Abstract

This specification defines operating semantics of the Cloud Tunneling Protocol (CTP). CTP enables a standardized mechanism for connecting Internet-of-Things (IoT) devices to hosted cloud services. CTP provides a network efficient means to facilitate multiple concurrent data conversations over the same channel by providing multiplexed virtual sockets. The protocol allows for services on either endpoint to be advertised and accessible to each endpoint of the CTP connection.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/1id-abstracts.html

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html

This Internet-Draft will expire on May 11, 2016.

Copyright Notice

Copyright (c) 2015 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents
1. Introduction

This document is a specification of the Cloud Tunneling Protocol (CTP). CTP communication normally occurs over a TCP/IP socket connection. CTP is an application layer protocol that can multiplex multiple TCP/IP [RFC793] sockets over a single connection. In the context of the Internet Suite (TCP/IP), the CTP protocol is located above the transport layer and below other application-layer protocols. This protocol describes a method of exposing and tunneling multiple services on both the client and server endpoints through a single TCP/IP connection. CTP is a departure from standard virtual private network (VPN) protocols as it does not define a method for accessing network based resources directly. Instead, resources are defined at the CTP application layer and resolved by either endpoint as configured.

The CTP protocol was developed by the CTP working group. It is created to operate specifically for the TCP/IP Internet environment, wrapping multiple bi-directional TCP/IP conversations within a single connection. The protocol was designed to facilitate the access of multiple services simultaneously between two endpoints, with data traversing complex networks interpolated with routers,
firewalls, and network address translation. Much work has been done
to produce an efficient protocol that can provide the type of
connectivity discussed.

[ Application ] <---Process to Process---> [ Application ]
[     CTP     ]                            [     CTP     ]
[ Transport ] <------Host to Host------> [ Transport ]
[ Internet ]                            [ Internet ]
[ Link ]                            [ Link ]

^                      ^
|__________________________________________|

Figure 1a: CTP in the Internet Protocol Suite

1.1 Protocol Overview

CTP assumes a long-lived TCP/IP connection established between a
client and a remote server. This connection is assumed to be
originated from the client. In general, a client establishes a
single connection to the remote server for efficiency. The CTP
protocol describes how to achieve multiple TCP/IP conversations over
this single connection.

The CTP protocol is a binary protocol which encapsulates application
layer data into frames for transport. These frames provide the
ability to multiplex multiple conversations on a single connection
by designating which conversation, or virtual socket, the frame is
associated with.

Once a connection is established, all communication between the two
endpoints is bidirectional for any TCP/IP conversation. Each
endpoint can run in either a client or server context using multiple
virtual sockets. As such, both the client and server endpoints can
establish virtual sockets over this single CTP connection. CTP
utilizes control messages to negotiate the establishment,
maintenance, and tear-down of virtual sockets over the lifetime of
this connection.

Establishing a virtual socket signals to the remote endpoint the
initiator’s intent to access a remote service. Once a virtual
socket is established, data can be transmitted over this channel.
Data is multiplexed and encapsulated into frames on the sending
endpoint. On the receiving endpoint, frames are demultiplexed and
de-encapsulated. All communication between endpoints is
deterministic with a fixed frame header specifying the payload size
to follow.

[ HTTP ] [ RTSP ] [ RTMP ] <-----------------> [ HTTP ] [ RTSP ] [ RTMP ]
[ Application ] <---Process to Process---> [ Application ]
[     CTP     ]                            [     CTP     ]
[ Transport ] <------Host to Host------> [ Transport ]
[ Internet ]                            [ Internet ]
[ Link ]                            [ Link ]

|__________________________________________|

Figure 1.1b: Example Applications Tunneled through CTP

Multiple application-layer processes with CTP within
the context of the Internet Protocol Suite.
By utilizing the protocol, both the TCP/IP client and
TCP/IP server can expose their own services.

1.2 Definitions of Commonly Used Terms

This section provides definitions for the terms that have a specific meaning to the CTP protocol. These terms are used throughout this specification.

Endpoint
One end of a TCP/IP socket connection. This term refers to either a TCP/IP server or client.

Local Endpoint
The local end of a TCP/IP socket connection from the reader’s perspective.

Remote Endpoint
The remote end of a TCP/IP socket connection from the reader’s perspective.

CTP Client
The endpoint from which the CTP connection originated.

CTP Server
The endpoint that received the CTP connection.

Frame
The basic unit of transmission in CTP. Encapsulates application layer data and specifies a particular TCP/IP conversation.

Control Frame
Extends a standard CTP frame to include a command, or action, as well as parameters in the frame’s payload.

String
A finite sequence of ASCII encoded characters as defined in RFC 20 [RFC20].

Command
An action represented as a 4 byte string in a control frame.

Tag
A key/value pair used to represent a parameter for a given control frame’s command.

Endpoint Service
A service or resource as advertised and exposed by a CTP endpoint.

Virtual Socket
A single TCP/IP conversation in a multiplexed CTP connection.

Service Label
A string that identifies a given endpoint service. Service labels are advertised by endpoints and used to specify which services endpoints are accessing through a virtual socket.

2. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this
3. Frame Encapsulation

CTP encapsulates and interleaves both data and control messages into transmission units called frames. All frames must begin with a fixed size header (8 bytes). The frame header provides the protocol with information about what data the frame encapsulates. This information is used to determine which TCP/IP conversation, or virtual socket, the frame is associated with.

The header also specifies the length of the data payload to follow. The maximum size of a frame is 65,543 bytes including the 8 byte header. The maximum frame payload size is 65,535 bytes. Frames can be split over multiple TCP/IP packets if necessary, but the entire frame must be sent in full before the next frame can be sent.

3.1 Frame Header

Following the TCP/IP header, the packet’s payload will contain one or more CTP frames. Each frame header must have the following structure:

```
0 0 1 2 3 4 5 6 7
+-----------------------+
| I | VERSION| VIRTUAL| R | PAYLOAD |
| D | SOCKET | S | SIZE |
|   | ID     | V |
+-----------------------+
```

Figure 3.1a: CTP Frame Header Layout

These frame fields are described in detail in Figure 3.1b. All numerical values in the frame header must be encoded as Big-Endian.

<table>
<thead>
<tr>
<th>FIELD</th>
<th>SIZE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID Tag</td>
<td>1 byte</td>
<td>Indicates the start of a frame. Set at 0x41 (character ‘A’)</td>
</tr>
<tr>
<td>Virtual Socket ID</td>
<td>2 bytes</td>
<td>Indicates the Virtual Socket ID which identifies a user-defined endpoint service.</td>
</tr>
<tr>
<td>Reserved</td>
<td>1 byte</td>
<td>Reserved for future protocol use.</td>
</tr>
<tr>
<td>Payload Size</td>
<td>2 bytes</td>
<td>Size of the application layer payload to follow.</td>
</tr>
</tbody>
</table>

Figure 3.1b: CTP Frame Header Fields

3.2 Control Frames
CTP control frames are sent using virtual sockets 0 and 1 for client and server originated commands, respectively. Control frames contain the same header as standard CTP frames, but contain additional command and tag data as part of the frame’s payload. All control frames must be acknowledged by the remote endpoint before a new control frame can be sent.

![Figure 3.2a: CTP Control Frame Header](image)

### 3.3 Commands and Tags

Control frames (Section 3.2) contain a 4 byte command string that indicates a particular action a remote endpoint must take. The command string is located in the 4 bytes following the payload size field. A control frame must not contain more than one command.

Many commands contain one or more parameters defined as tags. A tag is structured as 2 bytes of ASCII text for the tag name, followed by a 2 byte unsigned integer specifying the size of the value to follow. There is no limit on the number of tags that can follow a command. However, all tag data plus the command must be less than the maximum frame payload size. Control frames may contain tags that are not explicitly defined in this document, allowing an implementer to extend the functionality of the control channels.

![Figure 3.3a: CTP Tag](image)

Commands and tags are part of the frame’s payload. The length of the data in the command and tags section must be reflected in the frame header’s payload size field. All control frames must be acknowledged by the remote endpoint by sending an acknowledgement command (ACK) back to the sender with a status code tag (ST) indicating the status of the request (Section 3.4). The following commands and tags must be supported:

**AUTH (authenticate):** The authentication command is sent to authenticate a given CTP connection. The authentication command is optional, and only used for CTP connections that require authentication. The authentication command supports three forms of authentication: credentials (username/password), token, and client certificates. See Section 4.2 for more details on authorization. The following tags are supported by this command:
UN: username (string)
PW: password (string)
TK: token (string)
CR: client certificate (base64 encoded string)

A successful authentication request shall return with an OK status code (0x00). Requests that fail authentication shall return with an UNAUTHORIZED status code (0x40). See Section 3.4 for more details on response status codes.

OPVS (open virtual socket): Requests that a virtual socket be opened on the remote endpoint. The request specifies the endpoint service and virtual socket identifier that must be used for the virtual socket. See Section 6 for more details on virtual socket establishment. The following tags are required by this command:

SV: endpoint service (string)
VS: virtual socket identifier (unsigned 2 byte integer)

A successful open virtual socket acknowledgement frame shall contain an OK status code (0x00). Failure status codes may include PROTOCOL_NOT_SUPPORTED (0x60), VIRTUAL_SOCKET_ALREADY_OPEN (0x61), and VIRTUAL_SOCKET_UNAVAILABLE (0x63). See Section 3.4 for more details on response status codes.

CLVS (close virtual socket): Requests that a virtual socket be closed on the remote endpoint. The request specifies the virtual socket identifier which specifies the virtual socket to be closed. See Section 6.2 for more details on the virtual socket life-cycle. The following tags are required by this command:

VS: virtual socket identifier (unsigned 2 byte integer)

A successful close virtual socket acknowledgement frame shall contain an OK status code (0x00). Failure status codes may include VIRTUAL_SOCKET_ALREADY_CLOSED (0x62). See Section 3.4 for more details on response status codes.

ACK (acknowledgement): Acknowledgements are sent in response to control frames. All control frames must be acknowledged using the ACK command. The acknowledgement frame must contain a status tag (ST) indicating the return status of the request (Section 3.4). Additional trace data may be returned using the EX and TR tags. The following tags are supported by this command:

ST: status (unsigned 2 byte integer)
EX: exception message (string)
TR: stack trace (string)

SVLT (service list): Retrieves a list of endpoint services (Section 5) supported by the connected endpoint. There are no
tags associated with this command.

The remote endpoint shall return a list of SV tags indicating the service labels that are exposed through the endpoint. Multiple SV tags are supported in the acknowledgement response. An OK status code (0x00) tag must be included in the list of tags (Section 3.4).

SYNC (synchronize): Retrieves a list of virtual sockets that are currently allocated on the remote endpoint. There are no tags associated with this command.

The remote endpoint shall return a list of open virtual sockets using a SV tag to identify the service and a VS tag to identify the virtual socket identifier of the open virtual socket. Multiple SV/VS tags are supported in the acknowledgement response. An OK status code (0x00) must be included in the list of tags (Section 3.4).

PING (ping): Pings the remote endpoint to ensure CTP connectivity. This should be used for application-layer keepalive messages. There are no tags associated with this command.

The remote endpoint must return an acknowledgement with an OK status code (0x00) (Section 3.4).

3.4 Control Frame Acknowledgements

While standard CTP frames do not require responses, all CTP control frames must be acknowledged by the remote endpoint. Acknowledgement frames are control frames with the command field set to "ACK". All acknowledgement frames have a status tag (ST) set to indicate the status of the request. The status tag is represented as a 1 byte value corresponding to one of the following values:

OK (0x0): Indicates that the control frame was successfully processed.

ALREADY_AUTHENTICATED (0x01): Returned when an authentication command (AUTH) (Section 3.3) has been sent after the connection is already authenticated.

BUSY (0x07): Indicates the endpoint is currently busy processing a previously received control frame (Section 3.2).

UNAUTHORIZED (0x40): Indicates that the authentication command (AUTH) (Section 3.3) was received, but contained missing or incorrect authentication information.

FORBIDDEN (0x41): Returned when a command is sent to a remote endpoint before a successful authentication command (AUTH) (Section 3.3) was received or access to a requested resource was denied.

SERVICE_NOT_SUPPORTED (0x60): Indicates that the remote endpoint does not support the specified service or resource (Section 5) supplied with an open virtual socket (OPVS) command (Section 3.3).

VIRTUAL_SOCKET_ALREADY_OPEN (0x61): Returned when the remote endpoint already has a virtual socket with the specified
identifier open. This error code indicates that the server and client are not synchronized. Possibly returned due to a faulty CTP implementation.

VIRTUAL_SOCKET_ALREADY_CLOSED (0x62): Indicates that the close virtual socket command (CLVS) (Section 3.3) was not successful because the virtual socket is already closed on the remote endpoint.

VIRTUAL_SOCKET_UNAVAILABLE (0x63): Indicates that the remote endpoint has no further virtual sockets available. See Section 6.3 for more details on virtual socket allocation.

INVALID_TAG (0x80): Indicates that the remote endpoint was unable to process a command because it contained tags that were not valid for the command.

INVALID_STATUS (0x81): Indicates that a status code was not recognized by the remote endpoint.

INVALID_COMMAND (0x82): Returned when the remote endpoint receives a control command (Section 3.3) that it does not recognize.

TIMEOUT (0x90): Indicates that the command timed out during processing on the remote endpoint.

GENERAL_ERROR (0xFF): Indicates that an application-layer CTP error occurred on the endpoint.

Application layer errors (0xFF) are used to troubleshoot CTP software implementations and are not meant to be used to indicate an error with a service being tunneled through CTP. Optionally, applications can choose to supply more information to the remote endpoint by adding an EX tag for an error message and/or a TR tag with an attached stack trace. See Section 3.3 for more information about supported acknowledgement command (ACK) tags.

4. Initial Connection

A Cloud Tunneling Protocol connection is created when a TCP/IP socket is established between a CTP client and a CTP server. The originator of the TCP/IP socket is known as the CTP client. The receiver of the TCP/IP socket is known as the CTP server.

4.1 Establishment

A CTP connection is established by the CTP client establishing a TCP/IP socket connection to the CTP server. If the client does not require authentication, the connection is considered established. If the client is required to authenticate then the AUTH command (Section 3.3) must be sent and acknowledged successfully before the connection is considered established.

4.2 Authentication/Authorization

The protocol supports three different methods of authenticating a CTP connection: credentials (user/password), token, and client certificates. CTP authentication occurs when the CTP client sends the authorization command (AUTH) (Section 3.3). CTP does not provide a mechanism for authorizing access to specific resources. This should be handled at the CTP application layer or at the
service application layer if necessary.

<table>
<thead>
<tr>
<th>TYPE</th>
<th>TAGS</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Token</td>
<td>TR</td>
<td>String formatted token.</td>
</tr>
<tr>
<td>Credentials</td>
<td>UN</td>
<td>String formatted user name.</td>
</tr>
<tr>
<td></td>
<td>PW</td>
<td>String formatted password.</td>
</tr>
<tr>
<td>Client Certificate</td>
<td>CR</td>
<td>Base64 encoded string representation of binary certificate content.</td>
</tr>
</tbody>
</table>

Figure 4.2a: CTP Authentication Types

4.3 Transport Security

CTP does not define any encryption or transport security. Implementations of CTP that utilize authorization functionality or tunnel sensitive data should use SSL/TLS sockets in their implementation to remain secure. CTP sockets can be wrapped in a SSL/TLS connection for both encryption and transport layer security, making use of existing PKI infrastructure. See RFC 2246 [RFC2246] and RFC 6101 [RFC6101] for more information.

4.4 Keepalives

Determining when and where an endpoint is connected is a common problem driving many scalable server based architectures that exist today. CTP attempts to solve this problem with the addition of an optional keepalive mechanism on-top of the existing TCP/IP keepalive implementation. CTP will, by default, utilize the existing TCP/IP keepalive to determine if a connection is dropped unexpectedly. These keepalive timers should be set to a low interval to ensure speedy notification of a dropped connection.

CTP also provides an application layer keepalive through the ping command (PING) (Section 3.3). The ping command allows either endpoint in a CTP connection to ping the remote endpoint. Ping commands are replied with a standard CTP acknowledgement control frame (ACK) (Section 3.3). These application layer keepalives can optionally be used to ensure that the CTP stack is operating correctly. These events can be scheduled at periodic intervals and used to refresh statistics and status of a given connection.

5. Endpoint Services

Endpoint services are resources or services advertised by a remote endpoint. After a CTP connection has been established and validated (Section 4), either endpoint can request a list of supported services from the remote endpoint by issuing the service list command (SVLT) (Section 3.3). Endpoint services are represented as string based labels used by an endpoint to route a virtual socket to an associated service. Examples of protocol labels include:

1. HTTP
2. RTSP
3. 80
4. 1935
5. localhost:554
Each CTP endpoint determines how to route a given request to the appropriate service. Each implementation may handle label request routing differently. For most use cases, it is strongly recommended that the domain:port format is used as it provides the greatest amount of flexibility with minimal configuration.

6. Virtual Sockets

Virtual sockets are used to indicate a particular TCP/IP conversation being multiplexed over a CTP connection. Each frame specifies a particular conversation based on the value of the virtual socket ID field specified in the frame header (Section 3.1). The Cloud Tunneling Protocol supports up to 65,536 virtual sockets per TCP/IP connection.

6.1 Reserved Virtual Sockets

CTP reserves two channels for control communication between endpoints. Virtual socket identifier 0 and 1 are reserved for this use. When a CTP connection is initially established between two hosts, the initiator sends the authentication command (AUTH) (Section 3.3) over virtual socket 0 if authentication is enabled. All commands originating from the CTP client (the CTP connection originator) must be sent using virtual socket 0. Commands originating from the CTP server (the CTP connection receiver) must be sent using virtual socket 1.

6.2 Virtual Socket Life-Cycle

Virtual sockets can be opened and closed by either endpoint at any point after the initial connection is established if necessary. During virtual socket initialization, one endpoint issues an open virtual socket command (OPVS) (Section 3.3) which indicates the particular service or resource the endpoint is interested in accessing, as well as the virtual socket identifier to associate with this virtual connection. Once the remote endpoint processes the command and allocates the same virtual socket ID, the remote endpoint must reply to the request by sending an acknowledgement command (ACK) (Section 3.3) back on the same control virtual socket the request originated.

The virtual socket can remain open as long as necessary and even persist for the lifetime of the CTP connection. However, virtual sockets should be closed when no longer needed to conserve resources on both the client and server CTP endpoints. Virtual sockets can be closed by issuing the close virtual socket command (CLVS) (Section 3.3).

6.3 Virtual Socket Allocation

Virtual socket identifiers are allocated by the originating endpoint on demand when an open virtual socket request is issued. Virtual sockets opened from the CTP client endpoint must issue even numbered virtual socket identifiers, while virtual sockets originating from the CTP server endpoint must issue odd numbered virtual socket identifiers. This ensures that no identifier collisions occur if both the server and client attempt to open a virtual socket at the same time.
Client originated virtual sockets must start being assigned at 2. Server originated virtual sockets must start being assigned at 3. Each endpoint must keep track of the last allocated virtual socket identifier and increment (by two) from that point. Once the maximum number of virtual socket identifiers is reached, counting must roll-over and start again at the starting number. If no additional free virtual socket identifiers exist, an error condition must be raised to the caller.

7. IANA Considerations

A string for identifying CTP is entered into the "Application-Layer Protocol Negotiation (ALPN) Protocol IDs" registry established in [RFC7301].

7.1 Registration of CTP Identification Strings

This document creates a registration for the identification of CTP in the "Application-Layer Protocol Negotiation (ALPN) Protocol IDs" registry established in [RFC7301].

The "ctp/1" string identifies CTP when used over TLS:

Protocol:  CTP over TLS
Identification Sequence: 0x63 0x74 0x70 0x2f 0x31 0x73
Specification:  This document

8. Security Considerations

8.1 Denial of Service

CTP connections require some state management and therefore must allocate some amount of server resources to maintaining each connection's state (socket status and virtual sockets). When handling a large amount of concurrent connections it is important to manage resources efficiently. CTP connections are dependent on TCP/IP sockets, so the same care maintaining TCP/IP sockets must be taken for CTP connections.

8.1.1 Connection Starving

Connection starving occurs when a TCP connection is established from the CTP client to the CTP server, but no data is transferred over this connection. Resources are allocated by the server to handle the maintenance of this TCP socket even though it is not being utilized. A large amount of these inactive connections could cause a denial of service. Server implementations should terminate TCP/IP sockets that don’t send data after a given period of time to conserve server resources.

8.1.2 Zombie TCP/IP Connections

A zombie TCP/IP connection is a TCP/IP socket that has been properly established, but the client has unexpectedly terminated the socket without notifying the remote endpoint. The server implementation may never know that this socket has been terminated if data is not transmitted through it. Maintaining a large number of these zombie TCP/IP connections could consume significant server resources.
leading to a denial of service. It is recommended that TCP/IP keepalives be used for all CTP sockets to ensure that this situation is avoided. See Section 4.4 for more details on sending TCP/IP keepalives.

8.1.3 Zombie CTP Connections

A zombie CTP connection is a CTP connection that has been properly established and has an active TCP/IP socket, but the underlying CTP implementation is not transmitting data. This connection is actively using server resources, but is not functional. Server implementations that do not actively determine the validity of a CTP connection are at risk of a denial of service based on these conditions. It is recommended that server implementations implement a ping mechanism as discussed in Section 4.4.

8.1.4 Data Flooding

Data flooding can occur when one CTP connection sends a greater amount of data than other connections. The larger amount of data may be unfairly processed by the CTP server implementation, slowing down data processing of other connections. CTP server implementations should fairly read and process incoming data to ensure that all connections are not slowed down by a busy connection.

8.2 Command Length Security

It is possible that the sum of the tag lengths in a control frame exceeds the maximum frame length. As discussed in Section 3.3, a control frame’s command and tag data must never exceed the frame’s payload size, as such these frames are invalid. CTP implementations must respond with an acknowledgement command (ACK) (Section 3.3) with the status code set to INVALID_COMMAND (0x82) (Section 3.4). When processing control frames, implementations must not read beyond the frame’s payload size (Section 3.1).

8.3 Label Logic Considerations

Implementations should not parse endpoint service labels to determine what services to connect to. Endpoint services should be explicitly defined, specifying which endpoint services are exposed. Open virtual socket command (OPVS) (Section 3.3) can then be used to access these explicitly defined services.

9. References

9.1 Normative References


[RFC7301]
Diagram 1 below depicts the typical CTP initial connection dataflow. After standard TCP[1] and SSL/TLS[2] negotiations, the CTP client sends the AUTH authentication request[3]. The CTP server verifies the client credentials and responds with an acknowledgement command (ACK) [4]. At that point both connecting endpoints advertise their corresponding endpoint services via service list (SVLT) requests and acknowledgments. Note that the CTP client uses virtual socket (VS) 0 and the CTP server uses VS 1. As indicated, those virtual sockets IDs are reserved for CTP control. In the below diagram the CTP server advertises HTTP services [6] and the CTP client shows HTTP and RTSP internal server capabilities [8]. That concludes the initial CTP exchange. Both CTP endpoints are now ready to provide their services.
10.2 Appendix B. Virtual Socket Dataflow

Diagram 2 shows the typical virtual socket dataflow. As indicated in diagram 1, the CTP client broadcast HTTP service capability which is requested by the CTP server endpoint. The server initiates the virtual socket by sending the open virtual socket (OPVS) [1] command to TCP client indicating that the service endpoint (HTTP) and virtual socket ID (VS:3) has been allocated for that request. Note that control virtual socket 1 is used by TCP server. The TCP client responds with the acknowledgement [2]. At that point the HTTP request [3] and HTTP response [4] are sent from communicating endpoints on virtual socket 3 (VS:3).

Diagram 2: Initial Connection Data Flow

Diagram 2: Virtual Socket Data Flow - HTTP Data Example

Authors’ Addresses

Piotr Romanczyk
Honeywell
EMail: piotr.romanczyk@honeywell.com

Christopher Colicino
Honeywell
EMail: christopher.colicino@honeywell.com

Michael Landi
Honeywell
EMail: michael.landi@honeywell.com
Tomas Brodsky
Honeywell
EMail: tomas.brodsky@honeywell.com

IPR Disclosure

"By submitting this Internet-Draft, each author represents that any applicable patent or other IPR claims of which he or she is aware have been or will be disclosed, and any of which he or she becomes aware will be disclosed, in accordance with Section 6 of BCP 79."

Disclaimer

"This document and the information contained herein are provided on an "AS IS" basis and THE CONTRIBUTOR, THE ORGANIZATION HE/SHE REPRESENTS OR IS SPONSORED BY (IF ANY), THE INTERNET SOCIETY AND INTERNET ENGINEERING TASK FORCE DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE."

Expires May 11, 2016