MP Creation Zen
using the R2 MP Authoring Console

Part 1: Concepts and Application Modeling

successfully write Management Packs - even if you are not an application developer

Authors:
Pete Zerger (MVP)     Jaime Correia (MCSE)     Raphael Burri (MCTS)

Version: 1.0
August 2009

Some Rights Reserved: You are free to reference this document and its content, so long as when quoting you properly credit the authors and provide a link back to the original published source. No copies of this document may be published or hosted without the express written consent of the authors.
**Table of Contents**

Introduction ................................................................................................................................. 4

   Articles in the Series ................................................................................................................. 4

What You Will Learn in Part 1 ..................................................................................................... 5

Modeling the Application ............................................................................................................ 6

Exploring Application Architecture ........................................................................................... 6

What is a Class? ............................................................................................................................ 9

   Abstract Classes ....................................................................................................................... 9

   Concrete Classes ..................................................................................................................... 9

   Hosted and Non-Hosted Classes .......................................................................................... 10

   Singleton Classes .................................................................................................................. 10

Classes, Inheritance and Specialization ................................................................................. 11

What are relationships? ............................................................................................................. 13

   Reference Relationship ....................................................................................................... 14

   Containment Relationship ..................................................................................................... 14

   Hosting Relationship .............................................................................................................. 14

   The Health Model and Health Rollup ................................................................................ 15

Choosing a Base Class for your Object Classes ........................................................................ 16

   How do the Service Model and Health Model affect one another? .................................... 17

Mapping Components to Classes for the Certificates MP .................................................... 18

   Classes for the PKI Certificate Management Pack .......................................................... 18

Defining Properties for Classes ............................................................................................... 20

   What is a Key Property? ........................................................................................................ 20

   Data Types ............................................................................................................................. 20

   Certificate Store Class Properties ...................................................................................... 21

   Certificate Class Properties ............................................................................................... 21

Defining Relationships .............................................................................................................. 24

Defining the Health Model ........................................................................................................ 25

   Naming a Management Pack ............................................................................................... 26

Building the MP in the Authoring Console ............................................................................ 27

   Setting Compatibility ............................................................................................................ 27

Create the Object Classes and Relationships ......................................................................... 29

   Create the Certificate Services Component Class ............................................................ 29
Create the Certificate Store Class........................................................................................................... 30
Create the Certificate Class....................................................................................................................... 33
Testing Your Management Pack.................................................................................................................. 35
Conclusion.................................................................................................................................................. 36
Feedback.................................................................................................................................................... 37
Appendix A – MP Authoring Job Aids......................................................................................................... 38

Figures
Figure 1 - Targeting the Certificates MMC to Local Computer ................................................................. 8
Figure 2 - Personal Certificate Store in Certificates MMC Snap-in ............................................................. 8
Figure 3 - Clothing Class and it's Class Properties .................................................................................... 11
Figure 4 - "Specialized" Classes of Clothing ............................................................................................... 12
Figure 5 - Object Classes in the PKI Certificate Validation MP ................................................................. 19
Figure 6 - Class Properties in the MP Authoring Console ......................................................................... 20
Figure 7 – Certificate Properties .............................................................................................................. 21
Figure 8 - Classes in the PKI Certificate Validation MP ........................................................................... 23
Figure 9 - Health Model for PKI Certificates MP ..................................................................................... 25
Figure 10 - MP Authoring Console in OpsMgr 2007 R2 ......................................................................... 27
Figure 11 - MP References (in the MP Authoring Console) ...................................................................... 28
Figure 12 - Custom Class Interface in MP Authoring Console ................................................................. 30
Figure 13 - Class Properties Interface .................................................................................................... 31
Figure 14 - Property Display Name .......................................................................................................... 31
Figure 15 - Defining Property as a Key Property ....................................................................................... 32
Figure 16 - Hosting Relationship Properties for the Certificate Store Class .............................................. 33
Figure 17- Hosting Relationship Properties for the Certificate Class ...................................................... 35
Figure 18 - Certificate Validation MP Classes ........................................................................................... 36
Introduction

Creating a high quality Management Pack is considered by most to be a complex task, beyond the reach of most OpsMgr administrators. The goal of this articles series is to change this perception by explaining the process step-by-step, in terms everyone can understand...even if you are not a programmer.

Beginning with this article, the authors will guide you (the reader) through the process of building a Management Pack, explaining the important concepts and decisions to be taken at each step. The intended result is that you will understand not only “how” to design and build a Management Pack, but the rationale for each decision in the process.

However, this requires an understanding of certain core concepts of OpsMgr MP authoring functionality. It also requires knowledge of what questions have to be answered to make the best decision at each step in the process. This is the first of a six article series in which we will describe the necessary steps and best practices required to design and build a high quality Management Pack for Operations Manager 2007. The series will begin with simple authoring concepts and examples, moving on to progressively more advanced concepts in later articles.

This will not be a series of purely theoretical examples, but an end-to-end review of how a professionally-designed Management Pack – the PKI Certificate Validation MP, was created. The PKI Certificate Validation MP is designed to monitor the validity and configuration of digital certificates used by applications such IIS, OpsMgr and many others.

Articles in the Series
The articles in the series will cover the MP authoring process in the following installments:

- **Part 1 - MP Authoring Concepts and Application Modeling** – Overview of the MP design process. This installment will focus on object classes and relationships, and design of the MP Service Model and Health Model - everything required to effectively model the application targeted for monitoring.

- **Part 2 - Discovering Application Components** – Overview of discovery methodology, best practices and creation of discoveries for the Certificates MP.

- **Part 3 - Monitoring the Application** – Process for identifying how to best monitor your application, as well as implementation of the Health Model.

- **Part 4 - Advanced Monitoring** – This installment will provide a deep dive into modules and custom workflows.

- **Part 5 - Scripts in Rules and Monitors** – Scripting concepts and examples using the System Center Operations Manager 2007 Script API.
• **Part 6 – Presentation and Reporting** – This installment will focus on interaction with operators through the Operations console. Topics in this article will include how to create Views and Reports to be delivered in the MP. This will include hands-on examples of how to create “linked reports” to present performance and availability of your application.

**What You Will Learn in Part 1**

*The key takeaways from part 1 in the MP authoring series include everything you need to effectively “model the application” targeted for monitoring:*

- How to model your applications targeted for monitoring
- Classes and Relationships
- Class Specialization and Inheritance
- Class Properties (and Key Properties)
- Best practices when defining object classes, properties and relationships

Part 1 will focus on designing the **Service Model** for our application. The Service Model is a representation of the application components you intend to monitor, and how these components are related. In the process, you will also begin to make some decisions about the Health Model of our application. The **Health Model** of an application is a representation of how the health state of individual application components affect the overall health of the application as displayed in the Operations console.

**NOTE:**

*The MP Authoring process begins on paper (or note taking application of your choice). The modeling process should be complete before moving into the authoring console. Job aids are included in “Appendix A – Job Aids” to assist in you in taking appropriately detailed notes during the modeling process.*
Modeling the Application

To create the most effective service and Health Models for an application requires answering the following questions:

- **What are the components in the application and what do they do?** The process begins by understanding the application components and how it functions.

- **How are the components related to one another?** This requires delving into how the application components interact, and how the health of one component affects the others.

- **Which application components can be monitored and which need to be monitored?** Remember, you are only concerned with application components useful in monitoring application health. It is not the job of System Center Operations Manager 2007 to do inventory and configuration management.

- **How can the health of the components be measured?** Understanding the unique monitoring requirements of application components can be helpful in determining what distinct object classes are necessary.

Remember that the application model does not need to be complex. In fact, best practices for virtually every Microsoft application state that “the simplest design that meets the solution criteria is best”.

Exploring Application Architecture

For the purpose of this article, you are going to define a Service Model and Health Model for discovering and monitoring digital certificates. The goal is to identify digital certificates stored locally on a computer (in the registry) and monitor certificate health. Certificates can be used for a variety of functions (such as SSL encryption on web sites or mutual authentication in OpsMgr), but have similar requirements in order to function correctly. These include

- **Issuance from a valid (and trusted) certification authority** – Like a passport required for international travel, a certificate must be issued by a trusted and authorized entity. In this case, the authorized entity is a trusted Certification Authority (CA).

- **A future expiration date** – Every digital certificate issued has an expiration date. Expired certificates are not acceptable under most circumstances.

- **Revocation** – Certificates that have been revoked by the issuing CA are not accepted as trusted and valid.
To build a Management Pack to effectively monitor the health of digital certificates, or indeed any application, you must answer the following questions:

- **Which components are of interest for monitoring?** These will become *classes*.
- **How are the application components related to one another?** From this you will define *relationships* between the classes. This is known as the “*Service Model*” of the application.
- **How the health of each application component affects the health of other application components?** You will use this information to define the “*Health Model*” for the application...

You will find for many applications, you may not understand the application architecture and functionality well enough to answer these questions. This is the first rule of Management Pack authoring – “You cannot build an effective Management Pack if you do not understand how the application works”. Since nobody is an expert in all applications, this will often require you to work with a subject matter expert for the application to understand how the application functions. In this example, the authors will be your application experts and together we will explore how digital certificates work.

In order to describe the application in terms of classes and relationships and answer the questions above, you first have to understand the architecture of the application, its components and how they affect one another. You will begin by exploring the following components of certificate services, which are present any Windows computer:

- **Certificate Stores**
  - The “Personal” Stores (separate stores exist for User accounts and Local Computer account)
  - “Trusted Root Certification Authorities”
  - “Intermediate Certification Authorities” and

- **Certificates contained in the stores**

**How to view the certificate stores for the local computer**

Let's view the certificate store for the local computer to see how the certificate store is constructed. Using the Certificates mmc snap-in (certmgr.msc), you can set the console scope to display where the computer certificates are stored.

1. Click **Start**, click **Run**, type **mmc**, and then click **OK**.
2. On the **File** menu, click **Add/Remove Snap-in**, and then click **Add**.
3. Under **Snap-in**, double-click **Certificates**, click **Computer account**, and then click **Next**.
4. To manage certificates for the local computer, click **Local computer**, and then click **Finish**.
5. Click **Close. Certificates (Computer Name)** appears on the list of selected snap-ins for the new console.
6. If you have no more snap-ins to add to the console, click **OK**.
7. When you have completed these steps, select View → Options and verify the MMC snap-in you have created has the following settings:

![View Options](image)

**Figure 1 - Targeting the Certificates MMC to Local Computer**

As you can see there is a main store “Certificates Local Computer”, followed by a folder “Personal” and two subfolders “Registry” and “Certificates”. The information contained in these certificate stores is actually stored in the registry.

![Certificate MMC](image)

**Figure 2 - Personal Certificate Store in Certificates MMC Snap-in**

You will be focusing on the three stores that will be monitored by this MP:

- **Personal Certificate Store** – Used to store certificates issued to the local computer account
- **Trusted Root Certification Authorities** - Stores root certificates from a list of trusted certification authorities. The certificate of a root CA must be contained in this store in order for it to be trusted by the local computer.
- **Intermediate Certification Authorities** – An intermediate CA is a CA delegated the right to issue certificates on behalf of a root CA. The intermediate CA certificate must be present in order for certificates having been issued by this CA to be trusted.
What is a Class?

For the construction of the Management Pack, these folders (which are components of the application) must be represented as objects in OpsMgr. In OpsMgr, this is accomplished by creating a Class.

A class represents a type of object. A class will have one or more characteristics, called properties. For example, properties of the class representing the Windows Computer are NetBIOS name, FQDN, and OS Version among others. All objects of a particular type (called instances of the class) share common characteristics.

There are several terms used to differentiate classes in System Center Operations Manager 2007, including Abstract Class, Concrete Class, Hosted Class, Non-Hosted class each of which have defining characteristics.

NOTE: Classes are sometimes called Types or Targets. Just remember that Class = Type = Target. These are three terms that all mean the same thing.

Abstract Classes

The Abstract Class may be called a parent class because it exists with the sole purpose of being a class from which other classes can inherit properties. When you use an abstract class as the basis for your object class, all the properties of the abstract class are inherited by our class, and child classes of our class. The abstract class on which a newly created class specializes is known as its base class. Every class in OpsMgr will have a base class, except the root class at the top of the hierarchy. The root class is called Entity (System.Entity), and all classes inherit from it. The Entity class has a single property, which is “DisplayName”.

NOTE: The classes you create to represent the application components that will be discovered and monitored are sometimes referred to as concrete classes.

Concrete Classes

A Concrete Class is one created for the purpose of targeting application monitoring. Concrete classes are generated based on an abstract class. This will be discussed in greater detail later in this document.
**Hosted and Non-Hosted Classes**

All concrete classes in OpsMgr are defined as either **hosted** or **non-hosted**. Some classes can only exist when their parent class is present. Such classes are said to be **hosted** classes. For example, a Windows Service cannot exist without the Windows Operating System in which it resides. This concept will be covered at great depth throughout the series as necessary. A non-hosted class means that instances of the class are not hosted by an instance of another class. Non-hosted classes must have at least one key property defined (with a unique name) in order to uniquely identify the object class in Operations Manager. An example of a non-hosted class is the “Windows Computer” class.

**NOTE:**

If a class is hosted, it cannot exist without its host. *So, until the instances of the hosting class are discovered, instances of the hosted class cannot be discovered.*

**Singleton Classes**

The **Singleton Class** isn’t used as often as hosted classes in System Center Operations Manager 2007 because only a single instance of the class can exist in the entire management group. There is no discovery for this class, nor the ability to define a key property for it. The most common example of a singleton class in OpsMgr is a Group. Groups will be discussed in more detail later in the authoring process.
Classes, Inheritance and Specialization

Two more important concepts related to classes that you must understand are \textit{Specialization} and \textit{Inheritance}. Let us examine these in a non-technical example – Clothing. Everyone wears clothing.

All items of Clothing have a few characteristics (properties) in common:

- Material – Cotton, linen, leather, etc.
- Color – Black, brown, etc
- Gender - Men’s, women’s or unisex

Based on this information, you could define \textit{Clothing} as an \textit{abstract class} with all of these properties, as shown in the figure below:

![Figure 3 - Clothing Class and it's Class Properties](image)

There are several common types of clothing that can be more specifically defined. These items share all the properties of \textit{Clothing}, but have unique properties of their own:

- For example, a \textit{Shirt} can have long sleeve or short sleeves. It can also be a button down or pullover type.
- \textit{Pants} for example, can be long or short. Like shirts, they share all the common properties of \textit{Clothing} defined above.
- This exercise can be repeated for many types of clothing, including shoes, belts, undergarments, etc.

Because all items of clothing share common properties of \textit{Clothing}, you could create classes for Shirt, Pants, Shoes, etc. based on \textit{Clothing}. In other words, you could use \textit{Clothing} as the “base class” for all of these more specific types of clothing.
Because these object types (classes) define more specific types of Clothing, these classes are said to be “specialized from” the Clothing class. This is known as specialization.

Notice in the figure above that the Shirt, Pants and Belt classes automatically assume or “inherit” the properties of the Clothing class on which they are based. This is known as inheritance.

How many classes do you need? Should you use classes or properties?

How many object classes you create in your application model depends on your monitoring requirements. If long sleeve and short sleeve shirts have different monitoring requirements, they could be defined as separate object types. If not, they should be defined as a single object class. The same logic would carry over to a technical example. If you are monitoring 5 types of routers and each have different requirements for monitoring device health, 5 object classes could be necessary. On the other hand, if they can all be monitored in the same way, a single object class would suffice.
Properties to describe the unique device characteristics could be defined on the class to allow more specific identification of the object as necessary, while avoiding an unnecessarily complex class model.

**What can be learned from this example?**

There are two important lessons to be learned here. The first is the function of abstract classes. They serve as a “template”, allowing MP authors to easily create classes to represent their specific applications. All the classes have their distinct properties, but it is important to understand that the distinct or “specialized” classes you create to represent the components of your application inherit all the properties of the classes upon which they are based.

The second is that the simplest class model required to support your monitoring objectives is the best. Classes should be created only for objects targeted for monitoring. Additional classes are only needed for object types with distinct monitoring requirements.

**What are relationships?**

Now that classes and properties have been defined, it is an appropriate time to discuss relationships. Relationships are used to describe how object classes in a Management Pack are related to one another.

There are three types of relationships available in Operations Manager 2007:

<table>
<thead>
<tr>
<th>Relationship Type</th>
<th>Reference Type ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reference</td>
<td>System.Reference</td>
</tr>
<tr>
<td>Containment</td>
<td>System.Containment</td>
</tr>
<tr>
<td>Hosting</td>
<td>System.Hosting</td>
</tr>
</tbody>
</table>

**Table 1 – Relationship Types in OpsMgr 2007**

When a new relationship is declared in a Management Pack, it must use one of these three relationship types as its base. All relationship types are directional in nature and require a source and target class.
**Reference Relationship**
The reference relationship is the most general of the available relationship types. It is used when the classes are not dependant on each other. A good example is the relationship of two databases replicating with one another (log shipping or mirror).

Reference relationships are many-to-many; this means that you can have several targets for the same host.

**NOTE:**
The health state cannot be rolled up across a reference relationship. However, additional properties can be defined to the relationship.

**Containment Relationship**
The containment relationship type declares that one class can be contained within another class, although it is not required for the existence of the target class. An example for a containment relationship is a group: A computer can be contained in a group, but the same computer can exist even if the computer group doesn’t exist.

This is a many-to-many relationship. An object can be the target of multiple containment relationships. For example, a database object can be a member of multiple groups.

**NOTE:**
The health state can be rolled up across a containment relationship.

**Hosting Relationship**
The hosting relationship type is the most specific of the relationship types in Operations Manager. The hosting relationship type declares that an instance of a class can exist only when it is hosted within another class. An example of a hosting relationship is SQL databases. There is a class that represents SQL server instance and the class that represents the SQL database class that is hosted by the SQL server instance class.

The hosting relationship is a *one-to-many relationship*. For example, an instance of the SQL database class can be hosted by only one SQL DB Engine (SQL instance). As mentioned in the discussion of classes, when a class is defined in Operations Manager 2007, it is marked as either *hosted* or *non-hosted*. Classes that are the target of a hosting relationship are *hosted classes*.

**NOTE:**
The health state can be rolled up across a hosting relationship.
**The Health Model and Health Rollup**

The Health Model of the application is defined through health rollup. The health of a hosted object class is not automatically rolled up to the parent class – at least not in all cases as you’ll see in a moment. It is important to understand what is required to define the health rollup of the components in our Management Pack.

Health rollup requires two elements:

- **Relationships** – Health of the instances of one class can only be rolled up to another class if there is a hosting or containment relationship.
- **Rollup Monitors** – A Dependency Rollup Monitor must be created to roll up the health of one class to another.

**Should health be “rolled up” across classes?**

It does not always make sense to roll up the health of a hosted class to its parent. Whether or not health of one class should be rolled up to another should be given careful consideration by the MP author, as each monitoring scenario is unique. For example, just because a Certificate Store is unhealthy does not mean the Windows Computer that hosts the store is unhealthy. In this case, the MP author may choose not to roll up health of the child class to the parent class.

On the other hand, in the case of a server functioning solely as the host of a particular application (such as MS SQL Role), if the application role is unhealthy, the sole function of that computer is in an error state. In this case, health rollup of the child class to its host would be appropriate.

Determining when health roll up is or is not appropriate is at the discretion of the MP author and application experts assisting in the MP modeling process. Many MP authors choose to roll up health (even if strictly speaking it is not required), because operators tend to overlook unhealthy states if they are not reflected on the root object (the computer in this case).

The combination of classes, relationships and rollup monitors implement the Health Model of the application. The details of how to implement the Health Model will be discussed in greater detail in part 3 of this series as you implement monitors in the Management Pack.
Choosing a Base Class for your Object Classes

When you choose the base classes for the object classes in our Management Pack, it is important to understand that Microsoft has provided some common base classes for Windows applications to make this task much easier. Choosing a base class for your custom class is easier if you understand the role of the application on a computer and how the health of the application should affect (or not affect) the health state of the computer. To illustrate this point, let’s review some of the common base classes for Windows applications (and application components) in Operations Manager.

There are three commonly used base classes that can be used in a Windows application monitoring scenario such as this:

- **Windows Computer Role** – Which is the Primary function of server and it’s hosted by Windows Computer, and has an **Automatic Health Rollup to Computer**.
- **Windows Local Application** – that is where the applications are often installed and it’s hosted by Windows Computer, but **hasn’t got a Health Rollup to Computer**.
- **Windows Application Component** - Component of a local application or computer role, there aren’t any type of relationships here, they need to be created by hand.

Below is a more complete list of classes that are commonly used as base classes by authors creating custom Management Packs.

<table>
<thead>
<tr>
<th>Class</th>
<th>Management Pack</th>
<th>When to use it</th>
</tr>
</thead>
<tbody>
<tr>
<td>Microsoft.Windows.ComputerRole</td>
<td>Microsoft.Windows.Library</td>
<td>An application that is a Windows-based server role, such as SQL Server, DNS Server, Exchange Server</td>
</tr>
<tr>
<td>Microsoft.Windows.LocalApplication</td>
<td>Microsoft.Windows.Library</td>
<td>A standalone application that is not a server role</td>
</tr>
<tr>
<td>Microsoft.Windows.ApplicationComponent</td>
<td>Microsoft.Windows.Library</td>
<td>An application component that is part of a larger application.</td>
</tr>
<tr>
<td>Microsoft.SystemCenter.InstanceGroup</td>
<td>Microsoft.SystemCenter.Library</td>
<td>Groups, which are used for logical grouping of similar components These are Unhosted and Singleton</td>
</tr>
<tr>
<td>Microsoft.SystemCenter.ComputerGroup</td>
<td></td>
<td></td>
</tr>
<tr>
<td>System.ApplicationComponent</td>
<td>System.Library</td>
<td>Used for health rollup of other classes Unhosted and Singleton</td>
</tr>
</tbody>
</table>

Table 2 - Common Base Classes in OpsMgr 2007
What are the advantages of creating classes on these base classes? The primary difference has to do with *health rollup*.

If you use the **Windows Computer Role** as the **base class**, the class will automatically be hosted by the Windows Computer class. And the health state of this class will automatically roll up to the Windows Computer class. **Health Rollup to Computer**, this means that any instance of a class that is targeted to this main class will appear automatically on the health explorer of the windows computer, avoiding the need to create relationships or health rollups because they are already created. This base class is the most used in System Center Operations Manager 2007.

Any Class using **Windows Local Application** as its base class will also be hosted by the Windows Computer class. However, its health state will **NOT automatically roll up to the Windows Computer class**.

By targeting a Class to **Windows Application Component** as its base class, the resulting class will not inherit any Windows Computer properties or any **Health Rollup**.

**How do the Service Model and Health Model affect one another?**

Every time you add an object class, the Service Model for the MP becomes more complex. For every class, at least one more relationship must be defined, and the Health Model adjusted to define health rollup strategy to include the new class(es).
Mapping Components to Classes for the Certificates MP

With the discussion of Classes and Relationships out of the way, you are ready to change focus of the discussion to monitoring digital certificates. You will to start defining the logical model of the Management Pack, which begins with classes. The required classes are determined by the logical model of the application components you intend to monitor.

Please remember that you are still in the planning phase at this point. Once the application components, relationships and health rollup have been defined, you will move into the MP Authoring console and begin implementing our design.

Classes for the PKI Certificate Management Pack

You can begin by mapping the “Abstract Class” Certificate Services Application Component which will specialize from the Windows Local Application Component. This is done because all the relationships can be created manually between hosted and hosting instances. The Certificate Services Application Component class is an abstract class that was created to serve as the basis for other classes in this MP. Since several certificate types sharing common properties are possible, this will make it possible to create these classes with less work.

As the goal is monitoring of digital certificates, what application components should be targeted for monitoring?

- Windows Computer – That is where certificates and certificate stores reside
- Certificate Store – That is hosted on the Windows Computer
- Certificates – That are hosted on the Certificate Store

The Windows Computer class is one you don’t need to create it because it’s already exists in OpsMgr after importing the Operating System Management Pack. However, you do need classes to represent the other components:

- The Certificate Store and
- (Digital) Certificates,
This table shows the classes and their properties you are going to create in the Authoring console later in the process.

<table>
<thead>
<tr>
<th>ID</th>
<th>Display Name</th>
<th>Attributes</th>
<th>Accessibility</th>
</tr>
</thead>
</table>

Table 3 – Classes in PKI Certificate Validation MP

**NOTE:**
When creating classes in the MP Authoring console “Accessibility” by default is set to “private”, but for the class to be used on distributed application it must the set to “public”.

When creating classes that have hosting relationships, create the class with no attribute, then create the hosting relationship. This is only necessary when you choose the “Custom Class” option in the Authoring console. The wizards for some of the common base classes (mentioned in Table 2) will handle this for you.
Defining Properties for Classes

As the Classes are well underway, you need to start thinking of which properties they would have. For the Certificate Store, it is logical to define a Name property, which matches Windows name as it is seen in the OS. We have chosen the CertStoreID as the Key property (from the registry) as we need it later to monitor the store. This is a good transition to the next concept – key properties.

What is a Key Property?

A key property is a unique identifier that enables the discovery process to discover multiple instances of a class on a single host. A key property is not needed unless multiple instances of a class can be discovered on a single host.

Data Types

The Type field in a class property refers to the data type. While the Authoring console will always default to “string”, it is important to set the appropriate type based on the expected range of possible values. The data types available for class properties are shown in Table 4.

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>int</td>
<td>Contains an integer - which is a whole number without decimal places.</td>
</tr>
<tr>
<td>decimal</td>
<td>A number with one or more decimal places. For example, 5.25</td>
</tr>
<tr>
<td>double</td>
<td>A numeric data type representing very small to very large numbers, including fractions. For example, 5884444.24546.</td>
</tr>
<tr>
<td>string</td>
<td>Plain text</td>
</tr>
<tr>
<td>datetime</td>
<td>Date and time data</td>
</tr>
<tr>
<td>guid</td>
<td>A 32-digit alphanumeric identifier for objects represented in OpsMgr.</td>
</tr>
<tr>
<td>bool</td>
<td>True or false value (0 or 1)</td>
</tr>
</tbody>
</table>

Table 4 – Data Types
Certificate Store Class Properties

The Computer Store class will have the properties defined on (and inherited from) the Certificates Service Application Component class. The name of the certificate store as maintained in the physical location of the store (the Registry), which is guaranteed unique on a single computer, is used as the key property to uniquely identify the Certificate Store on a specific host.

<table>
<thead>
<tr>
<th>Name</th>
<th>Display Name</th>
<th>Key</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CertStoreName</td>
<td>Store Name</td>
<td></td>
<td>string</td>
<td>Friendly Certificate Store Name</td>
</tr>
<tr>
<td>CertStoreID</td>
<td>Store ID (Internal)</td>
<td>Yes</td>
<td>string</td>
<td>Operating System Certificate Store Key</td>
</tr>
</tbody>
</table>

Table 5 - Certificate Store Class Properties

Certificate Class Properties

For the Certificates class you need also to add properties relevant for unique identification and monitoring of certificates. If you open the console you can see many properties you can use.

Figure 7 – Certificate Properties
Of all the certificate properties available, the MP authors have chosen to include the following in the Management Pack:

<table>
<thead>
<tr>
<th>Name</th>
<th>Display Name</th>
<th>Key</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CertIssuedTo</td>
<td>Issued to</td>
<td></td>
<td>string</td>
<td>Issued To (Subject)</td>
</tr>
<tr>
<td>CertIssuedBy</td>
<td>Issued by</td>
<td></td>
<td>string</td>
<td>CA that issued the certificate</td>
</tr>
<tr>
<td>CertValidFrom</td>
<td>Valid from</td>
<td></td>
<td>string</td>
<td></td>
</tr>
<tr>
<td>CertValidTo</td>
<td>Valid to</td>
<td></td>
<td>string</td>
<td></td>
</tr>
<tr>
<td>CertTumbPrint</td>
<td>Thumbprint*</td>
<td>Yes</td>
<td>string</td>
<td></td>
</tr>
<tr>
<td>CertSerial</td>
<td>Serial N°</td>
<td></td>
<td>string</td>
<td></td>
</tr>
<tr>
<td>CertStatus</td>
<td>Status</td>
<td></td>
<td>string</td>
<td>Certificate status (valid, invalid, expired tec.)</td>
</tr>
<tr>
<td>CertStore</td>
<td>Certificate Store</td>
<td></td>
<td>string</td>
<td>Friendly Certificate Store Name</td>
</tr>
</tbody>
</table>

Table 6 - Computer Store Class Properties

The Thumbprint is particularly well-suited as a key property, as it is guaranteed to be unique across all certificates (in a certificate store) on a given host. The same is not true of the certificate serial number, as it is only unique per issuing CA. You could potentially have duplicate serials numbers on certificates issued by different CAs!

When defining the component properties of a class, ask yourself these questions:

- Will there be only one instance of the object in the Management Group?
- Will there be more than one instance of the object on the host?
- If so what key properties are required to uniquely identify the object for monitoring or reporting?

Remember, if there is only one instance of the object on a host, then the key property is not required.
This picture shows the modeling of the classes and their properties you are about to create.

**Certificate Services Application Component**
(*abstract class for inheritance only*)

**Certificate Store**
- CertStoreName
- CertStoreID (Key Property)

**Certificate**
- CertFriendlyName
- CertIssuedTo
- CertIssuedBy
- CertValidFrom
- CertValidTo
- CertThumbPrint (Key Property)
- CertSerial
- CertStatus
- CertStore

*Figure 8 - Classes in the PKI Certificate Validation MP*
Defining Relationships

Defining the object classes for the PKI Certificate Validation MP is only the first step. How objects relate to one another is every bit as important as the objects themselves. Due to the base classes chosen for the object classes in this MP (everything derives from the Windows Application Component), the relationships must be created manually.

Based on the information presented in “Choosing a Base Class for your Object Classes” earlier in this document, we know that the Windows Local Application Component class does not inherit any relationships by default. This was desirable in this situation, as it allows you as the MP author define the desired relationships between object classes.

- The Certificate Store cannot exist without a computer to host the operating system and Certificate Store. Since the Certificate Store cannot exist without the hosting component, a **Hosting** relationship is required.
- Likewise, a Certificate cannot exist without the Certificate Store in which it resides, thus another **Hosting** relationship is required.

<table>
<thead>
<tr>
<th>ID</th>
<th>Display Name</th>
<th>Source (Hosting Class)</th>
<th>Target (Hosted Class)</th>
<th>Accessibility (Internal or Public)</th>
</tr>
</thead>
</table>

Table 7 - Relationships for the PKI Certificate Validation MP
Defining the Health Model

The Health Model is represented by classes, relationships and health rollup. Because health rollup is implemented through a combination of relationships and rollup monitors, the health rollup model will not be fully implemented until the necessary rollup monitors are created in part 3 of this series.

Figure 9 - Health Model for PKI Certificates MP
Naming a Management Pack

The first step in creating a Management Pack in the Authoring console is assigning it a name.

There are a few best practices for naming a Management Pack which have been suggested by Microsoft. Usually the name is giving in three steps much like a FQDN name. In which the first part is the name of the vendor of the Management Pack, the second part the product we are monitoring, and finally the version of what we are monitoring if applicable. These three elements should be separated by periods.

Vendor.Application.Version

The name adopted for the sample Management Pack you develop in this series is,

SystemCenterCentral.Utilities.Articles

- Part one of the name (the Vendor) – There is no vendor here but the community web site SystemCenterCentral
- The second portion of the name (Product) - In this case the Certificates do not have a version as with server applications like Microsoft SQL, so Utilities was chosen, because it’s a utility Management Pack.
- For the third part (version) Articles was chosen. All components of the Management Pack should continue to follow the namespace provided for the MPs (class names, relationships, rules, monitors, etc).

Why “Articles”? This was chosen for one simple reason. So the samples you create through the course of this article series will not come into conflict with the actual PKI Certificate Validation MP if you have it installed in the same test environment.

NOTE:
Vendor.Product.Version is defined by Microsoft as the preferred naming convention for a Management Pack. You may choose your own naming convention, but it is important to stay to the chosen naming convention throughout the authoring process.
Building the MP in the Authoring Console

With the mapping of the classes and their properties done you can finally begin building the Management Pack. And the first step is to open the Authoring Console.

Start by creating the Management Pack:

1. From the Start Menu, select **Programs -> System Center Operations Manager R2 -> Authoring Console**.
2. From the **File** menu, select **New -> Empty Management Pack**.
3. In the Management Pack Identity field, enter **SystemCenterCentral.Utilities.Articles** and click **Next**.
4. In the **Display Name** field, enter **PKI Certificate Validation DEMO** and then click **Create**.

![MP Authoring Console in OpsMgr 2007 R2](Figure 10 - MP Authoring Console in OpsMgr 2007 R2)

Setting Compatibility

For the purpose of this article, we are going to use the R2 version of the Authoring Console. It is possible to author a Pre R2 compatible Management Pack with the R2 version of the Authoring Console. If you do author a Management Pack in the R2 console (but using no R2-specific modules), it will be possible to use this MP in an OpsMgr 2007 SP1 or even RTM environment.
However, to use the MP in a downlevel version of OpsMgr 2007, you will need to change the versions of the MPs referenced by this Management Pack back to their SP1 or RTM version numbers so it may be imported into a System Center Operations 2007 SP1 or RTM environment.

![Figure 11 - MP References (in the MP Authoring Console)](image)

**To change dependencies for OpsMgr RTM / SP1 Compatibility:**

Simply change the 6.1.xxxx values to 6.0.5000.0, and the MP is ready for testing in an RTM or SP1 environment. You can make this change in the MP Authoring Console by performing the following steps:

1. Select **File ➔ Management Pack Properties**.
2. On the **References** tab, click on each Management Pack to expand its properties.
3. Click on the **Values** and enter the desired minimum version requirement based on the versions of OpsMgr in which your MP should be functional.
OpsMgr 2007 versions are shown in Table 8.

<table>
<thead>
<tr>
<th>Version</th>
<th>Version Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>RTM</td>
<td>6.0.5000.0</td>
</tr>
<tr>
<td>SP1</td>
<td>6.0.6278.0</td>
</tr>
<tr>
<td>R2</td>
<td>6.1.7221.0</td>
</tr>
</tbody>
</table>

Table 8 – OpsMgr Version Numbers

**NOTE:**
If you lower the version requirement, make sure you have not used a feature only present in newer versions of the product. To make this determination, review the release notes for the SP1 and R2 versions of the product for a list of updates.

Create the Object Classes and Relationships
You will begin by creating the object classes used in the MP.

Create the Certificate Services Component Class
The first class you will define is the Certificate Services Component which is an abstract class that will serve as a base class for some of the concrete classes in this MP.

*To create the Certificate Services Component class, perform the following steps:*

1. In the MP Authoring console, select the Service Model workspace and then the Classes node.
2. In the Object pane (top center) of the Authoring Console, right-click and select New -> Custom Class. Notice the namespace established for this MP is carried forward automatically.
4. On the General tab, enter the following information
   - In the Base Class field, select Microsoft.Windows.ApplicationComponent.
   - In the Display Name field, enter Certificate Services Component.
   - In the Attributes section, select the Abstract Class checkbox, which defines Certificates as an abstract class.
5. Accept the default values on the Properties, Images and Relationships tabs and click OK to create the class. Since the base class is hosted by the Windows Computer class, the hosting relationship for this class has been defined automatically.
Create the Certificate Store Class

The next class you will define is the **Certificate Store** class, which represents the repository for digital certificates are stored on the Windows computer. In the process, you will also define the hosting relationship between the Certificate Store (the hosted class) and Windows Computer (hosting) class.

To create the Certificate Store class, perform the following steps:

1. In the MP Authoring console, select the **Service Model** workspace and then the **Classes** node.
2. In the Object pane (top center) of the Authoring Console, right-click and select **New -> Custom Class**.
4. On the **General** tab, enter the following information
   - In the Base Class field, select `SystemCenterCentral.Utilities.Articles.ApplicationComponent` which you created in the previous step.
   - In the **Display Name** field, enter **Certificate Store**.
   - In the Attributes section, change the Accessibility value to **Public**.
Next, add Properties to the Certificate Store class:

1. Move to the Properties tab on `SystemCenterCentral.Utilities.Articles.CertificateStore`. Right click on the on blank space in the left pane, select Add Property from the context menu and the Choose a unique identifier… dialogue will appear....
2. Enter the property Name in the field provided. Create the properties listed in Table 9 using the guidance below.
3. Accept the default values on the Images and Relationships tabs and click OK to create the class.

<table>
<thead>
<tr>
<th>Name</th>
<th>Display Name</th>
<th>Key</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CertStoreName</td>
<td>Store Name</td>
<td></td>
<td>string</td>
<td>Friendly Certificate Store Name</td>
</tr>
<tr>
<td>CertStoreID</td>
<td>Store ID (Internal)</td>
<td>Yes</td>
<td>string</td>
<td>Operating System Certificate Store Key</td>
</tr>
</tbody>
</table>

Table 9 - Certificate Store Class Properties

Tips when creating properties:
The naming process is the same as in the creation of a class, so be careful when defining the Name of the property. Spaces in the Name field are not allowed (but are allowed in Display Name and Description fields). The Name is the unique identifier for the property, and once created cannot be changed.
4. When creating the properties a window appears in the right pane where you can set Display Name, Description, Type values for the property. The process for creating a key property is the same as that for creating a non-key property, except the Key checkbox is selected.

<table>
<thead>
<tr>
<th>SystemCenterCentral.Utilities.CertificateStore</th>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CertStoreName</td>
<td>Name</td>
<td>CertStoreID</td>
</tr>
<tr>
<td>CertStoreD</td>
<td>Display Name</td>
<td>Store ID</td>
</tr>
<tr>
<td>System.Entity</td>
<td>Description</td>
<td>Certificate Store Identifier</td>
</tr>
<tr>
<td>DisplayName</td>
<td>Type</td>
<td>string</td>
</tr>
<tr>
<td></td>
<td>Case Sensitive</td>
<td>[ ]</td>
</tr>
<tr>
<td></td>
<td>Length</td>
<td>255</td>
</tr>
<tr>
<td></td>
<td>Key</td>
<td>[ ]</td>
</tr>
</tbody>
</table>

Figure 15 - Defining Property as a Key Property

As you can see, when the Key checkbox is selected, a key appears on the name of the property in the left pane.

**NOTE:**
When you create a Key property, remember to select the Key checkbox.

Finally, to create the relationship between the Certificate Store and Windows Computer classes, perform the following steps:

1. In the MP Authoring console, select the Service Model workspace and then the Relationships node.
2. In the Object pane (top center) of the Authoring Console, right-click and select New -> Custom Relationship.
4. On the General screen, enter the following values and click Next.
   a. Base Relationship: System.Hosting
   d. Display Name: Windows Computer Host Certificate Store
   e. Description: Defines the hosting relationship between the Windows Computer and Certificate Store classes
   f. In the Attributes section, change the Accessibility value to Public
Create the Object Classes and Relationships

1. In the MP Authoring console, select the Service Model workspace and then the Classes node.
2. In the Object pane (top center) of the Authoring Console, right-click and select New -> Custom Class.
4. On the General tab, enter the following information
   - In the Base Class field, select SystemCenterCentral.Utilities.Articles.ApplicationComponent.
   - In the Display Name field, enter Certificate.
   - In the Attributes section, change the Accessibility value to Public.

Next, add Properties to the Certificate class:

1. Move to the Properties tab on SystemCenterCentral.Utilities.Articles.Certificate. Right click on the on blank space in the left pane, select Add Property from the context menu and the Choose a unique identifier... dialogue will appear....
2. Enter the property Name in the field provided.

Figure 16 - Hosting Relationship Properties for the Certificate Store Class

4. Click Finish to create the relationship between the Windows Computer and Certificate Store classes.

Create the Certificate Class

The final class you will define is the Certificate class, which represents the digital certificates that will be discovered and monitored by the MP.

To create the certificate class, perform the following steps:

1. In the MP Authoring console, select the Service Model workspace and then the Classes node.
2. In the Object pane (top center) of the Authoring Console, right-click and select New -> Custom Class.
4. On the General tab, enter the following information
   - In the Base Class field, select SystemCenterCentral.Utilities.Articles.ApplicationComponent.
   - In the Display Name field, enter Certificate.
   - In the Attributes section, change the Accessibility value to Public.

Next, add Properties to the Certificate class:

1. Move to the Properties tab on SystemCenterCentral.Utilities.Articles.Certificate. Right click on the on blank space in the left pane, select Add Property from the context menu and the Choose a unique identifier... dialogue will appear....
2. Enter the property Name in the field provided.
3. In the properties screen provided, set the property values as shown in Table 9. If the Key column = True, be sure to check the Key checkbox for the property, which denotes the property as a key property of the class.

4. Repeat these steps to create all properties listed in Table 10 as illustrated for the Certificate Store class.

<table>
<thead>
<tr>
<th>Name</th>
<th>Display Name</th>
<th>Key</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CertIssuedTo</td>
<td>Issued to</td>
<td></td>
<td>string</td>
<td>Issued To (Subject)</td>
</tr>
<tr>
<td>CertIssuedBy</td>
<td>Issued by</td>
<td></td>
<td>string</td>
<td>CA that issued the certificate</td>
</tr>
<tr>
<td>CertValidFrom</td>
<td>Valid from</td>
<td></td>
<td>string</td>
<td></td>
</tr>
<tr>
<td>CertValidTo</td>
<td>Valid to</td>
<td></td>
<td>string</td>
<td></td>
</tr>
<tr>
<td>CertTumbPrint</td>
<td>Thumbprint</td>
<td>Yes</td>
<td>string</td>
<td></td>
</tr>
<tr>
<td>CertSerial</td>
<td>Serial N°</td>
<td></td>
<td>string</td>
<td></td>
</tr>
<tr>
<td>CertStatus</td>
<td>Status</td>
<td></td>
<td>string</td>
<td>Certificate status (valid, invalid, expired tec.)</td>
</tr>
<tr>
<td>CertStore</td>
<td>Certificate Store</td>
<td></td>
<td>string</td>
<td>Friendly Certificate Store Name</td>
</tr>
</tbody>
</table>

Table 10- Certificate Class Properties

5. When all properties have been added to the Certificate class, click OK to create the class.

*Finally, to create the relationship between the Certificate and Certificate Store classes, perform the following steps:*

1. In the MP Authoring console, select the Service Model workspace and then the Relationships node.
2. In the Object pane (top center) of the Authoring Console, right-click and select New -> Custom Relationship.
4. On the General screen, enter the following values and click Next.
   a. Base Relationship: System.Hosting
   d. Display Name: Certificate Store Hosts Certificate
   e. Description: Defines the hosting relationship between the Certificate Store and Certificate classes
   f. In the Attributes section, change the Accessibility value to Public
5. Click Finish to create the relationship between the Certificate and Certificate Store classes.

Congratulations! You have now successfully created the object classes and relationships for the PKI Certificate Validity MP.

**Testing Your Management Pack**

Now it is time to test your work. At this point, the MP consists of the classes and relationships that will be discovered and monitored when then MP is complete. If you were to import this MP into a live management group, nothing much would happen.

However, you should at least verify the MP is syntactically correct. To verify the MP is valid, you can run **MP Verify** from within the Authoring console. On the Tools menu, select **Verify**. The result is a report of any deficiencies in the MP that will prevent successful import into an OpsMgr management group.

If you want to go a step further:

- Import your MP into a live management group
- Open the Distributed Application Designer.
- Check for the classes, which should appear in as in the figure shown below.

![Hosting Relationship Properties for the Certificate Class](image)

**Figure 17- Hosting Relationship Properties for the Certificate Class**
Conclusion
In this article (which was a bit long) we covered the concepts, processes and decision criteria for authoring a Management Pack that will meet your monitoring objectives, while avoiding unnecessary complexity. Understanding the concepts of classes and relationships, as well as the commonly used base classes is the key here. Understanding the advantages and disadvantages and default behavior of each of the commonly used base classes avoids problems at later stages of the process. Without a good working knowledge of these concepts it is very difficult to use the MP Authoring Console to build a Management Pack.

A solid understanding of the concepts behind each step of the MP authoring process, along with a little planning will ensure predictable results in our production environment. Your results should become more predictable with practice.
In Part 2 - Discovering Application Components, we will drill down into authoring discoveries for the PKI Certificate Validation MP. In a mix of discussion and hands on demonstration, we will discuss:

- Deep dive into discovery methods in Operations Manager
- Designing a ‘layered’ discovery process to minimize resource consumption
- Hands on creation of discoveries for the Certificates MP

If you are looking for hands-on, real world demonstration of script-based discovery, then part 2 is what you are looking for.

**Feedback**

If you have a question regarding the contents of this article, please post your questions the MP Development Forum at System Center Central at the URL below.

**MP Development Forum at SystemCenterCentral.com**

## Appendix A – MP Authoring Job Aids

In this appendix there are some tables for helping in the process of creating classes and their properties.

### Tables for creating Classes

<table>
<thead>
<tr>
<th>ID</th>
<th>Display Name</th>
<th>Description</th>
<th>Attributes</th>
<th>Accessibility (Internal or Public)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Tables for Class properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Display Name</th>
<th>Key</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Tables for Relationships

<table>
<thead>
<tr>
<th>ID</th>
<th>Display Name</th>
<th>Source (Hosting Class)</th>
<th>Target (Hosted Class)</th>
<th>Accessibility (Internal or Public)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
