I

Dozer Configuration Guide

Computer Modules, Inc.
DVEO Division

11409 West Bernardo Court
San Diego, CA 92127, USA

Telephone: +1 858 613 1818
Fax: +1 858 613 1815

www.dveo.com

Introduction

This document provides an overview and usage guide as to how the Dozer protocol is initiated and used within a server operating system. Since operating systems may differ – linux vs. Windows, linux flavor vs. linux flavor – any command line or code samples provided may be different than those that might be required for your operating system. Therefore, please consider this document as a description of the pre-requisites and mechanics of the protocol.

Software Prerequisites

- The operating system should have the ability to create and configure a TAP adapter (one for each server and each client are necessary).

  linux: in most distributions the system should create and configure TAP adapters from a script or command line “on the fly” without requiring additional packages. The Dozer scripts are able to take advantage of this. In general, if you name your dozer configuration directories in a format such as dozerclient1, dozerclient2… dozerserver1, dozerserver2… the scripts will have no problems creating and configuring the TAP adapters.

  Windows: for Microsoft Windows, which provides a TAP adapter but does not install one by default, it is the responsibility of the system administrator to create as many TAP adapters as the expected number of DOZER tunnels prior to use. Unfortunately, there is no command line facility to create a TAP adapter. There is a command line utility (netsh) for configuration of an existing TAP adapters only.

Fundamental Concepts

The functions of Dozer are to ensure a media stream which traverses the public Internet with efficiency and accuracy, without adding jitter, and without dropouts.

The DOZER executables (one for server, one for client) are software which create an IP tunnel between a server on one side, and one or more clients on the other side. The Dozer processes provide an automatic re-request mechanism for dropped packets as well as configurable Forward Error Correction.

Dozer also provides a mechanism to reproduce on the destination side the instantaneous bitrate and packet spacing of a media stream, equal to that by which the source received the stream.

The Dozer client is the source side/transmitter of the media stream. The Dozer server is the destination side/receiver of the stream.

Server and client(s) must have a route by which they can initially reach

each other over public or private IP networking in order to establish the tunnel. Once established, each side addresses the other side via the IP addresses assigned to that tunnel. If, for example, the server side TAP adapter has an address of 1.1.1.1 and the client side 1.1.1.2, the server can ping 1.1.1.2 and vice versa. So long as each side knows the others TAP adapter address, any software running on either side just sees a normal network. Any packets going over that network will have the benefits of Dozer's automatic re-request mechanism for dropped packets, configurable Forward Error Correction and reproduction of the instantaneous bitrate and packet spacing.

Assume a source-side media server outputs from point A to point B; it should be configured to output to the TAP adapter address and to whatever port it normally would output to. On the destination side, the player should be configured to listen to its TAP adapter address on that port. Dozer does the rest. If any packets are dropped, for example, the Dozer receiver re-requests the packets.

The Dozer receiver also examines tags which the Dozer transmitter puts into the packet that provide information on the timing the packet was received on the transmitter side, and speed as regards to interpacket spacing. When you set up a Dozer tunnel, you specify a virtual size for that tunnel – as if it were a leased private line. Dozer calculates how many bytes it would take to fill that private line, and inserts a tag so that the receiving side can reconstruct the stream exactly as if it had traversed such a private line, reproducing the timing and speed at which those packets were originally received on the transmitting side. In this way, it prevents any jitter from being introduced on the receiving side.

There can be only one transmitter for a given Dozer link. There can be many receivers for the same transmitter.

The server hardware can run as many Dozer servers (and clients) as the hardware can support. Each must have a different IP/port assignment.

The administrator must write configuration files with the essential parameters, such as the IP and port to bind to, and then provide the configuration file locations to the Dozer executable as command line parameters.

DVEO products such as OnRamp provide a Web GUI for these configurations. This document assumes the Web GUI is not available to the administrator setting up Dozer on their own platform(s). In DVEO products, Dozer runs on a specialized linux distribution called Ammux OS. For the purposes of this document, we'll assume your platform is the latest Ubuntu Server LTS release.

The format of the configuration files is

    variable=value

One pair per line. For any variables not used (i.e., blank as in variable=), there is no need to include them in the file.

Compression and encryption are supported, and controlled via options in the configuration problems. (Note: in our experience, compressing one side, but not the other, or encrypting one side, but not the other are the two most common errors to look for when debugging a Dozer link which appears not to be functioning).

Dozer can also duplicate streams (redundancy in order to increase the probability of the stream reaching the destination properly) as well as split streams (multiple paths to help the data arrive at the destination faster.

The Server (Destination or Receiver) Configuration Files

The following documents the variables in the server.conf file. The server is the side receiving the media stream; the destination.

server.conf

servername	Provide a friendly name for the server for logging purposes; up to 16 characters.
port	Specify the port number (udp) for use by the Dozer server
cipher	Specify 128 for AES-128; 192 for AES-192; blank for none.
bindip	Specify the underlying device name to bind the TAP adapter to, for example, eth0.
enableclient2client	Specify 1 or 0 to enable or disable traffic between servers to traverse the tunnel to each other via the ports used to make the connection.
compression	Specify 1 or 0 to enable or disable lzo compression
sharedsecret	enter a string to verify both sides of the tunnel
remotepeerip	Specify an IP address for the client (source) side.
remotepeerport	Specify the port number (udp) for the client (source) side.
networkmode	Specify tap. No other choice is valid.
loglevel	Specify “error,” “warning,” “info” or “debug,” for increasing levels of verbosity, respectively.
configmode	Specify 1 for Point to Point with client callback (STB mode); 2 for Multipoint to Point Dozer.
username	Specify a user name, which the client must present to join the tunnel.
password	Specify a password, which the client must present to join the tunnel.

protocolversion	v2 in most, if not all, cases. There may be some cases in which for backward compatibility with older DVEO equipment v1 is necessary.
pinginterval	keepalive time. Enter a value in seconds.

The Client (Source or Transmitter) Configuration Files

The following documents the variables in the client.conf file.

client.conf

configmode	Leave blank/empty for Point to Point; 1 for Point to Multipoint / Advanced (for multiple receivers). Note: we suggest you start with one receiver until you are completely familiar with Dozer.
clientname	Provide a friendly name for the client for logging purposes; up to 16 characters.
cipher	Specify 128 for AES-128 or 192 for AES-192.
compression	Specify 1 or 0 to enable or disable lzo compression
sharedsecret	Enter a string to verify both sides of the tunnel
networkmode	Specify tap.
loglevel	Specify “error,” “warning,” “info” or “debug,” with increasing levels of verbosity, respectively.
tunnelindex	Specify 1
maxbitrate	Specify a size for the maximum bitrate of the tunnel. The virtual packet size calculator calculates a virtual packet size based on the bytes required to fill a private wire of the predefined bitrate capacity in a time equal to the inter-packet interval between the current and previous incoming datagrams. This provides the receiver with the data necessary to replicate the speed at which the datagrams containing the stream originally arrived at the transmitter. The size options are as follows: 1 1Mbps 5 5Mbps 10 10Mbps 20 20Mbps 50 50Mbps 100 100Mbps 200 200Mbps 500 500Mbps
protocolversion	v2 in most, if not all, cases. There may be some cases in which for backward compatibility with older DVEO equipment v1 is necessary.

For the following values substitute an integer for each peer to which the client is to transmit. For example: peer1_targetenabled

peer#_targetenabled	Specify 1 or 0 to enable or disable this target. This allows you to keep several target configurations in the configuration file, and if some should only be active at certain times or for certain events, you can disable them without having to erase the settings.
peer#_targetname	Specify a friendly name, up to 16 characters, for the target.
peer#_bindip	Specify the underlying network device for the TAP adapter, such as eth0.
peer#_bindport	Specify a port number or “auto” for the TAO adapter’s port.
peer#_dozermode	Specify “dozer” in all cases.
peer#_filtertype	Specify “IPV4” in all cases.
peer#_username	Specify a user name, which the peer must present to join the tunnel.
peer#_password	Specify a user name, which the peer must present to join the tunnel.
peer#_buffermin	Specify a minimum transmit buffer in milliseconds
peer#_buffermax	Specify a maximum transmit buffer in milliseconds
peer#_rttmin	Specify a minimum round trip transit time in milliseconds. This is used in calculations for filling the virtual tunnel.
peer#_rttmax	Specify a maximum round trip transit time in milliseconds. This is used in calculations for filling the virtual tunnel.
peer#_remotepeerip	Specify an IPv4 address for the peer.
peer#_remotepeerport	Specify the port on the peer side for the tunnel.
peer#_weight	Specify an integer weight for load balancing over several paths to the same target.
peer#_multipath	Specify balance for equal weighting of several paths to the same target..

