```
!
! (c) Copyright 2006-%(copyright-year)%, Sage Software Canada Ltd. (Ontario, Canada)
! $Id: set_mod_timestamp.pvc.pvt,v 1.5 2008/02/05 19:44:30 fred.mcguirk Exp $
/*
 * Build information:
! ** @Author Fred McGuirk
! ** @Date Nov 22, 2007
! ** @OutputFile set_mod_timestamp.pvc

! ** This observer will set the modification timestamp of the output file to be
! ** the same as the source file.
! **
! ** This program uses the Eclipse IResource class to get and set the timestamps.
 */

def class "set_mod_timestamp"
    like "EventManagerObserver"

    ! ** Private property to maintain the name of the temporary file that is created
    ! ** during the Pre-Process so that it can be deleted during the Post-Process.
    local tempSourceFile$

    ! ** This method is used to describe the purpose of this observer.  The
    ! ** text will be placed on the observers preference page to describe and
    ! ** should be limited to 80 characters (at most).
    function getDescription$()              getDescription

/*
    ! ** Get the flag that will be used to trigger the 'update()' method during
    ! ** execution of the built-in events for the plug-in.
    ! **
    ! ** If this observer is not a Post-process observer, the return value of
    ! ** this method should be set to one of the following values:
    ! ** @value _pvxConstants'_ideNone           Do not register as an observer
    ! ** @value _pvxConstants'_idePreProcess     Register: Pre-Process observer
    ! ** @value _pvxConstants'_idePrePostProcess Register: Pre/Post-Process observers
    ! ** @value _pvxConstants'_idePostProcess    Register: Post-Process observer
 */
    function getEventNotificationFlag()     getEventNotificationFlag
/*
    ! ** The logic to be executed when the observer is triggered.  This logic must
    ! ** check the major and minor codes to determine the current event and then
    ! ** decide what action is to be performed.
    ! **
    ! ** If this observer is set to watch both Pre-Process and Post-Process states
    ! ** for events, the logic in the 'update' method must check the state of the
    ! ** _pvxState'getArgument(_pvxConstants'_iEventNotificationFlag$) flag to
    ! ** determine the appropriate code to be executed.
    ! **
    ! ** @param state A reference to an object of class %PvxClass(PvxState)%
 */
    function update(initPvxState)           update
end def

getDescription:
    ! Change the value to be the actual description of this observer
    local theDescription$= \
        "Set modification timestamp of output file to be the same as source file."
return theDescription$

! register as a post-process observer
getEventNotificationFlag:
return _pvxConstants'_idePostProcess
```

```
/*
 * The update routine will be executed by the event manager during the
 * pre/post process logic depending on how this observer registers
 * itself through the getEventNotification() method.
 */
update:
enter aPvxState

    local psMajor$,psMinor$,source,dest$,enfState,x,destObj,fileObj,projObj,sysType$, \
        utcMTime,secondsIntoDay,dayHours,dayMinutes,daySeconds,julDay,theCmd$

    ! Get the Major/Minor codes that identify the current action
    psMajor$=aPvxState'getMajor$(), \
    psMinor$=aPvxState'getMinor$()

    /*
     * This observer is for the program build event only; all other events
     * should be ignored.
     */
    if (psMajor$=_pvxConstants'Incremental_Build$ \
            or psMajor$=_pvxConstants'Incremental_Build_Alt_Exe$) \
        and psMinor$=_pvxConstants'BuildType_BuildOne$ {

        ! Check the current state of processing for the ProvideX Event Manager
        enfState=aPvxState'getArgumentValue(_pvxConstants'_iEventNotificationFlag$)
        switch enfState
        case _pvxConstants'_idePostProcess

            ! Get the arguments from the current action that this logic will use
            eventSourceFile$=aPvxState'getArgumentValue$(_pvxConstants'event_SrcFile$), \
            dest$=aPvxState'getArgumentValue$(_pvxConstants'Dest$)

            ! Get the modification date of the original event source file and
            ! Set the modification date of the output file to be the same
            sysType$=sys
            if pos("LINUX"=ucs(sysType$))>0 then \
                sysType$="Linux"
            switch sysType$
            case "MS-WINDOWS"
                ! Set control values for DLL call
                access_rw=dec(ior($80000000$,$40000000$)), \
                share_rw=1+3, \
                open_existing=3, \
                file_attr_normal=128
                ! Load the MS-Windows 'kernel32.dll' into memory
                ! (since it will be used several times)
                aDLL=dll(addr "kernel32")
                ! Get a handle to the source file
                hSrc=dll(aDLL,"CreateFileA",eventSourceFile$+$00$,access_rw,share_rw,0, \
                        open_existing,file_attr_normal,0)
                dim aSrc$[1:3](8,$00$)
                ! Get creation, access, and modification times for source file
                status=dll(aDLL,"GetFileTime",hSrc,aSrc$[1],aSrc$[2],aSrc$[3])
                ! Close reference to file
                status=dll(aDLL,"CloseHandle",hSrc)

                if status=1 {
                    ! Get a handle to the destination file
                    hDest=dll(aDLL,"CreateFileA",dest$+$00$,access_rw,share_rw,0, \
                            open_existing,file_attr_normal,0)
                    dim aDest$[1:3](8,$00$)
                    ! Get creation, access, and modification times for destination file
```

```
                    status=dll(aDLL,"GetFileTime",hDest,aDest$[1],aDest$[2],aDest$[3])
                    ! change the modification time of the destination
                    aDest$[3]=aSrc$[3]
                    ! Set the new times for the destination file
                    status=dll(aDLL,"SetFileTime",hDest,aDest$[1],aDest$[2],aDest$[3])
                    ! Close reference to file
                    status=dll(aDLL,"CloseHandle",hDest)

                }
                ! remove the "kernel32.dll" from memory.
                s=dll(drop aDLL)
                break
            case "Linux"
                open input (hfn,isz=1,err=*break)eventSourceFile$
                utcMTime=num(fin(lfo,"UTC_Mtime",err=*next))
                close (lfo)
                if utcMTime>0 {
                    julDay=int(utcMTime/86400), \
                    secondsIntoDay=utcMTime-(julDay*86400)-tcb(44), \
                    dayHours=int(secondsIntoDay/3600), \
                    dayMinutes=int(secondsIntoDay/60)-(dayHours*60), \
                    daySeconds=secondsIntoDay-dayHours*3600-dayMinutes*60
                    theCmd$="touch -t "+dte(julDay:"%Y%Mz%Dz")+ \
                        str(dayHours:"00")+str(dayMinutes:"00")+ \
                        "."+str(daySeconds:"00")+" "+dest$
                    x=sys(theCmd$)
                }
                break
            end switch

            break
        end switch
    }
return 0

end
```