Abstract

This document defines a simple protocol - Trivial Object Transfer and Exchange (TOTE) - that provides bidirectional exchange of MIME objects over a TCP connection. It is used in conjunction with protocols based on the offer/answer model, such as the Session Initiation Protocol (SIP). TOTE can be used for any application requiring direct agent-to-agent data communication, such as picture exchange or camera controls.
Table of Contents

1. Introduction ................................................. 3
2. Overview of Operation ....................................... 4
3. Terminology .................................................. 5
4. Purpose Tokens ............................................... 5
5. Offer-Answer Procedures ...................................... 6
   5.1. Offerer Procedures ..................................... 7
   5.2. Answerer Procedures .................................... 7
6. Session Management ........................................... 8
7. Sending and Receiving Messages ............................... 8
8. Formal Syntax ................................................ 9
   8.1. SDP Extensions .......................................... 9
   8.2. TOTE Message Syntax .................................. 10
9. Possible Purposes ............................................ 11
   9.1. Session Mode IM ........................................ 11
   9.2. Shared Browsing ......................................... 11
   9.3. Forms and Application Interaction ....................... 11
   9.4. Conference Controls ................................... 12
   9.5. Picture Sharing ......................................... 12
10. Security Considerations ..................................... 13
11. IANA Considerations ........................................ 13
   11.1. TOTE Purpose Registry ................................ 13
   11.2. SDP Transport Protocol ................................. 14
   11.3. SDP Attribute Names .................................... 14
       11.3.1. Send Purposes .................................... 14
       11.3.2. Receive Purposes .................................. 14
12. Acknowledgements ........................................... 15
13. References .................................................. 15
   13.1. Normative References .................................. 15
   13.2. Informative References ................................ 16
Author’s Address ................................................ 16
Intellectual Property and Copyright Statements .................. 18
1. Introduction

The Session Initiation Protocol (SIP) [RFC3261] allows for the establishment, management, and termination of interactive sessions between user agents. These sessions are often used for the exchange of real time media, such as voice, video or text, using protocols such as the Real Time Transport Protocol (RTP).

Oftentimes, there is a need for agents to exchange non-real time application data as part of a communications session. Examples of such data include:

Camera Controls: Video conferencing systems often allow a participant in the session to control the camera on the far end of the call. This allows them to pan, tilt and zoom in order to see the person or people they are talking to.

Pictures: During a voice call, a user can snap a picture and send it to the other participant in the call. This requires a mechanism to transfer the picture from one end to the other.

Game Moves: Users engaging in a real time game may need to exchange game moves in addition to having a voice/video chat in concert with the game.

SIP provides three general purpose mechanisms that allow a pair of agents to communicate data during a session. These are:

Subscription: One user agent can send a SUBSCRIBE request to the other, using an event package specific to the application data that is desired. This subscription is ideally done as part of a separate dialog. This technique is recommended for application interaction, and is described in [I-D.ietf-sipping-app-interaction-framework].

Media: A user agent can use SIP to set up another media session whose purpose is to carry the application specific data.

INFO: A user agent can send the data in a SIP INFO message along the existing SIP dialog. This technique is described in [I-D.ietf-sip-info-events].

Currently, there is no general purpose protocol that allows agents to set up a media session for the exchange of application data. This specification fills that need. It defines a protocol called Trivial Object Transfer and Exchange (TOTE). TOTE is a TCP-based protocol that provides framing and content-typing. This specification also defines parameters for the Session Description Protocol (SDP).
so that TOTE sessions can be established using protocols based on the offer/answer model [RFC3264], such as SIP. These SDP extensions allow agents to negotiate the "purpose" of TOTE session, along with supported MIME types. The "purpose" is a parameter that operates similarly to the INFO events defined in [I-D.ietf-sip-info-events].

2. Overview of Operation

An agent wishing to establish a TOTE session includes a "message" media line in an SDP offer. This m-line indicates that the messaging session uses TOTE over TCP or TLS over TCP (TOTE is defined only for TCP or TLS/TCP). The m-line also includes a list of "purpose" parameters that the agent wishes to receive or wishes to send. TOTE sessions can be established before there is a need to actually send or receive data, or on-demand at the time the exchange is desired. Typically the agent that wishes to send will initiate the offer for the session.

The answerer includes the list of "purpose" parameters it wishes to send or receive. These need not be the inverse of those in the offer; the agent can indicate that it can receive with "purpose" values for which there was no corresponding send value in the offer, and vice-a-versa. This allows easy usage with third-party call control.

TOTE itself involves a direct TCP connection established between agents. All TOTE implementations support at least the lite version of Interactive Connectivity Establishment (ICE) for TCP [I-D.ietf-mmusic-ice-tcp]. ICE-tcp negotiates the directionality of connection establishment and provides firewall and NAT traversal. Secure TOTE is provided using [RFC4572].

The TOTE message itself is lightweight. It consists of a length, a purpose token, a MIME content type, a CRLF, and then the actual content. An example TOTE message is:

```
l:7655
p:pic
t:image/jpg

BINARY-IMAGE-DATA
```

Figure 1: Example TOTE Message
This message indicates that it is 7655 bytes long. Its purpose is "pic", which is used to exchange pictures for the purposes of image sharing in a communications session. Purpose tokens can either be global, using values registered through an IANA process, or can be vendor proprietary using vendor tags, like com.example.vendortag2. The content type is image/jpg, and the message contents are the binary image data. TOTE messages look similar to MIME and SIP and can be parsed using their parsers, but the syntax is a subset designed for low overhead.

3. 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 RFC 2119 [RFC2119].

4. Purpose Tokens

The purpose token plays an important role in TOTE. It conveys the context in which the exchange content is to be interpreted. If the content-type answers the question, "what", the purpose token answers the question "why". The purpose value tells a sender or recipient what it needs to do with the content it receives - how and if to render it, process it, store it, and so on.

As an example, the "bizcard" purpose token signifies that the content is being exchanged in order to convey contact information for the user. This knowledge allows a receiver to know that it should prompt the user and ask if the data should be imported, to render it to the user, and to allow a user to edit it. A particular purpose can be supported by a multiplicity of content types, and more importantly, a single content type could be used for many purposes. HTML is a good example of this. When it contains the hCard attributes in its elements, it can be used in conjunction with the "bizcard" purpose. It could also be used for a "form-entry" purpose, which could be used to push forms to a user for entry, rather than DTMF.

A purpose token can also be used to signal communication that goes beyond just object passing, but represents a full-blown protocol running between agents. For example, imagine that SOAP methods are developed to implement a shared whiteboard function. A purpose token called "whiteboard" could be used in conjunction with the content-type "application/soap". The agents can then run the SOAP-based protocol over TOTE in order to implement the whiteboard.

Purpose tokens exist in one of two namespaces. The global namespace
is a flat namespace, managed by IANA. New purpose tokens are added
to this namespace based on IETF consensus [RFC2434]. These tokens
are meant for common applications for which formal standardization is
desired. The other namespace is the vendor namespace. This
namespace is unmanaged. Vendors or other organizations can define
tokens by taking their domain name, reversing the order, and
appending additional tokens that signify the purpose. For example,
the example.com domain has complete control over the com.example
namespace, and can utilize purpose tokens like com.example.foo at its
discretion.

It is possible that purpose values that are defined initially in the
vendor namespace become sufficiently common and successful that there
is a desire to formalize the value and standardize it. In that case,
based on IETF consensus, the vendor token can be registered in the
global namespace.

Whether in the global namespace or the vendor namespace, each purpose
has to specify the following:

Baseline content type: Each purpose MUST specify a mandatory-to-
implement content type for use with that purpose. This ensures a
basic level of interoperability. Depending on the content type,
the purpose may also need to specify details on format of that
object, or usage of it, if the object is actually a protocol.

Additional content types: Each purpose MUST specify whether there
are additional content types that can be utilized for this
purpose, and if so, what they are. A purpose can be extended at a
later time by simply declaring support for additional types. Note
that, within a purpose token, the content-type has to uniquely
define what is to be sent and received. If two objects can be
exchanged for a particular purpose but would otherwise have
identical content types (for example, differently formatted text/
plain messages), a new content type MUST be defined for one of
them.

Semantics: Each purpose MUST clearly define what its used for. The
definition must be sufficiently clear to allow an implementation
to decide whether to render the object, what kind of UI to apply,
how and where to store the object, how to process it, and so on.

5. Offer-Answer Procedures
5.1. Offerer Procedures

An agent MAY send an offer at any time to add a tote session, remove a tote session, or modify a tote session. Removing a tote session is done by setting the port in the SDP m-line to zero, as is done for other media types as defined in [RFC3264].

A tote offer MUST utilize the "message" media type in the m-line, as defined in [RFC4566]. The port MUST be set as defined in [I-D.ietf-mmusic-ice-tcp], and the protocol MUST be "TOTE" for TOTE over TCP, or "TOTES" for TOTE over TCP/TLS. All agents MUST implement both TOTE and TOTES. However, the offerer MAY elect to use either. The format list MUST be "*".

A single TOTE session corresponds to a single connection (possibly through a relay) between agents. A single TOTE session can support a multiplicity of purposes. However, if a purpose represents a significant amount of data, such that head-of-line blocking might occur between purposes, an agent SHOULD utilize a separate TOTE session for that purpose.

The offerer MUST include one or more send-purp attributes and MUST include one or more recv-purp attributes. Each attribute contains a single purpose name, followed by a list of MIME content types that the agent supports for sending or receiving with that purpose parameter. This list MUST include the baseline mandatory content type for that purpose. For example:

a=send-purp:pic image/jpg image/tiff
a=recv-purp:pic image/jpg
a=recv-purp:bizcard text/x-vcard text/html

This indicates that the agent supports the "pic" purpose, to both send and receive, and the "bizcard" purpose, receiving only. For "bizcard", used to exchange contact information in a session, the agent supports the classic vcard format as well as the HTML microformat, also known as hCard.

5.2. Answerer Procedures

An agent receiving an offer with an m=message line including the protocol "TOTE" or "TOTES" is being offered the opportunity to set up a TOTE session for one or more purposes. If the agent is capable of receiving objects using at least one of the purpose tokens associated with the tote session, the agent SHOULD accept the session. If it can receive none of them, it MUST reject the session (by setting the port to zero).
Construction of the answer follows the same rules for construction of the offer. Note that the values of send-purp and recv-purp do not need to be in response to the values of recv-purp and send-purp in the offer. An agent MAY include send-purp values in the answer for which there was no corresponding recv-purp in the offer. Similarly, an agent MAY include recv-purp values in the answer for which there was no corresponding send-purp in the offer. The same applies for content types. Of course, since all agents have to support the baseline content type for a particular purpose, there will always be at least one type overlapping for each purpose between the offerer and answerer. In addition, if a particular purpose isn’t listed as a valid send purpose in one direction and receive in the other, it won’t end up being utilized for the TOTE session.

6. Session Management

A TOTE session persists for the duration that the offer/answer exchanges have signaled an active TOTE session. The actual TCP connection for the TOTE session is managed as defined in ICE-tcp [I-D.ietf-mmusic-ice-tcp]. This includes establishment, re-establishment, and termination of the TCP connection.

The usage of ICE-tcp implies that connections can be established end-to-end through firewalls and NATs, possibly with the aid of a TURN relay. ICE-tcp also allows for connections to be re-established mid-session without requiring additional signaling.

7. Sending and Receiving Messages

Once the TOTE session has been established, messages can be exchanged. Either agent can send a message at any time, with the following constraints:

- An agent MUST only send objects whose purpose was listed in a send-purp attribute placed by that agent in an SDP, and for which there was a recv-purp attribute in the SDP generated by its peer.

- An agent MUST only send objects whose content type was listed in a send-purp attribute placed by that agent in an SDP or the selected purpose, and for which there was a recv-purp attribute in the SDP generated by its peer for the selected purpose.

- The object is permitted to be sent at this time based on any constraints defined by the semantics of the purpose itself.

Sending an object via TOTE is easy. The purpose, length and content-
types are filled into the message. The length value represents the length of the entire TOTE message, in bytes, starting at the the first TOTE header after the length itself, and ending at the end of the body. Consequently, the length of the body is equal to the remaining length of the message after the headers.

For example, consider the following TOTE message:

```
1:37
p:name
t:text/plain

Jonathan Rosenberg

This TOTE message has a body containing a text/plain object, "Jonathan Rosenberg". The length field has a value of 37, which includes the 24 bytes of TOTE headers starting with the "p" and ending with, and including, the CRLF that terminates the headers, and then 18 bytes of body.

Note that the length value needs to be present and is used, even when TOTE is run in conjunction with ICE over TCP. ICE over TCP includes a shim layer, utilizing [RFC4571]. That shim defines a length field as well. However, that length is used only to separate the application data from the STUN messages. An actual TOTE message can span several RFC4571 frames; this is analogous in general to reads off of a TCP connection - the amount of data read at a time has no bearing on the length of the actual message. As a consequence of this, even though RFC4571 has a limit of 64k, there is no constraint on the length of a TOTE message.

When an agent receives an object, it processes it based on its purpose and content types.

8. Formal Syntax

8.1. SDP Extensions

This specification defines two new proto values for the m-line in SDP. These are "TOTE" and "TOTES". It also defines two new SDP attributes, send-purp and recv-purp:
8.2. TOTE Message Syntax

A TOTE message looks similar to a SIP message, but has a more restricted and constrained grammar:

```
tote-message    =    tote-header CRLF tote-body
        tote-header     =    len-hdr CRLF
                            purp-hdr CRLF
                            type-hdr CRLF
                            *(ext-hdr CRLF)
        len-hdr         =    "l" "" : 1*50DIGIT
        purp-hdr        =    "p" "" : purpose
        type-hdr        =    "t" "" : media-type
        ext-hdr         =    hdr-name "" : hdr-value
        hdr-name        =    token
                            ;token from RFC3261
        hdr-value       =    *(TEXT-UTF8char / UTF8-CONT)
                            ;TEXT-UTF8char from RFC3261
                            ;UTF8-CONT from RFC3261
        tote-body       =    *OCTET
```

In particular, it eliminates the gratuitous usage of whitespace that is used in SIP, and defines a fixed order for the mandatory headers.
It also defines a length that is always in a fixed position in the TOTE message (starting at the third byte). The length is in ASCII and can contain up to a 50 digit number.

9. Possible Purposes

This specification itself doesn't define any TOTE purposes. However, there are a few interesting possibilities that are worth mentioning.

9.1. Session Mode IM

It is possible to use TOTE as a mechanism for session mode IM. This would make it an alternative to MSRP [RFC4975]. The principle difference is that, in TOTE, message relays would be through TURN, rather than using an IM-specific relay (the MSRP relay). Furthermore, in sessions between different domains, there will always be a unique TCP connection per session that runs between the domains, rather than a shared connection, which is normally the case for MSRP. However, it is the use of a TCP connection per session which allows TOTE to work without much of the transport type of functionality present in MSRP (chunking, interruption, etc.).

The author is not, at this time, proposing to formally define a TOTE purpose for IM, but rather, is mentioning it for purposes of discussion.

9.2. Shared Browsing

Another possible application is shared web browsing. User A could connect to user B, and establish a TOTE session for shared browsing. Whenever user A goes to a website, the retrieved HTML file is sent to user B over TOTE. User B can then render the HTML. Whenever user A clicks on a hyperlink, or otherwise attempts an http query (for example as a result of some AJAX operations), instead of retrieving the URL, the actual URL is passed back to user A, and then user A actually performs the fetch.

9.3. Forms and Application Interaction

The application interaction framework [I-D.ietf-sipping-app-interaction-framework] defines a model where applications that wish to interact with users can REFER them to HTTP URLs, and then use traditional HTTP techniques to provide stimulus-based user interaction.

However, the app interaction framework has the problem that it doesn’t work effectively for applications which are behind firewalls
and NATs; this is because of the difficulties in extending HTTP connections to servers behind firewalls and NAT. Another practical drawback, is that it requires support for GRUU in endpoints, something that is not yet widely used.

An alternative solution is to use TOTE. This solution would only work when the application is the other participant in the session. The app-interaction framework is more general in this regard, allowing multiple applications to interact with the user, not just the one that is on the terminating side of the dialog.

With TOTE, the application would send to the user an HTML page to render. When the user enters form data or clicks on a hyperlink, instead of causing an HTTP post or GET to a URL, the actual URL is sent as an object to the application via TOTE. This is similar to the shared web browsing approach above. When the URL is received by the application, it is processed just as if it had been clicked by the user, and a new page is pushed to the client.

A significant benefit of this architecture is that the application can asynchronously push new web content to the user at any time; TOTE is a bidirectional channel, unlike HTTP which is client-server only.

### 9.4. Conference Controls

TOTE could be utilized as a control channel for conferences. For example, instead of using a separate conference event package subscription [RFC4575], a TOTE session can be used. The same XML object defined in [RFC4575] can be sent over TOTE. This would avoid the need to have a separate SIP exchange and SIP dialog. The tote session could be setup along with the media for the conference itself.

In addition, the client can send TOTE messages to invoke conference commands - requests to add or remove users, requests to mute or unmute, and so on. In essence, TOTE could be used as the transport for the conference control operations defined in [I-D.ietf-xcon-common-data-model].

### 9.5. Picture Sharing

A less controversial usage would be picture sharing. During a call, a user could send a picture to the person on the other end of the call, for purposes of rendering.
10. Security Considerations

TOTE could be used to exchange all kinds of data for different purposes. It is reasonable that many of these purposes will have requirements for authenticity, encryption, and message integrity. TOTE requires endpoints to support end-to-end TLS utilizing [RFC4572]. This provides all of those services. Furthermore, the end-to-end TLS works even in the presence of packet intermediaries (namely TURN servers) for the purposes of firewall and NAT traversal.

11. IANA Considerations

This specification has several IANA considerations.

11.1. TOTE Purpose Registry

This specification instructs IANA to create a new registry for TOTE purpose tokens. This registry is defined as a table that contains three columns:

<table>
<thead>
<tr>
<th>Purpose Token</th>
<th>Description</th>
<th>Document</th>
</tr>
</thead>
<tbody>
<tr>
<td>This will be a string provided in the IANA registrations into the registry.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>This is text that is supplied by the IANA registration into the registry.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>This is a reference to the RFC containing the registration.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

This specification instructs IANA to create this table with no initial entries. The resulting table would look like:

[[NOTE TO IANA/RFC-EDITOR: Please replace XXXX with the RFC number of this specification.]]

TOTE tokens are registered by the IANA when they are published in an RFC. The IANA Considerations section of the RFC must include the following information, which appears in the IANA registry along with the RFC number of the publication.
- Name of the TOTE Token. The name MAY be of any length, but SHOULD be no more than twenty characters long. The name MUST consist of alphanum and mark [RFC3261] characters only.

- Descriptive text that describes the purpose.

### 11.2. SDP Transport Protocol

TOTE defines the new SDP protocol field values "TOTE" and "TOTES", which should be registered in the sdp-parameters registry under "proto". This first value indicates the TOTE protocol when TCP is used as an underlying transport. The second indicates that TLS over TCP is used.

Specifications defining new protocol values must define the rules for the associated media format namespace. The "TOTE" and "TOTES" protocol values allow only one value in the format field (fmt), which is a single occurrence of "*". Actual format determination is made using the "send-purp" and "recv-purp" attributes.

### 11.3. SDP Attribute Names

This document registers the following SDP attribute parameter names in the sdp-parameters registry. These names are to be used in the SDP att-name field.

#### 11.3.1. Send Purposes

Contact Information: Jonathan Rosenberg, jdrosen@jdrosen.net

Attribute-name: send-purp

Long-form Attribute Name: TOTE Send Purposes

Type: Media level

Subject to Charset Attribute: No

Purpose and Appropriate Values: The "send-purp" attribute contains a list of media types that the endpoint is willing to send for a particular purpose. It contains one or more registered media-types.

#### 11.3.2. Receive Purposes
Contact Information: Jonathan Rosenberg, jdrosen@jdrosen.net

Attribute-name: recv-purp

Long-form Attribute Name: TOTE Receive Purposes

Type: Media level

Subject to Charset Attribute: No

Purpose and Appropriate Values: The "recv-purp" attribute contains a list of media types that the endpoint is willing to receive for a particular purpose. It contains one or more registered media-types.

12. Acknowledgements

I’d like to thank Jonathan Lennox for his comments and support.

13. References

13.1. Normative References


[I-D.ietf-mmusic-ice-tcp]

Description Protocol (SDP)", RFC 4572, July 2006.


13.2. Informative References

[I-D.ietf-sipping-app-interaction-framework]

[I-D.ietf-sip-info-events]


[I-D.ietf-xcon-common-data-model]
Author’s Address

Jonathan Rosenberg
Cisco
Iselin, NJ
US

Email: jdrosen@cisco.com
URI:  http://www.jdrosen.net