



XMPP

XEP-0220: Server Dialback

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

Peter Saint-Andre
<mailto:stpeter@jabber.org>
<xmpp:stpeter@jabber.org>
<https://stpeter.im/>

Philipp Hancke
<xmpp:fippo@psyced.org>

2011-09-19
Version 0.12

Status	Type	Short Name
Experimental	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 originating server, it does not process traffic over the connection until it has verified a key with an authoritative server for the domain asserted by the originating server. Although Server Dialback does not provide strong authentication or trusted federation and although it is subject to DNS poisoning attacks, it has effectively prevented most instances of address spoofing on the XMPP network since its development in the year 2000.

Legal

Copyright

This XMPP Extension Protocol is copyright © 1999 - 2011 by the [XMPP Standards Foundation](#) (XSF).

Permissions

Permission is hereby granted, free of charge, to any person obtaining a copy of this specification (the "Specification"), to make use of the Specification without restriction, including without limitation the rights to implement the Specification in a software program, deploy the Specification in a network service, and copy, modify, merge, publish, translate, distribute, sublicense, or sell copies of the Specification, and to permit persons to whom the Specification is furnished to do so, subject to the condition that the foregoing copyright notice and this permission notice shall be included in all copies or substantial portions of the Specification. Unless separate permission is granted, modified works that are redistributed shall not contain misleading information regarding the authors, title, number, or publisher of the Specification, and shall not claim endorsement of the modified works by the authors, any organization or project to which the authors belong, or the XMPP Standards Foundation.

Warranty

NOTE WELL: This Specification is provided on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE.

Liability

In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall the XMPP Standards Foundation or any author of this Specification be liable for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising from, out of, or in connection with the Specification or the implementation, deployment, or other use of the Specification (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if the XMPP Standards Foundation or such author has been advised of the possibility of such damages.

Conformance

This XMPP Extension Protocol has been contributed in full conformance with the XSF's Intellectual Property Rights Policy (a copy of which can be found at <http://xmpp.org/about-xmpp/xsf/xsf-ipr-policy/> or obtained by writing to XMPP Standards Foundation, 1899 Wynkoop Street, Suite 600, Denver, CO 80202 USA).

Contents

1	Introduction	1
1.1	Why Dialback?	1
1.2	What Dialback Accomplishes	1
1.3	When Dialback Is Used	2
1.4	How Dialback Works	2
2	Protocol	4
2.1	Outbound Connection	5
2.1.1	Originating Server Generates Outbound Request for Authorization by Receiving Server	5
2.1.2	Receiving Server Generates Outbound Request for Verification of Originating Server by Authoritative Server	7
2.2	Inbound Connection	9
2.2.1	Receiving Server Handles Inbound Authorization Request from Originating Server	9
2.2.2	Authoritative Server Handles Inbound Verification Request from Receiving Server	11
2.3	Directionality	13
2.4	Advertisement	13
2.4.1	Traditional Dialback	13
2.4.2	Dialback with Error Handling	13
2.5	Dialback Error Conditions	14
2.6	Multiplexing	16
2.6.1	Multiplexing Sender Domains	16
2.6.2	Multiplexing Target Domains	16
3	Security Considerations	17
4	IANA Considerations	17
5	XMPP Registrar Considerations	18
5.1	Protocol Namespaces	18
5.2	Stream Features	18
6	Acknowledgments	18
7	XML Schema	18
7.1	Dialback	18
7.2	Stream Feature	19

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; accepting a connection from any peer without determining its identity would result in the use of merely asserted identities and a completely uncontrolled approach to federation, which on the open Internet would rapidly devolve into chaos. Clearly such a state of affairs would be unsustainable for a network protocol aiming for widespread deployment.

Such potential chaos was the state of affairs on the Jabber network during the earliest releases of the original [jabberd](#)¹ server codebase (up through the 1.0 release in May 2000). Therefore the Jabber developer community designed a protocol ("Server Dialback") for weak 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.

When the early Jabber protocols were formalized by the XMPP Working Group of the [Internet Engineering Task Force \(IETF\)](#)² in 2002-2004, support for strong identity verification was added. 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 certificate authorities (CAs). However, the Server Dialback protocol is still in wide use, and probably will be for the foreseeable future given the perceived difficulty of obtaining digital certificates issued by common CAs. Therefore it is important to maintain accurate documentation of the Server Dialback protocol. Such documentation was originally provided in [RFC 3920](#)³. Although that documentation was removed from [RFC 6120](#)⁴, it is still provided in this specification for the sake of interoperability.

1.2 What Dialback Accomplishes

Server Dialback is a method for weak identity verification. Such verification depends 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.

Since October 2000, the use of Server Dialback has made it more difficult to spoof the host-

¹The jabberd server is the original server implementation of the Jabber/XMPP protocols, first developed by Jeremie Miller, inventor of Jabber. For further information, see <http://jabberd.org/>.

²The 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/>.

³RFC 3920: Extensible Messaging and Presence Protocol (XMPP): Core <http://tools.ietf.org/html/rfc3920>.

⁴RFC 6120: Extensible Messaging and Presence Protocol (XMPP): Core <http://tools.ietf.org/html/rfc6120>.

names of servers (and therefore the addresses of sent messages) on the XMPP network. However, Server Dialback does not provide authentication between servers and is not a security mechanism. Domains requiring high security are advised to use TLS and SASL with certificates issued by trusted roots.

Server Dialback is unidirectional, and results in weak identity verification for one XML stream in one direction. Because Server Dialback is not an authentication mechanism, mutual authentication is not possible via dialback. Therefore, Server Dialback needs to be completed in each direction in order to enable bidirectional communication between two domains.

Dialback does not verify that the IP address returned by a DNS lookup of the originating domain is the same as the source IP address of the inbound TCP connection. While this might often be true, not performing this check enables large deployments to separate inbound and outbound message routing.

1.3 When Dialback Is Used

Server Dialback is typically used in two scenarios:

1. When a peer service does not support XMPP 1.0 as defined in RFC 3920 or, more generally, does not offer negotiation of TLS.
2. When STARTTLS negotiation succeeds with a peer service but the peer's certificate cannot be used to establish the peer's identity.

Both of these scenarios result in an untrusted connection. However, depending on local security policies, a server might accept such an untrusted connection if the use of Server Dialback results in weak identity verification.

1.4 How Dialback Works

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.

A helpful analogy might be the following telephone scenario:

1. A worker from your electric utility company knocks on your front door and says he needs to enter your house to check your usage meter.
2. Rather than letting him in, you ask for his employee ID number and politely close the door for a few moments.

3. You open the phone book, find the authoritative phone number for the utility company's headquarters, and call them on the phone.
4. After being transferred to the customer service department, you ask if a worker with that particular ID number is authorized to be visiting your house.
5. The company tells you that the worker is authorized, so you thank them and hang up.
6. You then reopen the front door and allow the worker to enter your house.

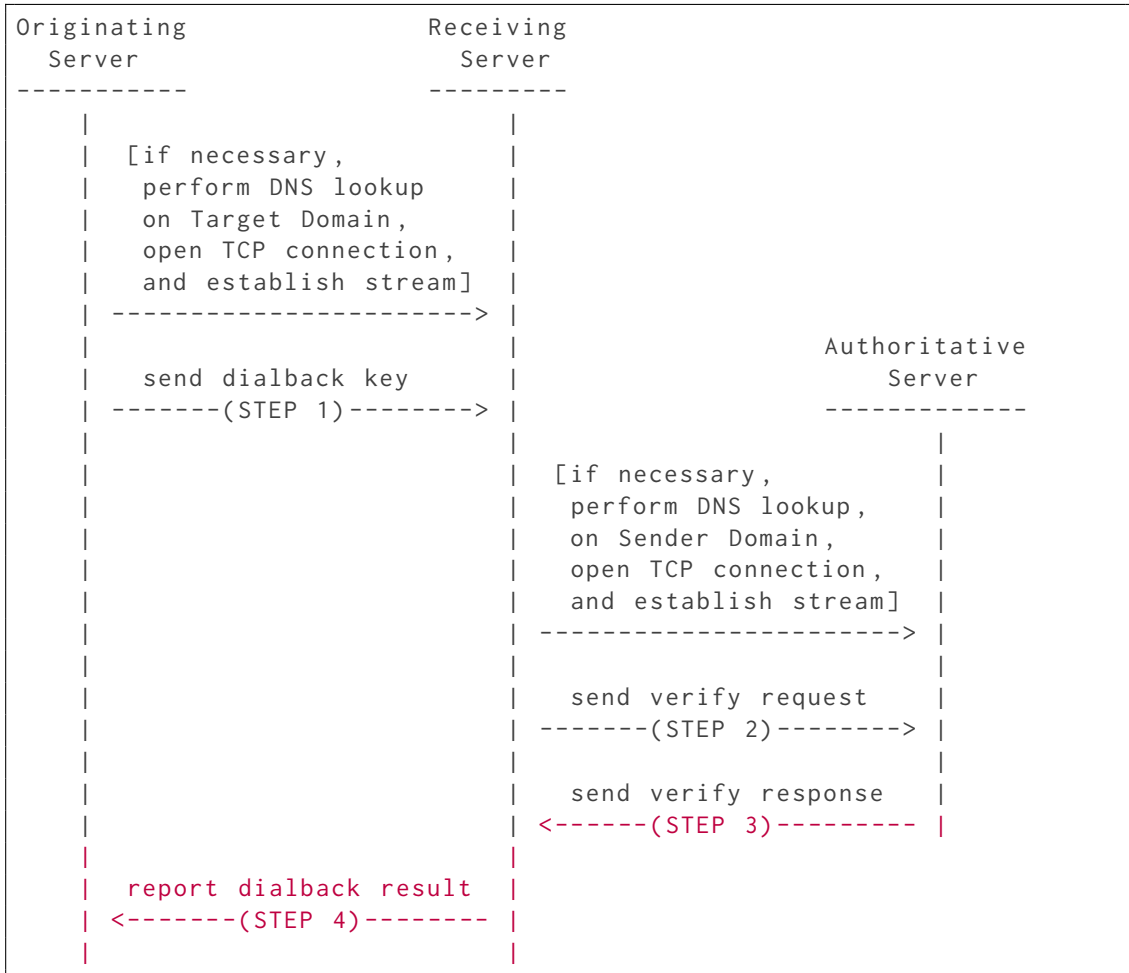
In Server Dialback, the equivalent of the utility company worker is the **ORIGINATING SERVER**, i.e., the machine that wants to send a message from an entity at the **SENDER DOMAIN** to an entity at the **TARGET DOMAIN** and thus is attempting to establish a connection for the **DOMAIN PAIR** (combination of Sender Domain and Target Domain). The equivalent of the person at the house is the **RECEIVING SERVER**, i.e., the machine to which the Originating 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 Originating Server represents the Sender Domain). And the equivalent of the company headquarters is the **AUTHORITATIVE SERVER**, i.e., the machine that is discovered from a DNS lookup for the Sender Domain; for simple deployments this will be the Originating Server, but it could be a separate machine in the Originating Server's network (where "network" is defined by knowledge of a shared secret for verification of dialback keys).

The basic flow of events in Server Dialback consists of the following four steps:

1. The Originating Server generates a dialback key and sends that value over its XML stream with the Receiving Server. (If the Originating 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 Originating 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 Originating Server whether its identity has been verified or not.

After Step 4, the Originating 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 a weak identity verification of the Sender Domain, this also ensures that the Receiving Server is accepting stanzas for the Target Domain. We can represent this flow of events graphically as follows.



2 Protocol

This section describes the protocol in detail.

Assumptions used in the examples:

- The Sender Domain is "sender.tld". A DNS lookup on this domain resolves to the machine "authority.sender.tld".
- The Target Domain is "target.tld". A DNS lookup on this domain resolves to the machine "receiver.target.tld".

- The stream ID of the stream from the Originating Server (some IP address) to the Receiving Server (receiver.target.tld) is "D6000229F".
- The shared secret of the Sender Domain (sender.tld) is "s3cr3tf0rd14lb4ck".

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. ⁵

Section 2.1 describes the protocol from the perspective of an active, outbound connection. Section 2.2 describes the protocol from the perspective of an inbound connection. Note that both parts can be implemented, tested, and used separately.

2.1 Outbound Connection

On an outbound connection there are two different tasks that the sending server can perform. The first task is to request authorization to send stanzas from the Sender Domain to the Target Domain, which is described under Section 2.1.1. The second task is to respond to requests on the validity of a given dialback key as described under Section 2.1.2.

2.1.1 Originating Server Generates Outbound Request for Authorization by Receiving Server

This subsection describes the interaction between the Originating Server and the Receiving Server, from the perspective of the Originating Server.

When the Originating Server has stanzas to send from the Sender Domain to the Target Domain, does not have a verified connection, or is currently attempting to get a verified connection for this domain pair, it sends a new 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: Originating Server Sends Dialback Key (Step 1)

```
send: <db:result
      from='sender.tld'
      to='target.tld'>
  1   e701f120f66824b57303384e83b51feba858024fd2221d39f7acc52dcf767a9
      </db:result>
```

⁵RFC 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 and is no longer necessary.

The key sent is generated as described in [Dialback Key Generation and Validation](#) ⁶:

```
key = HMAC-SHA256(
    SHA256('s3cr3tf0rd141b4ck'),
    { 'target.tld', '_', 'sender.tld', '_', 'D60000229F' }
)
= 1e701f120f66824b57303384e83b51feba858024fd2221d39f7acc52dcf767a9
```

Note: The Receiving Server MAY use any method to determine the validity of the dialback key and the identity of the Originating Server. The Originating Server MUST NOT make any assumptions about how the Receiving Server verifies the key. This includes the assumption that the key is ever verified by the Receiving Server.

After that, the Originating Server waits for the verification result. If the Originating 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. The Originating Server MUST NOT attempt to re-verify the domain pair on this TCP connection.

Note: While waiting for the verification result, the Originating 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](#) ⁷), the Originating Server MAY impose a timeout on key verification.

If the stream or the underlying TCP connection is closed by the remote side while waiting for the verification result, this is to be handled similar to receiving an error as described below.

After the Receiving Server has verified the request, the Originating Server receives the verification result.

The result is either valid...

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

```
recv: <db:result
    from='target.tld'
    to='sender.tld'
    type='valid' />
```

... or invalid ...

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

```
recv: <db:result
    from='target.tld'
    to='sender.tld'
    type='invalid' />
```

⁶XEP-0185: Dialback Key Generation and Validation <<http://xmpp.org/extensions/xep-0185.html>>.

⁷RFC 4732: Internet Denial-of-Service Considerations <<http://tools.ietf.org/html/rfc4732>>.

If the value of the 'type' attribute is "valid", then the connection between the domain pair is considered verified and the Originating Server can send any outbound stanzas it has queued up for routing to the Receiving Server for the domain pair.

If the value of the 'type' attribute is "invalid", then the Receiving Server is reporting that Originating Server's identity (as valid for the Sender Domain) could not be verified by the Authoritative Server. In this case, the Originating Server MUST NOT attempt to send any outbound stanzas it has queued up for routing to the Receiving Server for the domain pair. In addition, the Receiving Server MUST close the stream as described in Section 4.4 of RFC 6120. 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 to be considered non-fatal for the XML stream, but the Originating Server MUST return any queued stanzas to the respective senders at the Sender Domain with a <remote-server-timeout/> stanza error.

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

```
recv: <db:result
      from='target.tld'
      to='sender.tld'
      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 Originating Server by Authoritative Server

This subsection describes the interaction between the Receiving Server and the Authoritative Server, from the perspective of the Receiving Server.

To determine the validity of a dialback key received from the Originating Server, the Receiving Server needs to establish communications with the Authoritative 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 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 Originating Server (here the machine is "authority.sender.tld").

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 Originating 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 identifier of the Receiving Server's response stream header to the Originating Server. 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.

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

```
send: <db:verify
      from='target.tld'
      id='D60000229F'
      to='sender.tld'>
  1
      e701f120f66824b57303384e83b51feba858024fd2221d39f7acc52dcf767a9
</db:verify>
```

After that, the Receiving Server waits for the verification result. While doing so, it can still use the connection to send any dialback packets or 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='sender.tld'
      id='D60000229F'
      to='target.tld'
      type='valid'>
</db:verify>
```

... or invalid ...

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

```
recv: <db:verify
      from='sender.tld'
      id='D60000229F'
      to='target.tld'
      type='invalid'>
</db:verify>
```

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='sender.tld'
```

```

        id='D60000229F'
        to='target.tld'
        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.

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 MAY be dropped silently. If an inbound connection is found, the Receiving Server uses it to communicate the verification result to the Originating Server. A positive result indicates the readiness of the Receiving Server to accept stanzas from the Originating Server for this domain pair.

2.2 Inbound Connection

There are two different tasks on an inbound connection. The first task is to authorize inbound connections, which is described under Section 2.2.1. The second task is to answer requests for the validity of a dialback key, which is described under Section 2.2.2.

2.2.1 Receiving Server Handles Inbound Authorization Request from Originating Server

This subsection describes the interaction between the Originating Server and the Receiving Server, from the 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 Originating Server (Step 1)

```

recv: <db:result
      from='sender.tld'
      to='target.tld'>
  1
    e701f120f66824b57303384e83b51feba858024fd2221d39f7acc52dcf767a9
</db:result>

```

This key MUST be verified before the Originating Server is authorized to send stanzas from the Sender Domain ('sender.tld'). The verification process might fail prematurely, for example,

if the Receiving Server's policy states that connections from the Originating Server or the Sender Domain are not allowed.

The usual method for verifying that the Originating Server is authorized to send stanzas for the Sender Domain is to "dial back" the Authoritative Server for the Sender Domain and ask it to validate the dialback key which is contained in the XML character data of the request. Other methods can be used for verifying the identity of the Originating Server, but are out of scope for this document.

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 for sending XML stanzas for other domain pairs.

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

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

After the validity of the key has been established (for example, by the Authoritative Server), the domain pair is to be considered as verified and the Receiving Server MUST accept stanzas from the Originating Server for the verified domain pair.

In addition, the Originating Server is notified of the result. 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".

Therefore, here again the result is either valid...

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

```
send: <db:result
      from='target.tld'
      to='sender.tld'
      type='valid' />
```

... or invalid ...

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

```
send: <db:result
      from='target.tld'
      to='sender.tld'
      type='invalid'/>
```

If the type is 'invalid', the Originating Server is attempting to spoof the Sender Domain. The Receiving Server MUST NOT accept stanzas from the Originating Server for the Sender Domain, SHOULD log the attempt, and MUST close the XML stream (as described in Section 4.4 of RFC 6120).

As mentioned, Server Dialback results in weak identity verification of the Sender Domain by the Target Domain. In order to proceed with bidirectional communication so that the Target Domain can send XML stanzas to the Sender Domain, the Receiving Server needs to initiate a dialback negotiation with the Originating Server (i.e., assume the role of an originating server in a new dialback negotiation on a new TCP connection).

2.2.2 Authoritative Server Handles Inbound Verification Request from Receiving Server

This subsection describes the interaction between the Receiving Server and the Authoritative Server, from the perspective of the Authoritative Server (i.e., this section is the mirror image of Section 2.1.2).

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

```
recv: <db:verify
      from='target.tld'
      id='D60000229F'
      to='sender.tld'>
  1
    e701f120f66824b57303384e83b51feba858024fd2221d39f7acc52dcf767a9
</db:verify>
```

If the Target Domain as given in the 'to' attribute of the element does not match a configured local domain, 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.

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

```
send: <db:verify
      from='sender.tld'
      id='D60000229F'
      to='target.tld'
      type='error'>
```

```

<error type='cancel'>
  <item-not-found xmlns='urn:ietf:params:xml:ns:xmpp-stanzas' />
</error>
</db:result>

```

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('s3cr3tf0rd14lb4ck'),
  { 'target.tld', '_', 'sender.tld', '_', 'D60000229F' }
)
= 1e701f120f66824b57303384e83b51feba858024fd2221d39f7acc52dcf767a9

```

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...

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

```

send: <db:verify
      from='sender.tld'
      id='D60000229F'
      to='target.tld'
      type='valid' />

```

... or invalid ...

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

```

send: <db:verify
      from='sender.tld'
      id='D60000229F'
      to='target.tld'
      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 Receiving Server has verified the identity of the Originating Server, so that the Originating Server can send, and the Receiving Server can accept, XML stanzas over the "initial stream" (i.e., the stream from the Originating Server to the Receiving Server). In order to verify the identities of the entities using the "response stream" (i.e., the stream from the Receiving Server to the Originating Server), dialback MUST be completed in the opposite direction as well (i.e., with a reversal of roles so that the Receiving Server is now acting as an originating server and the Originating Server is now acting as a receiving server).

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

```
<stream:stream
  xmlns='jabber:server'
  xmlns:db='jabber:server:dialback'
  xmlns:stream='http://etherx.jabber.org/streams'
  from='target.tld'
  to='sender.tld'>
```

Although this method of advertising protocol support has been superseded by the use of stream features as originally defined in RFC 3920, the server dialback protocol predates the existence of stream features and therefore the namespace declaration method is still used in this instance.

2.4.2 Dialback with Error Handling

If a server supports graceful handling of dialback errors as described under Section 2.4, 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 compatibility 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 from 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 introduces a third value for the 'type' attribute, with the value "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. Furthermore, a server SHOULD send these errors only to XMPP 1.0 peers that advertise support for dialback errors as described under Section 2.3.2. Dialback errors are to be considered non-fatal for the XML stream, but queued stanzas MUST be returned to the respective senders with a <remote-server-timeout/> stanza error. If an error is encountered in Step 3, the Receiving Server MUST send a <remote-server-not-found/> error to the Originating Server.

When the <db:verify/> or <db:result/> element is of type "error", the element MUST contain an <error/> element, which is similar to a "stanza error" as specified in [XMPP Core](#)⁸. This specification re-uses the following stanza error conditions.

⁸RFC 6120: Extensible Messaging and Presence Protocol (XMPP): Core <<http://tools.ietf.org/html/rfc6120>>.

Condition	Description	When Occurs
<item-not-found/>	The domain given in the 'to' attribute of the request is not hosted on the Receiving Server. The domain is used in the 'from' attribute nonetheless with 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 Originating 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 which mandates usage of TLS before dialback and the Originating Server did send the dialback request without using TLS.	Step 3
<not-authorized/>	The Receiving Server enforces a policy which requires a valid x509 certificate containing the identity of the Sender Domain for dialback requests and the Originating Server did not provide a certificate with an identity that matches the Sender Domain.	Step 3

2.6 Multiplexing

A single XML stream between Originating Server and Receiving Server can be used to multiplex stanzas for more than one domain pair. This usage is for historical reasons also known as "PIGGYBACKING". One common motivation for this is virtual hosting, under which many domains are hosted on the same server. 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 "target.tld" and the "sender.tld" XMPP servers might host [Multi-User Chat](#)⁹ services at "chat.target.tld" and "chat.sender.tld" respectively. Without multiplexing, many server-to-server connections would be necessary to exchange stanzas between those domains. With more domains, 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](#)¹⁰. Multiplexing reduces the number of connections to two.

Note: Because dialback operates on domain pairs, a total of eight dialback negotiations is necessary for a bidirectional exchange of stanzas between two sending domains and two target domains. (Work is ongoing to overcome this multiplication of TCP connections and dialback negotiations; see [Domain Name Assertions](#)¹¹.)

2.6.1 Multiplexing Sender Domains

In order to accept XML stanzas from rooms at "chat.sender.tld" intended for addresses at "target.tld", the "target.tld" domain will need to validate the "chat.sender.tld" domain (just as it already did for the "sender.tld" domain). Thus the Originating Server would now initiate a dialback negotiation with "target.tld" but specify the Sender Domain as "chat.sender.tld". Specifying different Sender Domains is called "SENDER PIGGYBACKING" and MAY be used without further negotiation.

2.6.2 Multiplexing Target Domains

Likewise, to send stanzas to rooms at "chat.target.tld" from addresses at "sender.tld", the Originating Server would initiate dialback negotiation with "chat.target.tld" on the same connection that might already be used to send stanzas from "sender.tld" to "target.tld", specifying the Target Domain as "chat.target.tld". Specifying different target domains is called "TARGET PIGGYBACKING".

The Originating Server SHOULD NOT use Target Piggybacking unless the Receiving Server has signalled support for dialback error handling via <stream:features/> as described under [Dialback with Error Handling](#). The Originating Server MAY then attempt to multiplex a Sender Domain 'B' on the stream to the Receiving Server that is already used for Sender Domain 'A'

⁹XEP-0045: Multi-User Chat <<http://xmpp.org/extensions/xep-0045.html>>.

¹⁰XEP-0205: Best Practices to Discourage Denial of Service Attacks <<http://xmpp.org/extensions/xep-0205.html>>.

¹¹Domain Name Assertions <<http://tools.ietf.org/html/draft-ietf-xmpp-dna>>.

if the hostname and port resolution results in the same IP address and port combination. For example:

Listing 19: DNS SRV Record for the sender.tld Zone

_xmpp-server._tcp.target.tld.	86400	IN	SRV	10	0	5269	receiver.
target.tld							
_xmpp-server._tcp.chat.target.tld.	86400	IN	SRV	10	0	5269	receiver.
target.tld							
receiver.target.tld.	86400	IN	A				
10.44.0.4							

Because DNS lookups for both "target.tld" and "chat.target.tld" resolve to the same IP address (10.44.0.4) and port (5269), "sender.tld" MAY initiate a dialback negotiation from "sender.tld" to "chat.target.tld" over the same XML stream that is already used to send stanzas from "sender.tld" to "target.tld".

3 Security Considerations

Server Dialback helps protect against domain spoofing, thus making it more difficult to spoof XML stanzas. It is not a mechanism for authenticating, securing, or encrypting streams between servers as is done via SASL and TLS, and results in weak verification of server identities only. Furthermore, it is susceptible to DNS poisoning attacks unless DNSSEC (see [RFC 4033](#)¹²) is used. Even if the DNS information is accurate, Server Dialback cannot protect against attacks where the attacker is capable of hijacking the IP address of the remote domain. Domains requiring robust security SHOULD use TLS and SASL. If SASL is used for server-to-server authentication, Server Dialback SHOULD NOT be used since it is unnecessary.

4 IANA Considerations

This document requires no interaction with the [Internet Assigned Numbers Authority \(IANA\)](#)¹³.

¹²RFC 4033: DNS Security Introduction and Requirements <<http://tools.ietf.org/html/rfc4033>>.

¹³The 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/>>.

5 XMPP Registrar Considerations

5.1 Protocol Namespaces

The XMPP Registrar¹⁴ includes 'jabber:server:dialback' in its registry of protocol namespaces (see <<http://xmpp.org/registrar/namespaces.html>>).

5.2 Stream Features

The XMPP Registrar shall include 'urn:xmpp:features:dialback' in its registry of stream features (see <<http://xmpp.org/registrar/stream-features.html>>).

The submission is as follows:

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

6 Acknowledgments

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

7 XML Schema

7.1 Dialback

Note Well: the 'error' value for the 'type' attribute was added since RFC 3920.

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

<xs:schema
  xmlns:xs='http://www.w3.org/2001/XMLSchema'
  targetNamespace='jabber:server:dialback'
  xmlns='jabber:server:dialback'
```

¹⁴The 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 <<http://xmpp.org/registrar/>>.

```

    elementFormDefault='qualified'>

<xs:element name='result'>
  <xs:complexType>
    <xs:simpleContent>
      <xs:extension base='xs:NMTOKEN'>
        <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:extension>
    </xs:simpleContent>
  </xs:complexType>
</xs:element>

<xs:element name='verify'>
  <xs:complexType>
    <xs:simpleContent>
      <xs:extension base='xs:NMTOKEN'>
        <xs:attribute name='from' type='xs:string' use='required' />
        <xs:attribute name='id' type='xs:NMTOKEN' 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:extension>
    </xs:simpleContent>
  </xs:complexType>
</xs:element>

</xs:schema>

```

7.2 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:schema>
```