

# **Eidos Patch System**

*for Playstation®2 and Microsoft Windows*

## **Generating and Serving Patches**

*Version 1.2*

*Updated April 6, 2005*

Prepared by Pinniped Software

Pinniped Software



game and internet programming

Copyright © 2005, Eidos, Inc. All rights reserved.



## Table of Contents

Overview.....	1
What is a Patch?.....	1
Version Numbers.....	1
Generating a Patch.....	1
makepatch.....	2
DiffPatch.....	2
Authoring the Patch.....	3
Creating the Version Description File.....	3
VDF Fields.....	4
More Information for Playstation®2 Games.....	5
Serving the Patch.....	5
Important: Non-Chunked Encoding Required.....	5



# Eidos Patch System

## Generating and Serving Patches

Version 1.2

### Overview

---

This document describes how to generate a patch or update for a game which uses the Eidos patching system. It is intended to be used together with the document *Eidos Patch System Client-Side Programming Guide*.

### What is a Patch?

In this document, when we say *patch*, we are referring to any update of program code or data applied automatically for a user of your game. When the game downloads the update, it modifies, or patches, the original data as delivered. By using such a patching scheme (as opposed to replacing the original data completely), a smaller download can be used, and a small amount of security is introduced, because patching requires the original data to be present.

While patching generally refers to this process of modifying original assets, it is used somewhat interchangeably with the term *update* in this document.

### Version Numbers

Each game that uses the patching system must have defined a version numbering scheme which is used to test for the presence of patches, as well as possibly controlling entry to online games.

The patching system treats the version number as an arbitrary unsigned 32 bit number, whose interpretation is up to your game. It could be as simple as a sequential numbering scheme starting at zero for the original, unpatched game, and incrementing each time an update is made available. Or, for the more adventurous, it could be a calculated value, such as a CRC32, which may help in verifying the integrity of your application.

Regardless of how you choose to assign version numbers in your application, it is important to understand how they are used by the patching system. First of all, they are indicated in the *Version Description File* (VDF, generally called version.txt). When you ask the patching system to test for an update, it will download the VDF, and compare the version number in the VDF with that provided by your application. If they are different, the patching system will indicate to you application that an update is needed.

In addition to storing the version number in the VDF, the patch generation process will store it inside the patch itself, so it is important that the two version numbers are identical.

### Generating a Patch

---

A complete patch consists of the VDF described above, a description file for that patch,

and the associated patch file. The first step to generating the complete patch is to generate the patch file itself.

Two programs are provided to generate patch files. **DiffPatch** is an interactive Windows program which helps guide you through the process, while **makepatch** is a simple command-line utility to automatically generate a patch from batch command files.

Both programs work by comparing two directories (and all child directories), and generating a patch file. They both take version numbers as well, which are stored inside the patch.

## ***makepatch***

The makepatch program is a simple program which is likely to be the workhorse of your patch generation process. It is used as follows:

**makepatch** *original-dir modified-dir version-number output-file*

Compare the directory specified by *original-dir* with *modified-dir*, and for each file which is different, or exists only in *modified-dir*, create the patch commands to change the original to the modified version of that file. The patches for all files are stored together (with other information) in *output-file*. The *version-number* argument is required, and must match the version number which will be stored in the version definition file. It must also be unique; that is, it may not be the same as the version used for the original shipped game, and it may not match a previously released version.

## ***DiffPatch***

When you run the DiffPatch application you will get the dialog shown in Figure 1.

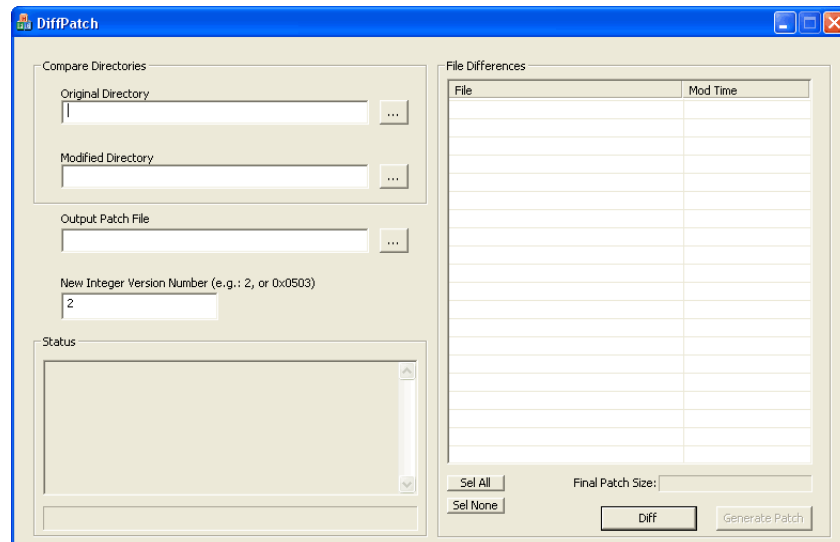
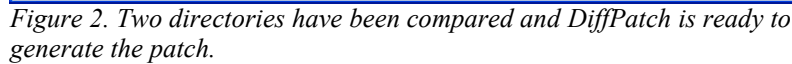


Figure 1. The DiffPatch Dialog

On the left side of the dialog are four input boxes where you can give the same

The result of pressing the “Diff” button for our test case is illustrated in Figure 2. Note that the Status pane on the lower left contains any information output from the procedure. Note also that a list of differences appears on the right hand side. The check boxes next to each file can be used to deselect the file so that it is not included in the final patch.



## Authoring the Patch

## Creating the Version Description File

3

would need to run on the Tool, and would need to access your game's development DNAS servers, and would thus need to be modified and built for each individual game.)

**Important limitation:** The VDF file **must** be under 512 bytes or the game will fail to download it.

## VDF Fields

The VDF is a fairly simple text file describing the patch. Here is the contents of a sample VDF:

```
# VDF for tst.patch

Version:    2                # 32 bit unsigned version number
Host:      patch.eidos.com   # http host for patch
Patch:     /gameps2/v02.enc  # Location of patch on host
RawSize:   420 # Size of DNAS authored file to be downloaded
StoreSize: 350 # Size of encrypted patch for memory card
ClearSize: 317 # Size of decrypted patch in memory
Description: /gameps2/v02desc.txt # What's in this patch
Mandatory: yes # This patch is required
```

The VDF file consists of *comments*, *fields*, and *values*. A comment is delineated by the “#” character, and includes all text from the “#” character to the end of the line.

Each line which is not blank and does not contain only a comment must have a field name, followed by a colon (":"), followed by the value of that field. The fields are indicated in bold above, and are described below.

- **Version** – This is the new version number for the game after the patch is applied. For proper functioning of the patch system, it *must* match the version specified when generating the patch.
- **Host** – The HTTP host from which the patch is to be fetched.
- **Patch** – The full path name of the patch on the above host. The initial slash ("/") is mandatory.
- **RawSize** – The size in bytes of the raw encrypted (DNAS-authored on PS2) file to be downloaded.
- **StoreSize** – For Windows titles, this size can be the same as RawSize. For Playstation®2 titles, StoreSize is the size in bytes of the personalized patch which will be stored on the memory card for Playstation®2. Experience has shown that this size is generally less than the raw size, so it is OK when creating a VDF to put the same value in this field as for RawSize. However, when you are testing the patch in the game, be on the lookout for debug output complaining that this value is wrong, and giving you the correct value to put in this field. When you have the correct value, you can edit the VDF and correct it, although as long as the correct StoreSize is less than RawSize the patch process should still proceed correctly.
- **ClearSize** – The size of the original unencrypted patch. This size is the size of the



patch file as originally output by `makepatch` or `DiffPatch`.

- **Description** – The location of a description file for this game. Normally the description file might be a simple text file describing what has changed with this version of the game, but your game is free to interpret it in any fashion. This file must reside on the same host as the patch itself (given by the **Host** field).
- **Mandatory** – Whether the patch should be considered mandatory. The exact interpretation of this field is left up to the game, but it generally indicates that a player must apply this patch before continuing. It should be set to `false` for patches which are optional (such as bonus assets). This parameter evaluates to `true` or `false` in the game, and can be set to **yes**, **true**, or **1** to specify the patch is mandatory, or **no**, **false**, or **0** to specify the patch is optional.

## ***More Information for Playstation®2 Games***

When testing your patch with your VDF, use a debug version of your game and look at the debug output for a message which looks like the following:

Warning: encrypted size 362 from DNAS is different from size 350 from version file.  
Please update version information file.

For our sample VDF given above, this would indicate that we want to edit the VDF and change **StoreSize** from 350 to 362. However, as long as **StoreSize** is less than **RawSize**, the game should still function properly.

In the (currently unobserved) case where **StoreSize** needs to be greater than **RawSize**, the debug version of your game should give a more forceful message:

PATCH ERROR: encrypted size 450 is larger than buffer size 420!

In this case, it is required for **StoreSize** to be corrected to the new larger value for your program to work correctly. Also make sure that the reported buffer size is at least as large as **RawSize**. If it is not, then there is likely another error in your program.

## **Serving the Patch**

---

Once the patch file has been authored by the DNAS system (for Playstation®2 games), and the VDF and patch description file have been created, they are ready to be served on your web site.

Your application will have stored in it an HTTP address for finding the VDF. It will then use HTTP 1.1 to download the the version description file. The version description file will then tell your application the location of the patch itself as described above. HTTP 1.1 is also used to download the patch description and the patch itself.

## ***Important: Non-Chunked Encoding Required***

The current version of the client-side software will not handle chunked encoding from the

server. That is, it depends on the server returning the `Content-Length` field, and not returning `Transfer-Encoding: chunked`. Normally, a server will only use chunked encoding for files where it cannot determine a length ahead of time, such as cgi scripts and html files (which can have server-side substitutions, changing the file size). As long as your files end with extensions that are unknown to the server (such as ".enc" for the authored file) or are known to be a fixed length (such as ".txt"), you should not encounter any problems in this area.

It is possible to modify the client-side software to handle chunked encoding, but it was decided that with a little care on the server side, a lot of code complexity and memory allocation issues could be avoided on the client side.

If the client application is having trouble with the server, there are a few methods with which you can test the transfer encoding of the server. First, if you have an interactive http analyzing program you should be able to check the headers returned when downloading the patch files (the VDF and the authored patch).

If you do not have a special program for this purpose, you can use the telnet program to do the same thing. First, connect to the http server on port 80 using telnet:

```
> telnet patchserver.company.com 80
```

Then request the header for the file you want to test with the HEAD command:

```
HEAD /gameps2/version.txt HTTP/1.1<cr>
Host: patchserver.company.com<cr>
<cr>
```

Note that you must type both lines above, followed by two carriage returns (indicated with the <cr> above).

After that, you should get back the header, similar to this sample:

```
HTTP/1.1 200 OK
Date: Wed, 12 Jan 2005 20:57:44 GMT
Server: Apache/1.3.27 (Unix) PHP/4.3.4
Last-Modified: Tue, 11 Jan 2005 20:16:10 GMT
ETag: "1fa284-1b9-41e4340a"
Accept-Ranges: bytes
Content-Length: 441
Content-Type: text/plain
```

In this case, the `Content-Length` field (noted in bold) came back OK, so we know the server is working properly for us. If the `Content-Length` field does not show up, there is a problem. In that case, try changing the extension of the file, say from .txt to .vdf, and see if the result changes. In the unlikely case that the result does not change, it may be necessary to either change HTTP servers or modify the client-side code patching code.