How to Use the Include Directive in PAL

PAL supports the Delphi *include* directive. This allows PAL from another file, such as custom functions or logic, to be included into another PAL for consistency and ease of maintenance.

Write once, use many times.

Table of Contents

How to Use the Include Directive in PAL....1
- Syntax....2
- Simple example....2
- Custom procedure example....3
- Permalink....5
- History....5

Syntax

```
{$include '<full path to pal>'}
```

The directive *must* be in the PAL *above* statements that use the included code.

Simple example

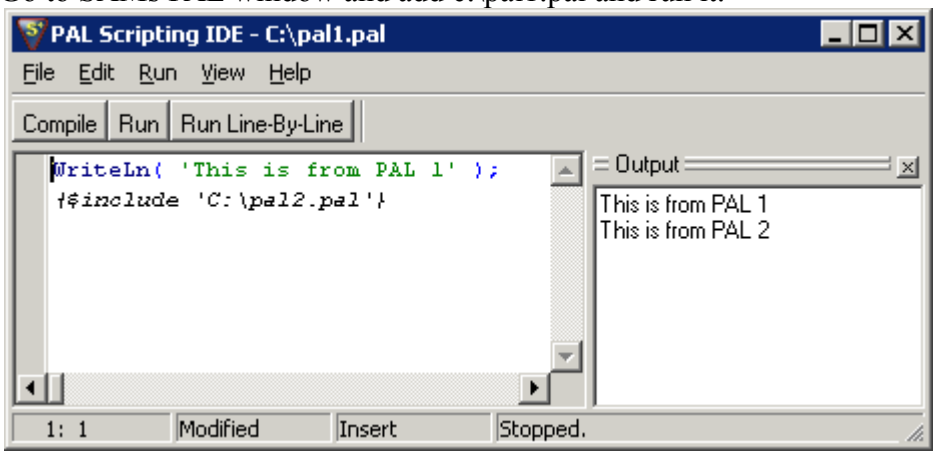
Save this as c:\pal2.pal:

```
WriteLn( 'This is from PAL 2' };
```

Save this as c:\pal1.pal:

```
WriteLn( 'This is from PAL 1' );
{$include 'C:\pal2.pal'}
```

Go to SAMs PAL window and add c:\pal1.pal and run it.



Custom procedure example

Save this as c:\write_log.pal

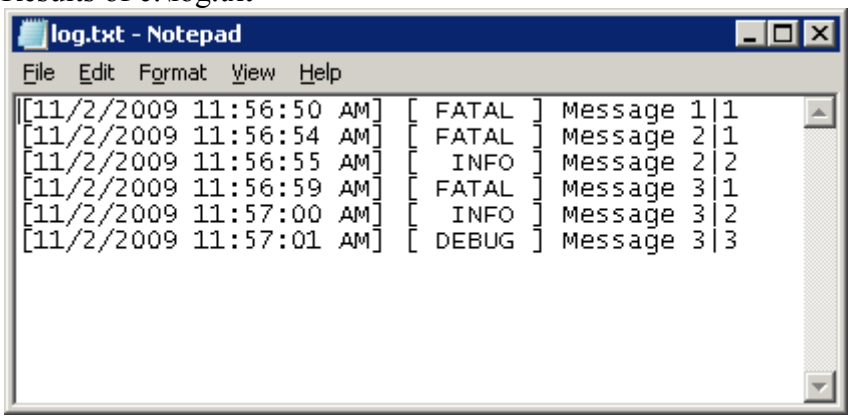
```
{******************************************************
 *
 * $Id: fhr_write_log.pal,v 1.1 2007/12/01 22:03:03 FesterHead Exp $
 *
 ******************************************************
 *
 *  Copyright Steve Kunitzer (FesterHead)
 *
 *  This program is free software: you can redistribute it and/or modify
 *  it under the terms of the GNU General Public License as published by
 *  the Free Software Foundation, either version 3 of the License, or
 *  (at your option) any later version.
 *
 *  This program is distributed in the hope that it will be useful,
 *  but WITHOUT ANY WARRANTY; without even the implied warranty of
 *  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 *  GNU General Public License for more details.
 *
 *  You should have received a copy of the GNU General Public License
 *  along with this program.  If not, see <http://www.gnu.org/licenses/>.
 *
 ******************************************************
 *
 * Procedure to log messages to a file by appending.
 * If the log file does not exist it will be created.
 *
 * Log/Message Levels are:
 *   0 = NO logging
 *   1 = Log FATAL messages
 *   2 = Log INFO messages
 *   3 = Log DEBUG messages
 *
 * A log level of DEBUG will log DEBUG, INFO, and FATAL messages.
 * A log level of INFO will log INFO and FATAL messages.
 * A log level of FATAL will log FATAL messages.
 *
 * Examples:
 *   WriteLog( 0, 2, 'Some informational message' );  // This will NOT log
 *   WriteLog( 1, 2, 'Some informational message' );  // This will NOT log
 *   WriteLog( 2, 2, 'Some informational message' );  // This will log
 *   WriteLog( 3, 1, 'Some fatal message' );          // This will log
 *
 * It is recommended to define a LogLevel in your PAL at the configuration layer
 *   and pass it as a parameter to this function.  This the logging level can be
 *   incremented or decremented as needed during development and production.
 *
 * Logged messages have the following format:
 *   [Date/Time] [LEVEL] message
 *
 * @param userLevel    the user log level.
 * @param messageLevel the message level.
 * @param theMessage   the message to log.
 * @param theLogFile   the log file to write the message by appending.
 *
 ******************************************************
 *
 * History:
 * [2007-12-01] : Initial version
 *
 ******************************************************}
procedure WriteLog( userLevel : Integer;
                    messageLevel : Integer;
                    theMessage : String;
                    theLogFile : String );
  begin
    var messageString : String = '';
    var crlf : String = Chr( 13 ) + Chr( 10 );

    case messageLevel of
      1 : messageString :=  ' FATAL ';
      2 : messageString :=  '  INFO ';
      3 : messageString :=  ' DEBUG ';
```

```
      else messageString := ' ERROR ';
    end;

    // check to see if we log the message
    if ( ( userLevel > 0 ) and
         ( userLevel >= messageLevel ) ) then
      begin
        if ( not FileExists( theLogFile ) ) then
          begin
            // Log file doesn't exist.  This will create the log file.
            StrToFile( theLogFile, '' );
          end;

        // Log the message
        AppendStringToFile( theLogFile, '[' + DateTimeToStr(Now)
                                            + '] ['
                                            + messageString
                                            + '] '
                                            + theMessage
                                            + crlf );
      end;
  end;
```

Save this as c:\pal3.pal:

```
{$include 'C:\write_log.pal'}

var log_file : String = 'c:\log.txt';

var user_level : Integer = 0;
WriteLog( user_level, 1, 'Message 0|1', log_file );  // will NOT log
WriteLog( user_level, 2, 'Message 0|2', log_file );  // will NOT log
WriteLog( user_level, 3, 'Message 0|3', log_file );  // will NOT log

user_level := 1;
WriteLog( user_level, 1, 'Message 1|1', log_file );  // will log FATAL
WriteLog( user_level, 2, 'Message 1|2', log_file );  // will NOT log
WriteLog( user_level, 3, 'Message 1|3', log_file );  // will NOT log

user_level := 2;
WriteLog( user_level, 1, 'Message 2|1', log_file );  // will log FATAL
WriteLog( user_level, 2, 'Message 2|2', log_file );  // will log INFO
WriteLog( user_level, 3, 'Message 2|3', log_file );  // will NOT log

user_level := 3;
WriteLog( user_level, 1, 'Message 3|1', log_file );  // will log FATAL
WriteLog( user_level, 2, 'Message 3|2', log_file );  // will log INFO
WriteLog( user_level, 3, 'Message 3|3', log_file );  // will log DEBUG
```

Results of c:\log.txt



Permalink

See the FesterHead Consulting Google group for the most current version of this document.

History

Date	Information
2009-11-02	Initial version