Önálló labor

Modellezés és analízis IBM Tivoli-val

összefoglaló, 1. félév

készítette: Présinger Zsolt
konzulens: dr Pataricza András

A feladat:

Megismerni az IBM Tivoli rendszer felügyeleti eszköz működését, üzemeltetését. Különös tekintettel a „Performance and Availability”, vagyis a teljesítmény és rendelkezésre állást vizsgáló komponensekre. Ezek közül legfontosabb a Monitoring, ami események gyűjtésére, és azok központi adatbázisába helyezésére alkalmas. Az adatbázisban később adatbányászkodni lehet, vizsgálva az események közötti összefüggéseket, előrejelezni esetleges jövőbeni eseményeket.

Miért jó a feladat?

A rendszerfelügyelet jelentős mértékű pénzt spórolhat meg üzleti és ipari vállalatoknak egyaránt az által, hogy a kiépített számítógépes infrastruktúrában keletkező hibákat, vagy potenciálisan hibához vezető állapotokat felismeri, és így időben javíthatóvá teszi. Biztonságkritikus rendszerek esetén ez esetleges baleset elkerülését is lehetővé teszi.
Továbbá a számítógépes rendszerek tervezhetősége -teljesítményre és rendelkezésre állásra- nagyon fontos dolog, mind máig pontosan nem megoldott probléma. Egy jól megtervezett rendszerben kevesebb a hibalehetőség, és stabilabb a működés.

Az első félév során elvégzett munka:

A félév elején letöltöttem az IBM szerveréről a Tivoli Enterprise Console-t és a Netview komponenst. A Tivoli installálásához hiányzott még ezen kívül a Tivoli Management Framework, ami az egész rendszer keretrendszere. Ez nem volt letölthető az IBM honlapjáról.
Így a félév eleje a honlapon található Redbook-ok tanulmányozásával telt. A Redbook-ok a http://www.redbooks.ibm.com/ internet címről érhetők el, ha a megjelenő oldalon a kereső mezőbe beírjuk, hogy „Tivoli”. Mivel ezen az oldalon rengeteg redbook található, így csak azokat olvastam, amik a Monitoring komponens működéséről szóltak.
A Management Framework beszerzésére november 8-án került sor, egy IBM konferencia alkalmával, ahol a Tivoli előadásokat tartó Tivoli szakember segített nekem a CD-k megszerzésében. Ezek után otthon installáltam egyetlen gépre a Framework-ot, ennek segítségével telepítettem az Enterprise Console-t, ami a Performance and Availability komponensek működését támogatja. Ekkor derült ki, hogy a Monitoring komponensek külön CD csomagot alkotnak, amik szintén nem tölthetők le az IBM oldaláról, és a beszerzésük a mai napig nem fejeződött be, bár folyamatban van.
Továbbá az Enterprise Console-nak szükséges adatbázis szerverrel való együttműködés még megoldandó probléma, mert a console-t nem engedi kapcsolódni a szerver. Valószínűleg az MSSQL adatbázis szerverrel való próbálkozást abba hagyom, és a szintén támogatott DB2 vagy Oracle szerverrel próbálkozok tovább.
A Monitoring CD-kből egyetlen apró kis programocska install-ja van meg, ez pedig a Workbench. Ezzel lehet tervezni az erőforrás modelleket, amiket a Monitoring használ a felügyelt gépekre telepítve. A félév hátralévő részében ezen Workbench-hez mellékelt minta modell-scripteket tanulmányoztam. A Workbench képes futtatni egy modellt, ami a működés során eseményeket generál a Monitoring számára, az adott gép erőforrásainak működése alapján. Ezen scriptek működését megértettem, de még nem tudom integrálni őket a rendszerbe, a Monitoring CD-k hiányában. Ezek a CD-k remélhetőleg hamarosan rendelkezésemre fognak állni.

Eredmények

A Tivoliról általában:

A Tivoli keretrendszer alapú, integrált rendszer-felügyeleti szoftver. A keretrendszernek köszönhetően könnyedén tud kezelni sokfajta platformot. Objektum orientált architektúrájú, és a rendszer részei Simple Network Managment Protocol (SNMP) vagy Common Object Request Broker Archictecture (CORBA) szabvány szerint kommunikálnak.
A Tivoli összes termékportfóliója 4 alapvető csoportba sorolható:

- Üzemeltetés és konfigurálás
- Teljesítmény és rendelkezésre állás (real time és historikus felügyelet)
- Tároló megoldások
- Biztonsági megoldások

Ezek közül az első kettő az amelyekbe az utóbbi időben beleintegrálódott az utolsó 2, és kialakították a PACO (*Performance and Availability, Coniguration and Operations*) portfóliót.
A rendszer a jó skálázhatóság érdekében un Management Agent struktúra szerint telepíthető sok gépes hálózatra. A struktúra hierarchia szintjei (egy fát alkot):

- Endpoint (futtatás, események gyűjtése, monitorozás, nem kezel adatbázist, sok van belőle)
- Gateway (kommunikációs szerep, összefog több endpoint-ot az Endpoint Manager fele)
- Endpoint Manager (általában egyetlen gép, ahonnan irányítani, felügyelni lehet a rendszert)

A keretrendszerre épülnek a modulok. Vannak Alkalmazási modulok, ilyen pl az Enterprise Console vagy a Monitoring, ezek szintén épülhetnek egymásra, vagyis az egyik működéséhez szükség van a másikra. Illetve vannak még Toolkitek, amikkel felhasználó barát módon menedzselni lehet a rendszert, illetve tervezni/módosítani a működését.

Performance and Availability:

Ennek segítségével optimalizálható a teljesítmény és rendelkezésre állás. A termékcsoport 3 részből áll:

- Monitoring Systems and Applications: A rendszer erőforrások és a taszkok monitorozása
- Event Correlation and Automation: Események közötti összefüggések felismerése és automatizálás
- Business Impact Management: Megmutatja hogy egy esemény milyen hatással van az egész rendszerre, kívülről nézve.

Monitoring:

Komplex rendszereknél monitorozás nélkül nagyon bonyolult lenne fenntartani a teljesítményt és rendelkezésre állást a kritikus szervereknél.
Az üzemen kívül töltött idő erős kihatással van egy e-Business vállalat vagy gyár működésére:

- Negatívan befolyásolja a jövedelmet és profitot
- Gátolja a vevők megszerzését és megtartását
- Károsítja egy márka hírnevét, cég jó hírét
- A kimaradó szolgáltatás hátrányos helyzetet eredményez

A termékcsoport Monitoring komponense adja a támogatást az Endpointok monitorozhatóságához. Mind real-time, mind historikus módon. Ez a szoftver használja az Enterprise console-t, és annak adatbázisát is. (*A Monitorozás: A monitorozó figyelmeztető jelzéseket generál minden esetben,*

amikor a teljesítmény egy bizonyos szint alá csökken. A jelzéseket egy központi adatbázisba is elküldi további feldolgozásra, és ha szükséges értesíti a rendszergazdát.)
Event Correlation and Automation:
Az akár több 1000 bekövetkező esemény közül, amit az egyes monitorozó komponensek továbbítottak az adatbázis szervernek, meg kell tudni állapítani, hogy mi a probléma gyökere, honnan indult a hiba. Továbbá az automatikusan kijavítható hibákat lehetőség szerint kijavítja a rendszer, ehhez meg kell adni tipikus hiba helyzeteket, és a kijavításukhoz szükséges lépéseket.

Telepítés:

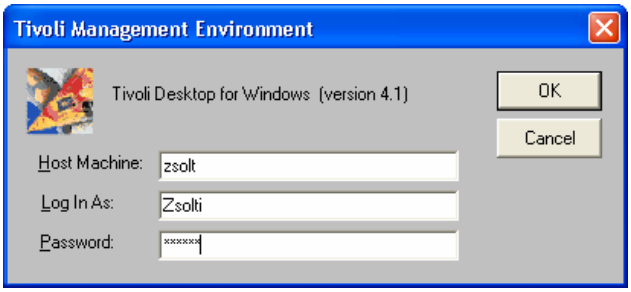
A félév folyamán a telepítés egyetlen gépre történt meg, amin egy Windows XP rendszer futott. A több gépes telepítés a jövő félévre tervezett a laborban.

A telepítés lépései (áttekintés):

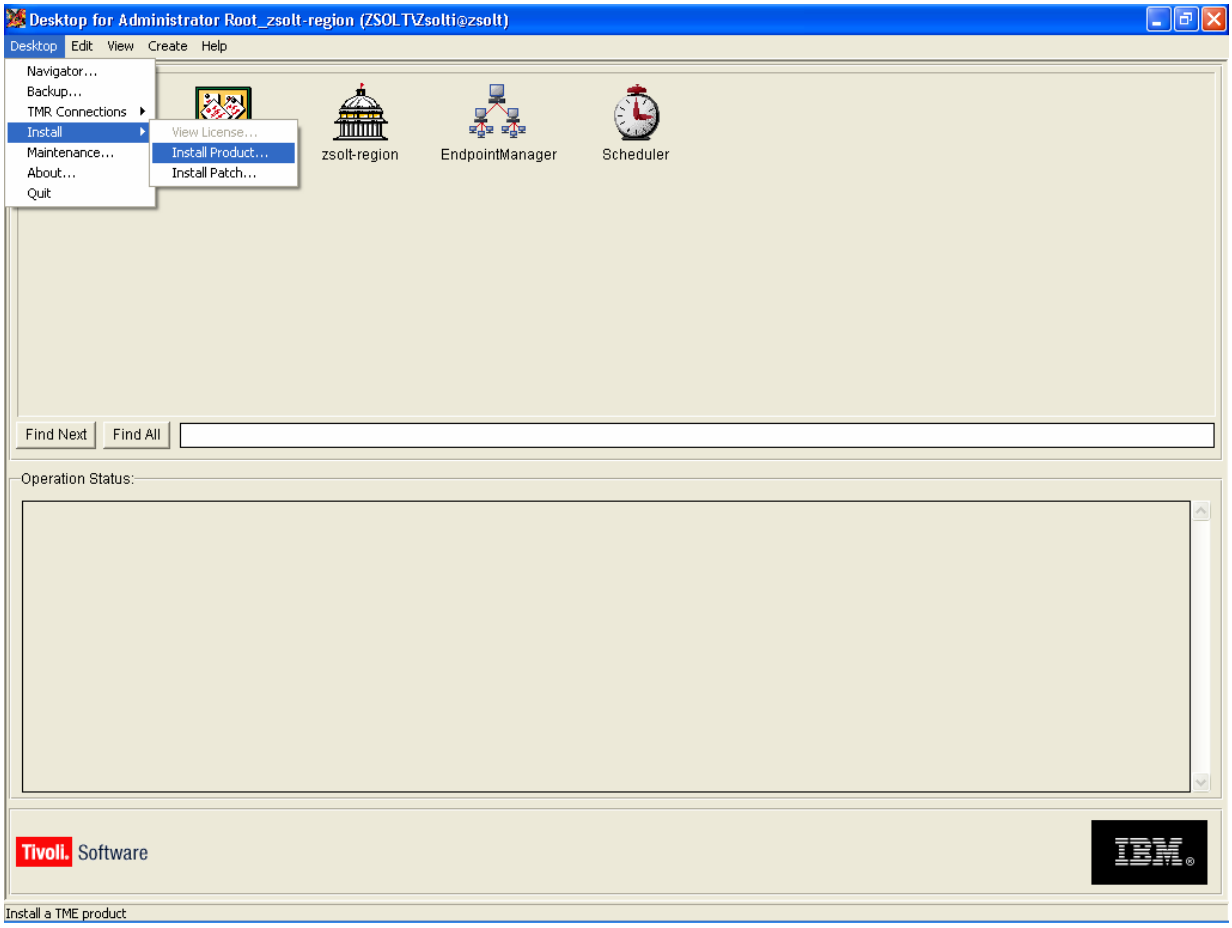
1. Kijelölni azokat a gépeket, amelyket szeretnénk a Tivoli Management Enviroment (TME) részévé tenni
2. Az előbb kijelölt gépekre telepíteni kell a Tivoli Management Framework (TMF) programot, ami támogatni fogja a Tivoli többi komponensének működését
3. Kijelölni azokat a gépeket, ahonnan elérhetővé akarjuk tenni az Endpoint Managert, akár rendszergazdai funkciókkal. Ezekre a gépekre telepíteni kell a Tivoli Desktop-ot, ami tulajdonképpen az Endpoint Manager adminisztrációs kliense
4. Ezek után telepítendő egy adatbázis szerver, ami majd a következő lépésben telepítendő Console Server által lesz használatba véve. A jelentősebb adatbázis kezelő szerverek támogatva vannak.
5. A Tivoli Enterprise Console (továbbiakban: TEC) telepítése. Ezzel a folyamattal kezdődően az összes tivoli komponens telepítése a Tivoli Desktopon keresztül történik.
6. A Monitoring komponensek telepítése.

A telepítés lépései (részletezve):

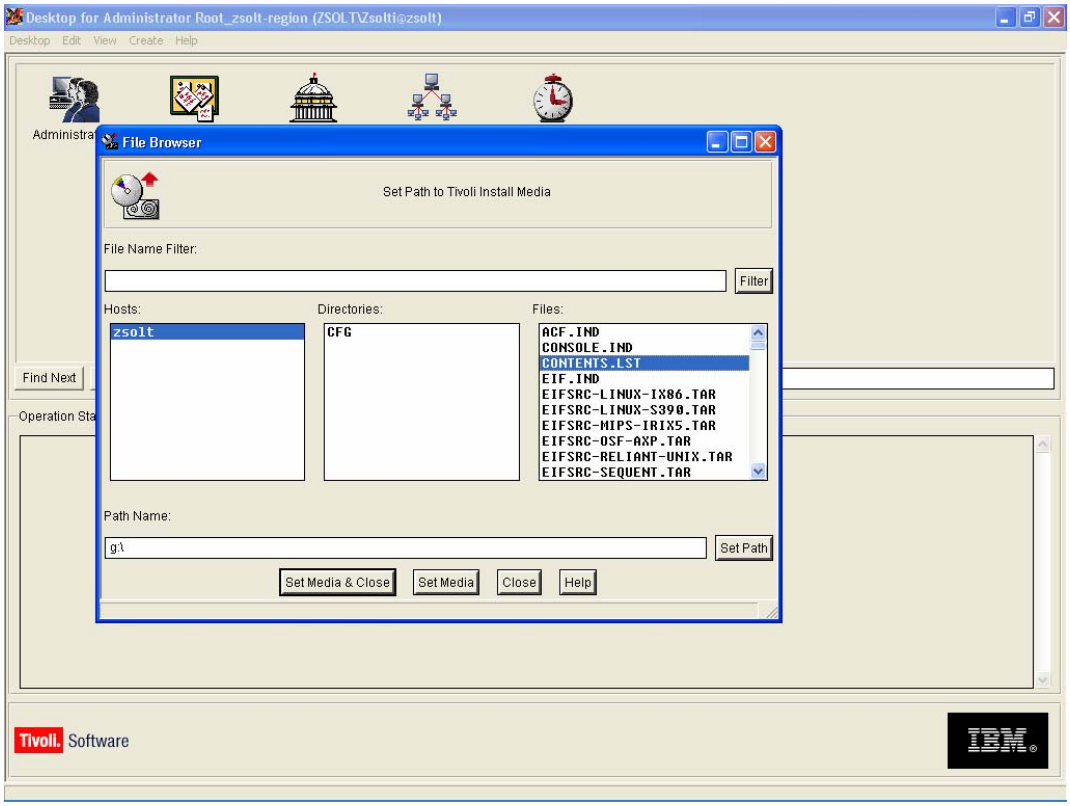
1. A vállalat nem mindegyik gépe kell, hogy a TME része legyen. Hiszen nem minden gép kritikus működésű, és nem minden gép esetében van értelme a Tivoli telepítésének. Ennek meghatározása a rendszergazda dolga. Általában azomban a legtöbb gépen megtörténik a telepítés
2. A telepítés egyszerűen a setup.exe program segítségével pár kattintással véghez vihető, de a telepítés végén mindenképpen szükséges a számítógép újraindítása, ez később nem pótolható művelet.
3. Ez általában csak néhány gépet jelent. Vállalati környezetben pl a rendszergazda gépe(i). Ennek telepítése is hasonlóan egyszerű mint a TMF telepítése.
4. Az adatbázis szerver ebben a félévben MS SQL volt. Ennek telepítése nagyon egyszerű, a telepítő varázsló végig vitelét jelenti. Bármely gépre telepíthető, nem muszáj a TME része legyen, csak elérhető kell legyen a TME-ből. A szerver telepítése után létre kell hozni egy TEC nevű adatbázist, és egy „tec” nevű felhasználót akinek teljes jogosultsága van a TEC adatbázishoz. Ezzel fog kapcsolódni a TEC szerver az adatbázishoz.
5. Ehhez be kell lépni a Tivoli Desktopon keresztül a TMF szerverére (TMR). A belépéskor kérni fogja a rendszer a jelszót, és azt hogy milyen host-on van a TMR. A felhasználónév és jelszó a TMR-t telepítő felhasználó neve és a jelszó az ő jelszava. Vigyázat, itt minden case-sensitive!



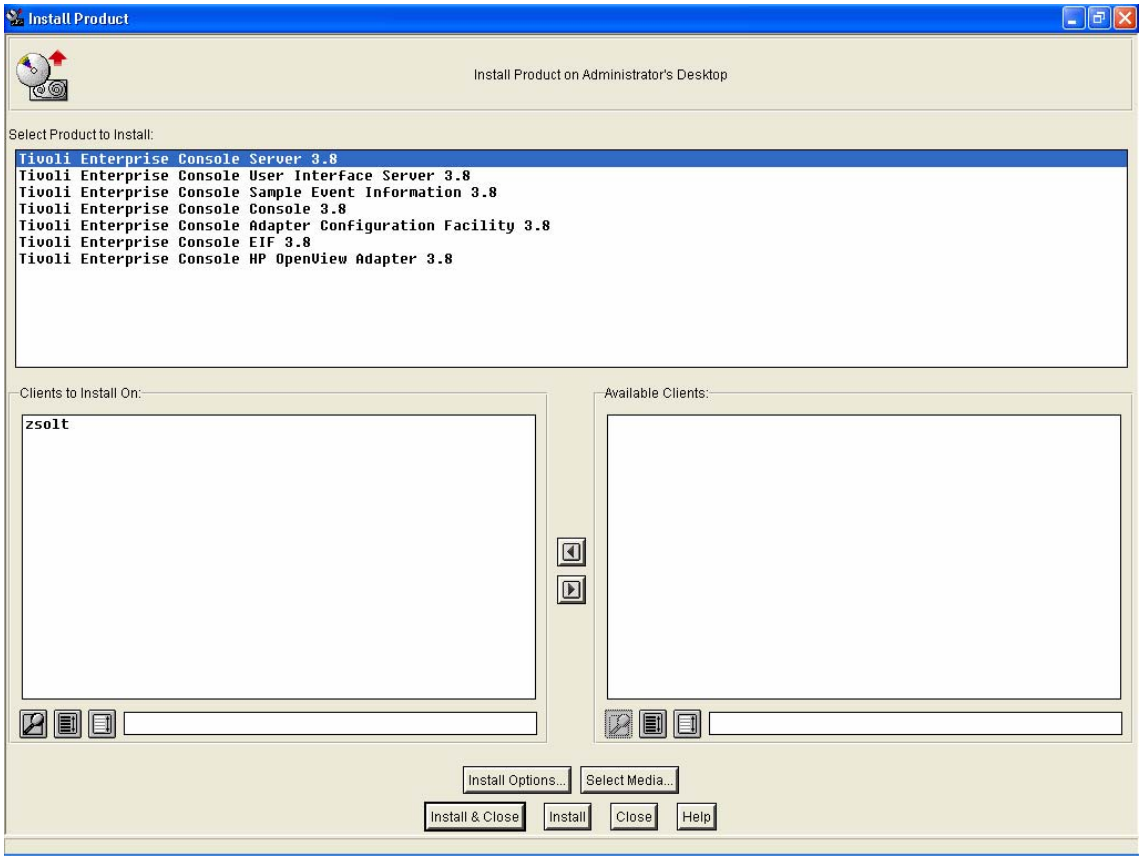
Ezután az elén ugró ablakban az alább látható menüpontot kell választanunk a TEC installálásához:

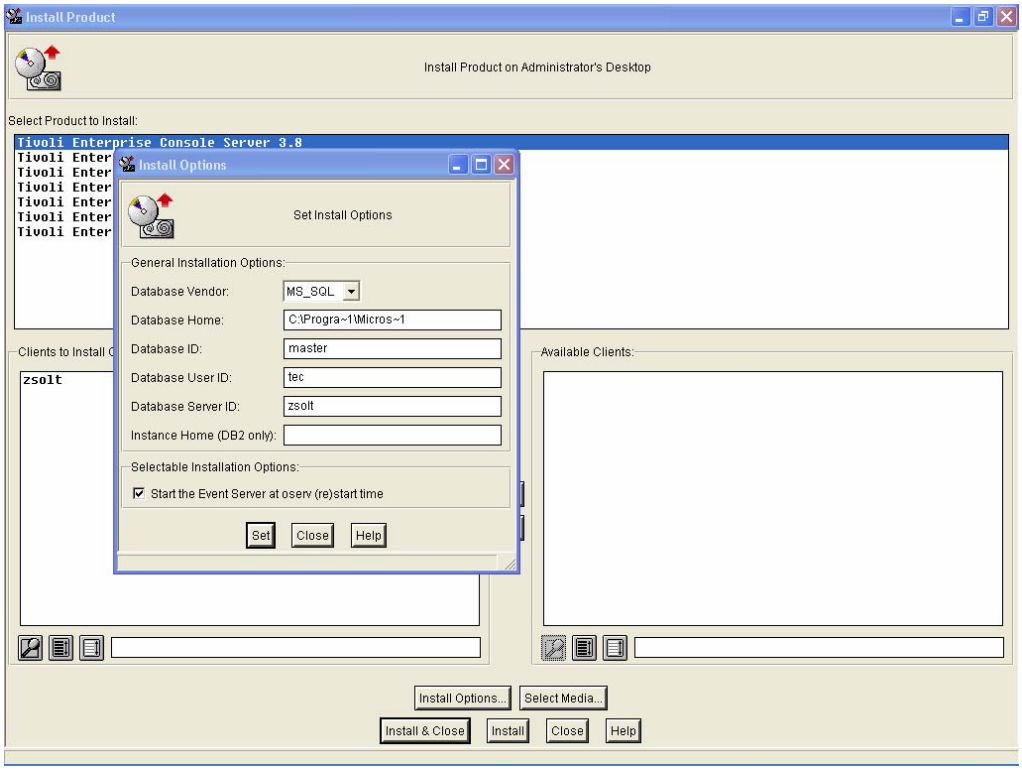


Aztán a TEC telepítő CD-jéről ki kell választani a COTENTS.LST-t:

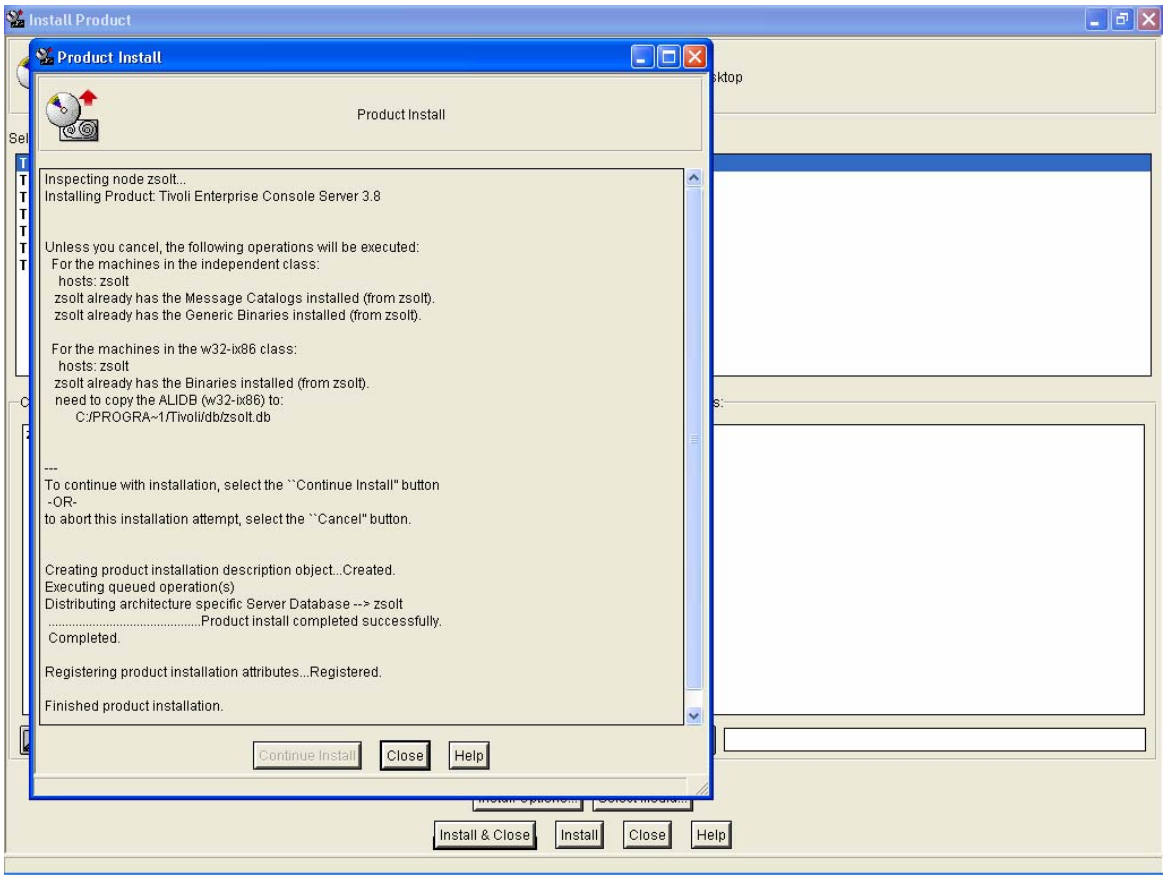


Aztán megjelenik egy ablak, ahonnan ki kell választani mit akarok telepíteni a CD-ről, amjd megjelenik az install options ablak. A TEC szerver telepítésénél meg kell adni az adatbázis szerver elérhetőségét.





És az install folyamat:

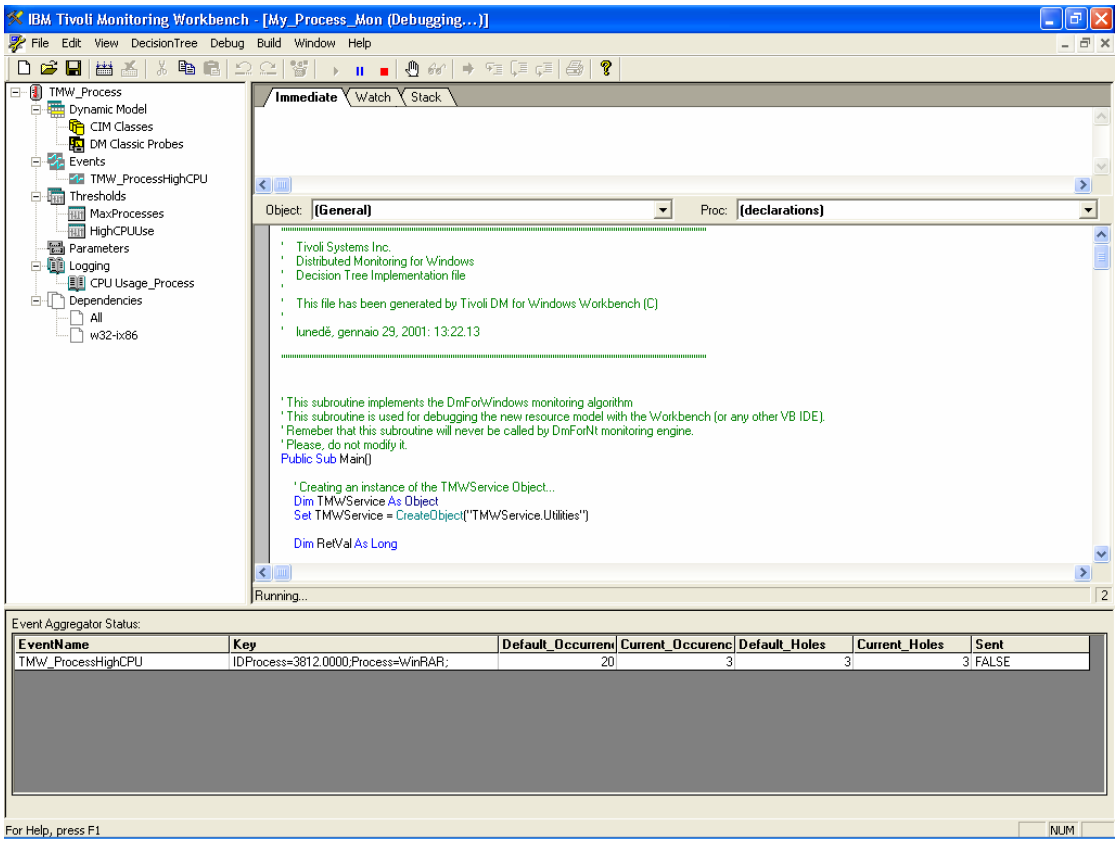


A TEC telepítésének befejezéséhez a CONTENTS.LST összes elemét telepíteni kell. Ezeknél az instal options tartalmát nem kell változtatni.

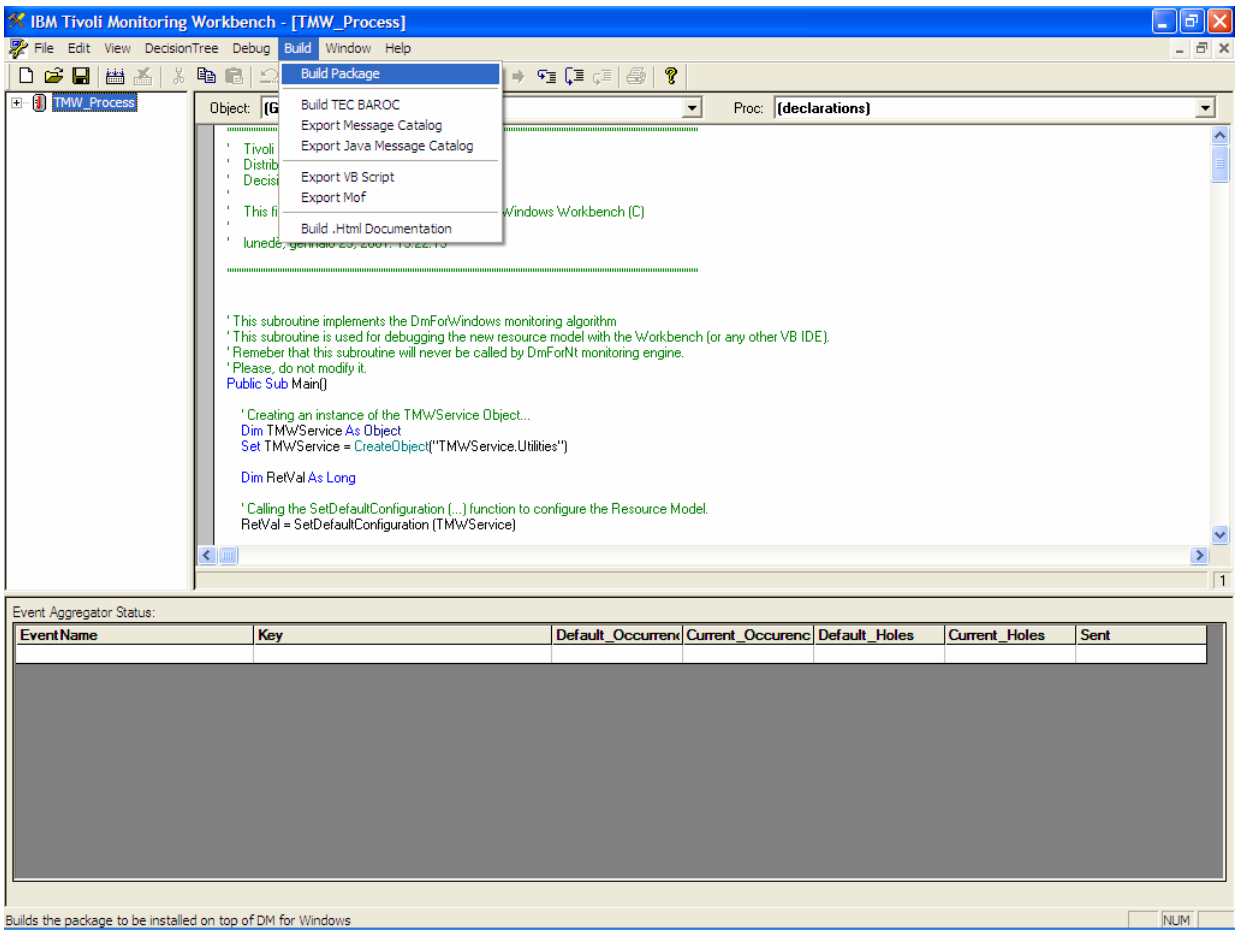
A Monitoring Workbench:

Ez egy Toolkit, aminek segítségével erőforrásokat modellező scripteket lehet írni. Ezen scriptek vagy VBA (csak Windows-on) vagy Java nyeléven írhatók, de ajánlott a varázslók használata, amik nagyrészt generálják a kódot.

A felhasználó felület:



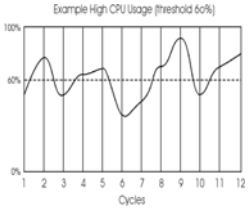
Ha az itt megszerkesztett resoucre modellt be akarjuk üzemelni a monitoringhoz, akkor az alábbi menüpontban:



lévő Bulid Package és Bulid Baroc file menüpontot választva létre kell hozni ezt a 2 fájlt. A package egy tar fájlt hoz létre, amit a Monitoring segítségével telepíteni lehet az Endpointokon, a „baroc" fájlok pedig a Monitoring szerver számára értelmezhetővé teszi a modell által küldött eseményeket.

Fogalmak (ezekre rákérdez a varázsló a modell készítésekor):

- Cycle time: Az az intervallum, amilyen időközönként egy modell futás közben snapshot-ot csinál az általa figyelt erőforrások állapotáról. Minden modell rendelkezik egy ilyen default érteékkel, ami módosítható szükség szerint
- Thresholds: Minden modell rendelkezik egy vagy több ilyennel, megmondja, mire reagál a modell. Pl:
  - High CPU Usage = 60%
  - Maximum Processes = 5
  - Low Disk Space threshold = 5%
- Minden ciklusban a modell küld információt arról, hogy
  - Occurrence (threshold esemény történt) vagy Hole (nem történt)
  - Occurrence Count (hányszor következett be a küszöb átlépése)
  -

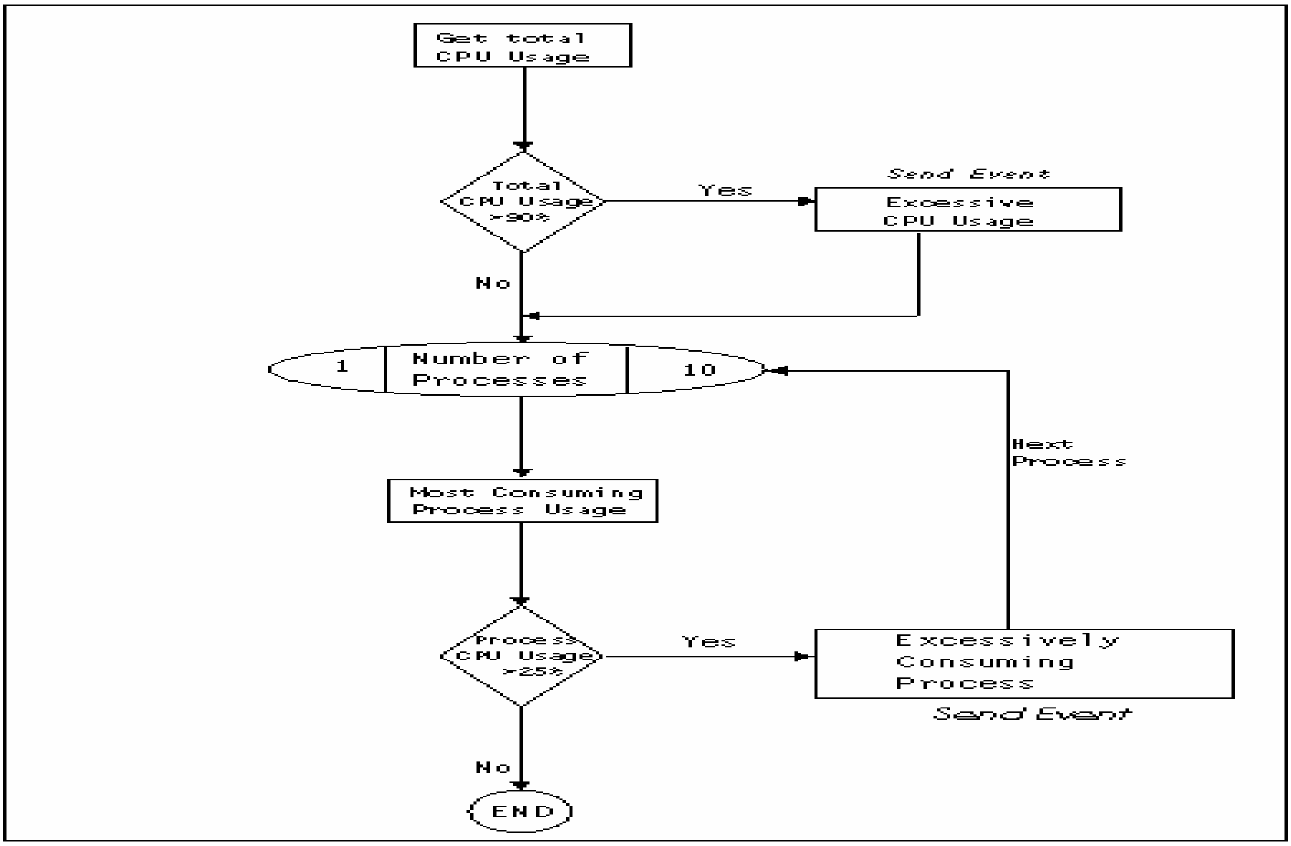


Data Collection: (adatgyűjtés)

- Két módja van:
  - Synchronous Data Collection: a begyűjtött adat továbbításra kerül, amint lehetséges, vagyis a következő ciklus végén
  - Asynchronous Data Collection: Elkezdődik egy ciklusban, és továbbnyúlik tetszőleges számú további ciklusba is, a váltás közben az összegyűjtött adatok cache-ben tárolódnak. Ez aszinkron forrásokhoz való, mint pl a Windows NT EventLog, és csak Windows platformokon támogatott, UNIX alatt nem.

Egy script, a processek CPU igényének monitorozásához VBA nyelven:

A folyamatábra:



A script:

```
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
'    Tivoli Systems Inc.
'    Distributed Monitoring for Windows
'    Decision Tree Implementation file
'
'    This file has been generated by Tivoli DM for Windows Workbench (C)
'
'    lunedě, gennaio 29, 2001: 13:22.13

''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''


' This subroutine implements the DmForWindows monitoring algorithm
```

```
' This subroutine is used for debugging the new resource model with the Workbench (or any other VB
IDE).
' Remeber that this subroutine will never be called by DmForNt monitoring engine.
' Please, do not modify it.
Public Sub Main()

	' Creating an instance of the TMWService Object...
	Dim TMWService As Object
	Set TMWService = CreateObject("TMWService.Utilities")

	Dim RetVal As Long

	' Calling the SetDefaultConfiguration (...) function to configure the Resource Model.
	RetVal = SetDefaultConfiguration (TMWService)

	If (RetVal = 0) Then

		' At this point DmForWindows Analyzer will override default settings.

		' Calling Init (...).
		RetVal = Init (TMWService)

		If (RetVal = 0) Then

			' Entering the monitoring loop...
			' It will loop forever.
			Do While (RetVal = 0)

				' Collect Data
				TMWService.CollectData

				' Visit the decision tree
				RetVal = VisitTree(TMWService)

				If (RetVal = 0) Then

					' Wait for a cycle time
					TMWService.EndVisit
					Wait TMWService.GetCycleTime()

				End If 'VisitTree = 0

			Loop

		End If 'Init = 0

	End If 'SetDefaultConfiguration = 0

	' Clean up used resources
	TMWService.Dispose

End Sub


' This function is used to define the settings of the resource model
' It is called only once, when the resource model is started.
' Don't modify remarks containing tags like <<....>> ... <<\...>>
' You can write additional initializing code in this function if required.
Public Function SetDefaultConfiguration (Svc As Object) As Long

	' General info section
	'<<GENERAL_INFO>>
	Svc.SetModelName "TMW_Process"
	Svc.SetProfileName "71611953"
	Svc.SetCycleTime 60
	'<<\GENERAL_INFO>>

	' Thresholds section
	'<<THRESHOLDS_INFO>>
	Svc.DefineThreshold "MaxProcesses", 57
	Svc.DefineThreshold "MaxHandles", 300
	Svc.DefineThreshold "HighCPUUse", 60
	'<<\THRESHOLDS_INFO>>

	' Parameters section
	'<<PARAMETERS_INFO>>
	'<<\PARAMETERS_INFO>>

	' Dynamic model section
	'<<DATA_INFO>>
```

```
        '<<\DATA_INFO>>

        ' Event definition section
        '<<EVENTS_INFO>>
        Svc.DefineEvent                                                    "TMW_ProcessHighCPU",
"IDProcess,PrcPercentUserTime,PrcPercentPriviledgedTime,PrcPriorityBase", "Process"
        Svc.DefineEvent "TMW_ProcessHandleLeak", "CurrentHanProcessID,CurrentHandleCount", "Process"
        '<<\EVENTS_INFO>>

        ' Logging definition section
        '<<LOGGING_INFO>>
        Svc.DefineLogInst "Handle Usage", "Process", "Process,ID", "HandleCount,ID", "Process"
        Svc.DefineLogInst          "CPU          Usage",          "Process",          "Process,ID",
"PercentUserTime,PercentPrivilegedTime,PercentProcessorTime,ID", "Process"
        '<<\LOGGING_INFO>>

        ' Place your additional intializing code below


        SetDefaultConfiguration = 0

End Function


' This function is called by the DM For Nt Analyzer after that
' the resource model default settings have been overridden
' It is called only once, when the resource model is started.
' You can write additional initializing code in this function if required
' that need to use the thresholds and parameters values
Public Function Init(Svc As Object) As Long


    Svc.DefineClass "CIM", "TMW_Process", "ROOT\CIMV2:TMW_Process", _
    "WHERE   ((ID   <>   0)   And   (Process   <>   ""_Total"")   And   (PercentProcessorTime   >"   +
Str(Svc.GetThreshold("HighCPUUse")) + "))", _
    "ID, HandleCount, PercentPrivilegedTime, PercentProcessorTime, PercentUserTime, PriorityBase", _
    "Process", _
    "ASCENDING", "PercentProcessorTime", Svc.GetThreshold("MaxProcesses") , 1

    Svc.DefineClass "CIM","TMW_ProcessHandles", "ROOT\CIMV2:TMW_Process", _
    "WHERE   ((ID   <>   0)   And   (Process   <>   ""_Total"")   And   (HandleCount   >   "   +
Str(Svc.GetThreshold("MaxHandles")) + "))", _
    "ID, HandleCount, PercentPrivilegedTime, PercentProcessorTime, PercentUserTime, PriorityBase", _
    "Process", _
    "ASCENDING", "HandleCount", Svc.GetThreshold("MaxProcesses") , 1

       Init = 0

End Function



' This function contains the monitoring algorithm
' It is called ciclically after a cycle time has elapsed
' Implement the the monitoring code here
Public Function VisitTree(Svc As Object) As Long

        Dim i As Long, numTotalProcesses As Long, numPrcPercentPrivTime As Long
    Dim k As Long, numIDProcess As Long
    Dim numPrcPercentUserTime As Long, numPrcPercentProcessorTime As Long, numPrcPriorityBase As
Long
    Dim numPrcPercentPriviledgedTime As Long
    Dim strProcess As String
    Dim found As Boolean
    Dim numTotalHandles_dinamyc As Long
    Static NumSlot(1 To 100) As Long
    Static IdSlot(1 To 100) As Long
    Static numTotalHandles As Long

    Static numCurrentCount As Long
    Static idCurrentID As Long
    Static flagNotFirstRun As Integer
    'I've changed the logic of the previous flag (before there was flagFirstRun)

    i = 0

    numTotalProcesses = Svc.GETNUMOFINST("TMW_Process")

    Do While i < numTotalProcesses
        'Need to check for a process that's hogging up the CPU
```

```
        'prcProcess = GETINST(TMW_Process, i)
        numPrcPercentProcessorTime = Svc.GetNumProperty("TMW_Process", i, "PercentProcessorTime")

        'I've commented out the next "if" because I put it in the "WHERE" clause of
        'the class definition
        'If numPrcPercentProcessorTime > Svc.GetThreshold("HighCPUUse") Then

            'send EVENT with prcid, prcname, percent usage, priority  base, %user and %priv
            'this information will help us determine what process is slamming the cpu and what
            'we can infer FROM the process.  IF the %user is high and %priv is low, we can assume
it'
            'an app running in the user space where the consiquences of killing the app arn't as
high
            'as it might IF it ran in the privlidged space.  Also, IF the priority base is high, we
            'can assume that killing the app is not a good solution and simply lowering the priority
            'may be a better soluiton IF that's possible

            strProcess = Svc.GetStrProperty("TMW_Process", i, "Process")
            numIDProcess = Svc.GetNumProperty("TMW_Process", i, "ID")
            numPrcPercentUserTime = Svc.GetNumProperty("TMW_Process", i, "PercentUserTime")
            numPrcPercentPriviledgedTime        =        Svc.GetNumProperty("TMW_Process",        i,
"PercentPrivilegedTime")
            numPrcPriorityBase = Svc.GetNumProperty("TMW_Process", i, "PriorityBase")

                     'Log Instance Data
            Svc.LogInst "CPU Usage", "Process", numPrcPercentUserTime, numPrcPercentPriviledgedTime,
numPrcPercentProcessorTime, numIDProcess, strProcess


            'send EVENT

            Svc.SENDEVENT "TMW_ProcessHighCPU", _
            numIDProcess, _
            numPrcPercentUserTime, _
            numPrcPercentPriviledgedTime, _
            numPrcPriorityBase, _
            strProcess
        'Else
            'i = numTotalProcesses
        'End If

            i = i + 1
    Loop 'end of the while loop for i



    'Check for handle leaks
    'We have to assume that the process with the highest number of handles will be the leaker. IF it
exist
    'in later revions, the USE of variable array types will help define this method and pinpoint
leaking
    'processes faster.

    'We'll use the same check method that was used in the memory model


        numTotalHandles_dinamyc = Svc.GETNUMOFINST("TMW_ProcessHandles")
        If flagNotFirstRun = 1 Then
            i = 0
            Do While (i < numTotalHandles_dinamyc)
                numCurrentCount = Svc.GetNumProperty("TMW_ProcessHandles", i, "HandleCount")
                idCurrentID = Svc.GetNumProperty("TMW_ProcessHandles", i, "ID")
                strProcess = Svc.GetStrProperty("TMW_ProcessHandles", i, "Process")

               'Log Instance Data
                Svc.LogInst "Handle Usage", "Process", numCurrentCount, idCurrentID, strProcess

                k = 1
                found = True
                Do While (k <= numTotalHandles)
                    If idCurrentID = IdSlot(k) Then
                        If numCurrentCount > NumSlot(k) Then
                            k = numTotalHandles
                            Svc.SENDEVENT "TMW_ProcessHandleLeak", _
                            idCurrentID, _
                            numCurrentCount, _
                            strProcess
                        End If
                    End If
                    k = k + 1
```

```
                Loop
                    i = i + 1
            Loop

        End If


        numTotalHandles = Svc.GETNUMOFINST("TMW_ProcessHandles")
        If numTotalHandles > 100 Then
               numTotalHandles = 100
        End If

        j = 0
        Do While (j < numTotalHandles)
            NumSlot(j+1) = Svc.GetNumProperty("TMW_ProcessHandles", j, "HandleCount")
            IdSlot(j+1) = Svc.GetNumProperty("TMW_ProcessHandles", j, "ID")
            j = j + 1
        Loop

        flagNotFirstRun = 1
      'End If

       VisitTree = 0

End Function
```