MMUSIC Working Group   M. Garcia-Martin
Internet-Draft   Nokia Siemens Networks
Intended status: Standards Track   J. Ott
Expires: August 21, 2008   Helsinki University of Technology
                                                      February 18, 2008

Session Description Protocol (SDP) Extensions and Conventions for Collaboration Applications
draft-garcia-mmusic-sdp-collaboration-00.txt

Abstract

The Session Description Protocol (SDP) is typically used in conjunction with the Session Initiation Protocol (SIP) for establishing multimedia sessions on the Internet. Typically these sessions include audio, video or messaging streams. Rich collaboration applications can complete these multimedia sessions, including view sharing with remote manipulation, co-web browsing, and file transfer.This document discusses rich collaboration between two endpoints and provides conventions and extension to SDP to enable these applications.

Table of Contents

1. Introduction

The Session Initiation Protocol (SIP) [RFC3261] is used for establishing and tearing down multimedia sessions in the Internet. Typically a SIP INVITE request carries an Session Description Protocol (SDP) [RFC4566] offer that describes the set of media streams that the caller wants to establish. The SDP offer gets answered with an SDP answer, as per procedures of the SDP offer/ answer model [RFC3264], and the multimedia session is eventually established. A session, according to RFC 4566, is a set of multimedia senders and receives and the data streams flowing from senders to receivers.

Media descriptions in SDP are signaled in the "m=" line. Each "m=" line contains a media type. SDP currently defines "audio", "video", "text", "application", and "message" media types. While establishing a session that contains audio, video, text, or message is straight forward, there are certain media streams that cannot be established without further specification. This memo tries to fill this gap.

Let us take a look at some use cases. Alice has already setup a multimedia session with Bob and they are exchanging audio and video. Then Alice requests Bob to help her setting up her computer. This is sometimes called "remote assistance". In practice, the use case can be implemented with a view sharing with remote manipulation protocol. It is not the scope of this memo to define a new view sharing with remote manipulation protocol, but instead, to re-use the rendezvous capabilities that SIP and SDP provide for setting up such media stream. Then Alice sends a re-INVITE request to Bob where she adds the description for a view sharing with remote manipulation protocol in the SDP. Once Bob accepts the session, an view sharing with remote manipulation application is launched at both endpoints. They exchange data over a given protocol. Then, Bob can assist Alice in setting up her computer.

In another scenario, Alice has established a multimedia session with Bob. That session includes audio and video media streams. Alice would like to browse a few web pages together, essentially, share with Bob the addresses of the web pages that she is visiting. This is sometimes referred to as co-web browsing.

Another common feature in rich collaboration includes the transfer of file. The file can include, for example, a screen shot, an image file, a document, etc.

NOTE: We are looking at a few examples of sharing content between two users. The list of examples and use cases might not be comprehensive enough at this stage. We need to find other

relevant formats.

The rest of this document is organized as follows: Section 3 provides an introduction to view sharing with remote manipulation media transport protocols. Section 4 provides the procedures to describe collaboration applications in SDP. Section 5 contains the format syntax.

2. 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 BCP 14, RFC 2119 [RFC2119] and indicate requirement levels for compliant implementations.

3. View Sharing with Remote Manipulation Media Transport Protocols

Many of the collaboration activities require some sort of view sharing with remote manipulation media protocol that is able to provide the required functionality. The approach taken by this memo is to re-use existing view sharing with remote manipulation media protocols whenever possible. On doing so, the scope is mostly reduced to specify the conventions for describing view sharing with remote manipulation protocols for the different use cases.

This section provides an overview of two set of protocols for which we provide descriptions in SDP later. These are the Remote Frame Buffer (RFB) protocol and ITU-T recommendation T.120 [ITU.T120.2007].

3.1. Overview of the Remote Frame Buffer (RFB) protocol

The Remote Frame Buffer (RFB) protocol [RFB] is a simple protocol for remote access to graphical user interfaces. RFB is used by Virtual Network Computing (VNC) applications and their successors. It works at the frame-buffer level, which roughly corresponds to the rendered screen image, which means that it can be applied to a variety of graphical systems. Remote Frame Buffer applications exist for many platforms, and can often be freely re-distributed.

The protocol operates with the classical client-server architecture. The application that runs on the machine where the user sits (containing the display, keyboard and pointer) is called the client. The application that runs on the machine where the framebuffer is located (which is running the windowing system and applications that the user is remotely controlling) is called the server. The client

accesses the remote graphical user interface and provides input events (keyboard, mouse), and the server provides the screen results.

RFB display protocol is based on a single primitive "put a rectangle of pixel data at a given x,7 position". The protocol allows different encodings for the pixel data, providing a large degree of flexibility in how to trade off various parameters, such as network bandwidth, client drawing speed and server processing speed.

A sequence of rectangles makes a framebuffer update, which represents a change from one valid framebuffer state to another. A sequence of framebuffer updates gives the user a perception similar to video.

The input side of the protocol is based on a standard workstation model of a keyboard an multi-button pointing device. Input events are simply sent to the server b the client whenever the user presses a key o pointer button, or whenever the pointing device is moved. These input events can also be synthesised from other non-standard input/output devices, for example, a pen-base handwriting device.

RFB includes a client server negotiation process that leads on the agreement of the format and encoding of the pixel data. Pixel format refers to the representation of individual colours by pixel values. The most common pixel formats are 24-bit or 16-bit "true colour", where 8-bit pixel format is typically used in scenarios with low bandwidth connectivity.

Encoding refers to how a rectangle of pixel data is sent on the wire. Every rectangle of pixel data is prefixed by a header giving the X,Y position of the rectangle on the screen, the width and height of the rectangle, and en encoding type which specifies the encoding of the pixel data. The data itself then follows using the specific encoding, for example, Raw, CopyRect, RRE, Hextile, and ZRLE.

RFB typically runs over TCP [RFC0793]. Once a TCP connection is establish, the client-server initialization takes place, followed by an authentication and authorization. Then the actual transfer of framebuffer updates can take place.

The protocol is extensible. New encoding, new pseudo-encoding, and new security types can be added when need arises.

3.2. Overview of T.120

T.120 [ITU.T120.2007] defines multipoint conferencing protocols and services for data applications including whiteboard (T.126 [ITU.T126.2007]), file transfer (T.127 [ITU.T127.2007]), application sharing (T.128 [ITU.T128.1998]), and text conversation (T.134

[ITU.T134.1998], T.140 [ITU.T135.2007]), among others.

In T.120, communication takes place in a hierarchical structure of T.120 entities interconnected by T.120 connections. In the simplest case, this may be just two entities interconnected via a T.120 connection, in a slightly more sophisticated one a single conference bridge (MCU) may be used to interconnect more than two entities. Each point-to-point T.120 connection uses TCP as underlying transport and TPKT (RFC 1006 [RFC1006]) framing on top. Up to four TCP connections may be established between any two T.120 entities to provide independent flow control for different transmission priorities.

The lowest layer above the point-to-point transport is the Multipoint Communication Service (MCS) (T.122 [ITU.T122.1998] and T.125 [ITU.T125.1998]). The MCS defines a connection setup procedure that allows to associate different transport connections with the same MCS Domain and to organize the MCS "providers" in a tree structure during connection setup. In the resulting tree, the MCS provides a multiplex for application data (using up to 64K channels) and simple means for synchronizations (tokens).

On top of MCS, the Generic Conference Control (GCC) [ITU.T124.2007] is responsible for controlling the connection setup and their association with a conference. Furthermore, GCC manages conference resources, provides capability exchange, allows for floor control, and provides some kind of a conference-wide registry. In GCC, conferences are identified by means of an octet string that is mapped to the MCS Domain identifier. GCC allows to create and destroy conferences as well as to inquire for, join, and leave existing ones.

T.120 data applications make use of the MCS communication platform to exchange information and of the GCC services to learn about each others existence and to find each other. In particular, GCC's capability exchange mechanism is used to discover commonly available applications and their respective features (with many tiny details being negotiable).

This memo is concerned with the definition of media stream descriptions in SDP that allows two nodes to learn about their respective T.120 capabilities and enable them to set up a single T.120 connection. Whether a single or more TCP connections are used and how those are associated if entirely left up to the respective T.120 entities, as is most of the T.120-specific capability exchange.

4. Collaboration applications

4.1. SDP Extensions for View Sharing with Remote Manipulation Applications

According to SDP [RFC4566], the media descriptions line is defined as follows:

```
m=<media> <port> <proto> <fmt> ...
```

Protocols used for view sharing with remote manipulation MUST use the media type "application" in the "m=" line. Several protocols for view sharing with remote manipulation are available today. We provide the means to describe the Remote Frame Buffer (RFB) protocol [RFB] and ITU-T T.120 [ITU.T120.2007] series of recommendations, in particular, application sharing as defined by ITU-T T.128 [ITU.T128.1998] recommendation. Other protocols can be added at a later time when need arises.

Both RFB and T.120 typically run over TCP [RFC0793], so, for the purpose of describing an RFB or T.120 session in SDP, it is required to use the TCP-based media transport extensions for SDP specified in RFC 4145 [RFC4145] for negotiating which endpoint establishes the TCP connection. Both protocols can also use TLS [RFC4346] as a secure transport protocol.

4.1.1. View Sharing with Remote Manipulation with RFB

An application that wants to describe a session where RFB is the protocol creates an SDP offer that contains a media description line "m=". The media type is set to "application". The port number is set to the client or server port number with the considerations indicated in RFC 4145 [RFC4145]. The <proto> MUST be set to the transport protocol mnemonic (e.g., "TCP") followed by a slash "/" and the mnemonic "RFB". The defined values in the protocol field are "TCP/RFB" for and "TCP/TLS/RFB" for TLS [RFC4346] over TCP. Since RFB contains built-in negotiation mechanisms for the different encodings, there is no need to signal a media format description, but for compatibility issues, the SDP offerer MUST include an asterisk sign "*" as a media format description at the end of the "m=" line. If TCP or other connection-oriented transport is used, the SDP offerer MUST add a 'setup' and 'connection' attributes as per procedures of RFC 4145 [RFC4145].

4.1.1.1. Example of RFB descriptions in SDP

The following is an example of an SDP offer for the RFB protocol:

```
v=0
o=alice 2890844526 2890844526 IN IP4 host.atlanta.example.com
s=
c=IN IP4 host.atlanta.example.com
t=0 0
m=application 42034 TCP/RFB *
a=setup:active
a=connection:new
```

Figure 1: Example of an SDP offer for setting up an RFB media stream

An example of an SDP answer to the above offer is:

```
v=0
o=bob 2890844527 2890844527 IN IP4 host.biloxy.example.com
s=
c=IN IP4 host.biloxy.example.com
t=0 0
m=application 5900 TCP/RFB *
a=setup:passive
a=connection:new
```

Figure 2: Example of an SDP answer for setting up an RFB media stream

4.1.2. View Sharing with Remote Manipulation with T.120

An application that wants to describe a session with T.120 as the protocol for view sharing with remote manipulation creates an SDP offer that contains a media description line "m=". The media type is set to "application". The port number is set to the client or server port number with the considerations indicated in RFC 4145 [RFC4145]. The <proto> MUST be set to the transport protocol mnemonic (e.g., "TCP"), followed by a slash "/" and the mnemonic "T120". The defined protocol field values are "TCP/T120" for TCP transport and "TCP/TLS/T120" for TLS [RFC4346] over TCP.

4.1.2.1. T.120-specific Attributes

T.120 connections established within the context of an SDP offer/answer exchange may need to be associated with a SIP dialog or a SIP conference. Therefore, the media-level 'confname' attribute is introduced. The content of the 'confname' attribute MUST be copied to the ConferenceName field used by the GCC entity. For a simple SIP dialog, the 'confname' attribute SHOULD be created with the concatenation of the SIP From tag, the To tag, the value of the Call-ID header field. For a SIP dialog in a conference, the 'confname' attribute SHOULD contain the focus URI. Whenever a character cannot be written according to the syntax of the 'confname'

attribute, that character should be percent encoded.

T.120 defines a number of applications. To determine from the offer/answer exchange whether or not at least the target application(s) are available at a peer, the optional "t120apps" attribute is introduced. If present, this attribute SHOULD contain a list of T.120 application protocols supported by the respective peer. The following application protocols are defined:

"t126": it indicates whiteboard sharing according to ITU-T T.126 [ITU.T126.2007].
"t127": it indicates file transfer according to ITU-T T.127 [ITU.T127.2007].
"t128": it indicates view sharing and remote manipulation according to ITU-T T.128 [ITU.T128.1998].
"t134": it indicates text conversation according to ITU-T T.134 [ITU.T134.1998].
"t136": it indicates remote device control according to ITU-T T.136 [ITU.T136.1999].
"t137": it indicates virtual room management according to ITU-T T.137 [ITU.T137.2000].

The fine-grained negotiation of capabilities within these application protocols is left up to the GCC operation after setup of a T.120 connection.

4.1.2.2. Example of T.120 descriptions in SDP

In the following example, an SDP offerer wants to set up an view sharing with remote manipulation stream.

```
v=0
o=alice 2890844526 2890844526 IN IP4 host.atlanta.example.com
s=
c=IN IP4 host.atlanta.example.com
t=0 0
m=application 23093 TCP/T120 *
a=setup:active
a=connection:new
a=confname:623847692,789234789,78687afded278bd@example.com
a=t120apps:t128
```

Figure 3: Example of an SDP offer for setting up a T.120 media stream

4.1.2.3. SDP Offer/Answer Operation with T.120

An SDP offerer wishing to set up a T.120 session MUST include an m= line according to Section 4.1 and include an appropriate c= line. The offerer MUST apply the procedures of RFC 4145 [RFC4145] for managing connection oriented protocols. The offerer MUST provide the 'confname' attribute as specified in Section 4.1.2.1 and SHOULD include the list of supported T.120 application protocols in a 't120apps' attribute.

An SDP answerer that receives an SDP offer for setting up a T.120 session MUST first examine that it supports T.120 protocol indicated in the 't120apps' attribute. If it does not support T.120 or does not wish to use T.120 in the present communication relationship, it MUST refuse the offered media stream by applying the procedures specified in RFC 3264 [RFC3264] Section 6. If the answerer supports T.120, it MUST validate the "confname" attribute. If no matching SIP dialog or focus URI can be found, the media stream offer SHOULD be refused by applying the procedures specified in RFC 3264 [RFC3264] Section 6. If a matching conference is found and the "t120apps" attribute is not present, the answerer MAY decide whether to accept the session and proceed with the transport connection setup as per RFC 4145 [RFC4145] or whether to refuse the media section. If the "t120apps" attribute is present, the answerer MUST examine its contents and match the applications listed by the offerer against its own capabilities. If the intersection of capabilities is empty, the answerer SHOULD refuse the media stream. If the intersection is not empty, the answerer SHOULD return the intersecting capabilities (in the order provided by the offerer) and then proceed with the transport connection setup as per RFC 4145 [RFC4145].

After successful establishment of a TCP connection as per RFC 4145 [RFC4145], the respective entities MUST proceed with the T.120 connection setup as defined in ITU-T T.123 [ITU.T123.2007], ITU-T T.124 [ITU.T124.2007], and ITU-T T.125 [ITU.T125.1998].

4.2. Co-Web Browsing

Co-web browsing allows endpoints in a multimedia session to visit the same web pages in nearly real-time. Co-web browsing could be implemented by sharing the web application, e.g., Alice could share her web browser or entire screen with Bob. However, this has some disadvantages. For example, since Bob is not running locally his web browser, he cannot save bookmarks, acquire cookies, record history of visited pages, etc. Bob might not be familiar with the exact web browser that Alice is using to surf the net, the size of the font might not be appropriate for Bob, or the color scheme that Alice is using. In some cases a better option might be more efficient and

convenient to just share the URL of the page that both either Alice or Bob are visiting, so that each endpoint becomes responsible for independently of each other downloading the content, and for rendering the web page in their respective web browser. This has advantages, such as the possibility of Alice and Bob to render the content in the capabilities of their browsers and according to the user preferences (size of font, colors) and also according to the capabilities of their devices (size of screen, number of colors of the screen, bandwidth, etc.). It also allows Alice and Bob to operate in their respective web browsers, such as record history, save bookmarks, acquire cookies, etc.

We acknowledge the fact that there is no guarantee that the same content is rendered equally in both web browsers. We merely try to achieve an automatic mechanism that simulates the fact that a user reads out or spells a URL, or copies the URL in a text messaging windows, sends it to the remote user, which also copies the URL and pastes it in their browser. If exact pixel-by-pixel copy of the display is needed, users should use the view sharing with remote manipulation feature described in Section 4.1.

To implement this use case we need to be able to signal the co-web browsing application in SDP. Additionally, we need a mechanism to transfer URLs in real-time between the two endpoints. This memo proposes to reuse Message Session Relay Protocol (MSRP) [RFC4975]. MSRP SEND messages can carry an XML document that includes the HTTP URL of the web page that Alice is visiting. Whenever Alice or Bob load a new web page, an MSRP SEND request conveys the URL to the remote endpoint. The co-web browsing application instructs then the web browser to load the received URL.

The co-web browsing application requires a protocol for transferring real-time messages that contain the URL that one of the endpoints is loading in the web browser. We use the Message Session Relay Protocol (MSRP) [RFC4975] as the real-time text message transport protocol. We define a new MIME body, using Extensible Data Format (XML), for wrapping the URL together with a time-stamp that indicates when the web XML document was created.

Since text messaging media streams established with MSRP can be already described in MSRP, this document does not introduce extensions to SDP to signal them.

This document introduces a new Co-Web Browsing XML document that encodes the URL linked to the web page rendered in a participant's web browser. Co-Web Browsing is an XML document [W3C.REC-xml-20060816] that MUST be well-formed and SHOULD be valid. Co-Web Browsing documents MUST be based on XML 1.0 and MUST be

encoded using UTF-8 [RFC3628]. This specification makes use of XML namespaces for identifying Co-Web Browsing documents. The namespace URI for elements defined by this specification is a URN [RFC2141], using the namespace identifier 'ietf' defined by RFC 2648 [RFC2648] and extended by RFC 3688 [RFC3688]. This URN is:

```
urn:ietf:params:xml:ns:co-web-browsing
```

Co-web browsing XML documents have an associated MIME content type of "application/co-web-browsing+xml". When a Co-web browsing XML document is included in an MSRP SEND message, the Content-Type header of the MSRP SEND request MUST be set to "application/co-web-browsing+xml".

A Co-Web Browser document begins with the root element tag <co-web-browsing>. It has two children elements. A <timestamp> element contains the time and date when the content was acquired. The recipient of the Co-web browsing document can use the <timestamp> element for correlation purposes, such as avoiding to download existing content that has not changed. For example, if the recipient has already downloaded the content, it can use a conditional HTTP request using the value in the <timestamp> in the If-Modified-Since and If-Unmodified-Since HTTP headers.

Last, a <url> element contains the URL of the content as seen from the sender.

Implementations according to the co-web browsing feature described in this memo MUST comply with the XML schema included in Section 5.2.1.

4.2.1. Example of a Co-Web Browsing document

The following is an example of a Co-Web Browsing document.

```
<?xml version="1.0" encoding="UTF-8"?>
<co-web-browsing>
    <timestamp>2008-01-27T16:49:29Z</timestamp>
    <url>http://wwww.example.com/index.html</url>
</co-web-browsing>
```

Figure 4: Example of a Co-Web Browsing document

4.3. File Transfer

The file transfer feature allows a user to offer a file to the remote user. The file is parametrized by its name, size, creation and modification dates, etc. The file offer can also include a small preview icon, for example, in case the file is an image or video

file. If the remote user accepts, then the file transfer operation takes place, typically using MSRP as the real-time transport protocol. The file transfer feature can be used, for example, to transfer screen shots that one user takes in his computer, or documents that the user wants to share with their correspondent.

File transfer is described in the memo SDP offer/answer to enable file transfer [I-D.ietf-mmusic-file-transfer-mech]. No further considerations are needed in this document.

Additionally, the T.120 group of specifications provides a file transfer capability: T.127 [ITU.T127.2007]. T.127 file transfer operations can be described in SDP as per procedures for T.120 applications.

4.4. Further Sharing Applications

The two application sharing approaches discussed above offer identical views to a single application or the entire remote desktop and hence are sensible for remote maintenance activities. However, they are inherently limited by offering only a single cursor, single undo history, etc. If collaboration inside a single document shall take place beyond this limited experience (e.g., allowing users to independently manipulate different parts of a document at the same time), the applications themselves must support sharing. Traditional examples have been the Mbone tools whiteboard (wb) and network text editor (nte) used with multicasting in the 1990s and the emacs support for multiple windows into a single buffer ("make-frame-on-display"). Also, presentation tools have offered "remote presentation" mechanisms in the past. Recent shared applications have been web-based, using a server as rendezvous point.

Note: In the following, we only give examples we are well aware of. The authors seek further input to expand this section and give proper coverage of other open tools.

Shared whiteboards have probably been among the most popular sharing applications. While a lot of research has gone into shared whiteboard applications particularly in the 1990s, not many open (standardized) solutions exist. ITU-T T.126 offers sophisticated image sharing, annotation, and pointing facilities and specifies a -- quite complex -- standardized protocol which requires the entire T.120 protocol suite underneath. T.126 is covered as per the above definition.

Web-based shared editors and similar tools (which run over HTTP) can use the mechanisms of co-web browsing described in the previous section.

Note: The use of web-based sharing applications raises an issue concerning the notion of a session in the context of a dialog. SDP usually uses m= lines to describe sessions and can hence open and close them as part of the offer/answer signaling. Conveying a URI of a shared editing application in an MSRP session, to a certain extent, also initiates a session in its own right which may be considered independent of a particular call.

5. Formal syntax

5.1. SDP extensions

We define a number of new SDP [RFC4566] attributes that provide the required information to describe the transfer of a file with MSRP. These are all media level only attributes in SDP. The following is the formal ABNF syntax [RFC5234] of these new attributes. It is built above the SDP [RFC4566] grammar.

```
attribute       = t120apps / confname
                  ;attribute is defined in RFC 4566

t120apps        = "t120apps:" t120appid *(SP t120appid)
t120appid       = "t126" / "t127" / "t128" / "t134" /
                  "t136" / "t137" / token
confname        = "confname:" token      ; token defined in RFC 4566
```

Figure 5: Syntax of the SDP extension

5.2. Co-Web browsing XML schema

Section 5.2.1 contains the XML schema of Co-web browsing XML documents. Implementations of such XML document MUST be compliant with that XML schema.

5.2.1. XML Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema
    targetNamespace="urn:ietf:params:xml:ns:co-web-browsing"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns="urn:ietf:params:xml:ns:co-web-browsing"
    elementFormDefault="qualified"
    attributeFormDefault="unqualified">
  <xs:import namespace="http://www.w3.org/XML/1998/namespace"/>
  <xs:annotation>
    <xs:documentation xml:lang="en">
        XML Schema Definition for Co-Web Browsing
    </xs:documentation>
  </xs:annotation>

  <xs:element name="co-web-browsing"
              type="co-web-browsingType"/>

  <xs:complexType name="co-web-browsingType">
    <xs:sequence>
       <xs:element name="timestamp"
                   type="xs:dateTime" minOccurs="0"/>
       <xs:element name="url" type="xs:anyURI" />
    </xs:sequence>
  </xs:complexType>
</xs:schema>
```

Figure 6: XML schema of Co-Web Browsing

6. Examples

TBD.

7. Security Considerations

This document defines additional attributes for SDP and conventions for using these with the offer/answer mechanism. The security considerations of SDP [RFC4566] and the offer/answer model [RFC3264] apply as do those for setting up TCP and TLS connections with SDP (RFC 4572 [RFC4572] and RFC 4145 [RFC4145].

Running (additional) application protocols inside a SIP dialog creates additonal targets for attack or potential leaks of private information. The security of the application part of the session depends on the security of the application protocols being used.

They may not only reveal information about the applications themselves, but may additionally leak information about other sessions inside the dialog and about the peers involved. Hence, it is suggested to use TLS protection for the application protocols if possible or turn on application-specific protocol mechanisms if available.

Furthermore, application sharing systems grant access to at least some parts (if not all) of the local environment (read-only, read/write). Co-web-browsing and whiteboard sessions may also do so to a limited extent. Therefore, security mechanisms (such as TLS) should be applied to application protocols and each peer should ensure that the respective connection truly originates and terminates at the intended remote party.

8. IANA Considerations

This document instructs IANA to register a number of SDP attributes according to the following:

8.1. Registration of new SDP attributes

This memo provides instructions to IANA to register a number of media level only attributes in the Session Description Protocol Parameters registry [1]. The registration data, according to RFC 4566 [RFC4566] follows.

Note to the RFC Editor: replace "RFC XXXX" with the RFC number of this specification.

8.1.1. Registration of the t120apps attribute

Contact: Miguel Garcia <miguel.garcia@nsn.com>
Phone: +358 71400 4000
Attribute name: t120apps
Long-form attribute name: T.120 Applications
Type of attribute: media level only
This attribute is subject to the charset attribute
Description: This attribute list the available T.120 applications.
Specification: RFC XXXX

8.1.2. Registration of the confname attribute

Contact: Miguel Garcia <miguel.garcia@nsn.com>
Phone: +358 71400 4000

Attribute name: confname
Long-form attribute name: Conference Name
Type of attribute: media level only
This attribute is subject to the charset attribute
Description: This attribute contains the conference name identifier
Specification: RFC XXXX

8.2. Registration of new SDP 'proto' values

This memo defines the new SDP protocol field values "TCP/RFB", "TCP/TLS/RFB", "TCP/T120", and "TCP/TLS/T120", which should be registered in the sdp-parameters registry under "proto".

8.3. MIME type registration

This specification requests IANA to register a new MIME type "application/co-web-browsing+xml" according to the procedures of RFC 4288 [RFC4288] and the guidelines in RFC 3023 [RFC3023].

MIME media type name: application

MIME subtype name: co-web-browsing+xml

Mandatory parameters: none

Optional parameters: Same as charset parameter application/xml as specified in RFC 3023 [RFC3023].

Encoding considerations: Same as encoding considerations of application/xml as specified in RFC 3023 [RFC3023].

Security considerations: See Section 10 of RFC 3023 [RFC3023].

Interoperability considerations: none

Published specification: RFC XXXX

Additional Information:

Magic Number: none
File Extension: none
Macintosh file type code: "TEXT"

Personal and email address for further information: Miguel Garcia, miguel.garcia@nsn.com

Intended usage: not common, restricted to sharing applications

Author/Change controller: The IETF.

8.4. URN Sub-Namespace Registration

This specification registers a new new XML namespace, as per the guidelines in RFC 3688 [RFC3688].

URI: urn:ietf:params:xml:ns:co-web-browsing

Registrant contact: IETF.

XML:

```
      BEGIN
      <?xml version="1.0"?>
      <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML Basic 1.0//EN"
         "http://www.w3.org/TR/xhtml-basic/xhtml-basic10.dtd">
      <html xmlns="http://www.w3.org/1999/xhtml">
      <head>
        <meta http-equiv="content-type"
           content="text/html;charset=iso-8859-1"/>
        <title>Co-web browsing namespace</title>
      </head>
      <body>
        <h1>Namespace for Co-web browsing XML Documents</h1>
        <h2>urn:ietf:params:xml:ns:co-web-browsing</h2>
        <p>See <a href="http://www.rfc-editor.org/rfc/rfc4825.txt">
           RFC4825</a></p>
      </body>
      </html>
      END
```

8.5. XML Schema Registration

This section registers a new XML schema per the procedures in RFC 3688 [RFC3688].

URI: urn:ietf:params:xml:schema:co-web-browsing

Registrant Contact: IETF

XML Schema: The XML for this schema can be found as the sole content of Section 5.2.1.

9. Acknowledgements

10. References

10.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC2141] Moats, R., "URN Syntax", RFC 2141, May 1997.

[RFC3264] Rosenberg, J. and H. Schulzrinne, "An Offer/Answer Model with Session Description Protocol (SDP)", RFC 3264, June 2002.

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, January 2004.

[RFC4145] Yon, D. and G. Camarillo, "TCP-Based Media Transport in the Session Description Protocol (SDP)", RFC 4145, September 2005.

[RFC4566] Handley, M., Jacobson, V., and C. Perkins, "SDP: Session Description Protocol", RFC 4566, July 2006.

[RFC4572] Lennox, J., "Connection-Oriented Media Transport over the Transport Layer Security (TLS) Protocol in the Session Description Protocol (SDP)", RFC 4572, July 2006.

[RFC5234] Crocker, D. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, January 2008.

[W3C.REC-xml-20060816] Maler, E., Paoli, J., Bray, T., Sperberg-McQueen, C., and F. Yergeau, "Extensible Markup Language (XML) 1.0 (Fourth Edition)", World Wide Web Consortium Recommendation REC-xml-20060816, August 2006, <http://www.w3.org/TR/2006/REC-xml-20060816>.

10.2. Informative References

[RFC0793] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, September 1981.

[RFC1006] Rose, M. and D. Cass, "ISO transport services on top of the TCP: Version 3", STD 35, RFC 1006, May 1987.

[RFC2648] Moats, R., "A URN Namespace for IETF Documents", RFC 2648, August 1999.

[RFC3023] Murata, M., St. Laurent, S., and D. Kohn, "XML Media Types", RFC 3023, January 2001.

[RFC3261] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, June 2002.

[RFC3628] Pinkas, D., Pope, N., and J. Ross, "Policy Requirements for Time-Stamping Authorities (TSAs)", RFC 3628, November 2003.

[RFC4288] Freed, N. and J. Klensin, "Media Type Specifications and Registration Procedures", BCP 13, RFC 4288, December 2005.

[RFC4346] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.1", RFC 4346, April 2006.

[RFC4975] Campbell, B., Mahy, R., and C. Jennings, "The Message Session Relay Protocol (MSRP)", RFC 4975, September 2007.

[I-D.ietf-mmusic-file-transfer-mech]
Garcia-Martin, M., Isomaki, M., Camarillo, G., Loreto, S., and P. Kyzivat, "A Session Description Protocol (SDP) Offer/Answer Mechanism to Enable File Transfer", draft-ietf-mmusic-file-transfer-mech-06 (work in progress), December 2007.

[RFB] Richardson, T., "The Remote Frame Buffer Protocol version 3.8", June 2007, <http://www.realvnc.com/docs/rfbproto.pdf>.

[ITU.T120.2007]
ITU-T, "Data Protocols for Multimedia Conferencing", ITU-T Recommendation T.120, January 2007.

[ITU.T122.1998]
ITU-T, "Multipoint communication service protocol specification - Service definition", ITU-T Recommendation T.122, February 1998.

[ITU.T123.2007]
ITU-T, "Network-specific data protocol stacks for multimedia conferencing", ITU-T Recommendation T.123, January 2007.

[ITU.T124.2007]
ITU-T, "Generic Conference Control", ITU-T Recommendation T.124, January 2007.

[ITU.T125.1998]

ITU-T, "Multipoint communication service protocol specification", ITU-T Recommendation T.125, August 2007.

[ITU.T126.2007]
ITU-T, "Multipoint still image and annotation protocol", ITU-T Recommendation T.126, August 2007.

[ITU.T127.2007]
ITU-T, "Multipoint binary file transfer protocol", ITU-T Recommendation T.128, August 2007.

[ITU.T128.1998]
ITU-T, "Multipoint Application Sharing", ITU-T Recommendation T.128, February 1998.

[ITU.T134.1998]
ITU-T, "Text chat application entity", ITU-T Recommendation T.134, February 1998.

[ITU.T135.2007]
ITU-T, "User-to-reservation system transactions within T.120 conferences", ITU-T Recommendation T.135, August 1998.

[ITU.T136.1999]
ITU-T, "Remote device control application protocol", ITU-T Recommendation T.136, May 1999.

[ITU.T137.2000]
ITU-T, "Virtual meeting room management for multimedia conferencing audio-visual control", ITU-T Recommendation T.137, February 2000.

URIs

[1] <http://www.iana.org/assignments/sdp-parameters>

Authors' Addresses

Miguel A. Garcia-Martin
Nokia Siemens Networks
P.O.Box 6
Nokia Siemens Networks, FIN 02022
Finland

Email: miguel.garcia@nsn.com

Joerg Ott
Helsinki University of Technology
Otakaari 5 A
Espoo, FIN 02150
Finland

Email: jo@netlab.hut.fi

Full Copyright Statement

Intellectual Property

Acknowledgment

Funding for the RFC Editor function is provided by the IETF Administrative Support Activity (IASA).