Stream: Internet Engineering Task Force (IETF)
RFC: 8688
Category: Standards Track
Published: December 2019
ISSN: 2070-1721
Authors: E.W. Burger, *Georgetown University*
B. Nagda, *Massachusetts Institute of Technology*

RFC 8688
A Session Initiation Protocol (SIP) Response Code for Rejected Calls

Abstract

This document defines the 608 (Rejected) Session Initiation Protocol (SIP) response code. This response code enables calling parties to learn that an intermediary rejected their call attempt. No one will deliver, and thus answer, the call. As a 6xx code, the caller will be aware that future attempts to contact the same User Agent Server will likely fail. The initial use case driving the need for the 608 response code is when the intermediary is an analytics engine. In this case, the rejection is by a machine or other process. This contrasts with the 607 (Unwanted) SIP response code in which a human at the target User Agent Server indicates the user did not want the call. In some jurisdictions, this distinction is important. This document also defines the use of the Call-Info header field in 608 responses to enable rejected callers to contact entities that blocked their calls in error. This provides a remediation mechanism for legal callers that find their calls blocked.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

The IETF has been addressing numerous issues surrounding how to handle unwanted and, depending on the jurisdiction, illegal calls [RFC5039]. Secure Telephone Identity Revisited (STIR) [RFC7340] and Signature-based Handling of Asserted information using toKENs (SHAKEN) [SHAKEN] address the cryptographic signing and attestation, respectively, of signaling to ensure the integrity and authenticity of the asserted caller identity.

This document describes a new Session Initiation Protocol (SIP) [RFC3261] response code, 608, which allows calling parties to learn that an intermediary rejected their call. As described below, we need a distinct indicator to differentiate between a user rejection and an intermediary's rejection of a call. In some jurisdictions, service providers may not be permitted to block calls, even if unwanted by the user, unless there is an explicit user request. Moreover, users may misidentify the nature of a caller.

For example, a legitimate caller may call a user who finds the call to be unwanted. However, instead of marking the call as unwanted, the user may mark the call as illegal. With that information, an analytics engine may determine to block all calls from that source. However, in some jurisdictions, blocking calls from that source for other users may not be legal. Likewise, one can envision jurisdictions that allow an operator to block such calls, but only if there is a remediation mechanism in place to address false positives.

Some call-blocking services may return responses such as 604 (Does Not Exist Anywhere). This might be a strategy to try to get a destination's address removed from a calling database. However, other network elements might also interpret this to mean the user truly does not exist, which might result in the user not being able to receive calls from anyone, even if they wanted to receive the calls. In many jurisdictions, providing such false signaling is also illegal.

The 608 response code addresses this need of remediating falsely blocked calls. Specifically, this code informs the SIP User Agent Client (UAC) that an intermediary blocked the call and provides a redress mechanism that allows callers to contact the operator of the intermediary.

In the current call handling ecosystem, users can explicitly reject a call or later mark a call as being unwanted by issuing a 607 SIP response code (Unwanted) [RFC8197]. Figures 1 and 2 show the operation of the 607 SIP response code. The User Agent Server (UAS) indicates the call was unwanted. As [RFC8197] explains, not only does the called party desire to reject that call, they can let their proxy know that they consider future calls from that source unwanted. Upon receipt of the 607 response from the UAS, the proxy may send unwanted call indicators, such as the value of the From header field and other information elements, to a call analytics engine. For various reasons described in [RFC8197], if a network operator receives multiple reports of unwanted calls, that may indicate that the entity placing the calls is likely to be a source of unwanted calls for many people. As such, other customers of the service provider may want the service provider to automatically reject calls on their behalf.

There is another value of the 607 rejection code. Presuming the proxy forwards the response code to the UAC, the calling UAC or intervening proxies will also learn the user is not interested in receiving calls from that sender.

```
                   +-----------+
                   |   Call    |
                   | Analytics |
                   |  Engine   |
                   +-----------+
                      ^     | (likely not SIP)
                      |     v
                   +-----------+
+-----+    607     |  Called   |    607    +-----+
| UAC | <--------- |  Party    |  <-------- | UAS |
+-----+            |  Proxy    |           +-----+
                   +-----------+
```

Figure 1: Unwanted (607) Call Flow

For calls rejected with a 607 from a legitimate caller, receiving a 607 response code can inform the caller to stop attempting to call the user. Moreover, if a legitimate caller believes the user is rejecting their calls in error, they can use other channels to contact the user. For example, if a pharmacy calls a user to let them know their prescription is available for pickup and the user mistakenly thinks the call is unwanted and issues a 607 response code, the pharmacy, having an existing relationship with the customer, can send the user an email or push a note to the pharmacist to ask the customer to consider not rejecting their calls in the future.

Many systems that allow the user to mark the call unwanted (e.g., with the 607 response code) also allow the user to change their mind and unmark such calls. This mechanism is relatively easy to implement as the user usually has a direct relationship with the service provider that is blocking calls.

However, things become more complicated if an intermediary, such as a third-party provider of call management services that classifies calls based on the relative likelihood that the call is unwanted, misidentifies the call as unwanted. Figure 3 shows this case. Note that the UAS typically does not receive an INVITE since the called party proxy rejects the call on behalf of the user. In this situation, it would be beneficial for the caller to learn who rejected the call so they can correct the misidentification.

```
                 +--------+         +-----------+
                 | Called |         |   Call    |
+-----+          | Party  |         | Analytics |   +-----+
| UAC |          | Proxy  |         |  Engine   |   | UAS |
+-----+          +--------+         +-----------+   +-----+
   |  INVITE         |                    |            |
   | --------------> |  Is call OK?       |            |
   |                 |------------------->|            |
   |                 |                    |            |
   |                 |               Yes  |            |
   |                 |<-------------------|            |
   |                 |                    |            |
   |                 | INVITE             |            |
   |                 | ------------------------------> |
   |                 |                    |            |
   |                 |                    |       607  |
   |                 | <------------------------------ |
   |                 |                    |            |
   |                 |  Unwanted call     |            |
   |            607  | -----------------> |            |
   | <-------------- |  indicators        |            |
   |                 |                    |            |
```

Figure 2: Unwanted (607) Ladder Diagram

```
                   +-----------+
                   |   Call    |
                   | Analytics |
                   |  Engine   |
                   +-----------+
                      ^     | (likely not SIP)
                      |     v
                   +-----------+
+-----+    608     |  Called   |           +-----+
| UAC | <--------- |  Party    |           | UAS |
+-----+            |  Proxy    |           +-----+
                   +-----------+
```

Figure 3: Rejected (608) Call Flow

In this situation, one might consider having the intermediary use the 607 response code. 607 indicates to the caller that the subscriber does not want the call. However, [RFC8197] specifies that one of the uses of 607 is to inform analytics engines that a user (human) has rejected a call. The problem here is that network elements downstream from the intermediary might interpret the 607 as coming from a user (human) who has marked the call as unwanted, as opposed to coming from an algorithm using statistics or machine learning to reject the call. An algorithm can be vulnerable to the base-rate fallacy [BaseRate] rejecting the call. In other words, those downstream entities should not rely on another entity "deciding" the call is unwanted. By distinguishing between a (human) user rejection and an intermediary engine's statistical rejection, a downstream network element that sees a 607 response code can weigh it as a human rejection in its call analytics, versus deciding whether to consider a 608 at all, and if so, weighing it appropriately.

It is useful for blocked callers to have a redress mechanism. One can imagine that some jurisdictions will require it. However, we must be mindful that most of the calls that intermediaries block will, in fact, be illegal and eligible for blocking. Thus, providing alternate contact information for a user would be counterproductive to protecting that user from illegal communications. This is another reason we do not propose to simply allow alternate contact information in a 607 response message.

Why do we not use the same mechanism an analytics service provider offers their customers? Specifically, why not have the analytics service provider allow the called party to correct a call blocked in error? The reason is that while there is an existing relationship between the customer (called party) and the analytics service provider, it is unlikely there is a relationship between the caller and the analytics service provider. Moreover, there are numerous call blocking providers in the ecosystem. Therefore, we need a mechanism for indicating an intermediary rejected a call that also provides contact information for the operator of that intermediary without exposing the target user's contact information.

The protocol described in this document uses existing SIP protocol mechanisms for specifying the redress mechanism. In the Call-Info header field passed back to the UAC, we send additional information specifying a redress address. We choose to encode the redress address using jCard [RFC7095]. As we will see later in this document, this information needs to have its own application-layer integrity protection. Thus, we use jCard rather than vCard [RFC6350], as we have a marshaling mechanism for creating a JavaScript Object Notation (JSON) [RFC8259] object, such as a jCard, and a standard integrity format for such an object, namely, JSON Web Signature (JWS) [RFC7515]. The SIP community is familiar with this concept as it is the mechanism used by STIR [RFC8224].

Integrity protecting the jCard with a cryptographic signature might seem unnecessary at first, but it is essential to preventing potential network attacks. Section 6 describes the attack and why we sign the jCard in more detail.

2. Terminology

The key words "**MUST**", "**MUST NOT**", "**REQUIRED**", "**SHALL**", "**SHALL NOT**", "**SHOULD**", "**SHOULD NOT**", "**RECOMMENDED**", "**NOT RECOMMENDED**", "**MAY**", and "**OPTIONAL**" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

3. Protocol Operation

This section uses the term "intermediary" to mean the entity that acts as a SIP UAS on behalf of the user in the network as opposed to the user's UAS (usually, but not necessarily, their phone). The intermediary could be a back-to-back user agent (B2BUA) or a SIP Proxy.

Figure 4 shows an overview of the call flow for a rejected call.

```
                    +--------+         +-----------+
                    | Called |         |   Call    |
   +-----+          | Party  |         | Analytics |   +-----+
   | UAC |          | Proxy  |         |  Engine   |   | UAS |
   +-----+          +--------+         +-----------+   +-----+
      |  INVITE         |                    |            |
      | -------------->  |  Is call OK?       |            |
      |                 |------------------->|            |
      |                 |                    |            |
      |                 |               Yes  |            |
      |                 |<-------------------|            |
      |                 |                    |            |
      |                 | INVITE             |            |
      |                 | ------------------------------> |
      |                 |                    |            |
      |                 |                    |       607  |
      |                 | <------------------------------ |
      |                 |                    |            |
      |                 |  Unwanted call     |            |
      |            607  | ----------------->  |            |
      | <-------------- |  indicators        |            |
      |                 |                    |            |
```

Figure 4: Rejected (608) Ladder Diagram

3.1. Intermediary Operation

An intermediary **MAY** issue the 608 response code in a failure response for an INVITE, MESSAGE, SUBSCRIBE, or other out-of-dialog SIP [RFC3261] request to indicate that an intermediary rejected the offered communication as unwanted by the user. An intermediary **MAY** issue the 608 as the value of the "cause" parameter of a SIP reason-value in a Reason header field [RFC3326].

If an intermediary issues a 608 code and there are no indicators the calling party will use the contents of the Call-Info header field for malicious purposes (see Section 6), the intermediary **MUST** include a Call-Info header field in the response.

If there is a Call-Info header field, it **MUST** have the "purpose" parameter of "jwscard". The value of the Call-Info header field **MUST** refer to a valid JSON Web Signature (JWS) [RFC7515] encoding of a jCard [RFC7095] object. The following section describes the construction of the JWS.

Proxies need to be mindful that a downstream intermediary may reject the attempt with a 608 while other paths may still be in progress. In this situation, the requirements stated in Section 16.7 of [RFC3261] apply. Specifically, the proxy should cancel pending transactions and must not create any new branches. Note this is not a new requirement but simply pointing out the existing 6xx protocol mechanism in SIP.

3.2. JWS Construction

The intermediary constructs the JWS of the jCard as follows.

3.2.1. JOSE Header

The Javascript Object Signing and Encryption (JOSE) header **MUST** include the typ, alg, and x5u parameters from JWS [RFC7515]. The typ parameter **MUST** have the value "vcard+json". Implementations **MUST** support ES256 as JSON Web Algorithms (JWA) [RFC7518] defines it and **MAY** support other registered signature algorithms. Finally, the x5u parameter **MUST** be a URI that resolves to the public key certificate corresponding to the key used to digitally sign the JWS.

3.2.2. JWT Payload

The payload contains two JSON values. The first JSON Web Token (JWT) claim that **MUST** be present is the "iat" (issued at) claim [RFC7519]. The "iat" **MUST** be set to the date and time of the issuance of the 608 response. This mandatory component protects the response from replay attacks.

The second JWT claim that **MUST** be present is the "jcard" claim. The value of the jcard [RFC7095] claim is a JSON array conforming to the JSON jCard data format defined in [RFC7095]. Section 5.3 describes the registration. In the construction of the jcard claim, the "jcard" **MUST** include at least one of the URL, EMAIL, TEL, or ADR properties. UACs supporting this specification **MUST** be prepared to receive a full jCard. Call originators (at the UAC) can use the information returned by the jCard to contact the intermediary that rejected the call to appeal the intermediary's blocking of the call attempt. What the intermediary does if the blocked caller contacts the intermediary is outside the scope of this document.

3.2.3. JWS Signature

JWS [RFC7515] specifies the procedure for calculating the signature over the jCard JWT. Section 4 of this document has a detailed example on constructing the JWS, including the signature.

3.3. UAC Operation

A UAC conforming to this specification **MUST** include the sip.608 feature-capability indicator in the Feature-Caps header field of the INVITE request.

Upon receiving a 608 response, UACs perform normal SIP processing for 6xx responses.

As for the disposition of the jCard itself, the UAC **MUST** check the "iat" claim in the JWT. As noted in Section 3.2.2, we are concerned about replay attacks. Therefore, the UAC **MUST** reject jCards that come with an expired "iat". The definition of "expired" is a matter of local policy. A reasonable value would be on the order of a minute due to clock drift and the possibility of the playing of an audio announcement before the delivery of the 608 response.

3.4. Legacy Interoperation

If the UAC indicates support for 608 and the intermediary issues a 608, life is good, as the UAC will receive all the information it needs to remediate an erroneous block by an intermediary. However, what if the UAC does not understand 608? For example, how can we support callers from a legacy, non-SIP, public-switched network connecting to the SIP network via a media gateway?

We address this situation by having the first network element that conforms with this specification play an announcement. See Section 3.5 for requirements on the announcement. The simple rule is a network element that inserts the sip.608 feature capability **MUST** be able to convey at a minimum how to contact the operator of the intermediary that rejected the call attempt.

The degenerate case is the intermediary is the only element that understands the semantics of the 608 response code. Obviously, any SIP device will understand that a 608 response code is a 6xx error. However, there are no other elements in the call path that understand the meaning of the value of the Call-Info header field. The intermediary knows this is the case as the INVITE request will not have the sip.608 feature capability. In this case, one can consider the intermediary to be the element "inserting" a virtual sip.608 feature capability. If the caveats described in Sections 3.5 and 6 do not hold, the intermediary **MUST** play the announcement.

Now we take the case where a network element that understands the 608 response code receives an INVITE for further processing. A network element conforming with this specification **MUST** insert the sip.608 feature capability per the behaviors described in Section 4.2 of [RFC6809].

Do note that even if a network element plays an announcement describing the contents of the 608 response message, the network element **MUST** forward the 608 response code message as the final response to the INVITE.

One aspect of using a feature capability is that only the network elements that will either consume (UAC) or play an announcement (media gateway, session border controller (SBC) [RFC7092], or proxy) need to understand the sip.608 feature capability. If the other network elements conform to Section 16.6 of [RFC3261], they will pass header fields such as "Feature-Caps: *;+sip.608" unmodified and without need for upgrade.

Because the ultimate disposition of the call attempt will be a 600-class response, the network element conveying the announcement in the legacy direction **MUST** use the 183 Session Progress response to establish the media session. Because of the small chance the UAC is an extremely old legacy device and is using UDP, the UAC **MUST** include support for 100rel [RFC3262] in its INVITE, the network element conveying the announcement **MUST** Require 100rel in the 183, and the UAC **MUST** issue a Provisional Response ACKnowledgement (PRACK) to which the network element **MUST** respond 200 OK PRACK.

3.5. Announcement Requirements

There are a few requirements on the element that handles the announcement for legacy interoperation.

As noted above, the element that inserts the sip.608 feature capability is responsible for conveying the information referenced by the Call-Info header field in the 608 response message. However, this specification does not mandate how to convey that information.

Let us take the case where a telecommunications service provider controls the element inserting the sip.608 feature capability. It would be reasonable to expect the service provider would play an announcement in the media path towards the UAC (caller). It is important to note the network element should be mindful of the media type requested by the UAC as it formulates the announcement. For example, it would make sense for an INVITE that only indicated audio codecs in the Session Description Protocol (SDP) [RFC4566] to result in an audio announcement. Likewise, if the INVITE only indicated real-time text [RFC4103] and the network element can render the information in the requested media format, the network element should send the information in a text format.

It is also possible for the network element inserting the sip.608 feature capability to be under the control of the same entity that controls the UAC. For example, a large call center might have legacy UACs, but have a modern outbound calling proxy that understands the full semantics of the 608 response code. In this case, it is enough for the outbound calling proxy to digest the Call-Info information and handle the information digitally rather than "transcoding" the Call-Info information for presentation to the caller.

4. Examples

These examples are not normative, do not include all protocol elements, and may have errors. Review the protocol documents for actual syntax and semantics of the protocol elements.

4.1. Full Exchange

Given an INVITE, shamelessly taken from [SHAKEN], with the line breaks in the Identity header field for display purposes only:

```
INVITE sip:+12155550113@tel.one.example.net SIP/2.0
Max-Forwards: 69
Contact: <sip:+12155550112@[2001:db8::12]:50207;rinstance=9da3088f3>
To: <sip:+12155550113@tel.one.example.net>
From: "Alice" <sip:+12155550112@tel.two.example.net>;tag=614bdb40
Call-ID: 79048YzkxNDA5NTI1MzA0OWFjOTFkMmFlODhiNTI2OWQ1ZTI
P-Asserted-Identity: "Alice"<sip:+12155550112@tel.two.example.net>,
    <tel:+12155550112>
CSeq: 2 INVITE
Allow: SUBSCRIBE, NOTIFY, INVITE, ACK, CANCEL, BYE, REFER, INFO,
    MESSAGE, OPTIONS
Content-Type: application/sdp
Date: Tue, 16 Aug 2016 19:23:38 GMT
Feature-Caps: *;+sip.608
Identity: eyJhbGciOiJFUzI1NiIsInR5cCI6InBhc3Nwb3J0IiwicHB0Ijoic2hha2V
uIiwieDV1IjoiaHR0cDovL2NlcnQuZXhhbXBsZTIubmV0L2V4YW1wbGUuY2VydCJ9.eyJ
hdHRlc3QiOiJBIiwiZGVzdCI6eyJ0biI6IisxMjE1NTU1MDExMyJ9LCJpYXQiOiIxNDcx
Mzc1NDE4Iiwib3JpZyI6eyJ0biI6IisxMjE1NTU1MDExMiJ9LCJvcmlnaWQiOiIxMjNlN
DU2Ny1lODliLTEyZDMtYTQ1Ni00MjY2NTU0NDAwMCJ9.QAht_eFqQlaoVrnEV56Qly-OU
tsDGifyCcpYjWcaR661Cz1hutFH2BzIlDswTahO7ujjqsWjeoOb4h97whTQJg;info=
<http://cert.example2.net/example.cert>;alg=ES256
Content-Length: 153

v=0
o=- 13103070023943130 1 IN IP6 2001:db8::177
c=IN IP6 2001:db8::177
t=0 0
m=audio 54242 RTP/AVP 0
a=sendrecv
```

An intermediary could reply:

```
SIP/2.0 608 Rejected
Via: SIP/2.0/UDP [2001:db8::177]:60012;branch=z9hG4bK-524287-1
From: "Alice" <sip:+12155550112@tel.two.example.net>;tag=614bdb40
To: <sip:+12155550113@tel.one.example.net>
Call-ID: 79048YzkxNDA5NTI1MzA0OWFjOTFkMmFlODhiNTI2OWQ1ZTI
CSeq: 2 INVITE
Call-Info: <https://block.example.net/complaint-jws>;purpose=jwscard
```

The location https://block.example.net/complaint-jws resolves to a JWS. One would construct the JWS as follows.

The JWS header of this example jCard could be:

```
{ "alg":"ES256",
  "typ":"vcard+json",
  "x5u":"https://certs.example.net/reject_key.cer"
}
```

Now, let us construct a minimal jCard. For this example, the jCard refers the caller to an email address, remediation@blocker.example.net:

```
["vcard",
  [
    ["version", {}, "text", "4.0"],
    ["fn", {}, "text", "Robocall Adjudication"],
    ["email", {"type":"work"}, "text",
     "remediation@blocker.example.net"]
  ]
]
```

With this jCard, we can now construct the JWT:

```
{
  "iat":1546008698,
  "jcard":["vcard",
    [
      ["version", {}, "text", "4.0"],
      ["fn", {}, "text", "Robocall Adjudication"],
      ["email", {"type":"work"},
       "text", "remediation@blocker.example.net"]
    ]
  ]
}
```

To calculate the signature, we need to encode the JSON Object Signing and Encryption (JOSE) header and JWT into base64url. As an implementation note, one can trim whitespace in the JSON objects to save a few bytes. UACs **MUST** be prepared to receive pretty-printed, compact, or bizarrely formatted JSON. For the purposes of this example, we leave the objects with pretty whitespace. Speaking of pretty vs. machine formatting, these examples have line breaks in the base64url encodings for ease of publication in the RFC format. The specification of base64url allows for these line breaks, and the decoded text works just fine. However, those extra line-break octets would affect the calculation of the signature. Implementations **MUST NOT** insert line breaks into the base64url encodings of the JOSE header or JWT. This also means UACs **MUST** be prepared to receive arbitrarily long octet streams from the URI referenced by the Call-Info header field.

base64url of JOSE header:

```
eyJhbGciOiJFUzI1NiIsInR5cCI6InZjYXJkK2pzb24iLCJ4NXUiOiJodHRwczov
L2NlcnRzLmV4YW1wbGUubmV0L3JlamVjdF9rZXkuY2VyIn0=
```

base64url of JWT:

```
eyJpYXQiOjE1NDYwMDg2OTgsImpjYXJkIjpbInZjYXJkIixbWyJ2ZXJzaW9uIix7
fSwidGV4dCIsIjQuMCJdLFsiZm4iLHt9LCJ0ZXh0IiwiUm9ib2NhbGwgQWRqdWRp
Y2F0aW9uIl0sWyJlbWFpbCIseyJ0eXBlIjoid29yayJ9LCJ0ZXh0IiwicmVtZWRp
YXRpb25AYmxvY2tlci5leGFtcGxlLm5ldCJdXV19
```

In this case, the object to sign (remembering this is just a single long line; the line breaks are for ease of review but do not appear in the actual object) is as follows:

```
eyJhbGciOiJFUzI1NiIsInR5cCI6InZjYXJk
K2pzb24iLCJ4NXUiOiJodHRwczovL2NlcnRzLmV4YW1wbGUubmV0L3JlamVjdF9r
ZXkuY2VyIn0.eyJpYXQiOjE1NDYwMDg2OTgsImpjYXJkIjpbInZjYXJkIixbWyJ2
ZXJzaW9uIix7fSwidGV4dCIsIjQuMCJdLFsiZm4iLHt9LCJ0ZXh0IiwiUm9ib2Nh
bGwgQWRqdWRpY2F0aW9uIl0sWyJlbWFpbCIseyJ0eXBlIjoid29yayJ9LCJ0ZXh0
IiwicmVtZWRpYXRpb25AYmxvY2tlci5leGFtcGxlLm5ldCJdXV19
```

We use the following X.509 PKCS #8-encoded Elliptic Curve Digital Signature Algorithm (ECDSA) key, also shamelessly taken from [SHAKEN], as an example key for signing the hash of the above text. Do NOT use this key in real life! It is for example purposes only. At the very least, we would strongly recommend encrypting the key at rest.

```
-----BEGIN PRIVATE KEY-----
MIGHAgEAMBMGByqGSM49AgEGCCqGSM49AwEHBG0wawIBAQQgi7q2TZvN9VDFg8Vy
qCP06bETrR2v8MRvr89rn4i+UAahRANCAAQWfaj1HUETpoNCrOtp9KA8o0V79IuW
ARKt9C1cFPkyd3FBP4SeiNZxQhDrD0tdBHls3/wFe8++K2FrPyQF9vuh
-----END PRIVATE KEY-----

-----BEGIN PUBLIC KEY-----
MFkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDQgAE8HNbQd/TmvCKwPKHkMF9fScavGeH
78YTU8qLS8I5HLHSSmlATLcslQMhNC/OhlWBYC626nIlo7XeebYS7Sb37g==
-----END PUBLIC KEY-----
```

The resulting JWS, using the above key on the above object, renders the following ECDSA P-256 SHA-256 digital signature.

```
7uz2SADRvPFOQOO_UgF2ZTUjPlDTegtPrYB04UHBMwBD6g9AmL
5harLJdTKDSTtH-LOV1jwJaGRUOUJiwP27ag
```

Thus, the JWS stored at https://blocker.example.net/complaints-jws would contain:

```
eyJhbGciOiJFUzI1NiIsInR5cCI6InZjYXJkK2pzb24iLCJ4NXUiOiJodHRwczovL
2NlcnRzLmV4YW1wbGUubmV0L3JlamVjdF9rZXkuY2VyIn0.eyJpYXQiOjE1NDYwMD
g2OTgsImpjYXJkIjpbInZjYXJkIixbWyJ2ZXJzaW9uIix7fSwidGV4dCIsIjQuMCJ
dLFsiZm4iLHt9LCJ0ZXh0IiwiUm9ib2NhbGwgQWRqdWRpY2F0aW9uIl0sWyJlbWFp
bCIseyJ0eXBlIjoid29yayJ9LCJ0ZXh0IiwicmVtZWRpYXRpb25AYmxvY2tlci5le
GFtcGxlLm5ldCJdXV19.7uz2SADRvPFOQOO_UgF2ZTUjPlDTegtPrYB04UHBMwBD6
g9AmL5harLJdTKDSTtH-LOV1jwJaGRUOUJiwP27ag
```

4.2. Web Site jCard

For an intermediary that provides a Web site for adjudication, the jCard could contain the following. Note that we do not show the calculation of the JWS; the URI reference in the Call-Info header field would be to the JWS of the signed jCard.

```
["vcard",
  [
    ["version", {}, "text", "4.0"],
    ["fn", {}, "text", "Robocall Adjudication"],
    ["url", {"type":"work"},
     "text", "https://blocker.example.net/adjudication-form"]
  ]
]
```

4.3. Multi-modal jCard

For an intermediary that provides a telephone number and a postal address, the jCard could contain the following. Note that we do not show the calculation of the JWS; the URI reference in the Call-Info header field would be to the JWS of the signed jCard.

```
["vcard",
  [
    ["version", {}, "text", "4.0"],
    ["fn", {}, "text", "Robocall Adjudication"],
    ["adr", {"type":"work"}, "text",
      ["Argument Clinic",
       "12 Main St","Anytown","AP","000000","Somecountry"]
    ]
    ["tel", {"type":"work"}, "uri", "tel:+1-555-555-0112"]
  ]
]
```

Note that it is up to the UAC to decide which jCard contact modality, if any, it will use.

4.4. Legacy Interoperability

Figure 5 depicts a call flow illustrating legacy interoperability. In this non-normative example, we see a UAC that does not support the full semantics for 608. However, there is an SBC that does support 608. Per [RFC6809], the SBC can insert "*;+sip.608" into the Feature-Caps header field for the INVITE. When the intermediary, labeled "Called Party Proxy" in the figure, rejects the call, it knows it can simply perform the processing described in this document. Since the intermediary saw the sip.608 feature capability, it knows it does not need to send any media describing whom to contact in the event of an erroneous rejection. For illustrative purposes, the figure shows generic SIP Proxies in the flow. Their presence or absence or the number of proxies is not relevant to the operation of the protocol. They are in the figure to show that proxies that do not understand the sip.608 feature capability can still participate in a network offering 608 services.

```
                                                  +---------+
                                                  |  Call   |
                                                  |Analytics|
                                                  | Engine  |
                                                  +--+--+---+
                                                     ^  |
                                                     |  |
                                                     |  v
                                                   +-+--+-+
+---+    +-----+    +---+    +-----+    +-----+    |Called|
|UAC+----+Proxy+----+SBC+----+Proxy+----+Proxy+----+Party |
+---+    +-----+    +---+    +-----+    +-----+    |Proxy |
  |                   |                            +------+
  | INVITE            |                               |
  |------------------>|                               |
  |                   | INVITE                        |
  |                   |------------------------------>|
  |                   | Feature-Caps: *;+sip.608      |
  |                   |                               |
  |                   |                  608 Rejected |
  |                   |<------------------------------|
  |               183 |              Call-Info: <...> |
  |<------------------|    [path for Call-Info elided |
  |     SDP for media |     for illustration purposes]|
  |                   |                               |
  | PRACK             |                               |
  |------------------>|                               |
  |                   |                               |
  |      200 OK PRACK |                               |
  |<------------------|                               |
  |                   |                               |
  |<== Announcement ==|                               |
  |                   |                               |
  |      608 Rejected |                               |
  |<------------------|                               |
  |  Call-Info: <...> |                               |
  |                   |                               |
```

Figure 5: Legacy Operation

When the SBC receives the 608 response code, it correlates that with the original INVITE from the UAC. The SBC remembers that it inserted the sip.608 feature capability, which means it is responsible for somehow alerting the UAC the call failed and disclosing whom to contact. At this point, the SBC can play a prompt, either natively or through a mechanism such as NETANN [RFC4240], that sends the relevant information in the appropriate media to the UAC. Since this is a potentially long transaction and there is a chance the UAC is using an unreliable transport protocol, the UAC will have indicated support for provisional responses, the SBC will indicate it requires a PRACK from the UAC in the 183 response, the UAC will provide the PRACK, and the SBC will acknowledge receipt of the PRACK before playing the announcement.

As an example, the SBC could extract the FN and TEL jCard fields and play something like a special information tone (see Section 6.21.2.1 of Telcordia [SR-2275] or Section 7 of ITU-T E.180 [ITU.E.180.1998]), followed by "Your call has been rejected by...", followed by a text-to-speech translation of the FN text, followed by "You can reach them on...", followed by a text-to-speech translation of the telephone number in the TEL field.

Note that the SBC also still sends the full 608 response code, including the Call-Info header field, towards the UAC.

5. IANA Considerations

5.1. SIP Response Code

This document defines a new SIP response code, 608, in the "Response Codes" subregistry of the "Session Initiation Protocol (SIP) Parameters" registry defined in [RFC3261].

Response code:	608
Description:	Rejected
Reference:	RFC 8688

5.2. SIP Feature-Capability Indicator

This document defines the feature capability, sip.608, in the "SIP Feature-Capability Indicator Registration Tree" registry defined in [RFC6809].

Name:	sip.608
Description:	This feature-capability indicator, when included in a Feature-Caps header field of an INVITE request, indicates that the entity associated with the indicator will be responsible for indicating to the caller any information contained in the 608 SIP response code, specifically, the value referenced by the Call-Info header field.
Reference:	RFC 8688

5.3. JSON Web Token Claim

This document defines the new JSON Web Token claim in the "JSON Web Token Claims" subregistry created by [RFC7519]. Section 3.2.2 defines the syntax. The required information is:

Claim Name:	jcard
Claim Description:	jCard data
Change Controller:	IESG
Reference:	RFC 8688, [RFC7095]

5.4. Call-Info Purpose

This document defines the new predefined value "jwscard" for the "purpose" header field parameter of the Call-Info header field. This modifies the "Header Field Parameters and Parameter Values" subregistry of the "Session Initiation Protocol (SIP) Parameters" registry by adding this RFC as a reference to the line for the header field "Call-Info" and parameter name "purpose":

Header Field:	Call-Info
Parameter Name:	purpose
Predefined Values:	Yes
Reference:	RFC 8688

6. Security Considerations

Intermediary operators need to be mindful to whom they are sending the 608 response. The intermediary could be rejecting a truly malicious caller. This raises two issues. The first is the caller, now alerted that an intermediary is automatically rejecting their call attempts, may change their call behavior to defeat call-blocking systems. The second, and more significant risk, is that by providing a contact in the Call-Info header field, the intermediary may be giving the malicious caller a vector for attack. In other words, the intermediary will be publishing an address that a malicious actor may use to launch an attack on the intermediary. Because of this, intermediary operators may wish to configure their response to only include a Call-Info header field for INVITE, or other signed initiating methods, that pass validation by STIR [RFC8224].

Another risk is as follows. Consider an attacker that floods a proxy that supports the sip.608 feature. However, the SDP in the INVITE request refers to a victim device. Moreover, the attacker somehow knows there is a 608-aware gateway connecting to the victim who is on a segment that lacks the sip.608 feature capability. Because the mechanism described here can result in sending an audio file to the target of the SDP, an attacker could use the mechanism described by this document as an amplification attack, given a SIP INVITE can be under 1 kilobyte and an audio file can be hundreds of kilobytes. One remediation for this is for devices that insert a sip.608 feature capability to only transmit media to what is highly likely to be the actual source of the call attempt. A method for this is to only play media in response to a STIR-signed INVITE that passes validation. Beyond requiring a valid STIR signature on the INVITE, the intermediary can also use remediation procedures such as doing the connectivity checks specified by Interactive Connectivity Establishment [RFC8445]. If the target did not request the media, the check will fail.

Yet another risk is a malicious intermediary that generates a malicious 608 response with a jCard referring to a malicious agent. For example, the recipient of a 608 may receive a TEL URI in the vCard. When the recipient calls that address, the malicious agent could ask for personally identifying information. However, instead of using that information to verify the recipient's identity, they are phishing the information for nefarious ends. A similar scenario can unfold if the malicious agent inserts a URI that points to a phishing or other site. As such, we strongly recommend the recipient validates to whom they are communicating with if asking to adjudicate an erroneously rejected call attempt. Since we may also be concerned about intermediate nodes modifying contact information, we can address both issues with a single solution. The remediation is to require the intermediary to sign the jCard. Signing the jCard provides integrity protection. In addition, one can imagine mechanisms such as used by [SHAKEN].

Similarly, one can imagine an adverse agent that maliciously spoofs a 608 response with a victim's contact address to many active callers who may then all send redress requests to the specified address (the basis for a denial-of-service attack). The process would occur as follows: (1) a malicious agent senses INVITE requests from a variety of UACs and (2) spoofs 608 responses with an unsigned redress address before the intended receivers can respond, causing (3) the UACs to all contact the redress address at once. The jCard encoding allows the UAC to verify the blocking intermediary's identity before contacting the redress address. Specifically, because the sender signs the jCard, we can cryptographically trace the sender of the jCard. Given the protocol machinery of having a signature, one can apply local policy to decide whether to believe that the sender of the jCard represents the owner of the contact information found in the jCard. This guards against a malicious agent spoofing 608 responses.

Specifically, one could use policies around signing certificate issuance as a mechanism for traceback to the entity issuing the jCard. One check could be verifying that the identity of the subject of the certificate relates to the To header field of the initial SIP request, similar to validating that the intermediary was vouching for the From header field of a SIP request with that identity. Note that we are only protecting against a malicious intermediary and not a hidden intermediary attack (formerly known as a "man-in-the-middle attack"). Thus, we only need to ensure the signature is fresh, which is why we include "iat". For most implementations, we assume that the intermediary has a single set of contact points and will generate the jCard on demand. As such, there is no need to directly correlate HTTPS fetches to specific calls. However, since the intermediary is in control of the jCard and Call-Info response, an intermediary may choose to encode per-call information in the URI returned in a given 608 response. However, if the intermediary does go that route, the intermediary **MUST** use a non-deterministic URI reference mechanism and be prepared to return dummy responses to URI requests referencing calls that do not exist so that attackers attempting to glean call metadata by guessing URIs (and thus calls) will not get any actionable information from the HTTPS GET.

Since the decision of whether to include Call-Info in the 608 response is a matter of policy, one thing to consider is whether a legitimate caller can ascertain whom to contact without including such information in the 608. For example, in some jurisdictions, if only the terminating service provider can be the intermediary, the caller can look up who the terminating service provider is based on the routing information for the dialed number. Thus, the Call-Info jCard could be redundant information. However, the factors going into a particular service provider's or jurisdiction's choice of whether to include Call-Info is outside the scope of this document.

7. References

7.1. Normative References

[RFC2119]

Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC3261] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, DOI 10.17487/RFC3261, June 2002, <https://www.rfc-editor.org/info/rfc3261>.

[RFC3262] Rosenberg, J. and H. Schulzrinne, "Reliability of Provisional Responses in Session Initiation Protocol (SIP)", RFC 3262, DOI 10.17487/RFC3262, June 2002, <https://www.rfc-editor.org/info/rfc3262>.

[RFC3326] Schulzrinne, H., Oran, D., and G. Camarillo, "The Reason Header Field for the Session Initiation Protocol (SIP)", RFC 3326, DOI 10.17487/RFC3326, December 2002, <https://www.rfc-editor.org/info/rfc3326>.

[RFC6809] Holmberg, C., Sedlacek, I., and H. Kaplan, "Mechanism to Indicate Support of Features and Capabilities in the Session Initiation Protocol (SIP)", RFC 6809, DOI 10.17487/RFC6809, November 2012, <https://www.rfc-editor.org/info/rfc6809>.

[RFC7095] Kewisch, P., "jCard: The JSON Format for vCard", RFC 7095, DOI 10.17487/RFC7095, January 2014, <https://www.rfc-editor.org/info/rfc7095>.

[RFC7515] Jones, M., Bradley, J., and N. Sakimura, "JSON Web Signature (JWS)", RFC 7515, DOI 10.17487/RFC7515, May 2015, <https://www.rfc-editor.org/info/rfc7515>.

[RFC7518] Jones, M., "JSON Web Algorithms (JWA)", RFC 7518, DOI 10.17487/RFC7518, May 2015, <https://www.rfc-editor.org/info/rfc7518>.

[RFC7519] Jones, M., Bradley, J., and N. Sakimura, "JSON Web Token (JWT)", RFC 7519, DOI 10.17487/RFC7519, May 2015, <https://www.rfc-editor.org/info/rfc7519>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

7.2. Informative References

[BaseRate] Bar-Hillel, M., "The Base-Rate Fallacy in Probability Judgements", April 1977, <https://apps.dtic.mil/docs/citations/ADA045772>.

[ITU.E.180.1998] ITU-T, "Technical characteristics of tones for the telephone service", ITU-T Recommendation E.180/Q.35, March 1998.

[RFC4103] Hellstrom, G. and P. Jones, "RTP Payload for Text Conversation", RFC 4103, DOI 10.17487/RFC4103, June 2005, <https://www.rfc-editor.org/info/rfc4103>.

[RFC4240] Burger, E., Ed., Van Dyke, J., and A. Spitzer, "Basic Network Media Services with SIP", RFC 4240, DOI 10.17487/RFC4240, December 2005, <https://www.rfc-editor.org/info/rfc4240>.

[RFC4566] Handley, M., Jacobson, V., and C. Perkins, "SDP: Session Description Protocol", RFC 4566, DOI 10.17487/RFC4566, July 2006, <https://www.rfc-editor.org/info/rfc4566>.

[RFC5039] Rosenberg, J. and C. Jennings, "The Session Initiation Protocol (SIP) and Spam", RFC 5039, DOI 10.17487/RFC5039, January 2008, <https://www.rfc-editor.org/info/rfc5039>.

[RFC6350] Perreault, S., "vCard Format Specification", RFC 6350, DOI 10.17487/RFC6350, August 2011, <https://www.rfc-editor.org/info/rfc6350>.

[RFC7092] Kaplan, H. and V. Pascual, "A Taxonomy of Session Initiation Protocol (SIP) Back-to-Back User Agents", RFC 7092, DOI 10.17487/RFC7092, December 2013, <https://www.rfc-editor.org/info/rfc7092>.

[RFC7340] Peterson, J., Schulzrinne, H., and H. Tschofenig, "Secure Telephone Identity Problem Statement and Requirements", RFC 7340, DOI 10.17487/RFC7340, September 2014, <https://www.rfc-editor.org/info/rfc7340>.

[RFC8197] Schulzrinne, H., "A SIP Response Code for Unwanted Calls", RFC 8197, DOI 10.17487/RFC8197, July 2017, <https://www.rfc-editor.org/info/rfc8197>.

[RFC8224] Peterson, J., Jennings, C., Rescorla, E., and C. Wendt, "Authenticated Identity Management in the Session Initiation Protocol (SIP)", RFC 8224, DOI 10.17487/RFC8224, February 2018, <https://www.rfc-editor.org/info/rfc8224>.

[RFC8259] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", STD 90, RFC 8259, DOI 10.17487/RFC8259, December 2017, <https://www.rfc-editor.org/info/rfc8259>.

[RFC8445] Keranen, A., Holmberg, C., and J. Rosenberg, "Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal", RFC 8445, DOI 10.17487/RFC8445, July 2018, <https://www.rfc-editor.org/info/rfc8445>.

[SHAKEN] ATIS/SIP Forum IP-INNI Task Group, "Signature-based Handling of Asserted information using toKENs (SHAKEN)", ATIS 1000074, January 2017, <https://www.sipforum.org/download/sip-forum-twg-10-signature-based-handling-of-asserted-information-using-tokens-shaken-pdf/?wpdmdl=2813>.

[SR-2275] Telcordia, "Telcordia Notes on the Networks", Telcordia SR-2275, October 2000.

Acknowledgements

This document liberally lifts from [RFC8197] in its text and structure. However, the mechanism and purpose of 608 is quite different than 607. Any errors are the current editor's and not the editor of RFC 8197. Thanks also go to Ken Carlberg of the FCC, Russ Housley, Paul Kyzivat, and Tolga Asveren for their suggestions on improving the document. Tolga's suggestion to provide a mechanism for legacy interoperability served to expand the document by 50%. In addition, Tolga came up with the jCard attack. Finally, Christer Holmberg, as always, provided a close reading and fixed a SIP feature-capability bug found by Yehoshua Gev.

Of course, we appreciated the close read and five pages of comments from our estimable Area Director, Adam Roach. In addition, we received valuable comments during IETF Last Call and JWT review from Ines Robles, Mike Jones, and Brian Campbell, and IESG review from Alissa Cooper, Eric Vyncke, Alexey Melnikov, Benjamin Kaduk, Barry Leiba, and with most glee, Warren Kumari.

Finally, Bhavik Nagda provided clarifying edits as well and, more especially, wrote and tested an implementation of the 608 response code in Kamailio. Code is available at https://github.com/nagdab/608_Implementation. Grace Chuan from MIT regenerated and verified the JWT while working at the FCC.

Authors' Addresses

Eric W. Burger
Georgetown University
37th & O St, NW
Washington, DC 20057
United States of America
Email: eburger@standardstrack.com

Bhavik Nagda
Massachusetts Institute of Technology
77 Massachusetts Avenue
Cambridge, MA 02139
United States of America
Email: nagdab@gmail.com