XEP-0220: Server Dialback

Jeremie Miller
mailto:jer@jabber.org
xmpp:jer@jabber.org

Peter Saint-Andre
mailto:xsf@stpeter.im
xmpp:peter@jabber.org
http://stpeter.im/

Philipp Hancke
mailto:fippo@andyet.com
xmpp:fippo@goodadvice.pages.de

2015-03-12
Version 1.1.1

Status	Type	Short Name
Draft	Standards Track	dialback

This specification defines the Server Dialback protocol, which is used between XMPP servers to provide identity verification. Server Dialback uses the Domain Name System (DNS) as the basis for verifying identity; the basic approach is that when a receiving server accepts a server-to-server connection from an initiating server, it does not process XMPP stanzas over the connection until it has verified the initiating server's identity. Additionally, the protocol is used to negotiate whether the receiving server is accepting stanzas for the target domain. Although Server Dialback does not provide strong authentication and is subject to DNS poisoning attacks, it has effectively prevented most address spoofing on the XMPP network since its development in the year 2000.

Legal

Copyright

Permissions

Warranty

Liability

Conformance

Contents

1 Introduction

1.1 Why Dialback?

When Jabber technologies were first developed in 1998, they were conceived of as a client-server system similar to email, wherein a client would connect to a server in order to communicate with other clients. Similarly, servers would connect with peer servers to provide inter-domain communication (often called "federation"). In a system that allows federation, it is important for a server to be able to determine the identity of a peer server. Therefore the Jabber developer community designed a protocol called "Server Dialback" for identity verification based on the Domain Name System (DNS), built support for that protocol into the jabberd 1.2 server (released in October 2000), and mandated support for that protocol on the emerging Jabber server network.

The basic idea behind Server Dialback is that a receiving server does not accept XMPP traffic from a sending server until it has (a) "called back" the authoritative server for the domain asserted by the sending server and (b) verified that the sending server is truly authorized to generate XMPP traffic for that domain. The protocol also ensures that the receiving server is accepting stanzas for the target domain.

When the early Jabber protocols were formalized by the XMPP Working Group of the Internet Engineering Task Force (IETF) [1] in 2002-2004, support for strong identity verification was added (see RFC 3920 [2], since updated by RFC 6120 [3]). That support takes the form of Transport Layer Security (TLS) for encryption of server-to-server XML streams and the Simple Authentication and Security Layer (SASL) for authentication of such streams, typically using digital certificates issued by trusted root certification authorities (CAs). However, the Server Dialback protocol is still in wide use. In addition, the slow but steady deployment of the DNS security extensions (DNSSEC) RFC 4033 [4] can provide a stronger basis for using Server Dialback.

1.2 What Dialback Accomplishes

Server Dialback is a method for identity verification: if the dialback negotiation succeeds, the receiving server for an XML stream can associate a pair of domain names with the stream; those two domain names are the sender domain asserted by the initiating server and the domain name at the receiving server that the initiating server has indicated it wishes to communicate with.

Traditionally, the verification accomplished in Server Dialback has depended on the Domain Name System (DNS) and the use of keys based on a shared secret known to all XMPP servers within a given administrative domain. It is a proof-of-possession protocol in the sense of RFC

[1] The Internet Engineering Task Force is the principal body engaged in the development of new Internet standard specifications, best known for its work on standards such as HTTP and SMTP. For further information, see <http://www.ietf.org/>.

[2] RFC 3920: Extensible Messaging and Presence Protocol (XMPP): Core <http://tools.ietf.org/html/rfc3920>.

[3] RFC 6120: Extensible Messaging and Presence Protocol (XMPP): Core <http://tools.ietf.org/html/rfc6120>.

[4] RFC 4033: DNS Security Introduction and Requirements <http://tools.ietf.org/html/rfc4033>.

4949 [5] which asserts that the initiating server and the authoritative server are associated with each other. The relative strength or weakness of the verification depends in part on the strength or weakness of the process for resolving the domain names of the authoritative server; in particular, if DNSSEC is not used then Server Dialback results in weak identity verification, whereas if DNSSEC is used then Server Dialback can result in fairly strong identity verification.

Since October 2000, the use of Server Dialback (even absent DNSSEC) has made it more difficult to spoof the hostnames of servers (and therefore the addresses of sent messages) on the XMPP network.

Server Dialback is unidirectional, and results in verification for one XML stream in one direction. Because traditionally Server-to-Server connections are used unidirectionally, Server Dialback needs to be completed in each direction in order to enable bidirectional communication between two domains (unless Bidirectional Server-to-Server Connections (XEP-0288) [6] is used).

Furthermore, because a separate TCP connection is mandated for each domain pair, the use of server dialback results in significant scalability challenges for large XMPP service providers that host many domains (see RFC 7712 [7] for a possible solution).

Finally, dialback signalling can be used without basing the identity verification on checking of the dialback key provided by the Initiating Server. As one example, if Transport Layer Security (TLS) is used then the Receiving Server can attempt to verify the certificate presented by the Initiating Server, either according to the PKIX-based rules specified in Best Practices for Use of SASL EXTERNAL (XEP-0178) [8], RFC 6120, and RFC 6125 [9] or by checking that the public key or certificate of the Initiating Server matches a public key or certificate obtained via POSH [10]. However, this technique of using dialback signalling without verifying the dialback key (sometimes called "dialback without dialing back" since the Receiving Server does not contact the Authoritative Server) is not described in this document.

1.3 Terminology

This document uses the following terms.

Authoritative Server The machine that is discovered by means of a DNS lookup for the Sender Domain; for simple deployments this will be the Initiating Server, but it could be a separate machine in the Initiating Server's network (where "network" is defined by knowledge of a shared secret for verification of dialback keys).

[5] RFC 4949: Internet Security Glossary, Version 2 <http://tools.ietf.org/html/rfc4949>.

[6] XEP-0288: Bidirectional Server-to-Server Connections <https://xmpp.org/extensions/xep-0288.html>.

[7] RFC 7712: Domain Name Associations (DNA) in the Extensible Messaging and Presence Protocol (XMPP)<http://tools.ietf.org/html/rfc7712>.

[8] XEP-0178: Best Practices for Use of SASL EXTERNAL <https://xmpp.org/extensions/xep-0178.html>.

[9] RFC 6125: Representation and Verification of Domain-Based Application Service Identity within Internet Public Key Infrastructure Using X.509 (PKIX) Certificates in the Context of Transport Layer Security (TLS) <http://tools.ietf.org/html/rfc6125>.

[10] PKIX Over Secure HTTP (POSH) <http://datatracker.ietf.org/doc/draft-miller-posh/>.

Domain Pair The combination of the Sender Domain and Target Domain.

Initiating Server The machine that wants to send a message from an entity at the Sender Domain to an entity at the Target Domain (and thus the machine that is attempting to establish a domain name association between (a) the Target Domain and (b) the XML stream from the Initiating Server to the Receiving Server). Note well that in older documentation of the Server Dialback protocol, this was called the Originating Server.

Receiving Server The machine to which the Initiating Server has opened a connection for the purpose of sending a message from the Sender Domain to the Target Domain (and thus the machine that is trying to verify that the Initiating Server represents the Sender Domain).

Sender Domain The domain name asserted by the Initiating Server as the domainpart of the XMPP 'from' address of stanzas that will flow over the XML stream from the Initiating Server to the Receiving Server.

Target Domain The domain name specified by the Initiating Server as the domainpart of the XMPP 'to' address of stanzas that will flow over the XML stream from the Initiating Server to the Receiving Server.

1.4 How Dialback Works

Server Dialback is used when a stanza that is to be sent from a Sender Domain must be routed to a Target Domain and there is not yet an established connection between the domains. The basic flow of events in Server Dialback consists of the following four steps:

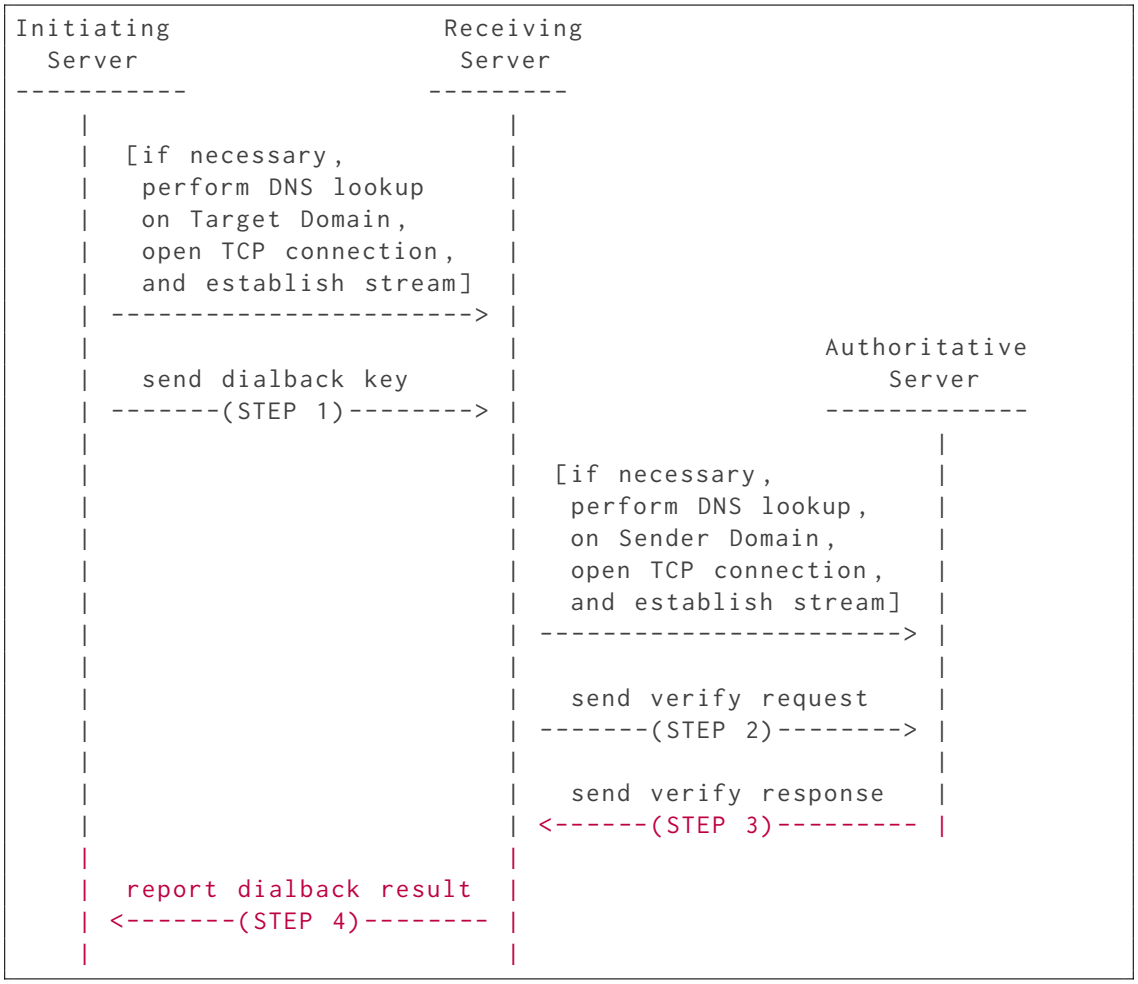
1. The Initiating Server generates a dialback key and sends that value over its XML stream with the Receiving Server. (If the Initiating Server does not yet have an XML stream to the Receiving Server, it will first need to perform a DNS lookup on the Target Domain and thus discover the Receiving Server, open a TCP connection to the discovered IP address and port, and establish an XML stream with the Receiving Server.)

2. Instead of immediately accepting XML stanzas on the connection from the Initiating Server, the Receiving Server sends the same dialback key over its XML stream with the Authoritative Server for verification. (If the Receiving Server does not yet have an XML stream to the Authoritative Server, it will first need to perform a DNS lookup on the Sender Domain and thus discover the Authoritative Server, open a TCP connection to the discovered IP address and port, and establish an XML stream with the Authoritative Server.)

3. The Authoritative Server informs the Receiving Server whether the key is valid or invalid.

4. The Receiving Server informs the Initiating Server whether its identity has been verified or not.

After Step 4, the Initiating Server is authorized to send stanzas from the Sender Domain to the Target Domain as communicated in the 'to' and 'from' attributes of the dialback negotiation. In addition to identity verification of the Sender Domain, this also ensures that the Receiving Server is accepting stanzas for the Target Domain.
We can represent the flow of events graphically as follows.



2 Protocol

This section describes the protocol in detail.
Assumptions used in the examples:

- The server hosting ”capulet.example” is acting as the Initiating Server in sections 2.1.1 and 2.2.1 and as the Receiving Server in sections 2.1.2 and 2.2.2. A DNS SRV lookup on ”capulet.example” resolves to ”orchard.capulet.example”.
- The server hosting ”montague.example” is acting as Receiving Server in sections 2.1.1 and 2.2.2 and as Authoritative Server in sections 2.1.2 and 2.2.2. A DNS SRV lookup on ”montague.example” resolves to the machine ”home.montague.example”
- The stream ID of the response stream header sent from ”capulet.example” to ”montague.example” is ”D60000229F”.
- The stream ID of the response stream header sent from ”montague.example” to ”capulet.example” is ”417GAF25”.
- The shared secret within the ”capulet.example” domain is ”s3cr3tf0rd14lb4ck”.
- The shared secret within the ”montague.example” domain is ”d14lb4ck43v3r”.

Note: All XML elements qualified by the Server Dialback namespace MUST be prefixed with the namespace prefix for the 'jabber:server:dialback' namespace as advertised on the stream header originally sent by the entity sending the element. [11]
This section can be read in two ways:

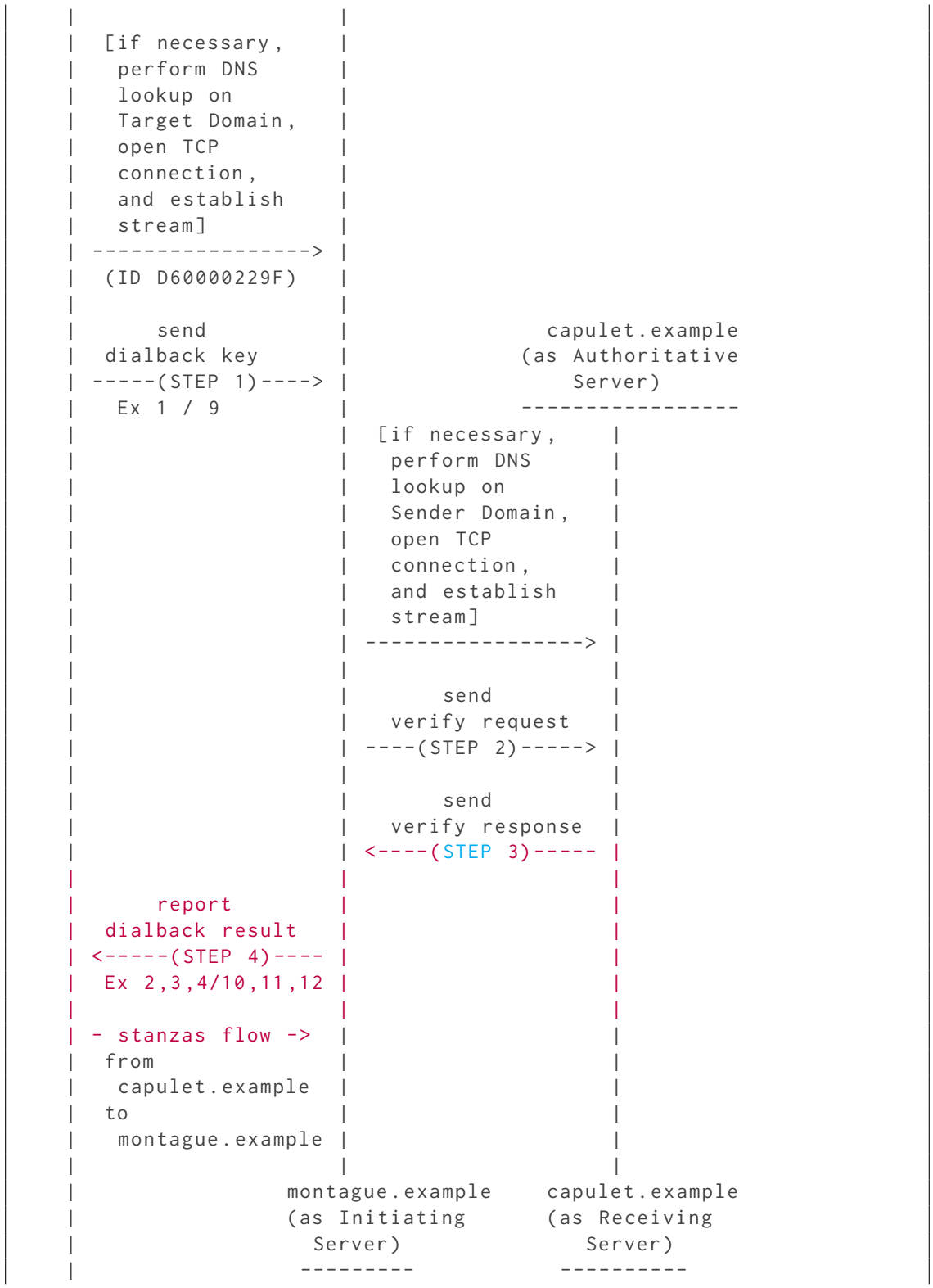
1. To understand the overall protocol flow of each dialback negotiation, read Section 2.1.1 and Section 2.2.1 (aspects of the dialback negotiation from capulet.example as Initiating Server to montague.example as Receiving Server), then Section 2.1.2 and 2.2.2 (aspects of the dialback negotiation from montague.example as Initiating Server to capulet.example as Receiving Server).

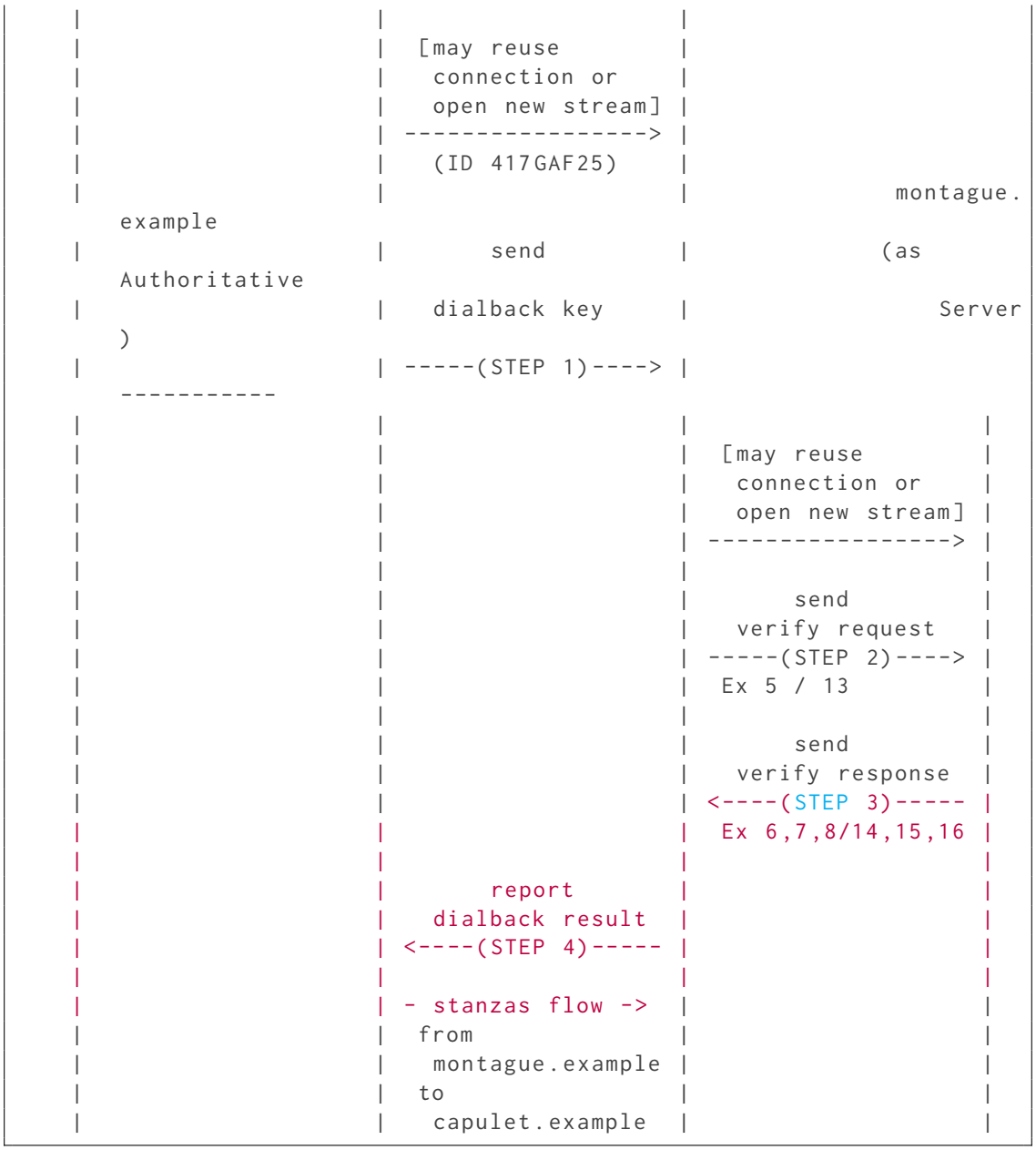
2. To implement the code for either an outbound connection or an inbound connection, read Section 2.1 (outbound) or Section 2.2 (inbound). Note that both parts can be implemented, tested, and used separately.

The following figure gives an overview of where each example is embedded in the process and illustrates the changing roles of each server.

```
orchard.capulet.      home.montague.
example               example
(as Initiating)       (as Receiving
   Server)               Server)
----------------      -------------
```

[11] RFC 3920 stipulated that ”an implementation SHOULD generate only the 'db:' prefix for such elements and MAY accept only the 'db:' prefix.” This restriction was included for the sake of backward compatibility with the jabberd 1.x codebase.


[if necessary, perform DNS lookup on Target Domain, open TCP connection, and establish stream]
(ID D60000229F)
send dialback key
-----(STEP 1)---->
Ex 1 / 9
capulet.example
(as Authoritative Server)
[if necessary, perform DNS lookup on Sender Domain, open TCP connection, and establish stream]
send verify request
----(STEP 2)----->
send verify response
<----(STEP 3)-----
report dialback result
<-----(STEP 4)----
Ex 2,3,4/10,11,12
- stanzas flow ->
from capulet.example
to montague.example
montague.example
(as Initiating Server)
capulet.example
(as Receiving Server)



2.1 Outbound Connection

On an outbound connection there are two different tasks:

1. Request authorization to send stanzas from a Sender Domain to a Target Domain, i.e., act as an Initiating Server in relation to a Receiving Server; this is is described under Section 2.1.1.

2. Generate verification requests about the validity of dialback keys, i.e., act as a Receiving Server in relation to an Authoritative Server; this is described under Section 2.1.2.

2.1.1 Initiating Server Generates Outbound Request for Authorization by Receiving Server

This subsection describes the interaction between the server hosting capulet.example (acting as an Initiating Server) and the server hosting montague.example (acting as a Receiving Server), from the outbound perspective of the Initiating Server.
When the Initiating Server has stanzas to send from the Sender Domain to the Target Domain, does not have a verified connection, is currently not attempting to get a verified connection for this domain pair, it sends a new dialback key to the Receiving Server.
To do so, either it can reuse an existing XML stream or it needs to establish a new connection. To establish a new connection, the Initiating Server performs a DNS lookup on the Target Domain, thus finding the IP address and port for server-to-server communication at an authoritative machine for the Target Domain (here that is "home.montague.example").
After the XML stream is established from the Initiating Server to the Receiving Server, the Initiating Server sends a dialback key to the Receiving Server. This is done by creating a <db:result/> element whose XML character data is the dialback key; the element MUST possess a 'from' attribute whose value is the Sender Domain and MUST possess a 'to' attribute whose value is the Target Domain.

Listing 1: Initiating Server Sends Dialback Key (Step 1)

```
send: <db:result
          from='capulet.example'
          to='montague.example'>
        b4835385f37fe2895af6c196b59097b16862406db80559900d96bf6fa7d23df3

      </db:result>
```

The key sent is generated as described in Dialback Key Generation and Validation (XEP-0185) [12]:

```
key = HMAC-SHA256(
        SHA256('s3cr3tf0rd14lb4ck'),
          { 'montague.example', '␣', 'capulet.example', '␣', '
              D60000229F' }
      )
    = b4835385f37fe2895af6c196b59097b16862406db80559900d96bf6fa7d23df3
```

Note: The Receiving Server MAY use any method to determine the validity of the dialback key and the identity of the Initiating Server. The Initiating Server MUST NOT make any assumptions about how the Receiving Server verifies the key, including even the assumption that the key is actively verified by the Receiving Server through communication with the

[12] XEP-0185: Dialback Key Generation and Validation <https://xmpp.org/extensions/xep-0185.html>.

Authoritative Server.
After sending the dialback key, the Initiating Server waits for the verification result from the Receiving Server. If the Initiating Server wishes to send any stanzas for this domain pair, it MUST queue them for sending after it has received authorization to send stanzas from the Receiving Server, and MUST NOT attempt to send stanzas until it has received such authorization.
Note: While waiting for the verification result, the Initiating Server SHOULD continue to send stanzas for any domain pair that has already been verified on that connection. It MAY send out additional dialback keys for different domain pairs and issue dialback verification requests as described under Section 2.1.2. To avoid denial of service attacks (RFC 4732 [13]) or deadlock situations, the Initiating Server MAY impose a timeout on dialback operations, i.e. it ought to consider dialback operations as having failed when there is no response after a certain amount of time.
If the stream or the underlying TCP connection is closed by the Receiving Server while the Initiating Server is waiting for the verification result, the Initiating Server shall behave as it does when receiving a dialback error as described below.
After the Receiving Server has verified the request, the Initiating Server receives the verification result in the form of a <db:result/> element with a the 'type' attribute whose value is "valid" or "invalid" (for the value of "error", see below). The Initiating Server MUST ensure that the 'from' and 'to' attributes in this response correlate to a request that was sent over the same XML stream (see Section 3.1).
Thus the result is either valid...

Listing 2: Initiating Server Receives Valid Verification Result from Receiving Server (Step 4)

```
recv: <db:result
          from='montague.example'
          to='capulet.example'
          type='valid'/>
```

... or invalid ...

Listing 3: Initiating Server Receives Invalid Verification Result from Receiving Server (Step 4)

```
recv: <db:result
          from='montague.example'
          to='capulet.example'
          type='invalid'/>
```

Note: There are no examples for Step 2 and Step 3 in this section of the document; see the examples under Sections 2.1.2 and 2.2.2.
If the value of the 'type' attribute is "valid", then the connection between the domain pair is considered verified and the Initiating Server can send any outbound stanzas it has queued up for routing to the Receiving Server for the domain pair (i.e., from the Sender Domain to the

[13] RFC 4732: Internet Denial-of-Service Considerations <http://tools.ietf.org/html/rfc4732>.

Target Domain). Naturally, the Initiating Server can also enable or negotiate other stream features at this point.
If the value of the 'type' attribute is "invalid", then the Receiving Server is reporting that that Initiating Server's identity (as valid for the Sender Domain) was deemed bogus by the Authoritative Server. In this case, the Initiating Server MUST NOT attempt to send any outbound stanzas it has queued up for routing to the Receiving Server for the domain pair but instead MUST return such stanzas to the respective senders at the Sender Domain with an <internal-server-error/> stanza error. Since the Receiving Server will most likely close the stream and the underlying TCP connection if that occurs (see Section 2.2.1), the Initiating Server SHOULD NOT attempt to send further stanzas for other domain pairs that have already been authorized.
If the value of the 'type' attribute is "error", this indicates a problem which is not related to the validity of the dialback key provided. The error conditions are explained in detail under Dialback with Error Handling. Such an error is non-fatal for the XML stream, but the Initiating Server MUST return any queued stanzas to the respective senders at the Sender Domain with a <remote-server-timeout/> stanza error.

Listing 4: Initiating Server Receives Dialback Error from Receiving Server (Step 4)

```
recv: <db:result
          from='montague.example'
          to='capulet.example'
          type='error'>
        <error type='cancel'>
          <item-not-found xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'/
              >
        </error>
      </db:result>
```

2.1.2 Receiving Server Generates Outbound Request for Verification of Initiating Server by Authoritative Server

This subsection describes the interaction between the server hosting capulet.example (acting as a Receiving Server) and the server hosting montague.example (acting as an Authoritative Server), from the outbound perspective of the Receiving Server.
To determine the validity of a dialback key received from the Initiating Server, the Receiving Server needs to establish communications with the Authoritative Server. To do so, it can reuse an existing XML stream or establish a new connection. To establish a new connection, the Receiving Server performs a DNS lookup on the Sender Domain, thus finding the IP address and port for server-to-server communication at an authoritative machine for the Sender Domain asserted by the Initiating Server (here the machine is "orchard.capulet.example").
After the XML stream is established from the Receiving Server to the Authoritative Server, the Receiving Server sends a verification request. This is done by creating a <db:verify/> element whose XML character data is the dialback key received from the Initiating Server; the element MUST possess a 'from' attribute whose value is the Target Domain, MUST possess

a 'to' attribute whose value is the Sender Domain as provided in the 'from' attribute of Step 1, and MUST possess an 'id' attribute whose value is the stream ID of the response stream header sent from the Receiving Server to the Initiating Server (here "417GAF25"). The combination of 'from', 'to', and 'id' attributes makes it possible for the Receiving Server to uniquely identify the TCP connection on which it received the original request in Step 1.
Note: An implementation MAY open a separate connection to the Authoritative Server for the sole purpose of doing key verification. Such an implementation SHOULD close the connection immediately after receiving the verification result. Not using TLS or any other stream features can reduce the number of round trips in that case.

Listing 5: Receiving Server Sends Verification Request to Authoritative Server (Step 2)

```
send: <db:verify
          from='capulet.example'
          id='417GAF25'
          to='montague.example'>
       225
          cc5aa6a071133249d25fef42ae516fc7a86c523aa1c6980a7f73e784c972d

     </db:verify>
```

After that, the Receiving Server waits for the verification result. While doing so, it can still use the connection to send dialback packets or to send stanzas for domain pairs that have already been validated.
Here again, the result is either valid...

Listing 6: Receiving Server is Informed by Authoritative Server that Key is Valid (Step 3)

```
recv: <db:verify
          from='montague.example'
          id='417GAF25'
          to='capulet.example'
          type='valid'/>
```

... or invalid ...

Listing 7: Receiving Server is Informed by Authoritative Server that Key is Invalid (Step 3)

```
recv: <db:verify
          from='montague.example'
          id='417GAF25'
          to='capulet.example'
          type='invalid'/>
```

In addition to the values "valid" and "invalid", the 'type' attribute can also have a value of "error"; see Dialback with Error Handling for a detailed explanation.

Listing 8: Receiving Server Receives Dialback Error from Authoritative Server (Step 3)

```
recv: <db:verify
          from='montague.example'
          id='417GAF25'
          to='capulet.example'
          type='error'>
        <error type='cancel'>
          <item-not-found xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'/
              >
        </error>
      </db:verify>
```

Note: If the underlying TCP connection is closed by the remote side while there are pending verification requests, those requests SHOULD be considered failed and therefore be treated like an error response.
The Receiving Server MUST ensure that the values of the 'from', 'to' and 'id attributes correlate to a request that was sent over the same XML stream (see Section 3.1).
After receiving the validation result from the Authoritative Server, the Receiving Server determines the inbound connection that the dialback key was originally received on. This connection is uniquely identified by the combination of the 'from', 'to', and 'id' attributes. If no inbound connection is found that matches this combination, the verification result SHOULD be dropped silently. If an inbound connection is found, the Receiving Server uses it to communicate the verification result to the Initiating Server. A positive result indicates the readiness of the Receiving Server to accept stanzas from the Initiating Server for this domain pair.
When receiving a verification result of type "invalid", the Receiving Server MAY choose not to relay this result to the Initiating Server. Instead, it might send a dialback error such as <forbidden/> to the Initiating Server. Compared to sending a result of type "invalid", this behavior will not result in the loss of the whole stream and any previously domain pairs previously negotiated, while at the same time not accepting stanzas from the spoofed domain. Even when not forwarding the "invalid" result, the incident ought to be logged.

2.2 Inbound Connection

There are two different tasks on an inbound connection:

1. Authorize inbound connections, i.e., act as a Receiving Server in relation to an Initiating Server; this is described under Section 2.2.1.

2. Answer verification requests about the validity of dialback keys, i.e., act as an Authoritative Server in relation to a Receiving Server; this is described under Section 2.2.2.

Note that, unless XEP-0288 is used, the 'type' should not be set on either <db:result/> or <db:verify/> elements received on an inbound connection.

2.2.1 Receiving Server Handles Inbound Authorization Request from Initiating Server

This subsection describes the interaction between the server hosting capulet.example (acting as an Initiating Server) and the server hosting montague.net (acting as a Receiving Server), from the inbound perspective of the Receiving Server (i.e., this section is the mirror image of Section 2.1.1).

Listing 9: Receiving Server Receives Dialback Key from Initiating Server (Step 1)

```
recv: <db:result
          from='capulet.example'
          to='montague.example'>
        b4835385f37fe2895af6c196b59097b16862406db80559900d96bf6fa7d23df3

      </db:result>
```

Before the Receiving Server allows the Initiating Server to send stanzas from the Sender Domain (here "capulet.example") to the Target Domain (here "montague.example"), it MUST verify the identity of the Initiating Server. Depending on how the server dialback protocol is used, this can be done by verifying the dialback key or by using some out-of-band method as in the POSH prooftype for XMPP domain name associations. Note that the verification process might fail prematurely, for example, if the Receiving Server's policy states that connections from the Initiating Server or the Sender Domain are not allowed.
Note: The Receiving Server MUST continue to accept and process stanzas for already verified domain pairs, and MUST continue to process both <db:result/> and <db:verify/> elements.
If the Target Domain as given in the 'to' attribute of the element is not a configured domain of the Receiving Server, this results in a dialback error. This error, which is explained further under Section 2.4.2, is not a stream error and therefore MUST NOT result in closing of the stream as described in Section 4.4 of RFC 6120, since the stream might already be used to exchange XML stanzas for other domain pairs.

Listing 10: Receiving Server Sends Dialback Error to Initiating Server (Step 4)

```
send: <db:result
          from='montague.example'
          to='capulet.example'
          type='error'>
        <error type='cancel'>
          <item-not-found xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'/
              >
        </error>
      </db:result>
```

After the validity of the dialback request has been established (for example, by the Authoritative Server), the Receiving Server can safely accept stanzas from the Initiating Server for the verified domain pair.
In addition, the Receiving Server SHALL notify the Initiating Server of the result and thus

signal its willingness to accept stanzas from the Initiating Server for the verified domain pair. This is done by creating a <db:result/> element which MUST possess a 'from' attribute whose value is the Target Domain, MUST possess a 'to' attribute whose value is the Sender Domain, and MUST possess a 'type' attribute whose value is either "valid" or "invalid" (or "error", as shown above).
Therefore, here again the result is either valid (this is the same as Example 2)...

Listing 11: Receiving Server Sends Valid Verification Result to Initiating Server (Step 4)

```
send: <db:result
          from='montague.example'
          to='capulet.example'
          type='valid'/>
```

... or invalid (this is the same as Example 3)...

Listing 12: Receiving Server Sends Invalid Verification Result to Initiating Server (Step 4)

```
send: <db:result
          from='montague.example'
          to='capulet.example'
          type='invalid'/>
```

If the result is "invalid", the Initiating Server is either attempting to spoof the Sender Domain or is misconfigured. The Receiving Server MUST NOT accept stanzas from the Initiating Server for the Sender Domain and ought to log the attempt. If no other valid domain pairs exist for this connection (i.e., if this is the first attempt), the Receiving Server SHOULD send a result with type "invalid" and MUST close the XML stream. If there exist other valid domain pairs for this connection, the Initiating Server might merely have a misconfiguration for the Sender Domain. In this case, the Receiving Server MAY (instead of closing the connection) return an error condition of <forbidden/> as described under Section 2.5 of this document.

2.2.2 Authoritative Server Handles Inbound Verification Request from Receiving Server

This subsection describes the interaction between the server hosting capulet.example (acting as a Receiving Server) and the server hosting montague.example (acting as an Authoritative Server), from the inbound perspective of the Authoritative Server (i.e., this section is the mirror image of Section 2.1.2 and the following example is the same as Example 5).

Listing 13: Authoritative Server Receives Verification Request from Receiving Server (Step 2)

```
recv: <db:verify
          from='capulet.example'
          id='417GAF25'
```

```
          to='montague.example'>
        225
           cc5aa6a071133249d25fef42ae516fc7a86c523aa1c6980a7f73e784c972d

      </db:verify>
```

If the Target Domain as given in the 'to' attribute of the element does not match a configured local domain according to the Authoritative Server, this results in a dialback error. This error, which is explained further under Section 2.4, is not a stream error and therefore MUST NOT result in closing of the stream (as described in Section 4.4 of RFC 6120), since the stream might already be used for sending XML stanzas for other domain pairs. (The following example is the same as Example 8.)

Listing 14: Authoritative Server Sends Dialback Error to Receiving Server (Step 3)

```
send: <db:verify
          from='montague.example'
          id='417GAF25'
          to='capulet.example'
          type='error'>
        <error type='cancel'>
          <item-not-found xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'/
              >
        </error>
      </db:verify>
```

Upon receiving this <db:verify/> element, the Authoritative Server determines the validity of the dialback key provided in the XML character data of the element. This can be achieved for example by comparing the character data with the output of applying the same key generation mechanism that was (presumably) used for the generation of the key, using as input the values of the 'from', 'to', and 'id' attributes contained in the verification request and the secret known only to the Sender Domain:

```
key = HMAC-SHA256(
        SHA256('d14lb4ck43v3r'),
          { 'capulet.example', '␣', 'montague.example', '␣', '417GAF25
              ' }
      )
    = 225cc5aa6a071133249d25fef42ae516fc7a86c523aa1c6980a7f73e784c972d
```

The Authoritative Server then notifies the Receiving Server whether the key is valid. This is done by creating a <db:verify/> element which MUST possess 'from' and 'to' attributes whose values are swapped from the request, MUST possess an 'id' attribute whose value is copied from the 'id' value of the request, and MUST possess a 'type' attribute whose value is either "valid" or "invalid".
Therefore, here again the result is either valid (this is the same as Example 6)...

Listing 15: Authoritative Server Informs Receiving Server that Key is Valid (Step 3)

```
send: <db:verify
          from='montague.example'
          id='417GAF25'
          to='capulet.example'
          type='valid'/>
```

... or invalid (this is the same as Example 7)...

Listing 16: Authoritative Server Informs Receiving Server that Key is Invalid (Step 3)

```
send: <db:verify
          from='montague.example'
          id='417GAF25'
          to='capulet.example'
          type='invalid'/>
```

There are several reasons why the key might be invalid (e.g., the Authoritative Server has a different secret key or the Authoritative Server doesn't know anything about the StreamID communicated in the <db:result/> element it received from the Receiving Server).

2.3 Directionality

The result of the protocol exchanges shown in the foregoing two sections is that the server hosting montague.example has verified the identity of the server hosting capulet.example. Since XMPP Server-to-Server connections are unidirectional (unless XEP-0288 is used), dialback needs to be completed in each direction before XML stanzas can be exchanged over the two TCP connections between the servers.

2.4 Advertisement

2.4.1 Traditional Dialback

Support for the traditional server dialback protocol (originally specified in RFC 3920) is indicated by inclusion of the dialback namespace declaration in the stream header.

Listing 17: Stream Header With Dialback Namespace Declaration

```
<stream:stream
    xmlns='jabber:server'
    xmlns:db='jabber:server:dialback'
    xmlns:stream='http://etherx.jabber.org/streams'
    from='montague.example'
    to='capulet.example'>
```

Note: Although in general advertising protocol support by means of an XML namespace declaration has been superseded by the use of stream features as originally defined in RFC 3920 and updated in RFC 6120, the server dialback protocol predates the existence of stream features and therefore the namespace declaration method is still used in this instance.
Note: It is conventional to use a namespace prefix of ”db” for Server Dialback elements. Although the prefix is allowed to be other than ”db” according to the XML namespaces specification (Namespaces in XML [14]), some existing implementations and deployments might accept only the ”db” prefix.

2.4.2 Dialback with Error Handling

If a server supports graceful handling of dialback errors as described in this document, it MUST advertise that via a stream feature which is a <dialback/> element qualified by the 'urn:xmpp:features:dialback' namespace, including an empty <errors/> element.

Listing 18: Stream Features With <errors/> Element

```
<stream:features>
  <dialback xmlns='urn:xmpp:features:dialback'>
    <errors/>
  </dialback>
</stream:features>
```

Note: As a general rule, stream feature elements containing child elements that advertise particular sub-features are not encouraged. The format shown above is used for the sake of backward compatiblity with existing implementations and deployments.

2.5 Dialback Error Conditions

RFC 3920 introduced stream errors for any errors related to dialback. However, this turned out to be overly aggressive, particularly if the XML stream was used to multiplex stanzas for more than one domain pair (since closing the stream would result in throwing away accumulated dialback state for a potentially large number of domain pairs). Therefore this specification defines a third value for the 'type' attribute: ”error”.
This usage of the 'error' value for the 'type' attribute is not fully backward compatible with RFC 3920. However, the server that generates the error SHOULD still attempt to send the dialback error instead of terminating the stream, as the worst thing that can happen is that the remote server terminates the stream if it does not understand the error or if it eventually times out the connection. Dialback errors are to be considered non-fatal for the XML stream, but the Initiating Server MUST return queued stanzas to the respective senders with a <remote-server-timeout/> stanza error. If an error is encountered in Step 3 of the dialback negotiation, the Receiving Server MUST send a <remote-server-not-found/> dialback error to

[14] Namespaces in XML <http://www.w3.org/TR/REC-xml-names/>.

the Initiating Server.
When the <db:verify/> or <db:result/> element is of type ”error”, the element MUST contain an <error/> element (implicitly qualified by the 'jabber:server' namespace), which MUST in turn contain an XML element qualified by the 'urn:ietf:params:xml:ns:xmpp-stanzas' namespace (i.e., a stanza error condition) as those errors are defined in RFC 6120. The following table provides additional guidance regarding the most relevant stanza error conditions:

Condition	Description	Occurs in
<item-not-found/>	The domain given in the 'to' attribute of the request is not hosted on the Receiving Server. Nonetheless, the domain is used in the 'from' attribute of the error packet, for the purpose of identifying the original request.	Step 3 or 4
<remote-connection-failed/>	The Receiving Server was unable to establish a connection to the Authoritative Server and therefore could not validate the identity of the Initiating Server.	Step 4
<remote-server-not-found/>	The Receiving Server encountered an <item-not-found/> error condition or a <host-unknown/> stream error when attempting to contact the Authoritative Server.	Step 4
<remote-server-timeout/>	The Receiving Server encountered a problem with the connection to the Authoritative Server, for example if the Authoritative Server unexpectedly closed the stream without verifying the dialback key.	Step 4
<policy-violation/>	The Receiving Server enforces a policy mandating usage of TLS before dialback and the Initiating Server sent the dialback request without using TLS.	Step 3 or 4

Condition	Description	Occurs in
<not-authorized/>	The Receiving Server enforces a policy requiring either a valid PKIX certificate containing the identity of the Sender Domain or some other proof of authorization (e.g., via POSH), and the Initiating Server did not provide proof of authorization.	Step 3
<forbidden/>	The Receiving Server received an "invalid" response when attempting to verify the dialback key with the Authoritative Server.	Step 4
<not-acceptable/>	The Receiving Server was unable to establish the asserted identity of the Initiating Server.	Step 4
<resource-constraint/>	The Receiving Server lacks the resources to add the requested domain pair to the list of connections authorized for this connection. The initiating server should attempt to establish a new TCP connection to the target domain using the process described in RFC 6120 and give up when receiving the same error on the new connection.	Step 4

2.6 Multiplexing

A single XML stream between Initiating Server and Receiving Server can be used to multiplex stanzas for more than one domain pair. We call this usage "multiplexing" (historically it has also been known as "piggybacking").
One common motivation for multiplexing is virtual hosting, under which many domains are hosted on the same server. This problem is described more fully in the Domain Name Associations specification, draft-ietf-xmpp-dna).
Another common motivation for such reuse is the existence of additional services associated with the Sender Domain but hosted at "subdomains" thereof. For example, both the

”montague.example” and the ”capulet.example” XMPP servers might host Multi-User Chat (XEP-0045) [15] services at ”chat.montague.example” and ”rooms.capulet.example” respectively. Because dialback operates on domain pairs, a total of eight dialback negotiations would be necessary for a bidirectional exchange of stanzas between two sending domains and two target domains.
If multiplexing is not used, the number of server-to-server connections needed to exchange stanzas between virtual hosting providers or multi-service XMPP servers can increase signficantly. Indeed, when the number of hosted domains becomes especially large, the number of connections might exceed the maximum number of connections allowed from a single IP address as explained in Best Practices to Discourage Denial of Service Attacks (XEP-0205) [16].
If multiplexing is used, the number of connections can be limited to only two (or, at the operator’s discretion, more than two for operational reasons).

2.6.1 Multiplexing Sender Domains

In order to accept XML stanzas from rooms at ”rooms.capulet.example” intended for addresses at ”montague.example”, the ”montague.example” domain will need to validate the ”rooms.capulet.example” domain (just as it already did for the ”capulet.example” domain). Thus the server hosting both capulet.example and rooms.capulet.example would now initiate a dialback negotiation with the server hosting montague.example but specify the Sender Domain as ”rooms.capulet.example”. Specifying different Sender Domains is called ”sender multiplexing” and MAY be used without further negotation.

2.6.2 Multiplexing Target Domains

Likewise, to send stanzas to rooms at ”chat.montague.example” from addresses at ”capulet.example”, the server hosting both capulet.example and rooms.capulet.example would initiate dialback negotiation with the server hosting chat.montague.example (probably on the same connection that is already used to send stanzas from ”capulet.example” to ”montague.example”), specifying the Target Domain as ”chat.montague.example”. Specifying different target domains is called ”target multiplexing”.
The Initiating Server SHOULD NOT use target multiplexing unless the Receiving Server has signalled support for dialback error handling via <stream:features/> as described under Dialback with Error Handling. The Initiating Server MAY then attempt to multiplex a new Sender Domain on the stream to the Receiving Server that is already used for another Sender Domain if the hostname and port resolution results in the same IP address and port combination. For example:

[15] XEP-0045: Multi-User Chat <https://xmpp.org/extensions/xep-0045.html>.
[16] XEP-0205: Best Practices to Discourage Denial of Service Attacks <https://xmpp.org/extensions/xep-0205.html>.

Listing 19: DNS SRV Record for the montague.example Zone

```
_xmpp-server._tcp.montague.example.       86400 IN SRV 10 0 5269 home.
    montague.example
_xmpp-server._tcp.chat.montague.example. 86400 IN SRV 10 0 5269 home.
    montague.example
home.montague.example.                    86400 IN A
                         10.44.0.4
```

Because DNS SRV lookups for both ”montague.example” and ”chat.montague.example” point to the same target (”home.montague.example”) and port (5269), or eventually resolve to the same IP address (10.44.0.4) and port (5269), ”capulet.example” MAY initiate a dialback negotation from ”capulet.example” to ”chat.montague.example” over the same XML stream that is already used to send stanzas from ”capulet.example” to ”montague.example”.
Bidirectional Server-to-Server Connections (XEP-0288) [17] extends those rules since any domain that has been used as a source domain can be used as a target domain without further negotiation.

3 Security Considerations

Server Dialback helps protect against domain spoofing, thus making it difficult to spoof the origin of XML stanzas. Absent the use of DNS security (DNSSEC, RFC 4033), Server Dialback does not provide a mechanism for authenticating a stream, as is done via TLS and SASL, and results in weak verification of server identities only. Furthermore, if DNSSEC is not used then it is susceptible to DNS poisoning attacks.
If DNSSEC is used, Server Dialback provides stream authentication only (i.e., a strong association between a domain name and an XML stream). However, Server Dialback by itself does not provide confidentiality, data integrity, or stream encryption. Some existing implementations are known to support dialback over TLS. In this case, Server Dialback is typically carried out the same way as without TLS, but gains from the use of channel encryption.

3.1 Unsolicited Dialback

In mid-2012, several implementations turned out to be vulnerable to a number of attacks against the protocol described in this document. These attacks were based on sending verify (STEP 3) or result (STEP 4) responses without an associated request, in an unexpected direction and/or on an unexpected XML stream. Failure to reject those ’unsolicited’ responses lets an attacker either spoof stanzas with an arbitrary sender domain or enables him to impersonate any target domain.

[17] XEP-0288: Bidirectional Server-to-Server Connections <https://xmpp.org/extensions/xep-0288.html>.

4 IANA Considerations

This document requires no interaction with the Internet Assigned Numbers Authority (IANA) [18].

5 XMPP Registrar Considerations

5.1 Protocol Namespaces

The XMPP Registrar [19] includes 'jabber:server:dialback' in its registry of protocol namespaces (see <https://xmpp.org/registrar/namespaces.html>).

5.2 Stream Features

The XMPP Registrar includes 'urn:xmpp:features:dialback' in its registry of stream features (see <https://xmpp.org/registrar/stream-features.html>).
The registration is as follows:

```
<feature>
  <ns>urn:xmpp:features:dialback</ns>
  <name>Server Dialback</name>
  <element>dialback</element>
  <desc>Support for Server Dialback with dialback errors</desc>
  <doc>XEP-0220</doc>
</feature>
```

6 XML Schema

6.1 Dialback

Note Well: the 'error' value for the 'type' attribute and the <error/> child element were added since RFC 3920.

```
<?xml version='1.0' encoding='UTF-8'?>

<xs:schema
```

[18] The Internet Assigned Numbers Authority (IANA) is the central coordinator for the assignment of unique parameter values for Internet protocols, such as port numbers and URI schemes. For further information, see <http://www.iana.org/>.

[19] The XMPP Registrar maintains a list of reserved protocol namespaces as well as registries of parameters used in the context of XMPP extension protocols approved by the XMPP Standards Foundation. For further information, see <https://xmpp.org/registrar/>.

```
  xmlns:xs='http://www.w3.org/2001/XMLSchema'
  targetNamespace='jabber:server:dialback'
  xmlns='jabber:server:dialback'
  elementFormDefault='qualified'>

<xs:annotation>
  <xs:documentation>
    The protocol documented by this schema is defined in
    XEP-0220: http://www.xmpp.org/extensions/xep-0220.html
  </xs:documentation>
</xs:annotation>

<xs:import namespace='urn:ietf:params:xml:ns:xmpp-stanzas'
           schemaLocation='http://xmpp.org/schemas/stanzaerror.xsd'/
              >

<xs:element name='result'>
  <xs:complexType mixed='true'>
    <xs:all>
      <xs:element name='error' type='errorType'/>
    </xs:all>
    <xs:attribute name='from' type='xs:string' use='required'/>
    <xs:attribute name='to' type='xs:string' use='required'/>
    <xs:attribute name='type' use='optional'>
      <xs:simpleType>
        <xs:restriction base='xs:NCName'>
          <xs:enumeration value='error'/>
          <xs:enumeration value='invalid'/>
          <xs:enumeration value='valid'/>
        </xs:restriction>
      </xs:simpleType>
    </xs:attribute>
  </xs:complexType>
</xs:element>

<xs:element name='verify'>
  <xs:complexType mixed='true'>
    <xs:all>
      <xs:element name='error' type='errorType'/>
    </xs:all>
    <xs:attribute name='from' type='xs:string' use='required'/>
    <xs:attribute name='to' type='xs:string' use='required'/>
    <xs:attribute name='id' type='xs:NMTOKEN' use='required'/>
    <xs:attribute name='type' use='optional'>
      <xs:simpleType>
        <xs:restriction base='xs:NCName'>
          <xs:enumeration value='error'/>
          <xs:enumeration value='invalid'/>
          <xs:enumeration value='valid'/>
```

```
          </xs:restriction>
        </xs:simpleType>
      </xs:attribute>
    </xs:complexType>
  </xs:element>

  <xs:complexType name='errorType'>
    <xs:sequence xmlns:err='urn:ietf:params:xml:ns:xmpp-stanzas'>
      <xs:group ref='err:stanzaErrorGroup'/>
    </xs:sequence>
  </xs:complexType>

</xs:schema>
```

6.2 Stream Feature

```
<?xml version='1.0' encoding='UTF-8'?>

<xs:schema
    xmlns:xs='http://www.w3.org/2001/XMLSchema'
    targetNamespace='urn:xmpp:features:dialback'
    xmlns='urn:xmpp:features:dialback'
    elementFormDefault='qualified'>

  <xs:element name='dialback'>
    <xs:complexType>
      <xs:sequence>
        <xs:element name='errors' minOccurs='0' type='empty'/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <xs:simpleType name='empty'>
    <xs:restriction base='xs:string'>
      <xs:enumeration value=''/>
    </xs:restriction>
  </xs:simpleType>

</xs:schema>
```

6.3 Application Specific Errors

```
<?xml version='1.0' encoding='UTF-8'?>

<xs:schema
    xmlns:xs='http://www.w3.org/2001/XMLSchema'
    targetNamespace='urn:xmpp:dialback:errors'
```

```
    xmlns='urn:xmpp:dialback:errors'
    elementFormDefault='qualified'>

  <xs:annotation>
    <xs:documentation>
      The protocol documented by this schema is defined in
      XEP-0220: http://www.xmpp.org/extensions/xep-0220.html
    </xs:documentation>
  </xs:annotation>
</xs:schema>
```

7 Acknowledgments

Thanks to Richard Barnes, Dave Cridland, Jack Erwin, Joe Hildebrand, Justin Karneges, Nina Kirchner, Carlo von Loesch, Ralph Meijer, Matthew Miller, Chris Newton, Rob Norris, Tory Patnoe, Dave Richards, Matthew Wild, and Matthias Wimmer for their comments.