To allow the recovery application to use the last valid data, the application must have the ability to change the location of the valid data. The setChangeLogFile method changes the value in $this->dog_data_xml. This property was originally set from locating information in the dog application XML file. However, that might not currently be the location of valid data. The code added to the destructor will use the new location of the valid data, apply the changes needed, and place valid data back into the original data file. There is no need to make any changes to the data_application XML file. After the destructor completes, the data file will now contain the most up-to-date valid data.

Example 6-3. The dogdata.php file with logging as well as backup and recovery processes

```
<?php
class dog_data {
private $dogs_array = array(); //defined as an empty array initially
private $dog_data_xml = "";
private $change_log_file = "change.log";
function __construct() {
        libxml:use_internal_errors(true);
        $xmlDoc = new DOMDocument();
        if ( file_exists("e5dog_applications.xml") )        {
        $xmlDoc->load( 'e5dog_applications.xml' );
        $searchNode = $xmlDoc->getElementsByTagName( "type" );
                foreach( $searchNode as $searchNode )        {
                        $valueID = $searchNode->getAttribute('ID');
                        if($valueID == "datastorage") {
                                $xmlLocation = $searchNode->getElementsByTagName( "location" );
                                $this->dog_data_xml = $xmlLocation->item(0)->nodeValue;

                                break;
                        }

                }
        }

        else { throw new Exception("Dog applications xml file missing or corrupt"); }
        $xmlfile = file_get_contents($this->dog_data_xml);
        $xmlstring = simplexml:load_string($xmlfile);
        if ($xmlstring === false) {
                $errorString = "Failed loading XML: ";
                foreach(libxml:get_errors() as $error) {
                        $errorString .= $error->message . " " ;  }
                throw new Exception($errorString); }
        $json = json_encode($xmlstring);
        $this->dogs_array = json_decode($json,TRUE);
}

function __destruct() {
        $xmlstring = '<?xml version="1.0" encoding="UTF-8"?>';
            $xmlstring .= "\n<dogs>\n";
            foreach ($this->dogs_array as $dogs=>$dogs_value) {
                foreach ($dogs_value as $dog => $dog_value) {
                        $xmlstring .="<$dogs>\n";
                        foreach ($dog_value as $column => $column_value)
```