SIP WG
Internet-Draft
Expires: April 28, 2003

C. Jennings
Cisco Systems
October 28, 2002

SIP Support for Application Initiation
draft-jennings-sip-app-info-00

Status of this Memo

Copyright Notice

Abstract

This document describes SIP extensions to allow network elements to request a UA to initiate a scripted application that is associated with a dialog. It provides a mechanism for the network elements to find out a UA's ability to fetch and execute scripts.

Table of Contents

1. Conventions

A "script" refers to some markup, program, or script that the UA can fetch and execute.

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 [2].

2. Introduction

The Session Initiation Protocol (SIP) [1] provides the ability for users to initiate, manage, and terminate communications sessions. Frequently, these sessions will involve a SIP application. A SIP application is defined as a program running on a SIP-based element (such as a proxy or user agent) that provides some value-added function to a user or system administrator. Examples of SIP applications include pre-paid calling card calls, conferencing, and presence-based [4] call routing.

In order for most applications to properly function, they need input from the user to guide their operation. For example, a pre-paid calling card application requires the user to input their calling card number, their PIN code, and the destination number they wish to reach. The process by which a user provides input to an application is referred to as "application interaction".

A set of high level requirements on a system for application interaction are described in [9]. To meet these requirements, a framework has been developed[11]. In this framework, applications can instantiate user interface components on client devices, for the purposes of interacting with the user. These user interface components are described using markup languages, such as VoiceXML and KPML [10]. The framework also defines a set of requirements for SIP extensions that allow for an application to discover the capabilities of the user device for supporting markup languages, for placing user interface components on the device, and for terminating the component. This document proposes a specific SIP extension that fulfills those requirements. This extension is the App-Info header.

3. Overview

The main mechanism of this draft is a new header field, called App-Info, that provides the UA with the URL of a script to execute. A network element can add this header field. The App-Info header field can occur in most SIP messages, including INVITE and MESSSAGE messages, as well as in reliable provisional responses.

This draft also defines several option tags for use in the supported header field to allow a UA to indicate which URL schemes are supported for fetching scripts. The Accept header field is used to indicate the types of markup that the UA can process.

An example App-Info header field is:

```
App-Info: "Bodgey Call Timer"
           <http://mediasvr.provider.net/calltimer.vxml>;
           id=app4323!sub4+svr56.provider.net;
```

This indicates that the UA should fetch and execute the script found at http://mediasvr.provider.net/calltimer.vxml. A key part of the header field value is an application id that consists of an application instance and an application class separated by a "!". In the example above, the instance is "app4323" and the class is "sub4+src56.provider.net". The combination of these two MUST make a globally unique identifier. There may be multiple user interface components running on a UA that are part of the same application instance, and therefore, share the same instance identifier. The instance identifier can be used to correlate the applications. The UA may use the display name for presentation purposes and for help in managing focus, but it has no other meaning. The formal syntax for the App-info header field is presented in Section 6.

This approach also uses the Supported and Accept header fields as well as the schemes mechanism from the caller prefs draft [7]. For example:

```
Supported: markup
Accept: multipart/mixed, application/vxml, text/html
  Contact: sip:1.2.3.4;schemes="http,cid,file"
```

This indicates that the UA can accept markup as defined in this draft. In particular the UA can accept VoiceXML and HTML markup and it is capable fetching scripts from using a http, cid, or file scheme. The cid scheme[8] fetches the content from an inline body in the same message. The http scheme and cid scheme SHOULD be supported.

4. User Agent Server Behavior

When a UAC sends a message it MUST include in the Supported header the script fetching URL types that it supports and the markup option tag. It MUST also put the markups or scripts that it can process in the Accept header field and indicate the schemes it can support.

When a UA receives a message that contains an App-Info header field, it must process each header field value and decide what to do with it. There are three cases: creating a new application, updating an existing application, and stopping a script that has been previously started.

In the case that the application identifier does not match any of the scripts that are currently running, a new application instance is created. The UA associates the application identifier with the dialog it was received on. The script is fetched. If the fetch fails for some reason, no error is reported by the UA. After fetching the script, execution starts in a context associated with the dialog.

If the application identifier matches an the identifier for a previously fetched script and the App-Info header field URL value has does not match the previous header field URL value for this script then the script is fetched and then used to replace the existing script. If the application identifier matches an existing script and the URL in the App-Info header field value is empty, then the existing script is terminated. If the URL has not changed, this header field value is ignored.

The UA fetches the script by using the URI found in the uri portion of the App-Info header field value. A UA which supports the App-Info header field SHOULD support fetching scripts from multipart MIME bodies using the cid scheme and SHOULD support the http[12] scheme.

When a dialog ends, all the applications associated with it SHOULD be immediately terminated.

A UA may add the App-Info header field to initiate an application on the other UA in the dialog. The UA SHOULD NOT request services that the other UA has not indicate it supports.

5. Proxy Behavior

Proxies may add header field values to the App-Info header field but they SHOULD NOT delete or modify any existing header field values that they did not originally add. App-Info header fields can be added to reliable provisional response. The proxy SHOULD NOT request services that the UA has not indicate it supports.

6. Formal Syntax

The following syntax specification uses the augmented Backus-Naur Form (BNF) as described in RFC-2234 [3].

6.1 The App-Info Header

```
App-Info    = "App-Info" HCOLON app *(COMMA app)
app         = [ display-name ] LAQUOT [absolute-uri] RAQUOT
                                      *(SEMI app-param)
app-param   = app-id-param / app-name-param / generic-param
app-id-param = "id" EQUAL app-id-value
app-id-value = app-instance-id "!" app-class-id
app-instance-id = app-token
app-class-id = app-token
app-token    = 1*(alphanum / "-" / "." / "%" / "*" / "_" / "+"
               / "'" / "`" / "~" ) ; this is a token with no "!"
app-name-param = "app-name" EQUAL gen-value
app-msg-id = msg-id ; as defined in RFC 2822
```

This document adds the following entry to Table 2 of RFC-3261 [1].

Header field	where	proxy	ACK	BYE	CAN	INV	OPT	REG
App-Info		adr	-	o	-	o	o	-

	SUB	NOT	REF	INF	UPD	PRA
	o	o	o	o	o	-

In addition it would be listed as an optional header for the MESSAGE message so this document adds the following line to Table 1 in draft-ietf-sip-message [6].

Header Field	where	proxy	MESSAGE
App-Info			o

7. Security Considerations

This document describes a mechanism that allows non trusted parties to request that a UA execute an arbitrary script. This mechanism should only be used to initiate scripts that are in scripting languages that are intended for situations in which scripts from non trusted parties are expected. HTML is a good example of a markup language that is considered safe to render content that is not trusted. A scripting language that allowed scripts that automatically caused the UA to hang up and then dial a toll service phone number would certainly not be appropriate for this mechanism.

The scripting language should not be able to access information on the UA that is not associated with the dialog, such as the user's address book. There is no reason the scripting mechanism could not have an authorization mechanism such as signed scripts, but this is not provided by the mechanisms in this document.

8. IANA Considerations

8.1 Registration of App-Info header

This document defines a new header field, "App-Info". As recommended by RFC-3261 [1] these headers fields should be registered by the IANA in the SIP header registry, using the RFC number of this document as its reference.

Name of Header:	App-Info
Short form:	none
Registrant:	Cullen Jennings fluffy@cisco.com
Normative description:	Section 6.1 of this document

8.2 IANA Registration of Option Tags

This specification registers a new option tags. The required information for this registration, as specified in RFC-3261 [1], is:

Name: markup
Description: This option tag is for fetching scripts into a UA. When present in a Supported header field, it indicates that the UA can supports the mechanism in RFC XXXX.
{NOTE to IANA: Please replace XXXX with the rfc number of this specification}

Registrant: Cullen Jennings
fluffy@cisco.com

9. Open Issues

If there is an error fetching a script, should any error be returned on the dialog?

If a dialog A exists or is being created, it should be possible to create a sidebar dialog B that can create an application instant that is associated with the main dialog A. We need to think about security for this.

When a dialog ends, should all the scripts associated with it be instantly terminated or should they be given a chance to notify some application server that they are ending? If the later is done, it introduces the possibility of the telephone equivalent to those HTML windows that just keep coming back no matter how many times you kill them.

Do we need a way to indicate that the UA can support forking media, local mixing, or media policy?

There are likely requirements for authenticating the application that are not addressed here. They should be addressed.

The schemes that are supported could be specified in a new header called Allow-Schemes instead of using the caller prefs mechanism. Which one is better?

10. Acknowledgements

Eric Burger, Robert Fairlie-Cuninghame, Jonathan Rosenberg, and I were the members of the Application Stimulus Signaling Design Team. All members of the team contributed significantly to this work. In addition, thanks to Bert Culpepper for previous work that has been reused here.

That said, the errors, misinterpretation, and typos in this document are my own.

Normative References

[1] 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.

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

[3] Crocker, D. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", RFC 2234, November 1997.

[4] Day, M., Rosenberg, J. and H. Sugano, "A Model for Presence and Instant Messaging", RFC 2778, February 2000.

[5] Resnick, P., "Internet Message Format", RFC 2822, April 2001.

[6] Rosenberg, J. and B. Campbell, "Session Initiation Protocol Extension for Instant Messaging", draft-ietf-sip-message-07 (work in progress), September 2002.

[7] Rosenberg, J. and H. Schulzrinne, "Session Initiation Protocol (SIP) Caller Preferences and Callee Capabilities", draft-ietf-sip-callerprefs-06 (work in progress), July 2002.

[8] Levinson, E., "Content-ID and Message-ID Uniform Resource Locators", RFC 2111, March 1997.

Informative References

[9] Culpepper, B. and R. Fairlie-Cuninghame, "Network Application Interaction Requirements", draft-culpepper-sipping-app-interact-reqs-02 (work in progress), October 2002.

[10] Burger, E., "Keypad Markup Language (KPML)", draft-burger-sipping-kpml-00 (work in progress), October 2002.

[11] Rosenberg, J., "A Framework for Application Interaction", draft-rosenberg-app-interaction-00 (work in progress), October 2002.

[12] Fielding, R., Gettys, J., Mogul, J., Nielsen, H., Masinter, L., Leach, P. and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, June 1999.

Author's Address

Cullen Jennings
Cisco Systems
170 West Tasman Drive
MS: SJC-21/3
San Jose, CA 95134
USA

Phone: +1 408 527-9132
EMail: fluffy@cisco.com

Full Copyright Statement

Acknowledgement

Funding for the RFC Editor function is currently provided by the Internet Society.