Network Working Group
Request for Comments: 5373
Category: Standards Track

D. Willis, Ed.
Softarmor Systems
A. Allen
Research in Motion (RIM)

November 2008

Requesting Answering Modes for the Session Initiation Protocol (SIP)

Status of This Memo

Copyright Notice

Abstract

This document extends SIP with two header fields and associated option tags that can be used in INVITE requests to convey the requester's preference for user-interface handling related to answering of that request. The first header, "Answer-Mode", expresses a preference as to whether the target node's user interface waits for user input before accepting the request or, instead, accepts the request without waiting on user input. The second header, "Priv-Answer-Mode", is similar to the first, except that it requests administrative-level access and has consequent additional authentication and authorization requirements. These behaviors have applicability to applications such as push-to-talk and to diagnostics like loop-back. Usage of each header field in a response to indicate how the request was handled is also defined.

Table of Contents

1. Background

The conventional model for session establishment using the Session Initiation Protocol (SIP, [RFC3261]) involves 1) sending a request for a session (a SIP INVITE) and notifying the user receiving the request, 2) acceptance of the request and of the session by that user, and 3) the sending of a response (SIP 200 OK) back to the requester before the session is established. Some usage scenarios deviate from this model, specifically with respect to the notification and acceptance phase. While it has always been possible for the node receiving the request to skip the notification and acceptance phases, there has been no standard mechanism for the party sending the request to specifically indicate a desire (or requirement) for this sort of treatment. This document defines a SIP extension header field that can be used to request specific treatment related to the notification and acceptance phase.

The first usage scenario is the requirement for diagnostic loopback calls. In this sort of scenario, a testing service sends an INVITE to a node being tested. The tested node accepts and a dialog is established. But rather than establishing a two-way media flow, the tested node loops back or "echoes" media received from the testing service back toward the testing service. The testing service can then analyze the media flow for quality and timing characteristics. Session Description Protocol (SDP) usage for this sort of flow is described in [LOOPBACK]. In this sort of application, it might not be necessary that the human using the tested node interact with the node in any way for the test to be satisfactorily executed. In some cases, it might be appropriate to alert the user to the ongoing test, and in other cases it might not be.

The second scenario is that of push-to-talk applications, which have been specified by the Open Mobile Alliance. In this sort of environment, SIP is used to establish a dialog supporting asynchronous delivery of unidirectional media flow, providing a user experience like that of a traditional two-way radio. It is conventional for the INVITES used to be automatically accepted by the called UA (User Agent), and the media is commonly played out on a loudspeaker. The called party’s UA’s microphone is not engaged until the user presses the local "talk" button to respond.

A third scenario is the Private Branch Exchange (PBX) attendant. Traditional office PBX systems often include intercom functionality. A typical use for the intercom function is to allow a receptionist to activate a loudspeaker on a desk telephone in order to announce a visitor. Not every caller can access the loudspeaker, only the

receptionist or operator, and it is not expected that these callers will always want "intercom" functionality -- they might instead want to make an ordinary call.

There are presumably many more use cases for the extensions defined in this specification, but this document was developed to specifically meet the requirements of these scenarios, or others with essentially similar properties.

These sorts of mechanisms are not required to provide the functionality of an "answering machine" or "voice mail recorder". Such a device knows that it is expected to answer and does not require a SIP extension to support its behavior.

Much of the discussion of this topic in working group meetings and on the mailing list dealt with differentiating "answering mode" from "alerting mode". Some early work did not make this distinction. We therefore proceed with the following definitions:

- Answering Mode includes behaviors in a SIP UA relating to acceptance or rejection of a request that are contingent on interaction between the UA and the user of that UA after the UA has received the request. We are principally concerned with the user interaction involved in accepting the request and initiating an active session. An example of this might be pressing the "yes" button on a mobile phone.

- Alerting Mode includes behaviors in a SIP UA relating to informing the user of the UA that a request to initiate a session has been received. An example of this might be activating the ring tone of a mobile phone.

This document deals only with "Answering Mode". Issues relating to "Alerting Mode" are outside its scope.

This document defines two SIP extension header fields: "Answer-Mode" and "Priv-Answer-Mode". These two extensions take the same parameters and operate in the same general way.

The distinction between Answer-Mode and Priv-Answer-Mode relates to the level of authorization claimed by the User Agent Client (UAC) and verified and policed by the User Agent Server (UAS). Requests are usually made using Answer-Mode. Requests made using Priv-Answer-Mode request "privileged" treatment from the UAS. This mechanism is discussed in greater detail below, in Section 4.1.

Priv-Answer-Mode is not an assertion of privilege. Instead, it is a request for privileged treatment. This is similar to the UNIX model, where a user might run a command normally or use "sudo" to request administrative privilege for the command. Including "Priv-" is equivalent to prefixing a UNIX command with "sudo". In other words, a separate policy table (like "/etc/sudoers") is consulted to determine whether the user may receive the requested treatment.

This distinction is discussed in greater detail in Section 4.1.

1.1. Requirements Language

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 [RFC2119].

2. Syntax of Header Fields and Option Tags

The following syntax uses ABNF as defined in [RFC5234]. Further, it relies on the syntax for SIP defined in [RFC3261].

The syntax for the header fields defined in this document is:

```
Answer-Mode = "Answer-Mode" HCOLON answer-mode-value
  *(SEMI answer-mode-param)

Priv-Answer-Mode = "Priv-Answer-Mode" HCOLON answer-mode-value
  *(SEMI answer-mode-param)

answer-mode-value = "Manual" / "Auto" / token

answer-mode-param= "require" / generic-param
```

The SIP option tag indicating support for this extension is "answermode".

> For implementors: SIP header field names and values are always compared in a case-insensitive manner. The pretty capitalization is just for readability.

This syntax includes extension hooks ("token" for answer-mode values and "generic-param" for optional parameters) that could be defined in future. This specification defines only the behavior for the values given explicitly above. In order to provide forward compatibility, implementations MUST ignore unknown values.

3. Usage of the Answer-Mode and Priv-Answer-Mode Header Fields

This document defines usage of the Answer-Mode and Priv-Answer-Mode header fields in initial (dialog-forming) SIP INVITE requests and in 200 (OK) responses to those requests. This document specifically does not define usage in any other sort of request or response, including but not limited to ACK, CANCEL, or any mid-dialog usage.

This limitation stems from the intended usage of this extension, which is to affect the way that users interact with communications devices when requesting new communications sessions and when responding to such requests. This sort of interaction occurs only during the formation of a dialog and its initial usage, not during subsequent operations such as re-INVITE. However, the security aspects of the session initiation must be applied to changes in media description introduced by re-INVITES or similar requests. See Section 7.1 for further discussion of this issue.

4. Usage of the Answer-Mode and Priv-Answer-Mode Header Fields in Requests

The Answer-Mode or Priv-Answer-Mode header field is used by a UAC in an INVITE request to invoke specific handling by the responding UAS; this handling is related to "automatic answering" functionality for any dialog resulting from that INVITE request. If no Answer-Mode or Priv-Answer-Mode header field is included in the request, answering behavior is at the discretion of the UAS, as it would be in the absence of this specification. The desired handling is indicated by the value of the Answer-Mode or Priv-Answer-Mode header field, as follows:

Manual: The UAS is asked to defer accepting the request until the user of the UAS has interacted with the user interface (UI) of the UAS in such a way as to indicate that the user desires the UAS to accept the request.

Auto: The UAS is asked to accept the request automatically, without waiting for the user of the UAS to interact with the UI of the UAS in such a way as to indicate that the user desires the UAS to accept the request.

Each value of the Answer-Mode or Priv-Answer-Mode header field can include an optional parameter, "require". If present, this parameter indicates that the UAC would prefer that the UAS reject the request if the UAS is unwilling (perhaps due to policy) to answer in the mode requested, rather than answering in another mode. For example, this

parameter could be used to make sure that a test "loopback" call doesn't disturb a user who has configured her phone to manually answer even if the caller requests an automatic answer.

The UAS is responsible for deciding how to honor this preference. In general, the UAS makes an authorization decision based on the authenticated identity presented in the request using authentication mechanisms such as SIP Digest Authentication [RFC3261], the SIP Identity mechanism [RFC4474], or (within the restricted networks for which it is suitable) the SIP mechanism for asserted identity within trusted networks [RFC3325]. When making an authorization decision, the UAS should also use authorization information or policy available to the UAS. This decision-making MUST consider the risk model of the media session corresponding to the request, and the UAS MUST NOT answer without user input in cases where the privacy or security of the user would be compromised as a result. Making this determination is a matter of system or application design, and cannot in general be addressed by having a set of functions that are configurable on or off. Specific discussion of media sessions and appropriate policy is discussed in Section 7.

4.1. The Difference Between Answer-Mode and Priv-Answer-Mode

The functions of the Answer-Mode and Priv-Answer-Mode header fields are similar; they both ask that the UAS handle the request as specified by the header field's value (automatic or manual). The difference is in the way the requests interact with the UAS's policy. A typical UAS will have different policies for handling each header field. For example, assume that the user of a UAS has placed that UAS into "meeting mode", indicating that she is engaged in an important activity and does not wish to be spuriously interrupted. The UAS might disallow automatic answering for Answer-Mode requests while in "meeting mode". However, that UAS might allow automatic answering for requests made with Priv-Answer-Mode. There will probably be differences in authorization policy. For example, a UAS might be configured such that callers on the "friends" list are allowed to make requests using Answer-Mode but not Priv-Answer-Mode. That same UAS might be configured to only allow callers on the "administrators" list to use Priv-Answer-Mode. This is different from always basing the behavior on the identity of the calling party. For example, assume caller "Bob" is on both the "friends" list and "administrators" list. If Bob wants his request to be processed according to the regular policy, he uses Answer-Mode. If Bob wants his request to be processed under the more restrictive "privileged" policy, he uses Priv-Answer-Mode.

A UAS SHOULD apply a stricter authorization policy to a request with Priv-Answer-Mode than it does to requests with Answer-Mode. The default policy SHOULD be to refuse requests containing Priv-Answer-Mode header fields unless the requester is authenticated and specifically authorized to make Priv-Answer-Mode requests. Failure to enforce such a policy leaves the user potentially vulnerable to abuses, as discussed in Section 7.

The use case envisioned for Priv-Answer-Mode relates to handling urgent requests from authorized callers. For example, assume Larry is a limousine driver working with a fleet dispatcher. Larry likes to provide a quiet environment for his car, so his communicator is configured for manual answer mode for all non-privileged calls, including push-to-talk (Answer-Mode: Auto) calls. Each time he gets a call, Larry's communicator chimes softly to alert him to the call. If the circumstances permit it, Larry presses the communicator in order to accept the call, the communicator sends a 200 (OK) response, and the calling party's talk-burst is played out through the communicator's loudspeaker. This treatment is delivered to incoming requests that have an Answer-Mode header field having values of "Manual" or "Auto" (or no Answer-Mode header field at all), no matter who the caller is.

Larry's fleet dispatch operator is familiar with this policy, and needs to inform Larry about a critical matter. The dispatch operator tries several times to push-to-talk call Larry (including Answer-Mode: Auto in the requests), but the calls aren't accepted because Larry has fallen asleep, and therefore isn't pressing his communicator to accept the call.

The operator then presses his "urgent" button and calls Larry again. This time, the INVITE request carries a "Priv-Answer-Mode: Auto" header field. Larry's communicator checks the identity of the caller (using a SIP Identity assertion or functionally equivalent mechanism), and matches the operator's identity against the list of users allowed to do Priv-Answer-Mode. Since the operator is listed, the communicator immediately returns a 200 (OK) response accepting the call. The operator speaks, and the resulting talk-burst is summarily played out the loudspeaker on Larry's communicator, waking him up.

The effect of requesting Priv-Answer-Mode is different than the effect of simply granting higher privilege to an Answer-Mode request based on the requester's identity and corresponding authorization level. This distinction is what allows the fleet operator to make polite (Answer-Mode: Auto) requests to Larry under normal conditions, and receive different handling (Priv-Answer-Mode: Auto) for a request having greater urgency.

In normal operations, only one of either Answer-Mode or Priv-Answer-Mode would be used in an INVITE request. If both are present, the UAS will first test the authorization of the requester for Priv-Answer-Mode and, if authorized, process the request as if only Priv-Answer-Mode had been included. If the requester is not authorized for Priv-Answer-Mode, then the UAS will process the request as if only Answer-Mode had been included.

4.2. The "require" Modifier

Both Answer-Mode and Priv-Answer-Mode allow a modifier of "require" (example: "Priv-Answer-Mode: Auto;require"). This modifier does not influence the UAS's policy in choosing whether to answer manually or automatically. The UAS decides whether or not to answer automatically based on other aspects of the request. The "require" modifier is only evaluated after the UAS has selected an answering mode. If the UAS's policy has resulted in an answering mode that is different from that specified in the request, the presence of the "require" modifier asks the UAS to reject the call. In the given example, the UAS is being asked to answer automatically if the caller is authorized for automatic answering under the "privileged" policy, and to reject the call (rather than answering manually) if the caller is not authorized for this mode. This is discussed in more depth in Section 4.5.

4.3. Procedures at User Agent Clients (UAC)

4.3.1. All Requests

A UA supporting the Answer-Mode and Priv-Answer-Mode header fields SHOULD indicate its support by including an option tag of "answermode" in the Supported header field of all requests it sends.

4.3.2. REGISTER Transactions

To indicate that it supports the answer-mode negotiation feature, a UA MAY include an extensions parameter with a value that includes "answermode". Example:

```
;extensions="answermode,100rel,gruu"
```

in the Contact header field of its REGISTER requests. This usage of feature tags is described in [RFC3840].

If a UA is dependent on support for callee capabilities in the registrar, it MAY include a Require header field with the value "pref" in its REGISTER request. This will cause the registrar to reject the request if the registrar does not support callee capabilities and caller preferences. Example:

```
Require: pref
```

4.3.3. INVITE Transactions

A UAC supporting this specification MAY include an Answer-Mode or Priv-Answer-Mode header field in an INVITE where it wishes to influence the answering mode of the responding UAS.

> Note: This is meaningful only in initial or dialog-forming INVITE requests. Answer-Mode and Priv-Answer-Mode header fields appearing in other requests are ignored. In general, if the request would not normally result in a notification to the user and acceptance by that user (for example, "ringing" and "answering"), then these extensions are not applicable.

To request that the UAS answer only after having interacted with its user and receiving an affirmative instruction from that user, the UAC includes an Answer-Mode or Priv-Answer-Mode header field having a value of "Manual". Example:

```
Answer-Mode: Manual
```

To request that the UAS answer manually, and ask that it reject the INVITE request if unable or unwilling to answer manually, the UAC includes an Answer-Mode or Priv-Answer-Mode header field having a value of "Manual" and a parameter of "require". Example:

```
Answer-Mode: Manual;require
```

To request that the UAS answer automatically without waiting for input from the user, the UAC includes an Answer-Mode or Priv-Answer-Mode header field having a value of "Auto". Example:

```
Answer-Mode: Auto
```

To request that the UAS answer automatically, and ask that it reject the INVITE request if unable or unwilling to answer automatically, the UAC includes an Answer-Mode or Priv-Answer-Mode header field having a value of "Auto" and a parameter of "require". Example:

```
Answer-Mode: Auto;require
```

To require that the UAS either support this extension or reject the request, the UAC includes a Require header field having the value "answermode". This does not actually force the UAS to automatically answer, it just requires that the UAS either understand this extension or reject the request. We do not have a SIP negotiation technique to force specific behavior. Rather, the desired behavior is indicated in the SIP extension itself. Example:

```
Require: answermode
```

To request that retargeting proxies in the path preferentially select targets that have indicated support for this extension in their registration, a UAC includes an Accept-Contact header field with an extensions parameter having a value of "answermode". This usage of Accept-Contact is described in [RFC3841]. This would normally be used in conjunction with the "Require: answermode" header field as described above. Example:

```
Require: answermode Accept-Contact:
          *;extensions="answermode";methods="INVITE"
```

To request that retargeting proxies in the path do not select targets that have indicated non-support for this extension in their registration, a UAC includes an Accept-Contact header field with an extensions parameter having a value of "answermode" and an option field of "require". This usage of Accept-Contact is described in [RFC3841]. This would normally be used in conjunction with the "Require: answermode" header field as described above. Example:

```
Require: answermode Accept-Contact:
        *;extensions="answermode"; methods="INVITE";require
```

To request that retargeting proxies in the path exclusively select targets that have indicated support for this extension in their registration, a UAC includes an Accept-Contact header field extensions parameter having a value of "answermode" and options of "require" and "explicit". This usage of Accept-Contact is described in [RFC3841]. This would normally be used in conjunction with the "Require: answermode" header field as described above. Example:

```
Require: answermode Accept-Contact:
        *;extensions="answermode";
        methods="INVITE";require;explicit
```

4.4. Procedures at Intermediate Proxies

4.4.1. General Proxy Behavior

The general procedure at all intermediate proxies, including the UAC's serving proxy or proxies and the UAS's serving proxy or proxies, is to ignore the Answer-Mode header field. However, the serving proxies (proxies responsible for resolving an address-of-record (AOR) into a registered contact) MAY exercise control over the requested answer mode, either inserting or deleting an Answer-Mode or Priv-Answer-Mode header field or altering the value of an existing header field, in accord with local policy. This could result in behavior that is inconsistent with user expectations (such as having a call that was intended to be a diagnostic loopback answered by a human) and consequently proxies MUST NOT insert, delete, or alter Answer-Mode or Priv-Answer-Mode header fields unless explicitly authorized to do so by an external agreement between the proxy operator and the user of the UA that the proxy is serving. These serving proxies MAY also reject a request according to local policy and, if they do so, SHOULD use the rejection codes as specified below for the UAS.

4.4.2. Issues with Automatic Answering and Forking

One of the well-known issues with forking is the problem of multiple acceptance. If an INVITE request is forked to several UASs and more than one replies with a 200 (OK) response, the conventional approach is to continue the dialog with the first respondent and tear down the dialog (using BYE requests) with all other respondents.

While this problem exists without an auto-answer negotiation capability, it is apparent that widespread adoption of UAs that engage in auto-answer behavior will exacerbate the multiple acceptance problem. Consequently, systems designers need to take this aspect into consideration. In general, auto-answer is NOT RECOMMENDED in environments that include parallel forking.

As an alternative, it might be reasonable to use a variation on manual-answer combined with no alerting and early media. In this approach, the initial message or talk-burst is transmitted as early media to all recipients, where it is displayed or played out. Any response utterance (pushing the transmit key and talking) from the user of a UAS following this would serve as an "acceptance", resulting in a 200 (OK) response being transmitted by their UAS. Consequently, the race-condition for acceptance would be limited to the subset of UAs actually responding under user control, rather than the full set of UAs to which the request was forked.

Another alternative would be to use dynamic conferencing instead of forking. In this approach, instead of forking the request, a conference would be initiated and all registered UAs invited into that conference. The mixer attached to the conference would then mediate traffic flows appropriately.

4.5. Procedures at User Agent Servers (UAS)

4.5.1. INVITE Transactions

For a request having an Answer-Mode value of "Manual" and not having an Answer-Mode parameter of "require", the UAS SHOULD defer accepting the request until the user of the UAS has confirmed willingness to accept the request. This behavior MAY be altered as needed for unattended UASs or other local characteristics or policy. For example, an auto-attendant or Public Switched Telephone Network (PSTN) gateway system that always answers automatically would go ahead and answer, despite the presence of the "Manual" Answer-Mode header field value.

For a request having an Answer-Mode value of "Manual" and an Answer-Mode parameter of "require", the UAS MUST defer accepting the request until the user of the UAS has confirmed willingness to accept the request. If the UAS is not capable of answering the request in this "Manual" mode or is unwilling to do so, it MUST reject the request, SHOULD do so with a "403 (Forbidden)" response, and MAY include a reason phrase of "manual answer forbidden".

For a request having an Answer-Mode value of "Auto", the UAS SHOULD, if the calling party is authenticated and authorized for automatic answering, accept the request without further user input. The UAS MAY, according to local policy or user preferences, treat this request as it would treat a request having an Answer-Mode with a value of "Manual" or having no Answer-Mode header field. If the calling party is not authenticated and authorized for automatic answer, the UAS MAY either handle the request as per "manual", or reject the request. If the UAS rejects the request, it SHOULD do so with a "403 (Forbidden)" response, and MAY include a reason phrase of "automatic answer forbidden". There may be an interaction with [RFC3261] section 23.2, which in some cases requires user validation of certificates used for S/MIME. Since this places the same interrupt burden on the user as would manually answering the request, a UAS experiencing this requirement for user validation of a request that requires automatic answering SHOULD reject the request with a "403 (Forbidden)" response and MAY include a reason phrase of "certificate validation requires user input not compatible with automatic answer."

For a request having an Answer-Mode value of "Auto" and an Answer-Mode parameter of "require", the UAS SHOULD, if the calling party is authenticated and authorized for automatic answering, accept the request. The UAS MUST NOT allow "manual" answer of this request, but MAY reject it. If, for whatever reason, the UAS chooses not to accept the request automatically, the UAS MUST reject the request, SHOULD do so with a "403 (Forbidden)" response, and MAY include a reason phrase of "automatic answer forbidden".

Similar behavior applies for Priv-Answer-Mode, except that the policy for authorization may be different (and generally more stringent).

5. Usage of the Answer-Mode and Priv-Answer-Mode Header Fields in Responses

The Answer-Mode or Priv-Answer-Mode header field can be inserted by a UAS into a response in order to indicate how it handled the associated request with respect to automatic answering functionality. The UAC might use this information to inform the user or otherwise adapt the behavior of the user interface. The handling is indicated by the value of the header field, as follows:

Manual: The UAS responded after the user of the UAS interacted with the user interface (UI) of the UAS in such a way as to indicate that the user desires the UAS to accept the request.

Auto: The UAS responded automatically, without waiting for the user of the UAS to interact with the UI of the UAS in such a way as to indicate that the user desires the UAS to accept the request.

The Answer-Mode and Priv-Answer-Mode header fields, when used in responses, are only valid in a 200 (OK) response to an INVITE request.

5.1. Procedures at the UAS

A UAS supporting this specification inserts an Answer-Mode or Priv-Answer-Mode header field into the 200 (OK) response to an INVITE request when it wishes to inform the UAC as to whether the request was answered manually or automatically. It is reasonable for a UAS to assume that if the UAC included an Answer-Mode header field in the request, it would probably like to see an Answer-Mode header field in the response. The full rationale for including or not including this header field in a response is outside of the scope of this specification, and is sensitive to the privacy concerns of the user of the UAS. For example, informing the calling party that a call was answered manually might reveal the presence of an "actual human" at the responding UAS. While in the general case the ensuing

conversation would also reveal this same information, there might be cases where this information might need to be protected. Consequently, UASs supporting this specification SHOULD include appropriately configurable policy mechanisms for making this determination, and the default configuration SHOULD be to exclude this header field from responses.

5.2. Procedures at the UAC

A UAC MAY use the value of the Answer-Mode or Priv-Answer-Mode header field, if present, to adapt the user interface and/or inform the user about the handling of the request. For example, the user of a push-to-talk system might speak differently if she knows that the called party answered "in person" vs. having the call blare out of an unattended speaker phone.

6. Examples of Usage

The following examples show Bob registering a contact that supports the negotiation of answering mode. Alice then calls Bob with an INVITE request, asking for automatic answering and explicitly asking that the request not be routed to contacts that have not indicated support for this extension. Further, Alice requires that the request be rejected if Bob's UA does not support negotiation of answering mode. Bob replies with a 200 (OK) response indicating that the call was answered automatically.

> The Content-Length header field shown in the examples contains a placeholder "..." instead of a valid Content-Length. Furthermore, the SDP bodies that would be expected in the INVITE requests and 200 (OK) responses are not shown.

6.1. REGISTER Request

In the following example, Bob's UA is registering and indicating that it supports the answermode extension.

```
REGISTER sip:example.com SIP/2.0
From: Bob<sip:bob@example.com>
To: Bob <sip:bob@example.com>
CallID: hh89as0d-asd88jkk@cell-phone.example.com
CSeq: 1 REGISTER
Contact: sip:cell-phone.example.com;
  ;audio
  ;+sip.extensions="answermode"
  ;methods="INVITE,BYE,OPTIONS,CANCEL,ACK"
  ;schemes="sip"
```

6.2. INVITE Request

In this example, Alice is calling Bob and asking Bob's UA to answer automatically. However, Alice is willing for Bob to answer manually if Bob's policy is to prefer manual answer, so Alice does not include a ";require" modifier on "Answer-Mode: Auto".

```
INVITE sip:bob@example.com SIP/2.0
Via: SIP/2.0/TCP client-alice.example.com:5060; branch=z9hG4bK74b43
Max-Forwards: 70
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@example.com>
Call-ID:3848276298220188511@client-alice.example.com
CSeq: 1 INVITE
Contact: <sip:alice@client.atlanta.example.com;transport=tcp>
Require: answermode
Accept-contact:*;require;explicit;extensions="answermode"
Answer-Mode: Auto
Content-Type: application/sdp
Content-Length: ...
```

6.3. 200 (OK) Response

Here, Bob has accepted the call and his UA has answered automatically, which it indicates in the 200 (OK) response.

```
SIP/2.0 200 OK
Via: SIP/2.0/TCP client-alice.example.com:5060; branch=z9hG4bK74b43
From: Alice <sip:alice@example.com>;tag=9fxced76sl
To: Bob <sip:bob@example.com>;tag=8321234356
Call-ID: 3848276298220188511@client-alice.example.com
CSeq: 1 INVITE
Contact: <sip:bob@client.biloxi.example.com;transport=tcp>
Answer-Mode: Auto
Content-Type: application/sdp
Content-Length: ...
```

7. Security Considerations

This specification adds the ability for a UAC to request potentially risky user interface behavior relating to the acceptance of an INVITE request by the UAS receiving the request. Specifically, the UAC can request that the UAS accept the request without input to the UAS by the user of the UAS (Answer-Mode: Auto).

There are several attacks possible here -- the most obvious being the ability to turn a phone into a remote listening device without its user being aware. Additional potential attacks include reverse

charge fraud, unsolicited push-to-talk communications (spam over push-to-talk (SPTT)), playout of obnoxious noises (the "whoopee cushion" attack), battery-rundown denial of service, "forced busy" denial of service, running up the victim's data transport bill, and phishing via session insertion (where an ongoing session is replaced by another without the victim's awareness).

Since SIP implementations do not commonly implement end-to-end message protections, this specification is completely dependent on transitive security across SIP proxies. Any misbehaving proxy can insert, delete, and/or alter the contents of the Answer-Mode and Priv-Answer-Mode header fields, and in general can do so without being noticed by either the UAC or UAS. Consequently, it is critical that any proxies in the path be not only trusted, but worthy of that trust. While proxies do not generally intentionally insert, delete, or alter the Answer-Mode and Priv-Answer-Mode header fields, this specification does note a use case for such manipulation by proxies acting on behalf of the user of a UAC or UAS that has limited support for the authentication or policy enforcement needed to securely exercise these extensions. Proxies that perform such extension-sensitive manipulation MUST therefore provide complete policy enforcement, as per the minimal policy discussed in Section 7.4.

The existing body of SIP work provides strong capabilities for authentication of requests, prevention of man-in-the-middle attacks, protection of the privacy and integrity of media flows, and so on (although as noted above, these capabilities usually rely on transitive trust across proxies). The behaviors added by the extensions in this document raise additional possibilities for attacks against media flows not completely addressed by existing SIP work, and therefore require analysis in this document.

Media attacks can be loosely categorized as:

Insertion: Media is inserted into and played out by the victim UA without consent of the UA's user.

Interception: The victim UA's media acquisition facility (such as a microphone or camera) is activated, producing a media stream, without the consent of the UA's user.

7.1. Attack Sensitivity Depends on Media Characteristics

The danger of abuse varies greatly depending on the media characteristics of the session being established. Since the expressive range of media sessions that can be established by SIP is

unbounded, we might find it more effective to model loose categories of media modality rather than explicitly describing every possible scenario. Security analysis can then be applied per modality.

The media modalities of interest appear to be:

UAC-sourced (Inbound) Unidirectional Media Insertion: Sensitive media flows from the UAC and is rendered by the UAS, annoying the user of the UAS or disrupting the function of the UAS. We refer to this as the "whoopee-cushion" attack because of its utility in replicating the rude-noise-making seat cushion. The danger of this attack is quite literally amplified by a loudspeaker apparatus attached to the victim UAS. Media that has minimal secondary implication (such as sending a move in a chess game to a computer that isn't running a chess game) is related, but of far less significance. This sort of attack can also have other consequences, such as discharging the victim's battery or increasing charges for data transport to be paid by the victim.

UAS-sourced (Outbound) Unidirectional Media Interception: Sensitive media flows from the UAS and is rendered by the UAC, violating the privacy of the user of the UAS. We refer to this as the "bug-my-phone" attack because that would appear to be the primary attack motivator.

Bidirectional Media Insertion or Interception: Bidirectional media is the common case when SIP is used in a voice-over-IP scenario or "traditional phone call". Once a media flow is established, both ends send and receive media without further engagement. The media information is presumed to be sensitive -- that is, if intercepted it damages the victim's privacy, and if inserted, it annoys or interferes with the recipient. Attacks of this sort might produce either the "whoopee-cushion" or "bug-my-phone" scenarios, potentially even simultaneously.

It seems reasonable to consider the "bug-my-phone" attack as being in a different class (potentially far more severe) than the "whoopee-cushion" attack. This distinction suggests that security policy could be established in different and presumably less restrictive fashion for inbound media flows than for outbound media flows. The set of callers from which a user would be willing to automatically accept inbound media is reasonably much broader than the set of callers to which a user would be willing to automatically grant outbound media access, although this may not be true in all environments, especially those where reception of unwanted media has unwanted financial consequences.

For example, assume a UA is designed such that it can be used to receive push-to-talk calls to a loudspeaker, and it can be used as a "baby monitor" (has an open mic and streams received audio to listeners). The policy for activating the push-to-talk loudspeaker would probably need to be reasonably broad (perhaps "all the user's buddies"). However, the policy for the baby monitor would need to be very narrow (perhaps "only the baby's mother") or even completely closed. The minimal policy defined in Section 7.4 explicitly forbids the "baby monitor" functionality.

7.2. Application Design Affects Attack Opportunity

In the most common use cases, the security aspects are somewhat mitigated by design aspects of the application. For example, in traditional telephony, the called party is alerted to the request (the phone rings), no media session is established without the acceptance of the called party (picking up the phone), and the media path is most commonly delivered to a single-user handset. Consequently, this application (although bidirectional) is relatively secure against both media insertion and media interception attacks of the sort enabled by the extensions in this document. The use of policy-free automatic-answering devices (like answering machines) and amplifiers (speakerphones and call-screening devices) weakens this defense.

In push-to-talk applications, media can be sent from UAC to UAS without user oversight, but no media is sent from the called UAS without user input (the "push" of "push-to-talk"). Consequently, there is no "bug-my-phone" attack opportunity. Further, screening of the UAC by eliminating UAC identities not on some sort of "white list" (often, a buddy list) reduces the threat of "whoopee cushion" attacks (except from one's buddies, of course).

Similar approaches apply to most applications. Insertion can be controlled (but not eliminated) by combining identity mechanisms with simple authorization policy, and interception can be effectively eliminated by combining strong identity mechanisms with aggressive authorization policy and/or user interaction.

7.3. Applying the Analysis

The extensions described in this document provide mechanisms by which a UAC can request that a UAS not deploy two of the five defensive mechanisms listed below -- user alerting and user acceptance. In order for this not to produce undue risk of insertion attacks or increased risk of interception attacks, we are therefore forced to rely on the remaining defensive mechanisms. This document defines a minimum threshold for satisfactory security. Certainly more

restrictive policies might reasonably be used, but any policy less restrictive than the approach described below is very likely to result in significant security issues.

From the previous discussion of risks, attacks, and vulnerabilities, we can derive five defensive mechanisms available at the application level:

1. Identity -- Know who the request came from.

2. Alerting -- Let the called user know what's happening. Some applications might use inbound media as an alert.

3. Acceptance -- Require called user to make run-time decision. Asking the user to make a run-time decision without alerting the user to the need to make a decision is generally infeasible. This will have implications for possible alerting options that are outside the scope of this document.

4. Limit the Input/Output (I/O) -- Turn off loudspeakers or microphone. This could be used to convert a bidirectional media session (very risky, possible "bug my phone") into a unidirectional, inbound-only (less risky, possible "spam" or "rundown", etc.) session while waiting for user acceptance.

5. Policy -- Rules about other factors, such as black- and whitelisting based on identity, disallowing acceptance without alerting, etc.

Since SIP and related work already provide several mechanisms (including SIP Digest Authentication [RFC3261], the SIP Identity mechanism [RFC4474], and the SIP mechanism for asserted identity within private networks [RFC3325], in networks for which it is suitable) for establishing the identity of the originator of a request, we presume that an appropriately selected mechanism is available for UAs implementing the extensions described in this document. In short, UAs implementing these extensions MUST be equipped with and MUST exercise a request-identity mechanism. The analysis below proceeds from an assumption that the identity of the sender of each request is either known or is known to be unknown, and can therefore be considered in related policy considerations. Failure to meet this identity requirement either opens the door to a wide range of attacks or requires operational policy so tight as to make these extensions useless.

We previously established a class distinction between inbound and outbound media flows, and can model bidirectional flows as "worst case" sums of the risks of the other two classes. Given this distinction, it seems reasonable to provide separate directionality policy classes for:

1. Inbound media flows.

2. Outbound media flows.

For each directionality policy class, we can divide the set of request identities into three classes:

1. Identities explicitly authorized for the class.

2. Identities explicitly denied for the class.

3. Identities for which we have no explicit policy and need the user to make a decision.

Note that not all combinations of policies possible in this decomposition are generally useful. Specifically, a policy of "inbound media denied, outbound media allowed" equates to a "bug my phone" attack, and is disallowed by the minimal policy of Section 7.4, which as written excludes all cases of "Outbound media explicitly authorized".

7.4. Minimal Policy Requirement

User agents implementing this specification SHOULD NOT establish a session providing inbound media without explicit user acceptance where the requesting user is unknown, or is known and has not been granted authorization for such a session. This requirement is intended to prevent "SPAM broadcast" attacks where unexpected and unwanted media is played out at a UAS .

User agents implementing this specification MUST NOT establish a session providing outbound or bidirectional media sourced from the user agent without explicit user acceptance. Loopback media used for connectivity testing is not constrained by this requirement. This requirement is intended to assure that this extension can not be used to turn a UAS into a remote-controlled microphone (or "bug") without the knowledge of its user. Since SIP allows for a session to be initially established with inbound-only media and then transitioned (via re-INVITE or UPDATE) to an outbound or bidirectional session,

enforcing this policy requires dialog-stateful inspection in the SIP UAS. In other words, if a session was initiated with automatic answering, the UAS MUST NOT transition to a mode that sends outbound media without explicit acceptance by the user of the UAS.

8. IANA Considerations

8.1. Registration of Header Fields

This document defines new SIP header fields named "Answer-Mode" and "Priv-Answer-Mode".

The following rows have been added to the "Header Fields" section of the SIP parameter registry:

Header Name	Compact Form	Reference
Answer-Mode		[RFC5373]
Priv-Answer-Mode		[RFC5373]

8.2. Registration of Header Field Parameters

This document defines parameters for the header fields defined in the preceding section. The header fields "Answer-Mode" and "Priv-Answer-Mode" can take the values "Manual" or "Auto".

The following rows have been added to the "Header Field Parameters and Parameter Values" section of the SIP parameter registry:

Header Field	Parameter Name	Predefined Values	Reference
Answer-Mode	require	No	[RFC5373]
Priv-Answer-Mode	require	No	[RFC5373]

8.3. Registration of SIP Option Tags

This document defines the SIP option tag "answermode".

The following row has been added to the "Option Tags" section of the SIP Parameter Registry:

Name	Description	Reference
answermode	This option tag is for support of the Answer-Mode and Priv-Answer-Mode extensions used to negotiate automatic or manual answering of a request.	[RFC5373]

9. Acknowledgements

This document draws requirements and a large part of its methodology from the work of the Open Mobile Alliance, and specifically from a document by Andrew Allen, Jan Holm, and Tom Hallin.

The editor would also like to recognize the contributions of David Oran and others who argued on the SIPPING mailing list and at the OMA ad-hoc meeting at IETF 62 that the underlying ideas of the above document were broadly applicable to the SIP community, and that the concepts of alerting and answering should be clearly delineated. Further, the security review provided by Sandy Murphy and the gen-art review by Suresh Krishnan were very helpful in improving the quality of this document.

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.

[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.

[RFC3840] Rosenberg, J., Schulzrinne, H., and P. Kyzivat, "Indicating User Agent Capabilities in the Session Initiation Protocol (SIP)", RFC 3840, August 2004.

[RFC3841] Rosenberg, J., Schulzrinne, H., and P. Kyzivat, "Caller Preferences for the Session Initiation Protocol (SIP)", RFC 3841, August 2004.

[RFC4474] Peterson, J. and C. Jennings, "Enhancements for Authenticated Identity Management in the Session Initiation Protocol (SIP)", RFC 4474, August 2006.

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

10.2. Informative References

[LOOPBACK] Hedayat, K., "An Extension to the Session Description Protocol (SDP) for Media Loopback", Work in Progress, August 2008.

[RFC3325] Jennings, C., Peterson, J., and M. Watson, "Private Extensions to the Session Initiation Protocol (SIP) for Asserted Identity within Trusted Networks", RFC 3325, November 2002.

Authors' Addresses

Dean Willis (editor)
Softarmor Systems
3100 Independence Pkwy #311-164
Plano, Texas 75075
USA

EMail: dean.willis@softarmor.com

Andrew Allen
Research in Motion (RIM)
300 Knightsbridge Parkway, Suite 360
Lincolnshire, Illinois 60069
USA

EMail: aallen@rim.com