Internet Engineering Task Force
Internet Draft

SIP
H. Schulzrinne
Columbia U.

draft-schulzrinne-sip-bind-00.txt
June 21, 2002
Expires: December 2002

Binding Generic URIs to SIP URIs

STATUS OF THIS MEMO

Abstract

For many services, it is useful to be able to associate data, represented by URLs, with SIP user identifiers, both address-of-records and host-based user names. This draft proposes a solution that addresses most of the requirements for such a mechanism.

1 Introduction

SIP elements need to interact with many other protocols. Examples include:

- SIP elements need to acquire scripts and other call processing logic, such as SIP-cgi scripts [1], SIP servlets [2] and CPL scripts [3].

- Multiparty conferences may be floor-controlled and configured by a non-SIP protocols such as SOAP [4].

- Users want to upload presence documents to presence agents [5].

- SIP elements may want to obtain configuration data from a domain-specific configuration server.

- New users need to be created and their properties configured ("user provisioning").

For many of these, this relationship can be expressed as a simple binding that maps a SIP URI to a labeled set of URIs which are then used to retrieve the associated data or affect the desired operations. This approach allows the entity retrieving the data to choose the appropriate mechanism, or delay updating until a more appropriate time. For small data elements, the data URI scheme [6] can be used to directly include the data in the binding update. Also, in some cases, the data can be included as multipart body, labeled with the mid and cid URIs [7]. Thus, the mechanism is general enough to handle different degrees of indirection.

We assume that such bindings may be needed for individual SIP URLs, typically associated with SIP UAs, as well as address-of-record URIs, typically associated with a SIP proxy and a registrar.

These bindings can be queried and updated by SIP elements or non-SIP elements. For non-SIP elements, the bindings might be queried and modified via a web page or a SOAP request, but such mechanisms are beyond the scope of this document. We believe that it is useful to have a mechanism that is closely tied to SIP, as SIP offers the ability to identify endpoint by a logical address not tied to a particular host or IP address. Also, for simplicity, it appears wise not to require user agents to implement a particular additional non-SIP "bootstrapping" protocol, probably tied to a different authentication and management mechanism. Using the mechanisms below, the amount of information added to SIP requests is very small.

This problem is related to the content indirection problem [8]. In both cases, a SIP message uses a URI to refer to external data, retrieved via a non-SIP protocol. However, the semantics are different: for content indirection, the data referenced by the URI replaces data normally carried in the SIP message itself, primarily for efficiency reasons. In this application, data is associated with the address-of-record or other SIP entity. The semantic differences lead to operations that allow adding and deleting bindings.

2 Terminology

In this document, the key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" are to be interpreted as described in RFC 2119 [9] and indicate requirement levels for compliant implementations. The terms are also used for protocol requirements, and then apply to the protocol properties, not the implementation.

3 The Binding Header Field

In a request, the Binding header field instructs the recipient, a registrar or UA, to update the bindings included in the header field. In a response, the header field values enumerate all or a subset of the bindings associated with the To header field value.

The binding expires after the number of seconds indicated in the ";expires" parameter. A special value of "1" indicates that the binding can only be used once. The value of "0" removes the binding.

The value "0" already had a special meaning.

The function of the binding is described by the ;disposition parameter. Values for the parameter are registered with IANA. In this document, we define the following

expires: The expires parameter describes the lifetime of the binding. Its usage and format is the same as the respective Contact parameter.

disposition: The disposition parameter describes the purpose of the URI.

etag: An entity tag, used to identify versions of the same resource. A user of a binding can compare the entity tag to see if the object needs to be retrieved again.

A registration may contain any number of bindings for each disposition. The disposition parameter is REQUIRED.

Example:

```
Binding: <http://www.tweak-me.com> ;disposition=configuration ;expires=0
Binding: <tftp://store.example.com> ;disposition=boot ;expires=36000
Binding: ;disposition=conference ;expires=0
```

The binding may be stored as a static table, or be generated dynamically. The precise meaning of the binding depends on the disposition parameter. The recipient (registrar or UA) does not have to act on the binding. It is up to the recipient of the binding whether it accepts the binding, at all or from a particular source. Registrars SHOULD accept all bindings, acting as a repository for information that user agents can then retrieve.

4 The Accept-Binding Header Field

We define a new header field, Accept-Binding, that enumerates all binding dispositions that the server will accept to store. It does not define the bindings that a client would like to obtain.

The special Accept-Binding value of "*" indicates that the SIP element will accept bindings with any disposition. An empty Accept-Binding indicates that the SIP element does not support any disposition.

5 The Use of REGISTER and OPTIONS

As discussed below, REGISTER is not appropriate for updating bindings. However, the Binding header field may appear in REGISTER responses to inform the registrand of the list of available bindings. This avoids the overhead of an extra SIP request.

The OPTIONS response MAY also contain Binding header fields. It can be used to obtain bindings for a particular SIP identifier, typically other than the AoR.

6 The BIND Method

6.1 Motivation

One of the problems with using REGISTER for maintaining bindings is that this works only reasonably well for conveying bindings from registrars to UAs, as that behavior adds no additional failure cases to the REGISTER behavior. Also, REGISTER is not appropriate for associating bindings with a conference, for example, since conference participants do not typically register for that address.

6.2 Overview

This document introduces the SIP BIND request method. Its behavior is similar to REGISTER, but it updates and retrieves generic URI bindings instead of Contact URIs. BIND requests are used by SIP UAs to query, update and delete bindings.

> The REGISTER mechanisms appear to be relatively well understood and easy to implement. It appears likely that the BIND method can be implemented in a very similar manner.

6.3 Operation

BIND is a non-INVITE method that can be forked.

Unlike REGISTER, BIND requests can be addressed either to the generic domain, as in sip:example.com, or to a specific instance, such as sip:alice@pc133.example.com. Thus, BIND can be supported by registrars and UAS.

6.4 Adding Bindings

Bindings are adding by addressing a BIND request to a SIP URI, with zero or more Binding header fields. The binding is identified by the To header field.

The BIND method is atomic. If any of the bindings fail, none of the other bindings are updated, added, deleted or replaced.

A server MAY offer other mechanisms, such as a web page or a database, to add, delete and update bindings.

If the receiver does not support a particular binding disposition, in general or from the issuer of the request, it responds with a 4?? (Binding refused) and includes an Accept-Binding header field that summarizes all binding dispositions that it does support for this particular user.

6.5 Obtaining bindings

The response to a BIND request enumerates the current bindings in the Binding header fields.

6.6 Deleting Bindings

A particular binding is deleted by refreshing it with an expiration interval of zero. All bindings having a particular disposition are deleted by omitting the specific URI, but including the disposition, in the Binding header field.

7 Requirements Analysis

The mechanism presented above is believed to satisfy the requirements for a publication mechanism detailed in [10]. We quote the

requirements below and indicate how they are met.

7.1 Publication URI

> The binding mechanism MUST allow a client to discover or infer a URI, or set of URIs, to which data may be published for a particular service. The mechanism MUST allow for any URI scheme, and MUST NOT make assumptions about how a specific publication mechanism interprets URIs.

The BIND and REGISTER responses contain Binding header fields that allow a client to discover the URI.

7.2 Publication Mechanism

> The binding mechanism MUST allow a client to discover or infer the mechanism, or set of mechanisms, to use to publish data to a particular publication URI.

The URI scheme indicates the protocol.

7.3 Address of Record

> The binding mechanism MUST allow clients to determine binding information based only on knowledge of an AoR. A client MAY allow provisioning of individual service publication bindings for an AoR. The binding mechanism MUST allow for multiple data-driven services for a single AoL, and MUST allow the client to distinguish one such service from another for publication purposes. (For example, the AoR "sip:alice@example.com" may have both presence and CPL services associated with it. A client must be able to avoid sending CPL to the presence service, or a presence document to the CPL services.)

The Binding header field is used to return information about the purpose of each URI and only requires the issuer of the BIND request to know the AoR.

7.4 Enumeration of Services

> The binding mechanism SHOULD offer a way for a client to determine a list of all services for a given AoR for which it can publish data.

The Binding header field enumerates all such services. The Accept-Binding allows the AoR to provide an indication of which service bindings can be updated.

7.5 Lifetime of Publication URIs

> The binding mechanism MUST allow a service element to manage the lifetime of a publication URI. It MUST allow long-lived publication URIs. It MAY also allow very short-lived publication URIs; for example, the URI may be single-use only.

URI bindings are similar to REGISTER bindings and can have any lifetime, subject to server policy constraints and the Min-Expires header field value.

7.6 Communication of Publication URIs

> The binding mechanism MUST allow a service provider to communicate publication URIs to a client, directly or through a third party. For example, the client might directly query a service element, or query a directory service. The mechanism MAY also provide a method for a client to infer publication URIs from an AoR without directly contacting the service elements, for example by using an algorithmic transformation, schema mapping, or the DNS.

The mechanism uses standard SIP name translation mechanisms to obtain the publication URIs.

7.7 Separate publication points

> The mechanism MUST NOT require all publication URIs for a given AoR to share the same host, address, or even domain. The mechanism MUST NOT require the publication point for a data-driven service to be colocated with the network element(s) that provide the service itself.

URIs in this mechanism can point to any network resource.

7.8 Publication URIs that are not easily guessable

> The binding mechanism SHOULD allow the use of publication

URIs that are not easily guessable.

In the context of this binding mechanism, it is up to the client or registrar to ensure that URIs are not guessable. The mechanism neither prevents or enforces this requirement.

Since this mechanism and content indirection employ URIs for indirection, some of the requirements in [8] are applicable here. For example, the following seem particularly relevant, quoted verbatim:

- It MUST be possible to specify the timespan for which a given URL is valid. Applications of this mechanism MUST specify a lifetime for the URL. This may or may not be the same as the lifetime for the content itself.

- It MUST be possible to specify the purpose and disposition of each URL independently.

- It MUST be possible to label each URL to identify if and when the content referred to by that URL has changed. Applications of this mechanism may send the same URL more than once. The intention of this requirement is to allow the receiving party to determine if the content referenced by the URL has changed without having to actually retrieve that content. Example ways the URL could be labelled include a sequence number, timestamp, version number, etc.

7.9 Security

The binding mechanism MUST allow a client to authenticate the source of a publication URI. The mechanism MAY allow a publication URI provider to authenticate clients. The mechanism MUST allow a client to ensure that publication URIs have not been tampered with.

Appropriate SIP security mechanisms can be used to ensure the integrity of the binding data and the identity of the requestor. Details are in Section 11.

8 Alternatives Considered

The original binding mechanism [11] included scripts in REGISTER requests. That mechanism had a number of problems that motivated the more general mechanism described here: Including the data in the REGISTER request made it difficult to distinguish the lifetime of the REGISTER Contact binding from the lifetime of the script. Typically, the script would need to be refreshed far less frequently. Also, if

the script update failed for some reasons, it was not clear if the REGISTER update itself failed or succeeded. The mechanism for removing scripts, XXX, was a bit of a kludge.

We assume that URLs are sufficient to obtain data or cause the desired actions to occur. Beyond URIs, there does not seem to be a more generic, but generally accepted means, of referencing data objects.

Stucker [12] proposed a new method, DATA, to push content to a service. A slightly different version of the approach presented here would be to combine DATA with the content indirection mechanism discussed earlier. One advantage of the approach in this document is that it is easy to update multiple bindings at once. The major disadvantage of the indirection approach of the BIND method described here (or the DATA method with content indirection) is that authentication is likely to be more difficult. For example, if a client wants to update the server binding for CPL scripts, the server has to be told what identity and secrets to use to access that resource, if that resource is read-protected. The BIND approach does allow carrying literal data as message bodies, mitigating this problem. It seems likely that random URIs, effectively embedding a secret in the URI, will be the only easily configured mechanism for publication. That approach only improves upon the deprecated Basic authentication of plain-text passwords if the BIND bindings are end-to-end encrypted.

There are other non-SIP mechanisms that can be used for this purpose. For example, we could map the SIP URI sip:alice@example.com to the DNS RR alice.example.com and then use recent DNS mechanisms to obtain the data. Practically speaking, administering such bindings appears to be significantly more complicated, since most DNS servers are not set up to allow users to modify DNS entries. DNSsec is not widely available, making protecting the information difficult in practice. Often, the DNS service is run by somebody other than the owner of the SIP AoR, further complicating the process. A simpler, per-domain mapping, for bindings that are the same or defaults for a whole domain may well be of interest. It does not offer any obvious functional advantages compared to the method described in this draft.

Alternatively, directory services such as LDAP [13] could be employed, with an appropriate transformation of the AoR into an LDAP DN. A common schema needs to be designed. Such an approach could be easily integrated into a general whitepages services that stores other information about users, but adds significant complexity to SIP UAs that now also need to support LDAP.

9 Open Issues

Do we need Min-Expires?

Does it make sense to add a q value for bindings for the same disposition? Should there be multiple bindings for the same disposition, e.g., with different URI schemes, or does each disposition have one binding? If there are multiple bindings, there needs to be a mechanism that uniquely identifies one, e.g., with some kind of random tag.

Do we need to be able to remove all bindings, e.g., using the "*" mechanism for REGISTER bindings? (I don't think this is particularly useful.)

For many applications, it is desirable if user agents can subscribe to changes in the bindings. Subscription should be possible for each disposition, as user agents are likely to be interested in updates to only a small subset of the bindings. Unlike REGISTER bindings [14], the presence package does not overlap with this functionality.

10 IANA Considerations

New disposition parameters need to be registered by IANA.

11 Security Considerations

The BIND method is subject to similar attacks as the REGISTER method. Given that BIND can be considered a generalization of the REGISTER method, this should not be surprising.

Since the nformation transported by this mechanism may control how a server directs private information intended for a user, the server MUST reject all unauthenticated attempts to update bindings, and SHOULD require that the authentication method used verifies the integrity of the submitted data. Note, in particular, that Digest authentication does not ensure the integrity of header fields. As proposed [15], the sensitive parts of the SIP message can be carried as a SIP message fragment.

A UA SHOULD only accept bindings with a disposition that it can handle, to prevent being abused as a data storage mechanism.

12 Acknowledgements

Many of the mechanisms above have been discussed informally within the SIP and SIPPING WG.

13 Bibliography

[1] J. Lennox, H. Schulzrinne, and J. Rosenberg, "Common gateway interface for SIP," RFC 3050, Internet Engineering Task Force, Jan. 2001.

[2] Java Community Process, "SIP servlet API," Java Specification Requests JSR 116, Java Community Process, May 2002.

[3] J. Lennox and H. Schulzrinne, "CPL: A language for user control of internet telephony services," Internet Draft, Internet Engineering Task Force, Nov. 2001. Work in progress.

[4] X. Wu et al., "Use SIP and SOAP for conference floor control," Internet Draft, Internet Engineering Task Force, Apr. 2002. Work in progress.

[5] J. Rosenberg, "Session initiation protocol (SIP) extensions for presence," Internet Draft, Internet Engineering Task Force, May 2002. Work in progress.

[6] L. Masinter, "The "data" URL scheme," RFC 2397, Internet Engineering Task Force, Aug. 1998.

[7] E. Levinson, "Content-id and message-id uniform resource locators," RFC 2392, Internet Engineering Task Force, Aug. 1998.

[8] S. Olson, "Requirements for content indirection in SIP messages," Internet Draft, Internet Engineering Task Force, June 2002. Work in progress.

[9] S. Bradner, "Key words for use in RFCs to indicate requirement levels," RFC 2119, Internet Engineering Task Force, Mar. 1997.

[10] B. Campbell, "Requirements for binding published data to SIP services," Internet Draft, Internet Engineering Task Force, May 2002. Work in progress.

[11] J. Lennox and H. Schulzrinne, "Transporting user control information in SIP REGISTER payloads," Internet Draft, Internet Engineering Task Force, Mar. 1999. Work in progress.

[12] B. Stucker, "SIP-specific network service publishing," Internet Draft, Internet Engineering Task Force, Nov. 2001. Work in progress.

[13] M. Wahl, T. Howes, and S. Kille, "Lightweight directory access protocol (v3)," RFC 2251, Internet Engineering Task Force, Dec. 1997.

[14] G. Mayer and M. Beckmann, "Registration event package," Internet Draft, Internet Engineering Task Force, May 2002. Work in progress.

[15] J. Undery, S. Sen, and V. Torvinen, "Enhanced usage of HTTP digest authentication for SIP," Internet Draft, Internet Engineering Task Force, June 2002. Work in progress.

14 Authors' Addresses

Henning Schulzrinne
Dept. of Computer Science
Columbia University
1214 Amsterdam Avenue
New York, NY 10027
USA
electronic mail: schulzrinne@cs.columbia.edu