FEATURE

Inside the Ops Manager Management Pack

A closer look at management pack design, function, and tuning

by Pete Zerger

System Center Operations Manager 2007 R2 is touted as the best-of-breed platform for application monitoring on Windows OSs. The "secret sauce" that separates Ops Manager from the competition is the management pack. Management packs provide the instructions for Ops Manager to discover and monitor specific applications. In June 2008, Microsoft declared that the company would deliver a management pack for every server application the company released, thus ensuring improved manageability across all Microsoft server applications. In this article, I explore the life cycle of a management pack, including design, function, and tuning.

Definition

A management pack is an XML file that contains all the elements necessary to discover and monitor an application with Ops Manager 2007. The management pack is imported into an Ops Manager 2007 management group and distributed to appropriate agents that use its contents to perform monitoring activities for a particular application.

Management packs contain product information, included by the management pack author, that provides common causes and resolutions for alerts that are raised during monitoring. This product information transfers the application experts' (i.e., the Microsoft application architects') knowledge to the operators responsible for responding to alert conditions. Management packs for Microsoft applications are designed primarily by the product teams themselves—the architects who arguably know more about the application than anyone—which gives Ops Manager an advantage over other enterprise monitoring platforms.

Management Pack Components

Management packs contain several components that help discover, describe, and monitor applications. I discuss the following components and related concepts later in the article.

- Attributes
- Classes
- Groups
- Health state
- Monitors (unit monitors, aggregate rollup monitors, dependency rollup monitors)
- Object discoveries
- Overrides
- Rules
- Tasks
- Views
- Reports

The basic components that comprise a management pack include the attributes, classes, groups, health state, and monitors. These elements (listed here in alphabetical order) are viewable in the Authoring space of the Operations console.

- Attributes: Attributes contain information that further defines an object type in Ops Manager. The Attributes node in Ops Manager's Authoring space displays a list of attributes for each object type in the management group, such as Windows Computer, Windows 2003 Operating System, SQL Server Database, or IIS Web Application.
- Classes: A class represents a type of object. Classes are defined in management packs and aren't limited to computers and groups. Classes can be defined to represent any component of an application or device, such as a SQL Server instance or database. Classes are sometimes called *object types* or *targets*.
- Groups: Groups are used to create collections of objects. Groups can contain collections of any type of object defined in a management pack and discovered and monitored by Ops Manager. Technically, every group is also a class (a special single-instance class called a singleton class). However, because groups can be used to create collections of subsets of the instances of a class for management pack tuning, they deserve special mention here.
- Health state: The current health of a monitored object (red, yellow, or green); sometimes simply called a state.
- Monitors: Monitors periodically assess the condition of specified objects, such as services and performance counters. As a result of the assessment, a monitor can change the health state of an object and can generate alerts. Only monitors understand health state (rules don't). As a result, monitors can be configured to automatically resolve an associated alert when the error condition improves and monitor state returns to healthy.

Ops Manager monitors include unit monitors, aggregate rollup monitors, and dependency rollup monitors.

- Unit monitors: These monitors are created to monitor specific aspects of applications, devices, and services, such as Windows events, services, and performance counters. They can also be used to monitor network devices through SNMP.
- Aggregate rollup monitors: You typically use an aggregate rollup monitor to group multiple like unit monitors into a single (aggregate) point, then use that monitor to set the health state (and generate an alert if desired). Aggregate monitors roll up the health of monitors beneath them according to a defined algorithm. These monitors use one of two algorithm elements: WorstOf or BestOf. With the WorstOf algorithm, if any unit monitor that's being aggregated is in a warning or error state, the aggregate rollup moves to a warning or error state. With the BestOf algorithm, if at least one of the unit monitors that's being aggregated remains in a healthy state, the aggregate monitor also remains in a healthy state.
- Dependency rollup monitors: These monitors roll up the health from an instance of another class linked by a hosting or containment relationship. For example, a monitor of this type rolls up the health of the IIS 7.0 Server Role class to the Windows Server 2008 instance hosting the role. Like aggregate rollup monitors, dependency rollup monitors support BestOf and WorstOf algorithms for health rollup. Dependency rollup monitors also support a third algorithm called WorstOfAPercentage. The monitor using this algorithm will transition to an unhealthy state when X percent of the monitors it aggregates are in an unhealthy state (where X is a percentage defined by the management pack author or later by Ops Manager administrators).

Additional management pack components include object discoveries, overrides, rules, tasks, views, and reports.

- Object discoveries: An object discovery is used to dynamically find objects and their properties on the network that need to be monitored. Object discoveries can use registry information, Windows Management Instrumentation (WMI) queries, SNMP, scripts (VBScript, JScript, or PowerShell), or managed code to identify applications running on a managed system. Sometimes simply called a discovery.
- Overrides: An override is an adjustment to the default settings of an object discovery, monitor, rule, or task. Overrides are how administrators tune management packs in Ops Manager.
- Rules: Rules collect data, such as performance and event information, generated by managed objects. Rules can be configured to generate alerts. However, rules don't affect the health state of the objects they monitor.
- Tasks: A task performs an administrative action on demand when an Ops Manager administrator selects the task. Some tasks run on the managed computer (console tasks), and some tasks run on the computer from which they were initiated (agent tasks).
- Views: Views display a particular aspect of monitoring settings. The displayed information in a view is the result of a query to the Ops Manager database. Ops Manager views include Alert, Event, Performance, State, Diagram, Dashboard, Task, and URL.
- Reports: Reports (delivered in many management packs) can display information about object availability, as well as event, alert, configuration, or performance data collected from the applications or devices monitored by the management pack.

Management Pack Design

A management pack's design determines how effective the management pack is in monitoring the application it targets. Although the experience level of the management pack author is important, it's equally important to involve a subject matter expert in the authoring process who truly understands the application to be monitored. A management pack created by someone who isn't familiar with the architecture and function of the target application won't result in an effective monitoring solution for that application.

How does Ops Manager dynamically discover and monitor applications? It all begins with the management pack design process, in which the authors define the application components of interest and how the components are related.

Service model. The combination of classes and their relationships within a management pack is referred to as the application's service model. Figure 1 shows the Windows Server 2008 Management Pack's service model.

The application components of interest for monitoring are defined as unique object types, or classes. An occurrence of a class discovered by Ops Manager is referred to as an instance of the class.

For example, the Windows Server 2008 Management Pack contains a class called Windows Server 2008 Computer, as Figure 1 shows. It also contains several classes representing the components of Windows Server 2008 Computer, such as Windows Server 2008 Operating System, Windows Server 2008 Logical Disk, Windows Server 2008 Physical Disk, and Windows Server 2008 Processor—object types of interest from a monitoring perspective.

However, it's not enough to describe and discover instances of these object classes. If every time we removed an Ops Manager agent from a Windows Server 2008 computer, its processor, disk, and other components remained in the Operations console, we'd have a serious administration headache. To dynamically discover, monitor, and (when necessary) remove instances, Ops Manager must also understand how the objects are related. To address this need, the management pack author creates relationships to describe the dependencies and interaction between object types. Objects can be connected through one of three types of relationships.

- Hosting relationship: For example, the Windows Server 2008 Management Pack contains a class called Windows Server 2008 Operating System. This class is said to be hosted by the Windows Computer class because the Windows OS can't exist without the computer that it runs on.
- Containment relationship: In this type of relationship, instances of the target class can exist even if they aren't part of a containment relationship. For example, a computer is contained within a computer group, but the computer can exist even if it isn't part of a computer group.
- Reference relationship: In this type of relationship, object types aren't dependent on each other. For example, a database could reference another database that it's replicating with. This is the least specific type of relationship (as well as the least often utilized by management pack authors).

After an application's object types (classes) and their relationships are defined (establishing the service model), the author(s) can add the logic to define how the health of one object type affects another related object type.

Health model. After the object types and their relationships are established, the author can define how the health of each object type affects the health of others in the management pack. An application's health model determines how the current state of an application or application component is represented in Ops Manager. This model shows how the health state of individual application components affects the overall health of the application as displayed in the Operations console.

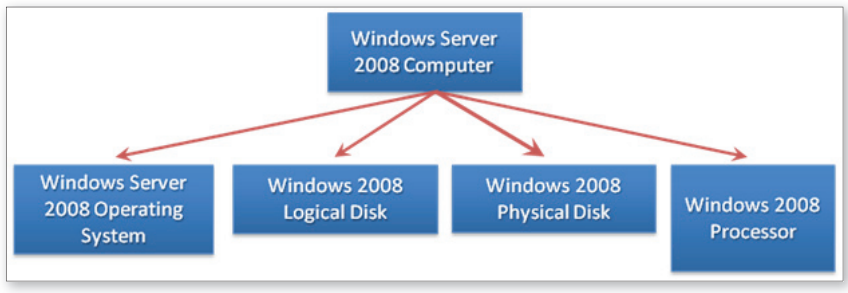

Figure 1: Windows Server 2008 Management Pack service model

The health of a hosted object class isn't automatically reflected (rolled up) in the health of the parent class—at least not in all cases. It's important to understand what's necessary to define the health rollup of the application components in a management pack.

Ops Manager health rollup requires two elements:

- Relationships: Health of the instances of one class can be rolled up to another class only if a hosting or containment relationship exists.
- Rollup monitors: A dependency rollup monitor must be created to roll up the health of one class to another.

You might wonder whether the health of a class *should* be rolled up to another class. This is a question the author must ask when designing a management pack's health model. The short answer is "it depends"—let's look at a couple of common examples to provide some perspective.

In the case of the Windows Server 2008 Management Pack, the health of the Windows Server 2008 Operating System class is rolled up to the Windows Server 2008 Computer class that hosts the OS. Because the primary function of the computer is to host the OS, it's logical to assume that if the OS is unhealthy, the computer won't be fully functional. The authors therefore included a dependency rollup monitor to transfer the health of the OS to the host computer.

In the case of the SQL Server 2008 Database Engine class (which represents a SQL Server instance), the SQL Server 2008 Database Engine class hosts the SQL Server 2008 Database class (SQL Server 2008 DB). Although these two classes share a hosting relationship (as in the previous example), the application health dynamics are unique. Even if one or more databases are unhealthy (e.g., offline, corrupt, suspect), this doesn't necessarily indicate a problem with the SQL Server database engine that hosts the database(s). In this case, the management pack authors chose not to roll up the health of the database class to its database engine host.

Discovering Applications and Components

Object discovery is the management pack element that's used to dynamically find objects and their properties that need to be monitored. Object discoveries can use registry information, WMI queries, SNMP, scripts, or managed code to identify applications running on a managed system. Application discovery is both a repetitive process and a progressive process in most cases. This is possible because every object discovery (and every rule, monitor, and task) has a target (class). The object discovery runs only on computers with one or more instances of the class targeted by the object discovery.

This concept is most easily explained by reviewing an example from an actual Microsoft management pack. Let's walk through the discovery process for the Windows Server 2008 IIS 7.0 management pack, from import of the management pack, transfer to the agent, loading of workflows, and the high-level discovery process. This example also illustrates the concept of progressive discovery, whereby layers of deeper discovery using scripts and other probes are performed only on systems on which the application is known to exist based on a lighter weight registry discovery.

1. First, the management pack is installed through the Import Management Packs wizard in the Operations console.
2. The management pack is delivered to all agents with at least one instance of the Windows Server class. (This action shows up as event 1201 in the Ops Manager event log on the monitored system.) In practical terms, every management pack is delivered to every properly functioning agent-managed system with at least one instance of the class targeted by a discovery, rule, monitor, etc., in the management pack. The seed discovery runs only on systems hosting an instance of the class targeted by the discovery.
3. Agents receive the management pack and load the workflows (discoveries, rules, monitors) into cache. (This action shows up as event 1210 in the Ops Manager event log on the monitored system.)
4. In the IIS 7.0 management pack, a discovery executes for all agents hosting an instance of the Windows Server 2008 class. This initial seed (root) discovery uses a simple registry probe to identify whether the IIS 7.0 role is installed on the system. Because discovery is a repetitive process, this discovery runs on a recurring basis every few hours.
5. An instance of the IIS 7.0 Server Role class is discovered on agents with IIS 7.0 installed. Likewise, if the IIS 7.0 role were uninstalled from the computer, the discovery would detect this and the IIS 7.0 Server Role instance, along with all the objects it hosts (e.g., websites, FTP sites, application pools), would be undiscovered (removed from) the management group automatically when the agent submitted an updated set of discovery data to the management group reporting this change.
6. Then, additional script-based discoveries targeting the IIS 7.0 Server Role class run only on those agents where an instance of the IIS 7.0 role was discovered, discovering specific IIS 7.0 components, such as the FTP server and SMTP server.
7. This process continues, with more discoveries targeting the Web, FTP, and SMTP Server classes to discover websites, FTP sites, and SMTP virtual servers hosted by the IIS 7.0 role on the server.

> Application discovery is both a repetitive process and a progressive process in most cases.

By using progressive layers of discovery in this manner, the management pack authors avoid running more detailed (and resource-intensive) scripts on systems where the target application or application components aren't installed, thus preventing unnecessary consumption of system resources.

On an ongoing basis, the agent runs all the applicable discoveries that it knows about when the health service starts up (e.g., when you restart the health service, or after a reboot). The agent sends this discovery data to the management server, then runs the discoveries again based on the interval specified on the object discovery—which might be as frequent as once per hour or as seldom as once per week. When the discovery runs, the agent inspects the discovery data it receives and compares it to the last discovery data it sent to the management server. If nothing changed since the last time discovery ran, the agent drops the discovery data and forwards nothing. If anything changed in the discovery data values, the agent resubmits the new data to the management server, which then submits the data to the operational database. The Root Management Server (RMS) detects the change and recalculates (and regenerates) the configuration. This action shows up on the RMS as event 21025 in the Ops Manager event log.

Event 21025 doesn't indicate a problem; it simply indicates that the System Center Management Configuration service (OMCFG, sometimes called the OpsMgr Config service) performed as expected. In this case, the service successfully regenerated the configuration file from the data in the operational database, writing it to the following file: \%ProgramFiles%\System Center Operations Manager 2007\Health Service State\Connector Configuration Cache*MGNAME*\OpsMgrConnector.Config.xml.

If changes in your management group are so frequent that the RMS is in a perpetual process of recalculating the configuration, you might encounter a condition known as config churn. For an in-depth explanation of config churn, including how to detect, troubleshoot, and prevent this undesirable condition, see the following blog posts: "Troubleshooting 21025 events—wrap up" by Daniele Grandini (nocentdocent.wordpress.com/2009/07/09/troubleshooting-21025-events-wrap-up) and "What is config churn?" by Kevin Holman (blogs.technet.com/b/kevinholman/archive/2009/10/05/what-is-config-churn.aspx).

Monitoring and Health State Calculation

After one or more instances of an application and its components are discovered (represented by class instances and

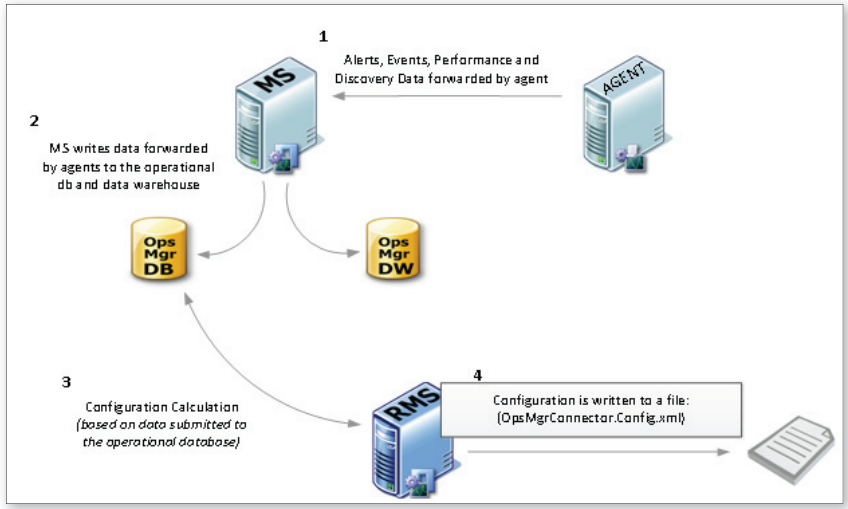

Figure 2: High-level data flow and process in configuration calculation

relationships), application monitoring begins. The rules and monitors contained within the management pack are loaded into cache by the agent as workflows. For example, in the management pack for IIS 7.0, unit monitors verify that Windows services related to SQL Server 2008 are running, confirm that databases are online and have adequate free space, and watch for error events in SQL Server and Windows event logs.

The agent runs these workflows and forwards this information to its designated management server. The management server writes this information to the operational database (Ops Manager), as well as to the data warehouse, if defined in the workflow. Data forwarded by the agent includes a number of different types, such as discovery, alert, event, and performance

The RMS then calculates the health state of monitored objects. After data is written to the Ops Manager database, the RMS watches and reads the instance space of the database and calculates the health state. As for configuration based on discovery data, calculation is controlled by the System Center Management Configuration service. Figure 2 illustrates the high-level flow of the configuration calculation process.

Tuning a Management Pack

Tuning a management pack involves using overrides to change the default settings. In sealed management packs, the management pack authors can control which discovery and monitoring parameters can be adjusted (tuned) by Ops Manager administrators by defining parameters as overridable parameters. Creating overrides is a relatively simple process, but knowing which overrides take precedence when competing or conflicting overrides exist can be confusing. However, you need to understand which overrides take precedence in order to avoid unnessesary problems.

The RMS determines which override takes precedence when multiple overrides apply to a workflow for a specific target, based on the following criteria:

- The first rule of overrides is that the most specific override takes precedence. For example, an override targeted to a group takes precedence over an override targeted to a class. Therefore, an override targeted to the Windows Server 2008 Computers group wins over an override targeted to the Windows Server 2008 class because the group, which contains a subset of the instances of the class, is the more specific target. As another example, an override targeted to an instance takes precedence over an override targeted to a group. Therefore, an override targeted to Server1 wins over an override targeted to the Windows Server 2008 Computers group because the instance is the more specific object.
- An enforced override takes precedence over all non-enforced overrides of the same type. For example, an enforced override targeted to a class wins over a non-enforced override targeted to a class.
- Overrides in unsealed management packs always take precedence over sealed overrides. This allows the administrator to better control settings in cases when an author has included overrides in a sealed management pack.
- Class overrides from hosted instances always take precedence over class overrides of the hosting instance.
- When all criteria for arbitration are exhausted, the RMS selects an override at random. It's important to never create so many like overrides that random selection is the only means by which Ops Manager can choose the effective override. One way to prevent this situation is to avoid creating group overrides for multiple groups that contain some of the same computers as members.

For a technical description of override precedence, see Jakub Olesky's "Overrides in SCOM 2007" at blogs.msdn.com/b/jakuboleksy/archive/2006/12/06/overrides-in-scom-2007.aspx. To learn more about Microsoft best practices for creating and storing overrides, see "Best practices to use when you configure overrides in System Center Operations Manager 2007" at support.microsoft.com/kb/943239.

Smooth Tuning

Tuning management packs can seem complicated. But knowing a management pack's design and function, understanding override precedence, and following Microsoft best practices for targeting and management pack tuning will go a long way in smoothing the tuning process.

InstantDoc ID 126053

Pete Zerger
(pzerger@akosts.com) is a consultant, blogger, author, speaker, and Microsoft MVP focusing on Microsoft System Center and virtualization technologies. He is a co-founder of systemcentercentral.com, a web community dedicated to System Center technologies.