```
!
! (c) Copyright 2006, Sage Software Canada Ltd. (Ontario, Canada)
! $Id: ExampleObserver.pvc,v 1.19 2008/02/05 19:44:30 fred.mcguirk Exp $
/*
 * Build information:
! ** @Author Fred McGuirk
! ** @Date Dec 4, 2006

! ** This is an example of the observer class.
! ** The name of the observer must be unique.  It must also be used when
! ** identifying this class in the 'classes.txt' file in the./ext folder by
! ** adding a line "eventName=ExampleObserver".
! **
! ** The entries in the "classes.txt" file are loaded automatically during start
! ** of the ProvideX Event Manager.  As each class is successfully loaded, it is
! ** added the the observers preference page.  This gives the user control of the
! ** external observers (if any) that are active.
 */

/*
 * The directory 'ext/' must not be included in the name of this class.
 * The directory is in the prefix that has been set by 'ide_events.pvx' so
 * the class will be found using the simple name.
 */
def class "ExampleObserver"
    like "EventManagerObserver"

    ! Override the value to be the actual description of this observer
    local theDescription$="Example observer"

    ! Override Value to register the observer
    local theNotificationFlag=_pvxConstants'_idePostProcess

/*
    ! ** 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
    ! ** aPvxState'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

update:
enter aPvxState

    local psMajor$,psMinor$,source,dest$

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

```
    /*
     * Turn on the event log to see a list of the arguments for a specific event, or
     * consult the documentation for building external observers for the list of
     * constants that have already been defined for this purpose.
     */

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

    /*
     * It is possible to override or change the value of an existing argument...
     */
    newDest$=dest$+".tmp"
    aPvxState'setArgument(_pvxConstants'Dest$,newDest$)

    /*
     * Add logic to the specific events that this observer is watching
     *
     * This logic can be added using a SWITCH / CASE structure or a simple IF-THEN.
     *
     * In either case, the action is identified by a major and minor combination code.
     * The full list of major/minor codes is listed in the "Plug-in Structure" section
     * of the documentation for "Extending the ProvideX Plug-in".
     *
     * To keep things simple, this example will use an IF-THEN to add new logic to
     * the Incremental Build post routines.
     *
     */
    if (psMajor$=_pvxConstants'Incremental_Build$ or
psMajor$=_pvxConstants'Incremental_Build_Alt_Exe$) \
        and psMinor$=_pvxConstants'BuildType_BuildOne$ {

        /*
         * The logic that is added cannot use any user-interaction directives since there
         * is usually no visible ProvideX window where this information can be displayed.
         * Also, the normal build process will be blocked until all processing has been
         * completed; a delay in an observer will affect the entire Eclipse session.
         */

        ! print a message to the CONSOLE view
        aRQ=new("pvxrequestclient",reqSocket$)
        aRQ'print("Hello, world ... in console view")
        drop object aRQ
    }
return 0

end
```