Operations Manager e il controllo della configurazione

#### di [Daniele Grandini](https://mvp.support.microsoft.com/profile=58922C7A-F40F-438B-93D1-7E4E3CF963AD) – Microsoft MVP ([blog](http://nocentdocent.wordpress.com))

* 1. 

*Maggio 2012*

## Introduzione

Il controllo della configurazione dei sistemi sotto monitor non è uno dei principali compiti di Operations Manager. All’interno di System Center, Configuration Manager e Service Manager, magari con il supporto di Orchestrator sono gli strumenti di elezione per realizzare un monitor della configurazione. Non sempre è però possibile implementare tutti i componenti di System Center. Questo articolo ha lo scopo di sviluppare un Management Pack che permetta di verificare i sistemi che sono conformi in termini di sistema operativo, service pack e hotfix installate.

Sebbene i moduli sviluppati saranno generici ed applicabili a diversi contesti, la realizzazione ha lo scopo di individuare i sistemi che non abbiano tutti i prerequisiti necessari al corretto funzionamento dell’agente Operations Manager. Si tratta di tutte quelle fix, esterne a Operations Manager, che fanno una drammatica differenza rispetto alla qualità e all’affidabilità del monitor. Per un elenco sempre aggiornato si può fare riferimento all’articolo “[Things to make and do for agent health](http://nocentdocent.wordpress.com/articles-and-series/things-to-make-and-do-for-agent-health/)”.

Ci addentreremo dunque nello sviluppo di un management pack che dovrà rispondere alle seguenti richieste:

* + Controllo del livello di service pack dato uno specifico sistema operativo
  + Controllo della presenza di una lista di patch dato uno specifico sistema operativo e livello di service pack
  + Opzionalmente dovrà integrarsi nell’health model dell’agente Operations Manager (HealthService class)
  + Dovrà segnalare in modo dettagliato le ragioni per cui un sistema è ritenuto non compliant (livello di service pack, quali patch sono mancanti e così via)

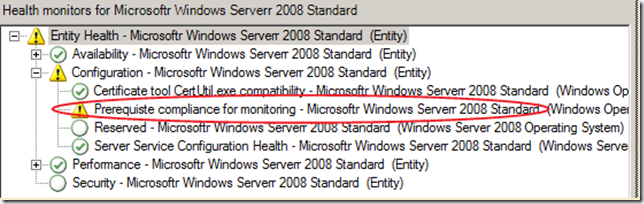
L’articolo presuppone che il lettore abbia familiarità con Operations Manager e il modello di monitor da esso sotteso.

Per ragioni di brevità e per non confondere il lettore con troppi dettagli mi concentrerò su Windows Server 2008 e Windows Server 2008 R2, ma la medesima tecnica può essere applicata a sistemi Windows 2003 o se per questo anche a sistemi operativi client controllati con Operations Manager. Questa è la ragione per cui, pur concentrandomi su questi sistemi operativi userò come linguaggio di scripting vbscript invece di powershell. L’utilizzo di moduli powershell è tipicamente più efficiente in quanto non è necessario uno spawn di un processo esterno come invece avviene con vbscript.

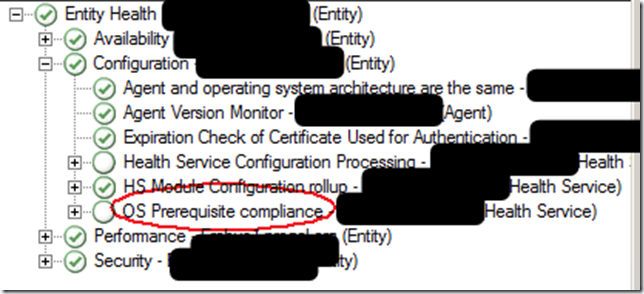
Non tratterò della parte di reporting, eventualmente lasciata d un articolo specifico.

## Health model

L’health model per la compliance deve integrarsi con quello del sistema operativo preso in esame, tutto questo è logico in quanto i parametri di configurazione che andremo a controllare sono specifici del sistema operativo. Trattandosi di configurazione la sua collocazione è semplice:

* 1. 
  2. Fig. 1 – Health model

Dal momento che, in questo specifico caso, la presenza o assenza di queste fix ha una ripercussione sul funzionamento dell’agente Operations Manager è possibile pensare di integrare anche l’health model dell’agente (classe HealthService). Il management pack definirà questa relazione, ma la lascerà disabilitata. Sarà possibile attivarla tramite uno specifico override.

* 1. 
  2. Fig. 2 – Integrazione con l’health model dell’HealthService

## Stabilire una relazione tra classi

Per ottenere questo il primo passaggio è definire una relazione, la più debole possibile, tra le classi che rappresentano il sistema operativo (Microsoft.Windows.OperatingSystem) e l’agente (Microsoft.SystemCenter.HealthService). Il tipo di relazione più debole a nostra disposizione è la System.Reference che non implica null’altro che una generica relazione tra le due classi. Questa relazione deve essere definita nella sezione dedicata ai tipi. Attenzione alla direzione della relazione, dal momento che sarà l’HealthService a dover “ereditare” lo stato di salute di un componente di sistema operativo, la relazione avrà questa direzione HealthService -> Operating System :

* 1. XML
  2. <TypeDefinitions>
  3. <EntityTypes>
  4. <RelationshipTypes>
  5. <RelationshipType ID="QND.Compliance.HealthServiceReferenceOperatingSystem" Accessibility="Internal" Abstract="false" Base="System!System.Reference">
  6. <Source>SC!Microsoft.SystemCenter.HealthService</Source>
  7. <Target>Windows!Microsoft.Windows.OperatingSystem</Target>
  8. </RelationshipType>
  9. </RelationshipTypes>
  10. </EntityTypes>
  11. </TypeDefinitions>

Definita la relazione sarà necessario definire un tipo particolare di monitor, ma lo vedremo a tempo opportuno, il punto da ricordare è che senza questa relazione non è possibile definire il monitor necessario alla rappresentazione di cui sopra.

Un ultima nota di attenzione per la proprietà Accessibility, come vedete l’hop definita “Internal” questo significa che la relazione sarà utilizzabile unicamente all’interno di questo Management Pack, se ritenete la relazione di utilità generale cambiate il codice per definirla “Public”.

## Baseline vecchie e nuove

Rimane un ultimo aspetto da indirizzare, ossia la gestione delle baseline. Quello che si vuole avere è fondamentalmente un controllo del livello di patching, ovviamente il livello deve essere relativo ad un punto di partenza o baseline, ai fini di questo articolo le baseline prese in considerazione saranno le seguenti:

* + Windows Server 2008
  + Windows Server 2008 R2 RTM
  + Windows Server 2008 R2 Service Pack 1

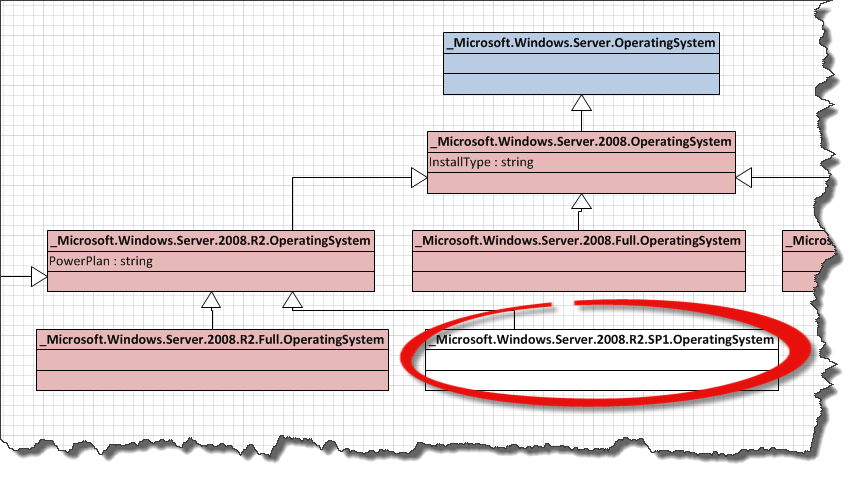
Nel codice mostrerò come definire una nuova baseline tramite una regola di discovery apposita per Windows Server 2008 R2 Service Pack 1. Ma dal momento che stiamo definendo i tipi che ci serviranno per lo sviluppo di questo Management Pack iniziamo con il definire la nuova classe per Windows 2008 R2 con Service Pack 1 (Microsoft.Windows.Server.2008.R2.SP1.OperatingSystem).

Il compito sembra semplice, ma la ricchezza del modello ci mette in realtà a disposizione diverse possibilità, soprattutto in termini di quale classe utilizzare come base per derivare la nostra. Possiamo restringere le scelte a due:

* + Windows 2008 R2 SP1 è un sistema operativo generico, nel qual caso lo deriviamo dalla classe Microsoft.Windows.OperatingSystem
  + Windows 2008 R2 SP1 è un Windows 2008 R2, in questo caso lo deriviamo da Microsoft.Windows.Server.2008.R2.OperatingSystem

La questione, come vedremo, non è di lana caprina perché ci sono implicazioni per entrambe le scelte. Nel mio caso ho preferito seguire il pattern impostato da Microsoft nel suo management pack per Windows 2008, dove Windows 2008 R2 è un Windows 2008, quindi deriva da Microsoft.Windows.Server.2008.OperatingSystem.

Come sempre un’immagine vale più di mille parole:

* 1. 

Come anticipato, questa scelta ha le sue implicazioni ed effetti collaterali. La prima implicazione è che tutte le regole di monitor che hanno come target Windows 2008 saranno applicate anche a Windows 2008 R2 e tutte quelle con target R2 saranno ereditate da Windows 2008 R2 SP1 (comprese quelle per Windows 2008). Cioè, come ci aspettiamo, le regole seguono l’ereditarietà delle classi. Per chi è nuovo nello sviluppo di Management Pack basta pensare alle regole come ai metodi delle classi, le regole vengono ereditate così come i metodi.

Bene, e quindi? Dal momento che, come vedremo, il nostro monitor per il check di compliance sarà genericamente applicato al Microsoft.Windows.Server.OperatingSystem, avremo un effetto collaterale che io chiamo di “convergenza lenta”, quello che accadrà per un nuovo sistema Windows 2008 R2 Service Pack 1 sarà:

* + Prima di tutto sarà indviduato come un Windows 2008 R2 e la nostra regola di compliance per quella baseline sarà applicata ad esso, questo causerà una non compliance in quanto le patch da applicare sono diverse tra Windows 2008 R2 e Windows 2008 R2 SP1 (altrimenti non avremmo avuto bisogno di differenziare le baseline). Il management pack per Windows 2008 ha un’unica regola di discovery che ritorna il sistema operativo corretto (Windows 2008 o 2008 R2), quindi questo effetto di convergenza lenta si ha solo tra R2 e R2 con SP1.
  + Quindi il sistema sarà classificato come Windows 2008 R2 SP1, a quel punto la regola più specifica che si applica a Windows 2008 R2 con SP1 sarà applicata ritornando l’effettivo e corretto stato di compliance

Penso che questa “convergenza lenta” sia accettabile, l’alternativa è derivare la classe Windows 2008 R2 SP1 da Windows.Server.OperatingSystem e riscrivere la regola di discovery che a questo punto dovrà essere più articolata.

Tornando al codice la definizione della nuova classe sarà piuttosto semplice:

* 1. XML
  2. <ClassTypes>
  3. <ClassType ID="Microsoft.Windows.Server.2008.R2.SP1.OperatingSystem" Abstract="false" Accessibility="Public" Base="Win2008!Microsoft.Windows.Server.2008.R2.OperatingSystem" Hosted="true" Singleton="false">
  4. </ClassType>
  5. </ClassTypes>

Non preoccupatevi se il nostro Management Pack non prende ancora una sua forma definitiva, la strada è ancora lunga e ci vuole pazienza. Per il momento abbiamo definito l’health model e le nuove erlazioni e classi necessarie.

## Lo script di monitor

Prima di addentrarci ancora nella struttura del management pack, voglio concentrarmi sullo script che dovrà fornire le informazioni necessarie al controllo della baseline. Come anticipato, per ragioni di supporto di Windows 2003, lo script sarà in vbscript. Per migliorarne la leggibilità ho rimosso tutti i controlli di errore e il logging diagnostico non indispensabili, in questo modo la logica con cui ritornare i dati a Operations Manager dovrebbe essere evidente. Questo implica però che prima di utilizzare lo script in un ambiente di produzione occorre introdurre un accurato controllo di errore.

Le specifiche sono le seguenti:

* + L’output deve poter essere utilizzato sia per il controllo della compliance sia per ritornare le informazioni su quali componenti sono eventualmente fuori compliance
  + Lo script deve usare interfacce che permettano il suo utilizzo su Windows 2003, 2008 e 2008 R2
  + Lo script dovrà controllare la compliance di una baseline data, per dirla con Operations Manager il target sarà una specifica baseline
  + La lista delle patch deve essere dinamica in modo da potersi adattare alle singole baseline e alle esigenze dei singoli sistemi (tramite override)
  + Lo script dovrà verificare la versione del sistema operativo, la versione di Windows Scripting Host e la presenza delle patch appropriate

Da queste specifiche derivano i parametri in input allo script:

* + Versione del sistema operativo desiderata (per esempio per Windows 2008 la service pack richiesta è la service pack 2)
  + Versione di Windows Scripting Host necessaria (tipicamente 5.7, se il parametro è impostato a IGNORE il check viene saltato)
  + Elenco separato da virgole delle patch richieste, anche in questo caso se il parametro vale IGNORE il controllo viene saltato

L’output dello script sarà una property bag che dovrà riportare dati di sintesi rispetto alla compliance e dati descrittivi da utilizzare per fornire maggiori informazioni agli operatori. I dati sintetici di tipo boolean saranno:

* + OS compliance
  + Patches compliance
  + WSH compliance

Le proprietà descrittive saranno:

* + Versione del sistema operativo
  + Versione di Windows Scripting Host
  + Una entry per ogni patch in input che riporti se la patch è presente oppure no
  + Una descrizione da utilizzare direttamente nell’alert che indichi la ragione della non compliance

Tutto questo premesso, la logica dello script è articolata su 5 passaggi:

* 1. Carica tutte le patch all’interno di un dizionario, al termine dei check il dizionario conterrà, per ogni voce, la sua presenza o assenza e sarà la base per la property bag ritornata
  2. Controlla la versione dell’OS tramite la classe WMI Win32\_OperatingSystem
  3. Controlla la versione di windows scripting host utilizzando la proprietà Version, se il parametro in input è “IGNORE” lo script assume che windows scripting host sia compliant
  4. Controlla la presenza delle patch utilizzando la classe WMI Win32\_QuickFixEngineering. Se date una scorsa al codice e ai commenti vedrete che prima viene cercato un match esatto e in assenza di questo vengono cercate eventuali versioni successive della fix usando il postfisso \_v<n>
  5. In ultimo tramite le informazioni raccolte viene costruita la property bag e assemblata la stringa descrittiva da usare negli alert
  6. Visual Basic
  7. Option Explicit
  8. SetLocale ("en-us")
  9. 'Globals
  10. Dim g\_API, g\_oXML
  11. Dim g\_StdErr
  12. Dim P\_TraceLevel, P\_OSVer, P\_QFEList, P\_WSHVersion
  13. 'On Error Resume Next
  14. Dim dtStart, oArgs
  15. Dim dicQFEs, qfe, oQFEs
  16. Dim i, oWMI, oWMIOS, OSVersion, Keys
  17. Dim oBag, bOSCompliance, os, bQFECompliance, bWshCompliance
  18. dtStart = Now
  19. Globals
  20. Set oArgs = WScript.Arguments
  21. P\_OSVer = oArgs(0)
  22. if UCASE(TRIM(oArgs(1)))="IGNORE" Then
  23. P\_QFEList = Split("", ",")
  24. else
  25. P\_QFEList = Split(oArgs(1),",")
  26. end if
  27. P\_WSHVersion = oArgs(2)
  28. Set dicQFEs = CreateObject("Scripting.Dictionary")
  29. dicQFEs.CompareMode=vbTextCompare
  30. for I=0 to ubound(P\_QFEList)
  31. dicQFEs.Add P\_QFEList(i), false
  32. next
  33. 'Now get the info we need via WMI
  34. Set oWMI = GetObject("winmgmts:\\.\root\cimv2")
  35. Set oWMIOS = oWMI.ExecQuery("Select Version from Win32\_OperatingSystem")
  36. for each os in oWMIOS
  37. OSVersion = os.Version
  38. next
  39. bOSCompliance = (OSVersion >= P\_OSVer)
  40. bWshCompliance = (WScript.Version >= P\_WSHVersion or UCASE(P\_WSHVersion)="IGNORE")
  41. Set oQFEs = oWMI.ExecQuery("Select \* from Win32\_QuickFixEngineering")
  42. for each qfe in oQFEs
  43. 'sadly I cannot use an exact match to take into account the KB123456-v? fix naming
  44. if (dicQFEs.Exists(qfe.HotFixID)) Then
  45. dicQFEs(qfe.HotFixID) = true
  46. else
  47. 'if I have not an exact match then check for patch versioning
  48. keys = dicQFEs.Keys
  49. for I=0 to dicQFEs.count -1
  50. if UCASE(keys(I) & "-v") = UCASE(Left(qfe.HotFixID, Len(keys(I)&"-v"))) then
  51. dicQFEs(keys(I)) = true
  52. exit for
  53. end if
  54. next
  55. end if
  56. next
  57. 'Now prepare the data to return
  58. Dim sMessage
  59. If Not bOSCompliance Then
  60. sMessage = sMessage & "OS is not compliant required version: " & P\_OSVer & " Detected Version: " & OSVersion & vbCrLf
  61. end if
  62. If Not bWshCompliance Then
  63. sMessage = sMessage & "WSH is not compliant required version: " & P\_WSHVersion & " Detected Version: " & WScript.Version & vbCrLf
  64. end if
  65. Set oBag = g\_API.CreateTypedPropertyBag(StateDataType)
  66. call oBag.AddValue("OSVersion", OSVersion)
  67. call oBag.AddValue("OSCompliant", bOSCompliance)
  68. bQFECompliance = True
  69. for each qfe in dicQFEs.Keys
  70. if false = dicQFEs(qfe) Then
  71. bQFECompliance = false
  72. sMessage = sMessage & qfe & " is missing" & vbCrLf
  73. end if
  74. call oBag.AddValue(qfe, dicQFEs(qfe))
  75. next
  76. call oBag.AddValue("QFECompliant", bQFECompliance)
  77. call oBag.AddValue("WSHCompliant", bWSHCompliance)
  78. call oBag.AddValue("Message", sMessage)
  79. 'finally resturn the property bag
  80. g\_API.AddItem oBag
  81. Call g\_API.ReturnItems
  82. ClearGlobals
  83. '\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*
  84. '\*\*\*\* HELPER FUNCTIONS SCOM sempre necessarie
  85. '\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*
  86. Sub Globals
  87. P\_TraceLevel = TRACE\_VERBOSE
  88. Set g\_API = CreateObject("MOM.ScriptAPI")
  89. Set g\_oXML = CreateObject("MSXML.DOMDocument")
  90. Set g\_StdErr = WScript.StdErr
  91. end Sub
  92. Sub ClearGlobals
  93. Set g\_API = Nothing
  94. Set g\_oXML = Nothing
  95. End Sub

## Definizione dei tipi di moduli

Ora che abbiamo lo script occorre utilizzarlo all’interno del management pack. Anche in questo caso ci sono diverse possibilità, ma per favorire la riusabilità dei componenti nei miei management pack utilizzo sempre uno schema tipo. Dal momento che il nostro fine ultimo sarà quello di costruire un monitor che verifichi la compliance di un sistema ecco il pattern che seguiremo

Percorso per la definizione di un monitor

* 1. Definire un ProbeModuleType
  2. Nella composition definire il modulo opportuno per il tipo di provider dati preparato (il nostro script)
  3. Definire un DataSourceModuleType
  4. Nella composition, oltre ai moduli aggiuntivi necessari (tipicamente uno scheduler), utilizzare il probe type definito al punto 1.
  5. Definire uno UnitMonitorType che utilizzerà il data source definito in 3.
  6. In ultimo definire uno UnitMonitor basato sul monitor type di cui al punto 5

Prima di procedere oltre facciamo un rapido elenco dei moduli che dovremo utilizzare nel nostro management pack, procedendo a ritroso:

* + Per il monitor di compliance avremo necessità di un **monitor type**, a sua volta questo necessiterà di un **datasource type** che, come da paragrafo precedente, scegliamo di comporre con un **probe type**
  + Per il discovery dei sistemi Windows 2008 R2 SP1 avremo bisogno di un data source, ma in questo caso utilizzando il registry per il discovery potremo usare un modulo standard senza doverne definire uno nostro
  + Per il discovery della relazione tra l’HealthService e il sistema operativo avremo invece bisogno di un **datasource type**

Ricapitolando dovremo definire i seguenti moduli:

* + Probe type per utilizzare lo script **QND.OSQFEGet.PT**
  + Data source type per il monitor di compliance **QND.OSQFEGet.DS**
  + Monitor type per la valutazione della compliance **QND.OSCompliance.MT**
  + Data source per il discovery della relazione tra l’HealthService e il sistema operativo **QND.Compliance.HSRefOS.DT**

## Il datasource per il monitor

Iniziamo con il definire il probe type, I management pack hanno una sezione apposite per la definizione di nuovi moduli <ModuleTypes> a loro volta contenuti nella sezione <TypeDefinitions> che abbiamo già usato per definire la classe Windows 2008 R2 SP1 e la relazione tra HealthService e sistema operativo.

Il probe utilizzerà un unico membro nella sua composizione uno ScriptPropertyBagProbe, la configurazione esporrà il parametro ScriptTimeout necessario al ScriptPropertyBagProbe e i parametri che abbiamo previsto per il nostro script:

* + Versione del sistema operativo desiderata - OSVersion
  + Versione di Windows Scripting Host necessaria - WSHVersion
  + Elenco separato da virgole delle patch richieste – QFEList

Non è necessario definire override perchè non useremo questo modulo direttamente nella sezione di moitoring, ma ovviamente farlo non ha implicazioni negative:

Un'altra accortezza che facilita e di molto il troubleshooting e il test è quello di includere tutti gli script in una sezione CDATA, in modo da evitare l’escaping HTML altrimenti necessario. I management Pack sono documenti XML e quindi non è possibile utilizzare caratteri riservati se non all’interno di sezioni CDATA.

Tutto questo è una mia convenzione, ma non è strettamente richiesto: potete tranquillamente fare l’embedding dello script direttamente all’interno del DataSourceModuleType facendo escaping dello script stesso. A voi la scelta.

* 1. XML
  2. <ModuleTypes>
  3. <ProbeActionModuleType ID="QND.OSQFEGet.PT" Accessibility="Public" Batching="false">
  4. <Configuration>
  5. <xsd:element name="OSVersion" type="xsd:string" />
  6. <xsd:element name="QFEList" type="xsd:string" />
  7. <xsd:element name="WSHVersion" type="xsd:string" />
  8. <xsd:element name="ScriptTimeout" type="xsd:integer" />
  9. </Configuration>
  10. <ModuleImplementation Isolation="Any">
  11. <Composite>
  12. <MemberModules>
  13. <ProbeAction ID="GetData" TypeID="Windows!Microsoft.Windows.ScriptPropertyBagProbe">
  14. <ScriptName>QND.GetOSCompliance.vbs</ScriptName>
  15. <Arguments>$Config/OSVersion$ "$Config/QFEList$" $Config/WSHVersion$</Arguments>
  16. <ScriptBody>
  17. <![CDATA[Inserire lo script qui]]>
  18. </ScriptBody>
  19. <TimeoutSeconds>$Config/ScriptTimeout$</TimeoutSeconds>
  20. </ProbeAction>
  21. </MemberModules>
  22. <Composition>
  23. <Node ID="GetData"/>
  24. </Composition>
  25. </Composite>
  26. </ModuleImplementation>
  27. <OutputType>System!System.PropertyBagData</OutputType>
  28. <InputType>System!System.BaseData</InputType>
  29. </ProbeActionModuleType>
  30. </ModuleTypes>

Costruito il probe, la definizione del nostro DataSource è questione di comporre il probe con un opportuno scheduler che ci permetta di eseguire lo script ad intervalli prestabiliti, volendo anche con la possibilità di sincronia ad un determinato orario. Come ho anticipato il data source così definito sarà poi utilizzato nella definizione del monitor.

Il data source dovrà essere aggiunto alla sezione ModuleTypes \*prima\* del ProbeActionModuleType.

I parametri del data source esporranno tutti quelli del probe e in aggiunta l’intervallo di esecuzione (IntervalSeconds) e l’eventuale orario di sincronizzazione (SyncTime) usati dal System.Scheduler. Anche in questo caso non è strettamente necessario definire i parametri overrideable.

* 1. XML
  2. <DataSourceModuleType ID="QND.OSQFEGet.DS" Accessibility="Public" Batching="false">
  3. <Configuration>
  4. <xsd:element name="OSVersion" type="xsd:string"/>
  5. <xsd:element name="QFEList" type="xsd:string" />
  6. <xsd:element name="WSHVersion" type="xsd:string" />
  7. <xsd:element name="ScriptTimeout" type="xsd:integer" />
  8. <xsd:element name="IntervalSeconds" type="xsd:integer" />
  9. <xsd:element name="SyncTime" type="xsd:string" />
  10. </Configuration>
  11. <ModuleImplementation>
  12. <Composite>
  13. <MemberModules>
  14. <DataSource ID="Scheduler" TypeID="System!System.Scheduler">
  15. <Scheduler>
  16. <SimpleReccuringSchedule>
  17. <Interval>$Config/IntervalSeconds$</Interval>
  18. <SyncTime>$Config/SyncTime$</SyncTime>
  19. </SimpleReccuringSchedule>
  20. <ExcludeDates />
  21. </Scheduler>
  22. </DataSource>
  23. <ProbeAction ID="GetData" TypeID="QND.OSQFEGet.PT">
  24. <OSVersion>$Config/OSVersion$</OSVersion>
  25. <QFEList>$Config/QFEList$</QFEList>
  26. <WSHVersion>$Config/WSHVersion$</WSHVersion>
  27. <ScriptTimeout>$Config/ScriptTimeout$</ScriptTimeout>
  28. </ProbeAction>
  29. </MemberModules>
  30. <Composition>
  31. <Node ID="GetData">
  32. <Node ID="Scheduler"/>
  33. </Node>
  34. </Composition>
  35. </Composite>
  36. </ModuleImplementation>
  37. <OutputType>System!System.PropertyBagData</OutputType>
  38. </DataSourceModuleType>

## Il monitor type

Il data source appena creato è funzionale alla definizione del nostro monitor type che a sua volta è funzionale al cuore di questo management pack, ossia il controllo della compliance dei sistemi. Se vi siete persi tornate al paragrafo iniziale per la definizione dei tipi di moduli.

Per questo monitor saranno sufficienti due stati, molto semplicemente “Compliant” e “NonCompliant”

* 1. XML
  2. <MonitorTypeStates>
  3. <MonitorTypeState ID="Compliant" NoDetection="false" />
  4. <MonitorTypeState ID="NonCompliant" NoDetection="false" />
  5. </MonitorTypeStates>

Questo stati vengono individuati in base ai valori ritornati dal data source, se ricordate il nostro script ritorna tre valori di tipo boolean:

* + OSCompliant
  + QFECompliant
  + WSHCompliant

Perchè il sistema sia compliant tutti e tre le proprietà devono essere vere, basta quindi definire una condition detection che metta in AND le tre proprietà, secondo il seguente pattern:

* 1. XML
  2. <Expression>
  3. <And>
  4. <Expression OS/>
  5. <Expression WSH/>
  6. <Expression QFE/>
  7. </And>
  8. </Expression>

Dove <Expression OS/WHS/QFE> seguono il seguente schema:

* 1. XML
  2. <Expression>
  3. <SimpleExpression>
  4. <ValueExpression>
  5. <XPathQuery Type="Boolean">Property[@Name="OSCompliant"]</XPathQuery>
  6. </ValueExpression>
  7. <Operator>Equal</Operator>
  8. <ValueExpression>
  9. <Value Type="Boolean">true</Value>
  10. </ValueExpression>
  11. </SimpleExpression>
  12. </Expression>

Il controllo di non compliance è simile ponendo in OR le tre variabili testate sul valore false.

Dal momento che questo modulo sarà utilizzato nella monitoring section è giunto il momento di definire i parametri overridable, mentre i parametri di configurazione saranno identici a quelli del data source, detto questo si tratta solo di comporre l’xml

* 1. XML
  2. <MonitorTypes>
  3. <UnitMonitorType ID="QND.OSCompliance.MT" Accessibility="Public">
  4. <MonitorTypeStates>
  5. <MonitorTypeState ID="Compliant" NoDetection="false" />
  6. <MonitorTypeState ID="NonCompliant" NoDetection="false" />
  7. </MonitorTypeStates>
  8. <Configuration>
  9. <xsd:element name="OSVersion" type="xsd:string"/>
  10. <xsd:element name="QFEList" type="xsd:string" />
  11. <xsd:element name="WSHVersion" type="xsd:string" />
  12. <xsd:element name="ScriptTimeout" type="xsd:integer" />
  13. <xsd:element name="IntervalSeconds" type="xsd:integer" />
  14. <xsd:element name="SyncTime" type="xsd:string" />
  15. </Configuration>
  16. <OverrideableParameters>
  17. <OverrideableParameter ID="IntervalSeconds" Selector="$Config/IntervalSeconds$" ParameterType="int" />
  18. <OverrideableParameter ID="SyncTime" Selector="$Config/SyncTime$" ParameterType="string" />
  19. <OverrideableParameter ID="OSVersion" Selector="$Config/OSVersion$" ParameterType="string" />
  20. <OverrideableParameter ID="QFEList" Selector="$Config/QFEList$" ParameterType="string" />
  21. <OverrideableParameter ID="WSHVersion" Selector="$Config/WSHVersion$" ParameterType="string" />
  22. <OverrideableParameter ID="ScriptTimeout" Selector="$Config/ScriptTimeout$" ParameterType="int" />
  23. </OverrideableParameters>
  24. <MonitorImplementation>
  25. <MemberModules>
  26. <DataSource ID="DS" TypeID="QND.OSQFEGet.DS">
  27. <OSVersion>$Config/OSVersion$</OSVersion>
  28. <QFEList>$Config/QFEList$</QFEList>
  29. <WSHVersion>$Config/WSHVersion$</WSHVersion>
  30. <ScriptTimeout>$Config/ScriptTimeout$</ScriptTimeout>
  31. <IntervalSeconds>$Config/IntervalSeconds$</IntervalSeconds>
  32. <SyncTime>$Config/SyncTime$</SyncTime>
  33. </DataSource>
  34. <ConditionDetection ID="CDCompliant" TypeID="System!System.ExpressionFilter">
  35. <Expression>
  36. <And>
  37. <Expression>
  38. <SimpleExpression>
  39. <ValueExpression>
  40. <XPathQuery Type="Boolean">Property[@Name="OSCompliant"]</XPathQuery>
  41. </ValueExpression>
  42. <Operator>Equal</Operator>
  43. <ValueExpression>
  44. <Value Type="Boolean">true</Value>
  45. </ValueExpression>
  46. </SimpleExpression>
  47. </Expression>
  48. <Expression>
  49. <SimpleExpression>
  50. <ValueExpression>
  51. <XPathQuery Type="Boolean">Property[@Name="QFECompliant"]</XPathQuery>
  52. </ValueExpression>
  53. <Operator>Equal</Operator>
  54. <ValueExpression>
  55. <Value Type="Boolean">true</Value>
  56. </ValueExpression>
  57. </SimpleExpression>
  58. </Expression>
  59. <Expression>
  60. <SimpleExpression>
  61. <ValueExpression>
  62. <XPathQuery Type="Boolean">Property[@Name="WSHCompliant"]</XPathQuery>
  63. </ValueExpression>
  64. <Operator>Equal</Operator>
  65. <ValueExpression>
  66. <Value Type="Boolean">true</Value>
  67. </ValueExpression>
  68. </SimpleExpression>
  69. </Expression>
  70. </And>
  71. </Expression>
  72. </ConditionDetection>
  73. <ConditionDetection ID="CDNonCompliant" TypeID="System!System.ExpressionFilter">
  74. <Expression>
  75. <Or>
  76. <Expression>
  77. <SimpleExpression>
  78. <ValueExpression>
  79. <XPathQuery Type="Boolean">Property[@Name="OSCompliant"]</XPathQuery>
  80. </ValueExpression>
  81. <Operator>Equal</Operator>
  82. <ValueExpression>
  83. <Value Type="Boolean">false</Value>
  84. </ValueExpression>
  85. </SimpleExpression>
  86. </Expression>
  87. <Expression>
  88. <SimpleExpression>
  89. <ValueExpression>
  90. <XPathQuery Type="Boolean">Property[@Name="QFECompliant"]</XPathQuery>
  91. </ValueExpression>
  92. <Operator>Equal</Operator>
  93. <ValueExpression>
  94. <Value Type="Boolean">false</Value>
  95. </ValueExpression>
  96. </SimpleExpression>
  97. </Expression>
  98. <Expression>
  99. <SimpleExpression>
  100. <ValueExpression>
  101. <XPathQuery Type="Boolean">Property[@Name="WSHCompliant"]</XPathQuery>
  102. </ValueExpression>
  103. <Operator>Equal</Operator>
  104. <ValueExpression>
  105. <Value Type="Boolean">false</Value>
  106. </ValueExpression>
  107. </SimpleExpression>
  108. </Expression>
  109. </Or>
  110. </Expression>
  111. </ConditionDetection>
  112. </MemberModules>
  113. <RegularDetections>
  114. <RegularDetection MonitorTypeStateID="Compliant">
  115. <Node ID="CDCompliant">
  116. <Node ID="DS" />
  117. </Node>
  118. </RegularDetection>
  119. <RegularDetection MonitorTypeStateID="NonCompliant">
  120. <Node ID="CDNonCompliant">
  121. <Node ID="DS" />
  122. </Node>
  123. </RegularDetection>
  124. </RegularDetections>
  125. </MonitorImplementation>
  126. </UnitMonitorType>
  127. </MonitorTypes>

## Il data source per il discovery della relazione

Classi e relazioni definite hanno necessità di un metodo per poter essere instanziate, ossia per poter determinare quali e quante istanze di queste classi o relazioni siano presenti nell’ambiente di cui effettuare il monitor. Per meglio intenderci: definire una classe Windows 2008 R2 Service Pack 1 significa solamente predisporre il sistema a ricevere oggetti di quella classe. L’individuazione degli oggetti è lasciata a regole di discovery.

Per l’individuazione della relazione tra HealthService e sistema operativo, di primo acchito, avevo pensato di scrivere uno script apposito, ma spulciando nella documentazione dei moduli disponibili ne ho trovato uno che poteva fare al caso nostro: [System.Discovery.RelationshipSnapshotDataMapper](http://msdn.microsoft.com/en-us/library/ee692973.aspx).

In verità la documentazione è piuttosto scarna e non ha esempi calzanti al nostro caso, ma verificando l’implementazione del modulo non è stato difficile trovare la sintassi corretta. Dal momento che questo modulo è una condition detection lo dovremo combinare con uno scheduler per ottenere il data source necessario al discovery della relazione. Come abbiamo già visto il data source esporrà i parametri dei moduli coinvolti e lascerà all’implementazione finale la loro valorizzazione. In questo caso la regola di discovery dovrà semplicemente indicare le proprietà chiave di Operating System e HealthService, lo vedremo in seguito, per il momento, con un atto di fiducia, teniamo questa semplice composition.

* 1. XML
  2. <DataSourceModuleType ID="QND.Compliance.HSRefOS.DT" Accessibility="Public" Batching="false">
  3. <Configuration>
  4. <IncludeSchemaTypes>
  5. <SchemaType>System!System.Discovery.MapperSchema</SchemaType>
  6. </IncludeSchemaTypes>
  7. <xsd:element minOccurs="1" name="Frequency" type="xsd:integer" />
  8. <xsd:element minOccurs="0" name="SyncTime" type="xsd:string" />
  9. <xsd:element name="RelationshipId" type="xsd:string" />
  10. <xsd:element name="SourceTypeId" minOccurs="0" maxOccurs="1" type="xsd:string" />
  11. <xsd:element name="TargetTypeId" minOccurs="0" maxOccurs="1" type="xsd:string" />
  12. <xsd:element name="SourceRoleSettings" type="SettingsType" />
  13. <xsd:element name="TargetRoleSettings" type="SettingsType" />
  14. <xsd:element name="InstanceSettings" minOccurs="0" maxOccurs="1" type="SettingsType" />
  15. </Configuration>
  16. <OverrideableParameters>
  17. <OverrideableParameter ID="Frequency" Selector="$Config/Frequency$" ParameterType="int" />
  18. <OverrideableParameter ID="SyncTime" Selector="$Config/SyncTime$" ParameterType="string" />
  19. </OverrideableParameters>
  20. <ModuleImplementation Isolation="Any">
  21. <Composite>
  22. <MemberModules>
  23. <DataSource ID="DS" TypeID="System!System.Discovery.Scheduler">
  24. <Scheduler>
  25. <SimpleReccuringSchedule>
  26. <Interval Unit="Seconds">$Config/Frequency$</Interval>
  27. <SyncTime>$Config/SyncTime$</SyncTime>
  28. </SimpleReccuringSchedule>
  29. <ExcludeDates />
  30. </Scheduler>
  31. </DataSource>
  32. <ConditionDetection ID="Mapping" TypeID="System!System.Discovery.RelationshipSnapshotDataMapper">
  33. <RelationshipId>$Config/RelationshipId$</RelationshipId>
  34. <SourceTypeId>$Config/SourceTypeId$</SourceTypeId>
  35. <TargetTypeId>$Config/TargetTypeId$</TargetTypeId>
  36. <SourceRoleSettings>$Config/SourceRoleSettings$</SourceRoleSettings>
  37. <TargetRoleSettings>$Config/TargetRoleSettings$</TargetRoleSettings>
  38. <InstanceSettings>$Config/InstanceSettings$</InstanceSettings>
  39. </ConditionDetection>
  40. </MemberModules>
  41. <Composition>
  42. <Node ID="Mapping">
  43. <Node ID="DS" />
  44. </Node>
  45. </Composition>
  46. </Composite>
  47. </ModuleImplementation>
  48. <OutputType>System!System.Discovery.Data</OutputType>
  49. </DataSourceModuleType>

## La struttura esterna del management pack

Prima di procedere oltre è giunto il tempo di definire lo scheletro del nostro management pack a partire con la definizione della sua identità, dare seguito con la sezione in cui definire tutte le dipendenze da management pack di sistema e finire l’importantissima sezione per le informazioni mostrate in console.

* 1. XML
  2. <?xml version="1.0" encoding="utf-8"?>
  3. <ManagementPack xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsl="http://www.w3.org/1999/XSL/Transform” xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  4. <Manifest>
  5. <Identity>
  6. <ID>QND.AgentCompliance</ID>
  7. <Version>6.1.7221.11</Version>
  8. </Identity>
  9. <Name>QND Agent Compliance</Name>
  10. <References>
  11. <Reference Alias="System">
  12. <ID>System.Library</ID>
  13. <Version>6.0.4941.0</Version>
  14. <PublicKeyToken>31bf3856ad364e35</PublicKeyToken>
  15. </Reference>
  16. <Reference Alias="Windows">
  17. <ID>Microsoft.Windows.Library</ID>
  18. <Version>6.0.4941.0</Version>
  19. <PublicKeyToken>31bf3856ad364e35</PublicKeyToken>
  20. </Reference>
  21. <Reference Alias="Win2003">
  22. <ID>Microsoft.Windows.Server.2003</ID>
  23. <Version>6.0.4941.0</Version>
  24. <PublicKeyToken>31bf3856ad364e35</PublicKeyToken>
  25. </Reference>
  26. <Reference Alias="Win2008">
  27. <ID>Microsoft.Windows.Server.2008.Discovery</ID>
  28. <Version>6.0.4941.0</Version>
  29. <PublicKeyToken>31bf3856ad364e35</PublicKeyToken>
  30. </Reference>
  31. <Reference Alias="SC">
  32. <ID>Microsoft.SystemCenter.Library</ID>
  33. <Version>6.0.4941.0</Version>
  34. <PublicKeyToken>31bf3856ad364e35</PublicKeyToken>
  35. </Reference>
  36. <Reference Alias="Health">
  37. <ID>System.Health.Library</ID>
  38. <Version>6.1.7043.0</Version>
  39. <PublicKeyToken>31bf3856ad364e35</PublicKeyToken>
  40. </Reference>
  41. </References>
  42. </Manifest>

Lo schema dei Management Pack prevede di concentrare tutte le stringhe in un’apposita sezione che supporta anche la localizzazione delle stesse. Dal momento che tutto il management pack è scritto in inglese anche le stringhe sono definite in inglese nel LanguagePack ENU, per chi voglila localizzarlo basta aggiungere un LanguagePack ITA con le medesime display string e ovviamente nome e descrizione tradotte in italiano. La definizione delle display string non è obbligatoria (a parte alcuni casi particolari come nel caso degli alert), ma rimane una pratica da perseguire per consentire agli operatori di ricevere informazioni intellegibili e non criptiche.

* 1. XML
  2. <LanguagePacks>
  3. <LanguagePack ID="ENU" IsDefault="true">
  4. <DisplayStrings>
  5. <DisplayString ElementID="QND.AgentCompliance">
  6. <Name>Quae Nocent Docent Agent Compliance</Name>
  7. <Description>Checks for OS compliance for monitoring</Description>
  8. </DisplayString>
  9. <DisplayString ElementID="QND.Compliance.HealthServiceReferenceOperatingSystem">
  10. <Name>HealthService references Operating System</Name>
  11. </DisplayString>
  12. <DisplayString ElementID="Microsoft.Windows.Server.2008.R2.SP1.OperatingSystem">
  13. <Name>Windows Server 2008 R2 Operating System with SP1</Name>
  14. </DisplayString>
  15. </DisplayStrings>
  16. </LanguagePack>
  17. </LanguagePacks>

Sempre nella sezione dei Language Pack è possibile definire articoli di Knowledge Base, anche qui non c’è nessun obbligo, ma è bene che almeno per il management pack stesso e per i monitor sia presente un articolo di Knowledge Base. Di seguito l’articolo per il management pack che illustra la sua funzione e le necessità di configurazione oltre che i link esterni per maggiori informazioni:

* 1. XML
  2. <KnowledgeArticle ElementID="QND.AgentCompliance" Visible="true">
  3. <MamlContent>
  4. <maml:section xmlns:maml="http://schemas.microsoft.com/maml/2004/10">
  5. <maml:title>Summary</maml:title>
  6. <maml:para>This set of rules checks for Operating System and Operations Manager Agent configuration compliance</maml:para>
  7. </maml:section>
  8. <maml:section xmlns:maml="http://schemas.microsoft.com/maml/2004/10">
  9. <maml:title>Configuration</maml:title>
  10. <maml:para>By default the non compliance status is not projected to the agent (health service) health state, to enable such a projection turn on the QND OS Prerequisite Compliance dependency monitor</maml:para>
  11. </maml:section>
  12. <maml:section xmlns:maml="http://schemas.microsoft.com/maml/2004/10">
  13. <maml:title>External</maml:title>
  14. <maml:para>
  15. <maml:navigationLink>
  16. <maml:linkText>Things to make and do for agent health</maml:linkText>
  17. <maml:uri href="http://nocentdocent.wordpress.com/articles-and-series/things-to-make-and-do-for-agent-health/" />
  18. </maml:navigationLink>
  19. </maml:para>
  20. </maml:section>
  21. </MamlContent>
  22. </KnowledgeArticle>

## La sezione di monitoring

Ora che le fondamenta sono gettate e che i moduli necessari sono stati sviluppati e inseriti nel Management Pack è ora di iniziare a costruire la parte di monitor vera e propria.

## Discovery

La nuova classe per Windows 2008 R2 Service Pack 1 e la definizione della relazione tra HealthService e sistema operativo devono trovare implementazione tramite la scrittura di apposite regole di discovery. Delle due la più interessante dal punto di vista formale è sicuramente quella che permette il discovery della relazione, mentre quella per Windows 2008 R2 SP1 è più comune e con diversi esempi nella documentazione.

Per il discovery della relationship useremo il data source definito appositamente, il target della rule è l’HealthService, il source della relazione, ma non fatevi confondere tutto questo è logico. Come vedete il datasource richiede il nome della relzione (QND.Compliance.HealthServiceReferenceOperatingSystem) e per le due entità coinvolte le proprietà definite come chiave nelle rispettive classi. Il nostro è un caso particolare perché per entrambe (HealthService e OperatingSystem) abbiamo un’unica proprietà chiave che è il nome del computer (PrincipalName).

* 1. XML
  2. <Discovery ID="QND.Compliance.HSRefOSDiscovery" Enabled="true" Target="SC!Microsoft.SystemCenter.HealthService" ConfirmDelivery="true" Remotable="true" Priority="Normal">
  3. <Category>Discovery</Category>
  4. <DiscoveryTypes>
  5. <DiscoveryRelationship TypeID="QND.Compliance.HealthServiceReferenceOperatingSystem" />
  6. </DiscoveryTypes>
  7. <DataSource ID="DS" TypeID="QND.Compliance.HSRefOS.DT">
  8. <Frequency>86400</Frequency>
  9. <SyncTime></SyncTime>
  10. <RelationshipId>$MPElement[Name="QND.Compliance.HealthServiceReferenceOperatingSystem"]$</RelationshipId>
  11. <SourceRoleSettings>
  12. <Settings>
  13. <Setting>
  14. <Name>$MPElement[Name="Windows!Microsoft.Windows.Computer"]/PrincipalName$</Name>
  15. <Value>$Target/Host/Property[Type="Windows!Microsoft.Windows.Computer"]/PrincipalName$</Value>
  16. </Setting>
  17. </Settings>
  18. </SourceRoleSettings>
  19. <TargetRoleSettings>
  20. <Settings>
  21. <Setting>
  22. <Name>$MPElement[Name="Windows!Microsoft.Windows.Computer"]/PrincipalName$</Name>
  23. <Value>$Target/Host/Property[Type="Windows!Microsoft.Windows.Computer"]/PrincipalName$</Value>
  24. </Setting>
  25. </Settings>
  26. </TargetRoleSettings>
  27. </DataSource>
  28. </Discovery>

Come anticipato per il discovery di Windows 2008 R2 SP1 useremo un semplice registry discovery verificando che la system build sia pari a 7601. Non aggiungo altro se non che di default questo discovery è disabilitato quindi se volete avere questa baseline dovrete abilitarlo.

* 1. XML
  2. <Discovery ID="QND.Compliance.Win2008R2SP1.Discovery" Enabled="false" Target="Win2008!Microsoft.Windows.Server.2008.R2.OperatingSystem" ConfirmDelivery="true" Remotable="true" Priority="Normal">
  3. <Category>Discovery</Category>
  4. <DiscoveryTypes>
  5. <DiscoveryClass TypeID="Microsoft.Windows.Server.2008.R2.SP1.OperatingSystem" />
  6. </DiscoveryTypes>
  7. <DataSource ID="DS" TypeID="Windows!Microsoft.Windows.FilteredRegistryDiscoveryProvider">
  8. <ComputerName>$Target/Host/Property[Type="Windows!Microsoft.Windows.Computer"]/NetworkName$</ComputerName>
  9. <RegistryAttributeDefinitions>
  10. <RegistryAttributeDefinition>
  11. <AttributeName>WindowsCurrentVersion</AttributeName>
  12. <Path>SOFTWARE\Microsoft\Windows NT\CurrentVersion\CurrentVersion</Path>
  13. <PathType>1</PathType>
  14. <AttributeType>1</AttributeType>
  15. </RegistryAttributeDefinition>
  16. <RegistryAttributeDefinition>
  17. <AttributeName>WindowsCurrentBuild</AttributeName>
  18. <Path>SOFTWARE\Microsoft\Windows NT\CurrentVersion\CurrentBuild</Path>
  19. <PathType>1</PathType>
  20. <AttributeType>1</AttributeType>
  21. </RegistryAttributeDefinition>
  22. </RegistryAttributeDefinitions>
  23. <Frequency>14400</Frequency>
  24. <ClassId>$MPElement[Name="Microsoft.Windows.Server.2008.R2.SP1.OperatingSystem"]$</ClassId>
  25. <InstanceSettings>
  26. <Settings>
  27. <Setting>
  28. <Name>$MPElement[Name='Windows!Microsoft.Windows.Computer']/PrincipalName$</Name>
  29. <Value>$Target/Host/Property[Type="Windows!Microsoft.Windows.Computer"]/PrincipalName$</Value>
  30. </Setting>
  31. </Settings>
  32. </InstanceSettings>
  33. <Expression>
  34. <And>
  35. <Expression>
  36. <SimpleExpression>
  37. <ValueExpression>
  38. <XPathQuery Type="String">Values/WindowsCurrentVersion</XPathQuery>
  39. </ValueExpression>
  40. <Operator>Equal</Operator>
  41. <ValueExpression>
  42. <Value Type="String">6.1</Value>
  43. </ValueExpression>
  44. </SimpleExpression>
  45. </Expression>
  46. <Expression>
  47. <SimpleExpression>
  48. <ValueExpression>
  49. <XPathQuery Type="String">Values/WindowsCurrentBuild</XPathQuery>
  50. </ValueExpression>
  51. <Operator>Equal</Operator>
  52. <ValueExpression>
  53. <Value Type="String">7601</Value>
  54. </ValueExpression>
  55. </SimpleExpression>
  56. </Expression>
  57. </And>
  58. </Expression>
  59. </DataSource>
  60. </Discovery>
  61. </Discoveries>

Ovviamente non dimentichiamo le rispettive display string da aggiungere nel language pack.

* + 1. XML
    2. <DisplayString ElementID="QND.Compliance.HSRefOSDiscovery">
    3. <Name>QND HealthService Reference OS Discovery</Name>
    4. <Description>Sets a reference to the OS for the HealthService needed for projecting the fix compliance to the agent state</Description>
    5. </DisplayString>
    6. <DisplayString ElementID="QND.Compliance.Win2008R2SP1.Discovery">
    7. <Name>QND Windows 2008 Server R2 with SP1 discovery</Name>
    8. </DisplayString>

## Monitoring

Eccoci arrivati al cuore del nostro management pack dove finalmente implementare il controllo di compliance, come vedrete dopo tutto quanto realizzato in precedenza la definizione del monitor sarà piuttosto semplice. Come sempre ci sono molte opzioni per l’implementazione del monitor, la scelta che ho preso prevede di avere un monitor generico con target un generico sistema operativo (Microsoft.Windows.OperatingSystem) e quindi implementare tramite override le casistiche specifiche per le singole baseline. In sintesi:

* + Lo unit monitor è generico, ha come target la classe Microsoft. Windows.OperatingSystem ed è disabilitato
  + Ogni baseline (es. Windows 2008 SP2) avrà un proprio override (in realtà una serie, ma logicamente sono come uno) che abiliterà il monitor ed imposterà i parametri in termini di versione di sistema operativo, elenco di patch richieste e così via
  + L’aggiunta di una baseline, posta che ci sia già il discovery del sistema operativo, è solo questione dell’aggiunta di un override (ancora un volta una serie, ma logicamente è come se fossero uno). In questo modo il management pack è estensibile ad altri sistemi operativi direttamente da Operations Console senza doverlo modificare

Lo unit monitor è dunque un’istanza del monitor type precedentemente definito che genera un alert utilizzando la proprietà Message ritornata dal nostro data source. È bene sottolineare che questo meccanismo non permette una localizzazione del messaggio usando i differenti language pack. Un approccio più strutturato avrebbe richiesto che il data source tornasse solo le parti variabili e non localizzabili del messaggio e quindi la definizione di un alert message con più parametri. Nella realtà dei fatti questo approccio non è sempre possibile perché manca, nella composizione dell’alert message, ogni possibilità di avere parametri condizionali. Questo significa che a fronte della necessità di avere un descrizione dell’alert fortemente dinamica, rischiamo di doverla creare o con informazioni inutili o con informazioni mancanti. Potremmo discuterne a lungo, ma fin tanto che l’implementazione dell’alert message è questa non vedo alternative a creare messaggi strutturati direttamente nel data source. Quindi il nostro alert avrà un unico parametro: la proprietà “Message”.

* 1. XML
  2. <AlertSettings AlertMessage="QND.Compliance.OS.AlertID">
  3. <AlertOnState>Warning</AlertOnState>
  4. <AutoResolve>true</AutoResolve>
  5. <AlertPriority>Normal</AlertPriority>
  6. <AlertSeverity>Warning</AlertSeverity>
  7. <AlertParameters>
  8. <AlertParameter1>$Data/Context/Property[@Name='Message']$</AlertParameter1>
  9. </AlertParameters>
  10. </AlertSettings>

Altra nota importante riguarda il controbuto dello unit monitor al modello di salute del sistema operativo, ovviamente, come abbiamo già definito nell’health model, il monitor modificherà lo stato dell’aspetto “configurazione” del sistema. La domanda è: quale criticità deve avere la non compliance? A mio modo di vedere non può essere altro che a livello “warning”, fin tanto che la mancata compliance non comporta un’interruzione del servizio non ha senso marcarla come “critical”. Ci sono troppi management pack sul mercato che utilizzano in modo non mediato le criticità degli alert, portando a console dove è difficile dare una prioritizzazione degli allarmi. In definitiva il nostro monitor in mancanza di compliance cambierà il suo stato in warning e genererà un alert parimenti di livello warning.

* 1. XML
  2. <OperationalStates>
  3. <OperationalState ID="Compliant" MonitorTypeStateID="Compliant" HealthState="Success" />
  4. <OperationalState ID="NonCompliant" MonitorTypeStateID="NonCompliant" HealthState="Warning" />
  5. </OperationalStates>

La composizione definitiva dell’unit monitor, da aggiungere nella sezione di Monitoring è dunque la seguente:

* 1. XML
  2. <UnitMonitor ID="QND.Compliance.WindowsOS" Accessibility="Public" Enabled="false" ConfirmDelivery="true" Priority="Normal" Remotable="false"
  3. ParentMonitorID="Health!System.Health.ConfigurationState" Target="Windows!Microsoft.Windows.OperatingSystem" TypeID="QND.OSCompliance.MT">
  4. <Category>ConfigurationHealth</Category>
  5. <AlertSettings AlertMessage="QND.Compliance.OS.AlertID">
  6. <AlertOnState>Warning</AlertOnState>
  7. <AutoResolve>true</AutoResolve>
  8. <AlertPriority>Normal</AlertPriority>
  9. <AlertSeverity>Warning</AlertSeverity>
  10. <AlertParameters>
  11. <AlertParameter1>$Data/Context/Property[@Name='Message']$</AlertParameter1>
  12. </AlertParameters>
  13. </AlertSettings>
  14. <OperationalStates>
  15. <OperationalState ID="Compliant" MonitorTypeStateID="Compliant" HealthState="Success" />
  16. <OperationalState ID="NonCompliant" MonitorTypeStateID="NonCompliant" HealthState="Warning" />
  17. </OperationalStates>
  18. <Configuration>
  19. <OSVersion></OSVersion>
  20. <QFEList>Ignore</QFEList>
  21. <WSHVersion>5.7</WSHVersion>
  22. <ScriptTimeout>300</ScriptTimeout>
  23. <IntervalSeconds>14400</IntervalSeconds>
  24. <SyncTime />
  25. </Configuration>
  26. </UnitMonitor>

Con i parametri di default il monitor, per altro disabilitato, controlla ogni 4 ore la versione del Windows Scripting Host. Per completare il monitor rimane da implementare la definizione dell’alert message e come sempre le display string utili a rendere più leggibile il monitor in console. Le stringhe per i messaggi di alert devono prima essere dichiarate nella sezione Presentation e quindi definite nel Language Pack:

* 1. XML
  2. <Presentation>
  3. <StringResources>
  4. <StringResource ID="QND.Compliance.OS.AlertID"/>
  5. </StringResources>
  6. </Presentation>

Nella display string attenzione alla inconsistenza nella sostituzione dei parametri, mentre nella definizione del monitor il primo parametro è definito come AlertParameter1 nella sostituzione i parametri iniziano da 0 ({0}). Purtroppo ci sono tante piccole inconsistenze che rendono più difficile scrivere management pack.

* 1. XML
  2. <DisplayString ElementID="QND.Compliance.OS.AlertID">
  3. <Name>Windows OS not compliant with minimum version and QFEs</Name>
  4. <Description>The Windows OS isn’t compliant with the minimum baseline required for monitoring.
  5. {0}
  6. Check the alert context for more information.</Description>
  7. </DisplayString>
  8. Finally don’t forget the display string for the monitor itself
  9. <DisplayString ElementID="QND.Compliance.WindowsOS">
  10. <Name>Prerequiste compliance for monitoring</Name>
  11. </DisplayString>

Come abbiamo già indicato nessun monitor dovrebbe mancare della propria knowledge base:

* 1. XML
  2. <KnowledgeArticle ElementID="QND.Compliance.WindowsOS" Visible="true">
  3. <MamlContent>
  4. <maml:section xmlns:maml="http://schemas.microsoft.com/maml/2004/10">
  5. <maml:title>Summary</maml:title>
  6. <maml:para>The monitor check for Operating System compliance in terms of:</maml:para>
  7. <maml:list>
  8. <maml:listItem>
  9. <maml:para>Operating System version</maml:para>
  10. </maml:listItem>
  11. <maml:listItem>
  12. <maml:para>Windows Scripting Host Version</maml:para>
  13. </maml:listItem>
  14. <maml:listItem>
  15. <maml:para>QFEs installed</maml:para>
  16. </maml:listItem>
  17. </maml:list>
  18. <maml:para>Missing requirements can impair the monitor capabilities and/or effect agent and operating system performance </maml:para>
  19. </maml:section>
  20. <maml:section xmlns:maml="http://schemas.microsoft.com/maml/2004/10">
  21. <maml:title>Configuration</maml:title>
  22. <maml:para>Proper overrides have been defined for each operating system version, the list of requirements can be adjusted using custom overrides</maml:para>
  23. </maml:section>
  24. <maml:section xmlns:maml="http://schemas.microsoft.com/maml/2004/10">
  25. <maml:title>Causes</maml:title>
  26. <maml:para>Fixes are not installed or operating system has not been updated to the proper service pack level</maml:para>
  27. </maml:section>
  28. <maml:section xmlns:maml="http://schemas.microsoft.com/maml/2004/10">
  29. <maml:title>Resolutions</maml:title>
  30. <maml:para>Install the missing prerequisites</maml:para>
  31. </maml:section>
  32. <maml:section xmlns:maml="http://schemas.microsoft.com/maml/2004/10">
  33. <maml:title>External</maml:title>
  34. <maml:para>
  35. For an up to date list of requirements check: <maml:navigationLink>
  36. <maml:linkText>Things to make and do for agent health</maml:linkText>
  37. <maml:uri href="http://nocentdocent.wordpress.com/articles-and-series/things-to-make-and-do-for-agent-health/" />
  38. </maml:navigationLink>
  39. </maml:para>
  40. </maml:section>
  41. </MamlContent>
  42. </KnowledgeArticle>

Finito? Non ancora. Se vogliamo rispettare il nostro health model dobbiamo definire come lo stato del monitor di compliance possa proiettarsi sullo stato dell’HealthService. Ricapitolando:

* + Il monitor ha come target il sistema operativo e contribuisce allo stato della sezione “Configuration”
  + Vogliamo che questo stato, trattandosi di fix che migliorano il funzionamento dell’agente Operations Manager, venga anche proiettato sull’health model dell’agente, o in altre parole vogliamo mettere una dipendenza al monitor di compliance per l’HealthService
  + Per fare questo occorre creare un DependencyMonitor
  + Per poter definire un DependencyMonitor tra due classi occorre che tra esse ci sia una relazione, questo è il motivo per cui abbiamo definito un relazione tra HealthService e sistema operativo

Tutto questo si traduce come segue all’interno del management pack, da notare che è disabilitato di default, per far sì che lo stato si propaghi all’HealthService occorre abilitarlo o direttamente nel management pack oppure tramite specifico override

* 1. XML
  2. <DependencyMonitor ID="QND.Compliance.OSPrereq.DM" Accessibility="Public" Enabled="false" Target="SC!Microsoft.SystemCenter.HealthService"
  3. ParentMonitorID="Health!System.Health.ConfigurationState" MemberMonitor="QND.Compliance.WindowsOS"
  4. RelationshipType="QND.Compliance.HealthServiceReferenceOperatingSystem" Priority="Normal" Remotable="true">
  5. <Category>ConfigurationHealth</Category>
  6. <Algorithm>WorstOf</Algorithm>
  7. </DependencyMonitor>

Infine occorre la display string che verrà mostrata nell’health model della classe HealthService

* 1. XML
  2. <DisplayString ElementID="QND.Compliance.OSPrereq.DM">
  3. <Name>OS Prerequisite compliance</Name>
  4. </DisplayString>

## Override

Come anticipato il monitor così come definito ha poca utilità e in effetti è disabilitato. Per portarlo alla vita occorrono almeno un paio di override il cui ambito di applicazione (context) può essere una qualsiasi classe derivata da Microsoft.Windows.OperatingSystem (oppure gruppi di oggetti o singole istanze degli stessi):

* + Un override che abiliti il monitor
  + Uno o più override per i parametri specifici (versione del sistema operativo, liste delle fix, versione di windows scripting host minima)

Ad esempio per abilitare il monitor per i Server Windows 2008 occorrerà scrivere

* 1. XML
  2. <MonitorPropertyOverride ID="QND.Compliance.Win2008.Enable.OR" Context="Win2008!Microsoft.Windows.Server.2008.OperatingSystem" Enforced="false"
  3. Monitor="QND.Compliance.WindowsOS" Property="Enabled">
  4. <Value>true</Value>
  5. </MonitorPropertyOverride>

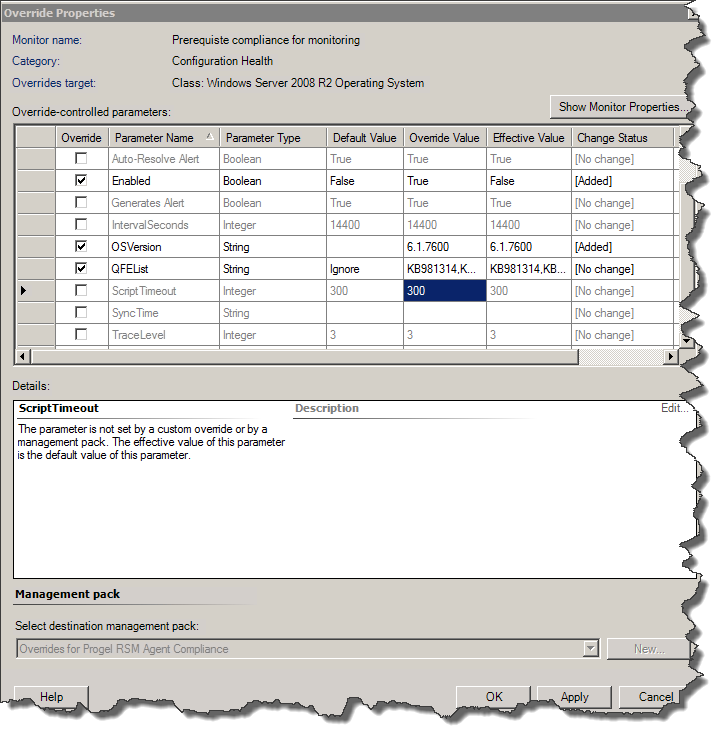
Per impostare la versione minima del sistema operativo, diciamo per Windows 2008 la Service Pack 2, ovvero la versione 6.0.6002 occorrerà indicare

* 1. XML
  2. <MonitorConfigurationOverride ID="QND.Compliance.Win2008.OSVer.OR" Context="Win2008!Microsoft.Windows.Server.2008.OperatingSystem" Enforced="false"
  3. Monitor="QND.Compliance.WindowsOS" Parameter="OSVersion">
  4. <Value>6.0.6002</Value>
  5. </MonitorConfigurationOverride>

Analogamente per le fix:

* 1. XML
  2. <MonitorConfigurationOverride ID="QND.Compliance.Win2008.QFE.OR" Context="Win2008!Microsoft.Windows.Server.2008.OperatingSystem" Enforced="false"
  3. Monitor="QND.Compliance.WindowsOS" Parameter="QFEList">
  4. <Value>KB968967,KB981936,KB2458331</Value>
  5. </MonitorConfigurationOverride>

Se vi siete stancati di scrivere XML tutto questo può essere fatto anche tramite UI, una volta che il management pack è stato importato:

* 1. 

Per convenienza il Management Pack che potrete scaricare conterrà tutti gli override necessari per le baseline oggetto di questo articolo, la sezione degli override completa sarà:

* 1. XML
  2. <Overrides>
  3. <MonitorPropertyOverride ID="QND.Compliance.Win2008.Enable.OR" Context="Win2008!Microsoft.Windows.Server.2008.OperatingSystem" Enforced="false"
  4. Monitor="QND.Compliance.WindowsOS" Property="Enabled">
  5. <Value>true</Value>
  6. </MonitorPropertyOverride>
  7. <MonitorConfigurationOverride ID="QND.Compliance.Win2008.OSVer.OR" Context="Win2008!Microsoft.Windows.Server.2008.OperatingSystem" Enforced="false"
  8. Monitor="QND.Compliance.WindowsOS" Parameter="OSVersion">
  9. <Value>6.0.6002</Value>
  10. </MonitorConfigurationOverride>
  11. <MonitorConfigurationOverride ID="QND.Compliance.Win2008.QFE.OR" Context="Win2008!Microsoft.Windows.Server.2008.OperatingSystem" Enforced="false"
  12. Monitor="QND.Compliance.WindowsOS" Parameter="QFEList">
  13. <Value>KB968967,KB981936,KB2458331</Value>
  14. </MonitorConfigurationOverride>
  15. <MonitorPropertyOverride ID="QND.Compliance.Win2008R2.Enable.OR" Context="Win2008!Microsoft.Windows.Server.2008.R2.OperatingSystem" Enforced="false"
  16. Monitor="QND.Compliance.WindowsOS" Property="Enabled">
  17. <Value>true</Value>
  18. </MonitorPropertyOverride>
  19. <MonitorConfigurationOverride ID="QND.Compliance.Win2008R2.OSVer.OR" Context="Win2008!Microsoft.Windows.Server.2008.R2.OperatingSystem" Enforced="false"
  20. Monitor="QND.Compliance.WindowsOS" Parameter="OSVersion">
  21. <Value>6.1.7600</Value>
  22. </MonitorConfigurationOverride>
  23. <MonitorConfigurationOverride ID="QND.Compliance.Win2008R2.QFE.OR" Context="Win2008!Microsoft.Windows.Server.2008.R2.OperatingSystem" Enforced="false"
  24. Monitor="QND.Compliance.WindowsOS" Parameter="QFEList">
  25. <Value>KB981314,KB981936</Value>
  26. </MonitorConfigurationOverride>
  27. <MonitorPropertyOverride ID="QND.Compliance.Win2008R2SP1.Enable.OR" Context="Microsoft.Windows.Server.2008.R2.SP1.OperatingSystem" Enforced="false"
  28. Monitor="QND.Compliance.WindowsOS" Property="Enabled">
  29. <Value>true</Value>
  30. </MonitorPropertyOverride>
  31. <MonitorConfigurationOverride ID="QND.Compliance.Win2008R2SP1.OSVer.OR" Context="Microsoft.Windows.Server.2008.R2.SP1.OperatingSystem" Enforced="false"
  32. Monitor="QND.Compliance.WindowsOS" Parameter="OSVersion">
  33. <Value>6.1.7601</Value>
  34. </MonitorConfigurationOverride>
  35. <MonitorConfigurationOverride ID="QND.Compliance.Win2008R2SP1.QFE.OR" Context="Microsoft.Windows.Server.2008.R2.SP1.OperatingSystem" Enforced="false"
  36. Monitor="QND.Compliance.WindowsOS" Parameter="QFEList">
  37. <Value>ignore</Value>
  38. </MonitorConfigurationOverride>
  39. </Overrides>

Se volete che l’override per Windows 2008 R2 Service Pack 1 sia utile occorre abilitare il discovery di questa versione del sistema operativo:

* 1. XML
  2. <DiscoveryPropertyOverride ID="QND.Compliance.Win2008R2SP1Discovery.Enable.OR" Context="Win2008!Microsoft.Windows.Server.2008.R2.OperatingSystem" Enforced="false"
  3. Discovery="QND.Compliance.Win2008R2SP1.Discovery" Property="Enabled">
  4. <Value>true</Value>
  5. </DiscoveryPropertyOverride>

## Conclusioni

Spero con questo articolo di aver raggiunto due risultati, da un lato dare risposta ad una esigenza molto sentita come quella di verificare se le fix consigliate per il corretto funzionamento degli agenti Operations Manager siano state installate, dall’altro aver mostrato la flessibilità permessa dai Management Pack.

A onor del vero l’intero sistema può sembrare complesso. Non voglio negare una certa complessità del modello dei Management Pack, ma è bene precisare che molti ambiti di monitor sono risolti tramite facili wizard o tramite schemi Visio che generano in automatico le regole di monitor necessarie, l’accesso al codice come mostrato in questo articolo è necessario solo per le situazioni più complesse e al di fuori dell’ordinario.

Per comodità il Management Pack può essere scaricato [qui](https://skydrive.live.com/?cid=69A8661ACB441E62&id=69A8661ACB441E62%21124).

Ovviamente il management pack è un esempio e non deve essere usato in ambiente di produzione.

La stessa tecnica e il medesimo data source può essere utilizzato per indirizzare altri ambiti di compliance, sempre relativamente alle fix da installare. Ad esempio è stato utilizzato con successo per verificare la presenza di fix ritenute critiche in ambienti hyper-v mission critical dove, per scelta non tecnica, l’agente Configuration Manager non poteva essere installato.

#### di [Daniele Grandini](https://mvp.support.microsoft.com/profile=58922C7A-F40F-438B-93D1-7E4E3CF963AD) – Microsoft MVP ([blog](http://nocentdocent.wordpress.com))

* 1. [*Altri articoli di Daniele Grandini nella Library*](http://sxp.microsoft.com/feeds/3.0/msdntn/TA_MSDN_ITA?contenttype=Article&author=Daniele%20Grandini) [](http://sxp.microsoft.com/feeds/3.0/msdntn/TA_MSDN_ITA?contenttype=Article&author=Matteo%20Emili)