTTP/TLS as described in this specification, the default JSON serialization described in [OpenC2-Lang-v1.0] MUST be supported.

When OpenC2 Messages are sent over HTTPS using the default JSON serialization the message MUST specify the content type "`application/openc2+json;version=1.0`".

3.3.2 OpenC2 Message Structure

OpenC2 messages transferred using HTTPS utilize the `OpenC2-Message` structure defined in Section 3.2 of OpenC2-Lang-v1.0.

```
Message = Record
 1 headers       Headers optional
 2 body          Body
 3 signature     String optional

Headers = Map{1..*}
 1 request_id    String optional
 2 created       ls:Date-Time optional
 3 from          String optional
```



```
 4 to            String [0..*]

Body = Choice
 1 openc2        OpenC2-Content

OpenC2-Content = Choice
 1 request       OpenC2-Command
 2 response      OpenC2-Response
 3 notification  OpenC2-Event
```

Since HTTPS provides a point-to-point connection between an OpenC2 Producer and Consumer, the message `from` and `to` fields are not needed for addressing. However, OpenC2 Producers and Consumers MAY populate the message headers `from` and `to` fields.

3.3.3 Message Identification

OpenC2 Producers and Consumers need the ability to identify requests and corresponding responses in order to correlate activity. In addition, it is common for network monitoring tools to similarly track HTTP requests and responses. The OpenC2 message format includes a header field, `request_id` that is the OpenC2 mechanism for message correlation. The extension header `X-Request-ID` is commonly used in HTTP traffic for the same purpose. This specification supports the use of these mechanisms in parallel, with the following requirements:

- OpenC2 Producers MUST generate a unique identifying value for OpenC2 Command (i.e., request) messages, and SHOULD use an [RFC4122] UUID_v4 identifier for that purpose.
- OpenC2 Producers MUST populate either the HTTP `X-Request-ID` header or the OpenC2 message header `request_id` field with the Command message's unique identifying value, and SHOULD populate both locations.
- OpenC2 Consumers MUST return the Command message's unique identifying value in corresponding Response messages.
- When responding to a Command message that did not contain an OpenC2 `request_id`, the OpenC2 Consumer MUST populate the Response message `request_id` header field with the unique identifying value from the Command message's `X-Request-ID` header.

3.4 OpenC2 Consumer as HTTP/TLS Server

This section defines HTTP requirements that apply to the OpenC2 Consumer operating as the HTTP server.

As the OpenC2 Consumer is the HTTP server, the Producer initiates a connection to a specific Consumer and directly transmits OpenC2 Messages containing Commands; the HTTP POST method is used, with the OpenC2 Command body contained in the POST body.

The following HTTP request headers MUST be populated when transferring OpenC2 Commands:

- Host: host name of HTTP server:listening port number (if other than port 80 [testing] / 443 [operations])
- Content-type: `application/openc2+json;version=1.0` (when using the default JSON serialization)
- Accept: `application/openc2+json;version=1.0` (when using the default JSON serialization)

The following HTTP response headers MUST be populated when transferring OpenC2 Responses:

- Content-type: `application/openc2+json;version=1.0` (when using the default JSON serialization)

The following HTTP request and response headers SHOULD be populated when transferring OpenC2 Commands and Responses when the Consumer is the HTTP/TLS server:

- Date: date-time in the preferred IMF-fixdate format as defined by Section 7.1.1.1 of RFC 7231; the conditions for populating the `Date:` header specified in Section 7.1.1.2 of RFC 7231 SHALL be followed

Example messages can be found in Appendix E, section E.1.

4 Conformance

This specification defines conformance targets and requirements for

1. Testing: HTTP without TLS, using port 80
2. Operations: HTTP with TLS, using port 443

4.1 Conformance Targets

The conformance targets for this specification are:

- **Testing**: this conformance target applies to the transfer of OpenC2 messages over HTTP in environments where interoperability is a primary concern and security protections are optional. For example, the Testing conformance target is appropriate for prototyping and "plug fest" interoperability demonstration situations.
- **Operations**: this conformance target applies to the transfer of OpenC2 messages over HTTPS in environments where security or the demonstration of secure interoperability is a primary concern.

The Testing and Operations targets MUST NOT be concurrently available; an OpenC2 Consumer MUST reject non-secure connections when conforming to the Operations target.

4.2 Conformance Requirements

This section defines the conformance requirements for the two conformance targets identified in Section 4.1.

4.2.1 Testing Target Conformance Requirements

A conformant implementation of this transfer specification for the **Testing** conformance target MUST:

1. Support JSON serialization as specified in Section 3.3.1.
2. Transfer OpenC2 Messages using the content type defined in Section 3.3.1.
3. Listen for HTTP connections on port 80 as specified in Section 3.2.1.
4. Use HTTP POST method as specified in Sections 3.2.1, and 3.4, and no other HTTP methods.
5. Transfer OpenC2 command messages to the HTTP listener using the URI scheme specified in Section 3.2.2.
6. Ensure HTTP request and response messages only contain a single OpenC2 message, as specified in Section 3.2.1.
7. Employ HTTP methods to send and receive OpenC2 Messages as specified in Section 3.4.
8. Employ only the HTTP response codes specified in [OpenC2-Lang-v1.0], Section 3.3.2.1.
9. Utilize the OpenC2 message format specified in Section 3.3.2.
10. Support the handling of unique message identifiers as specified in Section 3.3.3

4.2.2 Operations Target Conformance Requirements

A conformant implementation of this transfer specification for the **Operations** conformance target MUST:

1. Satisfy all of the requirements for the **Testing** conformance target as specified in Section 4.2.1.
2. Listen for HTTPS connections on port 443 as specified in Section 3.2.1, and ignore HTTP connection requests on port 80. NOTE: This requirement supercedes conformance requirement #3 in the Testing conformance list in Section 4.2.1.
3. Implement TLS in accordance with the requirements and restrictions specified in Section 3.2.2.
4. Support authentication of remote parties as specified in Section 3.2.4.

Appendix A. References

A.1 Normative References

[RFC2119]

Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

[RFC2818]

Rescorla, E., "HTTP Over TLS", RFC 2818, DOI 10.17487/RFC2818, May 2000, <https://www.rfc-editor.org/info/rfc2818>.

[RFC4122]

Leach, P., Mealling, M., and R. Salz, "A Universally Unique IDentifier (UUID) URN Namespace", RFC 4122, DOI 10.17487/RFC4122, July 2005, https://www.rfc-editor.org/info/rfc4122.

[RFC5246]

Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, DOI 10.17487/RFC5246, August 2008, <https://www.rfc-editor.org/info/rfc5246>.

[RFC7230]

Fielding, R., Ed., and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing", RFC 7230, DOI 10.17487/RFC7230, June 2014, https://www.rfc-editor.org/info/rfc7230.

[RFC7231]

Fielding, R., Ed., and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content", RFC 7231, DOI 10.17487/RFC7231, June 2014, https://www.rfc-editor.org/info/rfc7231.

[RFC7235]

Fielding, R., Ed., and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Authentication", RFC 7235, DOI 10.17487/RFC7235, June 2014, https://www.rfc-editor.org/info/rfc7235.

[RFC7540]

Belshe, M., Peon, R., and M. Thomson, Ed., "Hypertext Transfer Protocol Version 2 (HTTP/2)", RFC 7540, DOI 10.17487/RFC7540, May 2015, https://www.rfc-editor.org/info/rfc7540.

[RFC8174]

Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <http://www.rfc-editor.org/info/rfc8174>.

[RFC8446]

Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <http://www.rfc-editor.org/info/rfc8446>

[RFC8615]

Nottingham, M., "Well-Known Uniform Resource Identifiers (URIs)", RFC 8615, DOI 10.17487/RFC8615, May 2019, https://www.rfc-editor.org/info/rfc8615

[OpenC2-Lang-v1.0]

Open Command and Control (OpenC2) Language Specification Version 1.0. Edited by Jason Romano and Duncan Sparrell. Latest version: http://docs.oasis-open.org/openc2/oc2ls/v1.0/oc2ls-v1.0.html.

A.2 Informative References

[RFC3205]

Moore, K., "On the use of HTTP as a Substrate", BCP 56, RFC 3205, DOI 10.17487/RFC3205, February 2002, https://www.rfc-editor.org/info/rfc3205.

[RFC7525]

Sheffer, Y., Holz, R., and P. Saint-Andre, "Recommendations for Secure Use of Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)", BCP 195, RFC 7525, DOI 10.17487/RFC7525, May 2015, https://www.rfc-editor.org/info/rfc7525.

[RFC8259]

Bray, T., ed., "The JavaScript Object Notation (JSON) Data Interchange Format", STD 90, RFC 8259, DOI 10.17487/RFC8259, December 2017, http://www.rfc-editor.org/info/rfc8259

[SLPF]

Open Command and Control (OpenC2) Profile for Stateless Packet Filtering Version 1.0. Edited by Joe Brule, Duncan Sparrell and Alex Everett. Latest version: http://docs.oasis-open.org/openc2/oc2slpf/v1.0/oc2slpf-v1.0.html

[IACD]

M. J. Herring, K. D. Willett, "Active Cyber Defense: A Vision for Real-Time Cyber Defense," Journal of Information Warfare, vol. 13, Issue 2, p. 80, April 2014.https://www.semanticscholar.org/paper/Active-Cyber-Defense-%3A-A-Vision-for-Real-Time-Cyber-Herring-Willett/7c128468ae42584f282578b86439dbe9e8c904a8.

Willett, Keith D., "Integrated Adaptive Cyberspace Defense: Secure Orchestration", International Command and Control Research and Technology Symposium, June 2015 https://www.semanticscholar.org/paper/Integrated-Adaptive-Cyberspace-Defense-%3A-Secure-by-Willett/a22881b8a046e7eab11acf647d530c2a3b03b762.

Appendix B. Safety, Security and Privacy Considerations

Security considerations are addressed in Section 3.2.3 TLS Usage.

Appendix C. Acknowledgements

The Implementation Considerations Subcommittee was tasked by the OASIS Open Command and Control Technical Committee (OpenC2 TC) which at the time of this submission, had 79 members. The editor wishes to express their gratitude to the members of the OpenC2 TC.

C.1 Special Thanks

The editor also thanks Jerome Czachor, of Huntington-Ingalls Industries, for assistance with incorporating the new `OpenC2-Message` structure into this specification.

C.2 Participants

The following individuals are acknowledged for providing comments, suggested text, and/or participation in CSD ballots or face-to-face meetings:

- Michelle Barry, AT&T
- Toby Considine, University of North Carolina at Chapel Hill
- Martin Evandt, University of Oslo
- Alex Everett, University of North Carolina at Chapel Hill
- John-Mark Gurney, Copado
- Christian Hunt, Copado
- Dan Johnson, sFractal Consulting LLC
- Takahiro Kakumaru, NEC Corporation
- David Kemp, National Security Agency
- Lauri Korts-Pärn, NECAM
- Anthony Librera, AT&T
- Patrick Maroney, AT&T
- Vasileios Mavroeidis, University of Oslo
- Chris Ricard, FS-ISAC
- Daniel Riedel, New Context Services, Inc.
- Michael Rosa, National Security Agency
- Duane Skeen, Northrop Grumman
- Duncan Sparrell, sFractal Consulting LLC
- Michael Stair, AT&T
- Andrew Storms, New Context Services, Inc.
- Gerald Stueve, Fornetix
- Takayuki Tachihara, Trend Micro
- Bill Trost, AT&T
- Drew Varner, NineFX

Appendix D. Revision History

Revision	Date	Editor	Changes Made
v1.0-wd01-wip	6/15/2018	Lemire	Initial working draft
v1.0-wd01-wip	6/29/2018	Lemire	1) Added Suitability section (1.6) 2) Responded to SC member comments
v1.0-wd01-wip	7/20/2018	Lemire	1) Additional responses to member comments 2) Formatting clean-up for easier conversion to Markdown
v1.0-wd01-wip	8/9/2018	Lemire	Implementing feedback from the July 2018 face-to-face meeting and resolving other comments to reach WD01 version to submit for CSD ballot
v1.0-wd02-wip	8/24/2018	Lemire	1) Various edits to clarify interactions when the producer is HTTP listener 2) Other edits and cleanup in response to document comments and Slack forum discussions
v1.0-wd02-wip	8/29/2018	Lemire	1) Adjustments to content type definitions to distinguish commands and responses 2) Made corresponding adjustments to message flow descriptions and sample messages 3) Added acknowledgements
v1.0-wd02-wip	8/30/2018	Lemire	Inserted proposed replacements for sequence diagrams (Figures 2 and 3)
v1.0-wd02-wip	8/31/2018	Lemire	1) Inserted initial draft conformance language (section 4) 2) Revised Section 1 content for greater consistency with related OpenC2 specifications 3) Revised section 2.1 to merge proposed endpoint role descriptions 4) General edit for formatting, readability, consistency, etc.
v1.0-wd02-wip	9/11/2018	Lemire	1) Reviewed and accepted / rejected comments 2) Added placeholders for addressing use of "From" field 3) Added statements about using Cache-control
v1.0-wd02-wip	9/17/2018	Lemire	1) Added table to conformance section specifying mapping of Language Spec message elements. 2) Clarified certificate mutual authentication requirement. 3) Removed language about unsolicited responses from Consumers 4) Numbered the conformance items
v1.0-wd02-wip	9/17/2018	Lemire	1) Removed used of the HTTP "From:" field, and mapped the OpenC2 "from" message element to the authenticated identity of the peer entity 2) Updated examples to remove HTTP From:
v1.0-wd02-wip	9/19/2018	Lemire	1) Final clean-up of residual comments and edits to create WD02 package for CSD ballot 2) Renamed document to WD03-wip
v1.0-wd03-wip	10/15/2018	Lemire	1) Reorganized section 1 to align with other OpenC2 specifications 2) Reworded section 3.3.1 to properly use MUST / SHALL language 3) Clarified requirements wording section 3.2.2 to better indicate TLS version requirements and preferences, and authentication requirements 4) Updated Table 4-1 to align with changes to Language Specification Table 3-1
v1.0-wd03-wip	10/16/2018	Lemire	Final clean-up of residual edits to create WD03 package for CSD approval and release for public review
v1.0-wd03-wip	3/27/2019	Lemire	Resolution of issues from public review 1
v1.0-wd03-wip	3/28/2019	Lemire	Incremented WD version number to 05 prior to CSD ballot to eliminate ambiguity

Revision	Date	Editor	Changes Made
v1.0-wd06-wip	5/14/2019	Lemire	Resolution of issues from public review 2 and adjustments for consistency across the suite of specifications
v1.0-wd07	6/23/2021	Lemire	Minor corrections and changes from January 2020 Plug Fest experience and other miscellaneous updates; captures state of working draft prior to reorganization against new OASIS template
v1.0-wd08	7/15/2021	Lemire	Reorganizes specification to use the new OASIS template
v1.1-wd01	9/08/2021	Lemire	1) Defines a standard Uniform Resource Identifier (URI) scheme and added a corresponding conformance requirement 2) Specifies the use of the atomic OpenC2 message structure, updated content-type accordingly, and adjusted examples to match 3) Defines Testing and Operations conformance targets and requirements to support both secure (HTTPS) and non-secure (HTTP) message transfers with a single specification 4) Restructured document using the updated OASIS work products outline 5) Added new section with requirements for message correlation 6) Other minor changes and corrections based on plug fest and interoperability testing experiences

Appendix E. Examples

This section is non-normative.

OpenC2 Messages consist of a set of "message elements" defined in Section 3.2 of [OpenC2-Lang-v1.0]. The example messages below illustrate how this is handled in practice.

A Request-URI ending in `/.well-known/openc2` is used in all example HTTP requests.

E.1 HTTP Request / Response Examples: Consumer as HTTP Server

This section presents the HTTP message structures used when the OpenC2 Consumer acts as the HTTP listener.

E.1.1 Producer HTTP POST with OpenC2 Command

Example message:

```
POST /.well-known/openc2 HTTP/1.1
Content-type: application/openc2+json;version=1.0
Date: Wed, 19 Dec 2018 22:15:00 GMT
X-Request-ID: d1ac0489-ed51-4345-9175-f3078f30afe5

{
  "headers": {
    "request_id": "d1ac0489-ed51-4345-9175-f3078f30afe5",
    "created": 1545257700000,
    "from": "oc2producer.company.net",
    "to": [
      "oc2consumer.company.net"
    ]
  },
  "body": {
    "openc2": {
      "request": {
        "action": "...",
        "target": "...",
        "args": "..."
      }
    }
  }
}
```

E.1.2 Consumer HTTP Response with OpenC2 Response

Example message:

```
HTTP/1.1 200 OK
Date: Wed, 19 Dec 2018 22:15:10 GMT
Content-type: application/openc2+json;version=1.0
X-Request-ID: d1ac0489-ed51-4345-9175-f3078f30afe5

{
  "headers": {
    "request_id": "d1ac0489-ed51-4345-9175-f3078f30afe5",
    "created": 1545257710000,
    "from": "oc2consumer.company.net",
    "to": [
      "oc2producer.company.net"
    ]
  },
  "body": {
    "openc2": {
```



```
      "response": {
        "status": 200,
        "status_text": "...",
        "results": "..."
      }
    }
  }
}
```


Appendix F. Notices