SMS 2003 Recipes

A Problem-Solution Approach

Greg Ramsey
and Warren Byle

Apress®

SMS 2003 Recipes: A Problem-Solution Approach

ISBN-13 (pbk): 978-1-59059-712-5

ISBN-10 (pbk): 1-59059-712-5

Printed and bound in the United States of America 9 8 7 6 5 4 3 2 1

Lead Editor: Jonathan Gennick
Technical Reviewers: Chris Minaugh, Michael Niehaus, Duncan Mcalynn
Editorial Board: Steve Anglin, Ewan Buckingham, Gary Cornell, Jason Gilmore, Jonathan Gennick, Jonathan Hassell, James Huddleston, Chris Mills, Matthew Moodie, Dominic Shakeshaft, Jim Sumser, Keir Thomas, Matt Wade
Project Manager: Richard Dal Porto
Copy Edit Manager: Nicole LeClerc
Copy Editors: Heather Lang, Nicole LeClerc
Assistant Production Director: Kari Brooks-Copony
Senior Production Editor: Laura Cheu
Compositor: Kinetic Publishing Services, LLC
Proofreader: April Eddy
Indexer: Present Day Indexing
Cover Designer: Kurt Krames
Manufacturing Director: Tom Debolski

Distributed to the book trade worldwide by Springer-Verlag New York, Inc., 233 Spring Street, 6th Floor, New York, NY 10013. Phone 1-800-SPRINGER, fax 201-348-4505, e-mail `orders-ny@springer-sbm.com`, or visit `http://www.springeronline.com`.

For information on translations, please contact Apress directly at 2560 Ninth Street, Suite 219, Berkeley, CA 94710. Phone 510-549-5930, fax 510-549-5939, e-mail `info@apress.com`, or visit `http://www.apress.com`.

The source code for this book is available to readers at `http://www.apress.com` in the Source Code section.

CHAPTER 3

Collections

Collections are an integral part of SMS. Without properly defined collections, you run the risk of installing software (and rebooting systems) on systems you did not intend to target. SMS collections may appear to be simple to the novice SMS administrator, but the more you use SMS, the more you will realize the power and complexity of collections.

Collections are primarily used for software distribution and SMS web reports. Collections of systems or users are created using direct membership rules, query membership rules, or a combination of both.

A *direct membership* rule explicitly names a single resource and is a *static* member of the collection. For example, if a request to install Microsoft Office Visio Standard 2003 is approved for computer A, you could add a direct membership rule to add computer A to your `VisioStd2003` collection. Provided you already have an advertisement for installing Visio Standard 2003 to that collection, the software would automatically install, minimizing the effort required by the SMS administrator.

A *query membership* rule can contain multiple resources, and it can also be *dynamic*, in that the collection membership can be scheduled to refresh on a defined schedule. Say, for example, that you want to ensure all PCs have the latest version of Adobe Acrobat Reader. You could simply create a query-based collection that contains all systems that do not have the latest version of Adobe Acrobat Reader and advertise the installation to this collection. When software inventory, hardware inventory, or both (depending on how the query-based collection is written) are updated on the client and forwarded to the SMS primary site, the updated data will be available for the next time the collection updates its membership. (By default, collection membership information is updated every 24 hours.) When collection membership has been updated, the collection will no longer contain systems that received the updated installation, provided updated inventory has been received from the client. Eventually, this collection would be reduced to only a few systems that have not yet run the installation (typically, mobile clients take a little longer to run the installations and report updated inventory to the SMS site).

Remember that many collections may never be used for software distribution purposes, but they are equally as important in your SMS environment. Web reports can be created that prompt the user at runtime to select an appropriate collection, which will *filter* the report to only show data related to the selected query. For example, you could create a web report to show all applicable security patches for a specific collection, or you could allow the user to select the desired collection just before executing the report.

3-1. Creating a Collection

Problem

You want to create a collection.

Solution: Using a Graphical Interface

1. Open the SMS Administrator console.
2. From the SMS Administrator console, expand Site Database (<*Site Code*>) and then Collections.
3. Right-click Collections and select New ➤ Collection from the menu.
4. On the General tab, enter the name of the new collection and any comments.
5. Use the Membership rules tab to create query or direct membership rules now, or you can add them later.
6. Click OK to create the new collection.

Solution: Using VBScript

Example 3-1 demonstrates how to create an empty SMS collection and associate it with a parent collection in the tree.

Example 3-1. *CreateCollection.vbs*

```
 strSMSServer = <SMSServer>
strParentCollID = "COLLROOT"
'This example creates the collection in
'the collection root.  Replace COLLROOT with the CollectionID
'of an existing collection to make the new collection a child

strCollectionName = "Systems Without Windows Update Agent"
strCollectionComment = "This collection contains all systems " & _
    "that do not have the Windows Update Agent installed."

Set objLoc =  CreateObject("WbemScripting.SWbemLocator")
Set objSMS= objLoc.ConnectServer(strSMSServer, "root\sms")
Set Results = objSMS.ExecQuery _
    ("SELECT * From SMS_ProviderLocation WHERE ProviderForLocalSite = true")
For each Loc in Results
    If Loc.ProviderForLocalSite = True Then
        Set objSMS = objLoc.ConnectServer(Loc.Machine, "root\sms\site_" & _
            Loc.SiteCode)
    end if
Next

Set newCollection = objSMS.Get("SMS_Collection").SpawnInstance_()

newCollection.Name = strCollectionName
newCollection.OwnedByThisSite = True
newCollection.Comment = strCollectionComment
path=newCollection.Put_

'the following two lines are used to obtain the CollectionID
'of the collection we just created
Set Collection=objSMS.Get(path)
strCollID= Collection.CollectionID
'now we create a relationship between the new collection
'and its parent.
```

```
Set newCollectionRelation = objSMS.Get _
    ( "SMS_CollectToSubCollect" ).SpawnInstance_()
newCollectionRelation.parentCollectionID = strParentCollID
newCollectionRelation.subCollectionID = strCollID
newCollectionRelation.Put_
```

Discussion

Creating a collection is a basic function of the SMS administrator and may be the very first step in every software distribution that you do. Don't think that a collection has to be perfect before you click OK; many of our collections begin as empty "placeholder" collections that have queries added or imported at a later date.

Example 3-1 demonstrates how to create a collection. Granted, the code appears to be a bit bulky, but when you create a collection, more steps are involved than just creating it. First, we set the name and description of the new collection, and then we save the collection (`newCollection.Put_`). Then we use `Get(path)` to obtain the `CollectionID` of the collection we just created. Next, we create a relationship between the new collection and its parent. In Example 3-1, we associate the new collection with the parent `CollectionID` `COLLROOT`, which is the root collection. By associating the new collection with `COLLROOT`, the collection will appear in the root of the `Collections` node. To make your new collection a subcollection of an existing collection, simply replace `COLLROOT` in Example 3-1 with the desired `CollectionID`. The next recipe describes how to obtain the `CollectionID` of an existing collection.

■ **Caution** You must associate each collection you create to a parent. If you do not set a parent, the collection will not appear in the SMS Administrator console and will be unusable.

■ **Note** The SMS Administrator console will not allow you to create a duplicate collection name. However, when you create a new collection programmatically, there is no automatic check to verify the collection does not currently exist. You should verify that the collection name does not exist before creating a new collection programmatically.

See Also

- Recipe 3-5, "Creating a Linked Collection," describes how to create a linked collection.
- Recipe 3-7, "Creating a Direct Membership Rule," describes how to create a direct membership collection.
- Recipe 3-8, "Creating a Query-Based Membership Rule," describes how to create a query-based membership rule.
- Recipe 3-9, "Creating a Collection Based on an Existing SMS Query," demonstrates how to import an existing SMS query statement as the criteria for a new query-based collection.
- Recipe 4-1, "Creating a Package," describes how to create a package.
- Recipe 5-1, "Creating a Program," describes how to create a program.
- Recipe 6-1, "Creating an Advertisement," describes how to create an advertisement.

- The *SMS 2003 Operations Guide* and the *Microsoft Systems Management Server 2003 Administrator's Companion* by Steven Kaczmarek (Microsoft Press, 2004) provide additional information on creating a collection.
- Appendix C of the *SMS 2003 Operations Guide* provides an additional example of creating a collection using VBScript.

3-2. Listing Subcollections

Problem

You want to enumerate subcollections of a given `CollectionID` from the command line.

Solution: Using VBScript

Example 3-2 demonstrates how to list all subcollections of the desired collection, given the `CollectionID`.

Example 3-2. *ListSubcollections.vbs*

```
 strSMSServer = <SMSServer>

'If you want to start from the root collection, set
'   strCollID = "COLLROOT"
strCollID = "LAB00080"

Set objLoc =  CreateObject("WbemScripting.SWbemLocator")
Set objSMS= objLoc.ConnectServer(strSMSServer, "root\sms")
Set Results = objSMS.ExecQuery _
    ("SELECT * From SMS_ProviderLocation WHERE ProviderForLocalSite = true")
For each Loc in Results
    If Loc.ProviderForLocalSite = True Then
        Set objSMS = objLoc.ConnectServer(Loc.Machine, "root\sms\site_" & _
            Loc.SiteCode)
    end if
Next

wscript.echo strCollID & vbTAB & GetCollectionName(strCollID)

DisplaySubCollections strCollID, 3

Sub DisplaySubCollections(strCollID, intSpace)

strWQL ="SELECT col.* FROM SMS_Collection as col " & _
        "INNER JOIN SMS_CollectToSubCollect as ctsc " & _
        "ON col.CollectionID = ctsc.subCollectionID " & _
        "WHERE ctsc.parentCollectionID='" & strcollID & "' " & _
        "ORDER by col.Name"

    Set colSubCollections = objSMS.ExecQuery(strWQL) _

    For each objSubCollection in colSubCollections
        wscript.echo space(intSpace) & objSubCollection.CollectionID & _
            vbTAB & objSubCollection.Name
```

```
        DisplaySubCollections strSubCollID, intSpace + 3
    Next
End Sub

Function GetCollectionName(strCollID)
    Set objCollection = objSMS.Get _
    ("SMS_Collection.CollectionID='" & strCollID & "'")
    GetCollectionName = objCollection.Name
End Function
```

Discussion

By default, when creating an SMS advertisement and selecting a collection for deployment, all subcollections of the selected collection are also targeted. Example 3-2 demonstrates how to list all subcollections for a specified collection. This VBScript is a good example of using *recursion*. We start by displaying the base collection defined in `strCollID`, and then call `DisplaySubCollections`, passing the base collection of interest. In `DisplaySubCollections`, we query `SMS_CollectToSubCollect` to identify all collections, of which the base collection is the parent. And for each subcollection identified, we display the `CollectionID` and collection name, and then call the `DisplaySubCollections` subroutine again, but this time pass the `CollectionID` of the subcollection. By calling it this way, we can use the same subroutine to find all subcollections. The `GetCollectionName` function is used to display the collection name for each `CollectionID` displayed.

Recursion is a great tool for situations like this—where you know that you will reach an ending to the "loop." When you work with collections, the number of subcollections will eventually come to an end. SMS doesn't allow for endless loops of CollectionA referring to CollectionB, which refers to CollectionC, which refers to CollectionA, which refers to . . . well, you get the idea.

■ **Caution** When programming, only use recursion when you are certain that your basis for recursion will come to an end. Otherwise, your code may recurse infinitely, causing much the same problem as an endless loop.

See Also

- Recipe 4-3, "Enumerating Package and Package-Folder Structure," describes how to list all SMS packages.
- Recipe 5-3, "Viewing All Programs in a Package," describes how to list all programs for a package.
- Recipe 6-3, "Viewing All Advertisements," describes how to list all advertisements.

3-3. Identifying the CollectionID of a Collection

Problem

You want to identify the `CollectionID` of a collection.

Solution: Using a Graphical Interface

1. Open the SMS administrator console.
2. From the SMS Administrator console, expand Site Database (<*Site Code*>) and then Collections.
3. In the right pane, you will see the `CollectionID` listed next to the collection name.

Solution: Using VBScript

Example 3-3 demonstrates how to list all SMS collections in an SMS site.

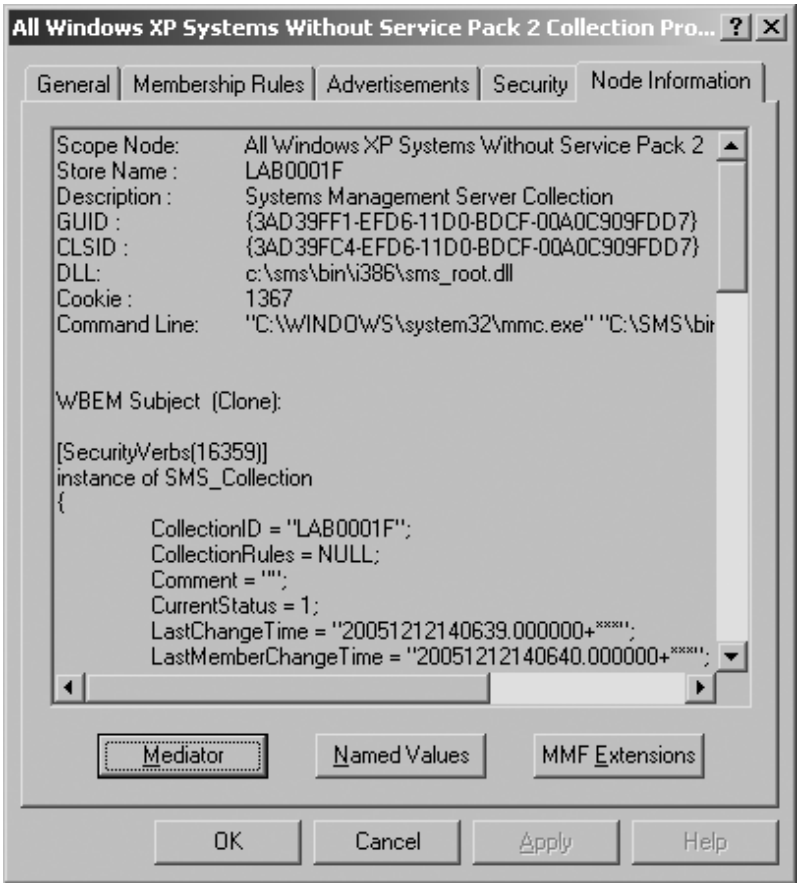
Example 3-3. *ListCollectionIDs.vbs*

```
 strSMSServer = <SMSServer>

Set objLoc =  CreateObject("WbemScripting.SWbemLocator")
Set objSMS= objLoc.ConnectServer(strSMSServer, "root\sms")
Set Results = objSMS.ExecQuery _
    ("SELECT * From SMS_ProviderLocation WHERE ProviderForLocalSite = true")
For each Loc in Results
    If Loc.ProviderForLocalSite = True Then
        Set objSMS = objLoc.ConnectServer(Loc.Machine, "root\sms\site_" & _
            Loc.SiteCode)
    end if
Next

Set colCollections = objSMS.ExecQuery _
("select * from SMS_Collection order by Name")
for each objCollection in colCollections
    wscript.echo objCollection.Name & vbTAB & _
        objCollection.CollectionID
next
```

Discussion

If you're managing SMS from only the SMS Administrator console, the `CollectionID` may be of little use to you. If, however, you are ready to extend SMS by leveraging some of the VBScript scripts in this book, then the SMS `CollectionID` will be one of your best friends. Most of the scripts in this book that require a `CollectionID` will assume that you know it. You can use either method described in this recipe to obtain the `CollectionID` you need. If you plan to be a frequent scripter, we suggest paying attention to the following hot tip.

■ **Note** The easiest way to find an object ID is to modify the shortcut for your SMS Administrator console. For example, instead of the shortcut launching `C:\SMSAdmin\bin\i386\sms.msc`, have it launch `C:\SMSAdmin\bin\i386\sms.msc /sms:nodeinfo=1`. Now when you launch your SMS Administrator console, you can right-click any node, select Properties, and view the Node Information tab. For example, when you view the Node Information tab on a Collection, look for `CollectionID="LAB0001F"`, where `"LAB0001F"` is the `CollectionID` you would use. As an example, consider the following figure.



Example 3-3 is a good example of querying a class in the SMS database. In this solution, we display all collections in alphabetical order by name, and we also display the `CollectionID`. Other properties (such as `LastChangeTime`, `LastRefreshTime`, `ReplicateToSubSites`, etc.) are also available for display from the `SMS_Collection` class.

See Also

- Recipe 4-2, "Determining a Package's PackageID," describes how to identify the `PackageID` for a package.

3-4. Deleting a Collection

Problem

You want to delete a collection.

Solution: Using a Graphical Interface

1. Open the SMS Administrator console.
2. From the SMS Administrator console, expand Site Database (<*Site Code*>) and then Collections.
3. Right-click the collection that you want to delete and select Delete from the menu.
4. Click Next to go to the welcome screen of the Delete Collection Wizard.
5. Click Next to see additional information on the effect of deleting this collection. If you are absolutely sure that you want to delete it, then select that option before proceeding.
6. Review subcollections that will be affected. Click Next
7. Review advertisements that will be affected. Click Next.
8. Review queries that will be affected. Click Next.
9. Review membership rules that will be affected. Click Next.
10. Review administrative rights that will be affected. Click Next.
11. Confirm your choice to delete the collection and click Next.
12. Click Finish to exit the Delete Collection Wizard.

Solution: Using VBScript

Example 3-4 demonstrates how to delete a collection.

Example 3-4. *DeleteCollection.vbs*

```
strSMSServer = <SMSServer>

strCollID = "LAB001A7"

Set objLoc =  CreateObject("WbemScripting.SWbemLocator")
Set objSMS= objLoc.ConnectServer(strSMSServer, "root\sms")
Set Results = objSMS.ExecQuery _
    ("SELECT * From SMS_ProviderLocation WHERE ProviderForLocalSite = true")
For each Loc in Results
    If Loc.ProviderForLocalSite = True Then
        Set objSMS = objLoc.ConnectServer(Loc.Machine, "root\sms\site_" & _
            Loc.SiteCode)
    end if
Next

    Set col = objSMS.Get _
    ("SMS_Collection.CollectionID='" & strCollID & "'")
Col.Delete_
```

Discussion

Deleting collections should not be an afterthought in your understanding of SMS. The Delete Collection Wizard does an excellent job of showing you exactly what components of SMS will be impacted

by the deletion of a collection. Also remember that some items will be deleted automatically if you choose to delete a collection, so don't move too quickly through the seemingly endless Next, Next, Next button-clicking of the Delete Collection Wizard.

Example 3-4 demonstrates how to delete a collection programmatically using VBScript. If you delete a collection that is directly targeted from an advertisement, SMS automatically deletes the advertisement too.

■ **Caution** When deleting a collection programmatically, be sure to delete all subcollections for the collection and all parent-child relationships for that collection. Otherwise, those objects will become *orphaned* and not accessible from the SMS Administrator console.

See Also

- Recipe 3-6, "Deleting a Linked Collection," describes how to delete a linked collection.
- Recipe 3-14, "Removing Direct Members from a Collection," describes how to remove direct members from a collection.
- Recipe 3-15, "Deleting Resources from SMS," describes how to permanently remove resources (computers and users) from SMS.
- Recipe 6-2, "Deleting an Advertisement," describes how to delete an advertisement.
- Recipe 5-2, "Deleting a Program," describes how to delete a program.
- Recipe 4-4, "Deleting a Package," describes how to delete a package.

3-5. Creating a Linked Collection

Problem

You want to create a collection linked under an existing collection.

Solution: Using a Graphical Interface

1. Open the SMS Administrator console.
2. From the SMS Administrator console, expand Site Database (<*Site Code*>), expand Collections, and then select a collection you want to link to another collection underneath it.
3. Right-click the collection and select New ➤ Link to collection from the menu.
4. Select the collection that you want to link under your initial collection and click OK. The linked collection should show up underneath the first collection if you expand it.

Solution: Using VBScript

Example 3-5 demonstrates how to create a linked collection.

Example 3-5. *CreateCollectionLink.vbs*

```
strSMSServer = <SMSServer>

strParentColl = "LAB0000D" 'parent collection ID
strSubColl = "LAB0000E"  'collection ID to link
```

```
Set objLoc =  CreateObject("WbemScripting.SWbemLocator")
Set objSMS= objLoc.ConnectServer(strSMSServer, "root\sms")
Set Results = objSMS.ExecQuery _
    ("SELECT * From SMS_ProviderLocation WHERE ProviderForLocalSite = true")
For each Loc in Results
    If Loc.ProviderForLocalSite = True Then
        Set objSMS = objLoc.ConnectServer(Loc.Machine, "root\sms\site_" & _
            Loc.SiteCode)
    end if
Next

Set ColSvcs = objSMS.Get("SMS_Collection")
    ColSvcs.VerifyNoLoops "SMS_Collection.CollectionID=""" & _
    strParentColl & """", "SMS_Collection.CollectionID=""" & _
    strSubColl & """", Result

if Result = 0 then
    wscript.echo "Link would cause looping condition, exiting"
else
    Set objCol = objSMS.Get _
        ("SMS_CollectToSubCollect").SpawnInstance_()
    objCol.parentCollectionID = strParentColl
    objCol.subCollectionID = strSubColl
    objCol.Put_
    wscript.echo "Created Collection Link!"
end if
```

Discussion

Linked collections are a very handy feature of SMS. When advertising a new product or patch, you can start by sending the advertisement to a small group, and next to a beta group, followed by your desired distribution process. If you have an advertisement advertised to a collection (and the advertisement is also configured to include members of subcollections), use collection linking to easily add various groups to your advertisement.

Example 3-5 programmatically creates a parent-child relationship between two collections. Notice that we call the `VerifyNoLoops` method to verify that the relationship will not create an infinite parent-child relationship. If you later decide to remove this specific linked collection, you may do so safely with VBScript since this parent-child relationship is unique.

See Also

- Recipe 3-1, "Creating a Collection," describes how to create a collection.
- Recipe 3-7, "Creating a Direct Membership Rule," describes how to create a direct membership collection.
- Recipe 3-8, "Creating a Query-Based Membership Rule," describes how to create a query-based membership collection rule.
- Chapter 4 of the *SMS 2003 Operations Guide* demonstrates how to create a linked collection.
- Chapter 11 of the *Microsoft Systems Management Server 2003 Administrator's Companion* by Steven Kaczmarek (Microsoft Press, 2004) details linking collections.

3-6. Deleting a Linked Collection

Problem

You want to delete a linked collection.

Solution

1. Open the SMS Administrator console.
2. From the SMS Administrator console, expand Site Database (<*Site Code*>), expand Collections, and then expand the collection that contains the linked collection.
3. Right-click the linked collection that you want to delete and select Delete from the menu.
4. Click Next to go to the welcome screen of the Delete Collection Wizard.
5. Confirm your choice to delete the linked instance of the collection and click Next.
6. Click Finish to exit the Delete Collection Wizard.

Discussion

The process for deleting a linked collection may be a little confusing since you are not actually deleting the linked collection at all—more accurately, you are removing the link between the two collections. This also explains why you don't have to go through numerous screens in the Delete Collections Wizard, because nothing else is affected when you break the link between the collections except what was happening as a result of the link.

See Also

- Recipe 3-4, "Deleting a Collection," describes how to delete a collection.

3-7. Creating a Direct Membership Rule

Problem

You want to add a system to a collection using direct membership.

Solution: Using a Graphical Interface

1. Open the SMS Administrator console.
2. From the SMS Administrator console, expand Site Database (<*Site Code*>), expand Collections, and then select a collection to add a system by direct membership rule.
3. Right-click the collection, select Properties from the menu, and select the Membership Rules tab in the Collection Properties dialog box.
4. Click the New Direct Membership button (the icon of a computer with a sunburst) to create a new direct membership. This action opens the Create Direct Membership Rule Wizard.
5. Click Next.
6. Select a resource class, attribute name, and value for your search criteria. For example, to search for PC1234, you could select the System Resource class, Name attribute, and PC1234 for a search value. You can also use wildcard characters to expand your search. Click Next.

7. Click Next if you have access to all resources; otherwise, select a collection to limit your search and click Next.
8. Select the check box next to the resource that you want added to the collection. If no available resources are presented to you, go back and expand your search criteria until your desired resources are found. Then click Next.
9. Click Finish to exit the wizard. Your direct membership rule will now be in the membership rules list.
10. Click OK to apply the settings.

Solution: Using Free Tools

1. Open the SMS Administrator console.
2. From the SMS Administrator console, expand Site Database (*<Site Code>*), expand Collections, and then select a collection to add a system by direct membership.
3. Right-click the collection and select SMS Tools ➤ Add PCs to this Collection.
4. Enter the PC names or copy and paste them from another source, and then click Add PCs to Collection.
5. The tool will process the list and return any systems that could not be added.
6. If you need to add more systems, click Reset Form; otherwise, click Exit.

Solution: Using VBScript

Example 3-6 demonstrates how to create a direct membership rule.

Example 3-6. *CreateDirectMembership.vbs*

```
strSMSServer = <SMSServer>

strCollID = "LAB0000F" 'ID of the collection
strComputerName = "2kPro" 'computer name to add

Set objLoc =  CreateObject("WbemScripting.SWbemLocator")
Set objSMS= objLoc.ConnectServer(strSMSServer, "root\sms")
Set Results = objSMS.ExecQuery _
    ("SELECT * From SMS_ProviderLocation WHERE ProviderForLocalSite = true")
For each Loc in Results
    If Loc.ProviderForLocalSite = True Then
        Set objSMS = objLoc.ConnectServer(Loc.Machine, "root\sms\site_" & _
            Loc.SiteCode)
    end if
Next

'obtain the ResourceID for strComputerName
Set colResourceIDs=objSMS.ExecQuery _
    ("SELECT ResourceId FROM SMS_R_System WHERE NetbiosName ='" & _
         strComputerName & "'")
For each insResource in colResourceIDs
        strNewResourceID = insResource.ResourceID
```

```
Next
'add the ResourceID to the collection
Set instColl = objSMS.Get _
    ("SMS_Collection.CollectionID=""" & strCollID & """")
Set instDirectRule = objSMS.Get _
    ("SMS_CollectionRuleDirect").SpawnInstance_ ()
instDirectRule.ResourceClassName = "SMS_R_System"
instDirectRule.ResourceID = strNewResourceID
instDirectRule.RuleName = strComputerName
instColl.AddMembershipRule instDirectRule
```

Discussion

If the graphical interface method seems to be rather involved for adding a single system to a collection, don't worry. Members of the SMS community have created tools that make the process much easier. For example, Greg Ramsey has created one of these tools specifically to add a single system to a collection. You can get a free copy of his "Right-Click, Add PCs to Collection" tool at myITforum.com (`http://www.myitforum.com/articles/1/view.asp?id=7609`). The steps to use it are described in the "Right-click Tools" section of Appendix A. All of the free tools discussed in this book are listed in Appendix A, along with a brief description.

Example 3-6 assumes that you know the `CollectionID` of the affected collection. Review Recipe 3-3, "Identifying the CollectionID of a Collection," for more information. When you know the `CollectionID` of the affected collection, you can proceed through the script. Collections in SMS use `ResourceIDs` to identify membership, so first obtain the `ResourceID` for the computer to add to the collection. Next, create a new instance of a direct membership rule (`("SMS_CollectionRuleDirect").SpawnInstance_ ()`) and set the appropriate properties for the rule.

See Also

- Recipe 3-1, "Creating a Collection," describes how to create a collection.
- Recipe 3-8, "Creating a Query-Based Membership Rule," describes how to create a query-based collection rule.
- The SMS 2003 SDK provides more information about adding members to a collection.
- Appendix C of the *SMS 2003 Operations Guide* provides an additional example of creating a collection using VBScript.

3-8. Creating a Query-Based Membership Rule

Problem

You want to create a query-based rule for a collection based on dynamic data, such as a program listed in Add or Remove Programs.

Solution: Using a Graphical Interface

The solution is to add a new query rule to the query-based collection, ensuring that the new rule encompasses the new systems or users that you wish to add. Here are the steps to follow:

1. Open the SMS Administrator console.
2. From the SMS Administrator console, expand Site Database (<*Site Code*>), expand Collections, and then select a collection in preparation for adding a query-based membership rule.

3. Right-click the collection, select Properties from the menu, and select the Membership Rules tab in the Collection Properties dialog box.
4. Click the New Query-based Membership button (the icon of a database with a sunburst) to create a new query rule. This will open the Query Rule Properties dialog box.
5. Enter a name for your query rule and click Edit Query Statement to open the Query Statement Properties dialog box.
6. Click the New button to open the Criterion Properties dialog box.
7. Select your criterion type and click the Select button to open the Select Attribute dialog box.
8. Select an attribute class and attribute to use in your query criteria and click OK. For example, if you want to query for systems with WinZip installed, you could select the Add/Remove Programs attribute class and Display name attribute.
9. In the Criterion Properties dialog box, select your operator and enter a search value or click the Values button to return real values from the SMS database for you to choose from and click OK. Continuing our previous example, use the "is equal to" operator and enter WinZip in the value string box.
10. You will now see your search selection in the criteria box. You can now add additional search criteria or click OK.
11. Click OK to the Query Rules properties box after you have named the query rule and defined your query limitations.
12. Your query will now appear in the list of membership rules. Click OK for the settings to be applied.

Solution: Using VBScript

Example 3-7 demonstrates how to create a query-based membership rule to encompass new systems or users that you wish to add.

Example 3-7. *CreateQueryMembershipRule.vbs*

```
strSMSServer = <SMSServer>

strCollID = "LAB0000F"
strQuery = "select * from SMS_R_System inner join " & _
    "SMS_G_System_ADD_REMOVE_PROGRAMS on " & _
    "SMS_G_System_ADD_REMOVE_PROGRAMS.ResourceID = " & _
    "SMS_R_System.ResourceId where " & _
    "SMS_G_System_ADD_REMOVE_PROGRAMS.DisplayName = 'SMS View'"
strQueryName = "Systems that have SMSView Installed"

Set objLoc =  CreateObject("WbemScripting.SWbemLocator")
Set objSMS= objLoc.ConnectServer(strSMSServer, "root\sms")
Set Results = objSMS.ExecQuery _
    ("SELECT * From SMS_ProviderLocation WHERE ProviderForLocalSite = true")
For each Loc in Results
    If Loc.ProviderForLocalSite = True Then
        Set objSMS = objLoc.ConnectServer(Loc.Machine, "root\sms\site_" & _
            Loc.SiteCode)
    end if
Next
```

```
    Set instCollection = objSMS.Get _
    ("SMS_Collection.CollectionID='" & strCollID & "'")
Set clsQueryRule = objSMS.Get _
    ("SMS_CollectionRuleQuery")

'next we need to validate the query
ValidQuery = clsQueryRule.ValidateQuery(strQuery)

If ValidQuery Then
    Set instQueryRule = clsQueryRule.SpawnInstance_
    instQueryRule.QueryExpression = strQuery
    instQueryRule.RuleName = strQueryName
    instCollection.AddMembershipRule instQueryRule
End If
```

Discussion

The full power of SMS is not unleashed until you begin to use collections based on queries. We attempt to use query-based collections whenever possible purely for the dynamic nature that they provide. If a new SMS client is placed on your network, it will be added to the appropriate collections (based on client inventory information) and receive the associated advertisements without requiring any administrative intervention. That powerful automation may seem scary at first, but you will quickly see how dependable and trouble-free it is. Just be careful how you wield that power by creating your collections first, before you advertise a package to them. You want to make sure that you are targeting correctly before you pull the trigger!

Example 3-7 assumes you know the `CollectionID` of the affected collection. Review Recipe 3-3, "Identifying the CollectionID of a Collection," for more information. `StrQuery` contains a WBEM Query Language (WQL) query to identify all systems that have the program `DisplayName` of `SMSView` in Add or Remove Programs (use the GUI to create proper queries with the wizard). You can then obtain the desired collection (using the `CollectionID`), and verify that the query is valid by calling `ValidateQuery(strQuery)`. Provided the query is valid, create a new instance of a query rule, and add the membership rule.

See Also

- Recipe 3-1, "Creating a Collection," describes how to create a new collection.
- Recipe 3-7, "Creating a Direct Membership Rule," describes how to create a direct membership rule for a collection.
- The SMS 2003 SDK provides more information about adding members to a collection.
- Appendix C of the *SMS 2003 Operations Guide* provides an additional example of creating a collection using VBScript.

3-9. Creating a Collection Based on an Existing SMS Query

Problem

You want to create a collection based on an existing SMS query.

Solution: Using a Graphical Interface

1. Open the SMS Administrator console.
2. From the SMS Administrator console, expand Site Database (<*Site Code*>), expand Collections, and then select a collection to add a query-based membership rule.
3. Right-click the collection, select Properties from the menu, and select the Membership Rules tab in the Collection Properties dialog box.
4. Click the New Query-based Membership button (the icon of a database with a sunburst) to create a new query rule. This action opens the Query Rule Properties dialog box.
5. Enter a name for your query rule and click Import Query Statement. Browse the existing queries and select the one that you would like to import, and then click OK.
6. Click OK to the Query Rules properties box after you have named the query rule and defined your query limitations.
7. Your query will now appear in the list of membership rules. Click OK to apply the settings.

Discussion

You may find it easier to build your query first so you can fine-tune it before using it in a collection, or you may have a basic query that you plan on using for a number of collections with only minor modifications. In both situations, building the query first will save you some time when you get around to building your collections. However, the SMS query and the imported query are not linked in any way. If you make a change in the SMS query, you will have to re-import it, as the import function simply copies the query attributes.

See Also

- Recipe 3-1, "Creating a Collection," describes how to create a new collection.
- Recipe 3-7, "Creating a Direct Membership Rule," describes how to create a direct membership rule for a collection.
- Recipe 3-8, "Creating a Query-Based Membership Rule," describes how to create a query-based membership rule.
- The SMS 2003 SDK provides more information about the `SMS_Collection` class.
- Appendix C of the *SMS 2003 Operations Guide* provides an additional example of creating a collection using VBScript.

3-10. Creating a Collection That Uses a Subselect Query

Problem

You want to create a collection that uses a subselect query.

Solution: Using a Graphical Interface

Use a subselect query when you want to query for objects that are missing something. For example, the query in Example 3-8 displays all systems that do not have the Windows Update file `wuauclt.exe` in the `system32` directory of the system. Here are the steps to follow:

1. Open the SMS Administrator console.
2. From the SMS Administrator console, expand Site Database (<*Site Code*>), expand Collections, and then select a collection to add a query-based membership rule.

3. Right-click the collection, select Properties from the menu, and select the Membership Rules tab in the Collection Properties dialog box.
4. Click the New Query-based Membership button (the icon of a database with a sunburst) to create a new query rule. This action opens the Query Rule Properties dialog box.
5. Enter a name for your query rule and click Edit Query Statement to open the Query Statement Properties dialog box.
6. Click the New button to open the Criterion Properties dialog box.
7. Select Sub-selected values for your Criterion type and click the Select button to open the Select Attribute dialog box.
8. Select an attribute class and attribute to use in your query criteria and click OK.
9. In the Criterion Properties dialog box, select your operator and enter a subselect query, or click the Browse button to select an existing query. Click OK to close the Criterion Properties dialog box.
10. You will now see your search selection in the criteria box. You can add additional search criteria or click OK.
11. Click OK on the Query Rules properties box after you have named the query rule and defined your query limitations.
12. Your query will now appear in the list of membership rules. Click OK to apply the settings.

Solution: Using VBScript

Example 3-8 demonstrates how to create a subselect query rule.

Example 3-8. *CreateSubSelectQueryRule.vbs*

```
strSMSServer = <SMSServer>

strCollID = "LAB0002B"

strQuery = "select SMS_R_System.ResourceID," & _
    "SMS_R_System.ResourceType,SMS_R_System.Name," & _
    "SMS_R_System.SMSUniqueIdentifier," & _
    "SMS_R_System.ResourceDomainORWorkgroup," & _
    "SMS_R_System.Client from SMS_R_System inner join " & _
    "SMS_G_System_SYSTEM on " &_
    "SMS_G_System_SYSTEM.ResourceID = " & _
    "SMS_R_System.ResourceId where " & _
    "SMS_G_System_SYSTEM.Name not in (select " & _
    "SMS_G_System_SYSTEM.Name from SMS_R_System inner " & _
    "join SMS_G_System_SoftwareFile on " & _
    "SMS_G_System_SoftwareFile.ResourceID = " & _
    "SMS_R_System.ResourceId inner join " & _
    "SMS_G_System_SYSTEM on " & _
    "SMS_G_System_SYSTEM.ResourceID =" & _
    "SMS_R_System.ResourceId where " & _
    "SMS_G_System_SoftwareFile.FileName = 'wuauclt.exe' " & _
    "and SMS_G_System_SoftwareFile.FilePath like '%system32\\%')"
strQueryName = "Systems That Need Windows Update Agent"
```

```
Set objLoc =  CreateObject("WbemScripting.SWbemLocator")
Set objSMS= objLoc.ConnectServer(strSMSServer, "root\sms")
Set Results = objSMS.ExecQuery _
    ("SELECT * From SMS_ProviderLocation WHERE ProviderForLocalSite = true")
For each Loc in Results
    If Loc.ProviderForLocalSite = True Then
        Set objSMS = objLoc.ConnectServer(Loc.Machine, "root\sms\site_" & _
            Loc.SiteCode)
    end if
Next

Set instCollection = objSMS.Get _
     ("SMS_Collection.CollectionID='" & strCollID & "'")
Set clsQueryRule = objSMS.Get _
     ("SMS_CollectionRuleQuery")
'make sure we have a valid query
ValidQuery = clsQueryRule.ValidateQuery(strQuery)
If ValidQuery Then
    Set instQueryRule = clsQueryRule.SpawnInstance_
    instQueryRule.QueryExpression = strQuery
    instQueryRule.RuleName = strQueryName
    instCollection.AddMembershipRule instQueryRule
End If
```

Discussion

There are some things a simple query can't do, and for everything else there's a subselect query (sorry, MasterCard!). As you spend more time building queries, you may hit the simple query wall. For example, you may need to find all systems that have application A and not application B, and a simple query just won't do. If you aren't a programmer, the term "subselect" may not be in your vocabulary, and it may be easier to understand subselect as "select then select," for example:

> *Select all systems with application B and then select all systems with application A that aren't in the group with application B.*

If you are wondering where to get the subselect query to paste into the subselect window, try to create the subselect portion of your query as a simple query first. For example, begin with a query to select all systems with application B. Then open the Show SQL query window and copy and paste the query into the subselect window. Finally, apply the `Not in` operator.

■ **Note** Example 3-8 is the same as Recipe 3-8, "Creating a Query-Based Membership Rule." The only difference is the WQL query.

See Also

- Recipe 3-1, "Creating a Collection," describes how to create a collection.
- Recipe 3-7, "Creating a Direct Membership Rule," describes how to create a direct membership rule for a collection.
- Recipe 3-8, "Creating a Query-Based Membership Rule," describes how to create a query-based membership rule.

- The SMS 2003 SDK provides more information about the `SMS_Collection` class.
- Appendix C of the *SMS 2003 Operations Guide* provides an additional example of creating a collection using VBScript.

3-11. Updating Collection Membership

Problem

You want to immediately update the query-based membership of a collection.

Solution: Using a Graphical Interface

1. Open the SMS Administrator console.
2. From the SMS Administrator console, expand Site Database (<*Site Code*>) and then Collections.
3. Right-click the collection you want to update the membership of and select All Tasks ➤ Update Collection Membership.
4. Choose whether to update subcollections and click OK.

Solution: Using VBScript

Example 3-9 demonstrates how to update collection membership.

Example 3-9. *UpdateCollectionMembeship.vbs*

```
strSMSServer = <SMSServer>
strSMSServer = <SMSServer>

strCollID = "LAB00017"

Set objLoc =  CreateObject("WbemScripting.SWbemLocator")
Set objSMS= objLoc.ConnectServer(strSMSServer, "root\sms")
Set Results = objSMS.ExecQuery _
    ("SELECT * From SMS_ProviderLocation WHERE ProviderForLocalSite = true")
For each Loc in Results
    If Loc.ProviderForLocalSite = True Then

            strSMSSiteCode = Loc.Sitecode
    end if
Next

Set objCollection = GetObject( "WinMgmts:!\\" & strSMSServer & _
     "\root\SMS\site_" & strSMSSiteCode & _
    ":SMS_Collection.CollectionID='" & strCollID & "'")
objCollection.RequestRefresh False
```

Discussion

Most likely, you do not have your collections set to update every 15 minutes (as it is not that practical), but you may have the need on occasion to update query-based collection memberships faster than defined by your refresh schedule. For example, immediately updating collection membership becomes very important when a new system is being built and you need to have all of the advertisements

run in a short amount of time. By updating query-based memberships, new systems that meet the query-based criteria will be added to the collection (and, of course, systems that were in the collection that no longer meet the criteria will be removed from the collection). New clients that appear in the collection will obtain advertisement policy that is targeted to the collection on the next client-polling interval. At this time, the client will execute any mandatory advertisements that are past the mandatory start time, but have not yet expired.

Updating the collection membership is not the same as updating the current view in the console. After you update the collection membership, you will see a small hourglass next to the collection icon, and you won't see the new members in the right pane. The small hourglass indicates that the collection membership is being updated but the view is not current. Refresh the collection view by either using the refresh button or pressing F5. If the collection is large or has an intricate query, it may take a while for the update to process. Refresh the collection view until the small hourglass goes away; then you can be sure that the console is displaying the current membership.

Example 3-9 is straightforward. We connect to the appropriate collection in SMS and refresh the collection. Notice the last line of the script:

```
objCollection.RequestRefresh False
```

This code refreshes the current collection membership, but does not refresh the collection membership of subcollections. To refresh collection membership of all subcollections, change `False` to `True`.

See Also

- Recipe 3-8, "Creating a Query-Based Membership Rule," describes how to create a query-based membership rule.
- Recipe 3-12, "Setting the Update Collection Interval," describes how to set the collection refresh interval.
- The SMS 2003 SDK contains a section titled "Adding, Deleting, and Refreshing Members of a Collection."
- Appendix C of the *SMS 2003 Operations Guide* provides an additional example of creating a collection using VBScript.

3-12. Setting the Update Collection Interval

Problem

You want to set the interval for a collection to evaluate its membership.

Solution: Using a Graphical Interface

1. Open the SMS Administrator console.
2. From the SMS Administrator console, expand Site Database (<*Site Code*>), expand Collections, and then select a collection to modify its update schedule.
3. Right-click the collection, select Properties from the menu, and select the Membership Rules tab in the Collection Properties dialog box.
4. Click the Schedule button to adjust the frequency with which the collection updates its membership.
5. Click OK. The new settings appear in the Collection Properties dialog box.
6. Click OK to apply the settings.

Solution: Using VBScript

Example 3-10 demonstrates how to set the collection membership to update daily (`Token.DaySpan=1`). `Token.StartTime` is a WMI date string.

Example 3-10. *SetUpdateCollInterval.vbs*

```
strSMSServer = <SMSServer>
strCollID = "LAB0002B"

Set objLoc =  CreateObject("WbemScripting.SWbemLocator")
Set objSMS= objLoc.ConnectServer(strSMSServer, "root\sms")
Set Results = objSMS.ExecQuery _
    ("SELECT * From SMS_ProviderLocation WHERE ProviderForLocalSite = true")
For each Loc in Results
    If Loc.ProviderForLocalSite = True Then
        Set objSMS = objLoc.ConnectServer(Loc.Machine, "root\sms\site_" & _
            Loc.SiteCode)
    end if
Next

Set Token = objSMS.Get("SMS_ST_RecurInterval")
Token.DaySpan = 1
Token.StartTime = "20051202103000.000000+***" 'wmi date-string
'If omitted, StartTime = Jan 1, 1990 - this shouldn't
'cause any issues
Set objCollection = objSMS.Get _
    ("SMS_Collection.CollectionID='"  & strCollID & "'")
objCollection.RefreshSchedule = Array(Token)
objCollection.RefreshType = 2  'Periodic refresh
objCollection.Put_
```

Discussion

You can use the collection update interval to your advantage, and it does not need to stay at the default 24 hours. You may have some query-based collections that you don't use on a regular basis. For example, perhaps you only use them at the end of the month for reporting purposes. In this situation, you may configure the collection interval to update at the end of every month (instead of the default of 24 hours). Conversely, if you have an important advertisement that needs to be delivered in a timely manner, then you may want to shrink the associated collection refresh interval to an hour or so to give you a swifter response. New clients that appear in the collection after a collection update refresh will obtain an advertisement policy targeted to the collection on the next client-polling interval. At this time, the client will execute any mandatory advertisements past the mandatory start time, but not yet expired.

■ **Note** If you have a collection that is made of only direct memberships, then you do not need to refresh the collection at all.

Example 3-11 demonstrates how to convert a normal date-time into a WMI date-time.

Example 3-11. *ConvertToWMIDate.vbs*

```
wscript.echo ConvertToWMIDate(Now())

Function ConvertToWMIDate(strDate)
    strYear = year(strDate)
    strMonth = month(strDate)
    strDay = day(strDate)
    strHour = hour(strDate)
    strMinute = minute(strDate)
    'pad date appropriately
    if len(strmonth) = 1 then strMonth = "0" & strMonth
    if len(strDay) = 1 then strDay = "0" & strDay
    if len(strHour) = 1 then strHour = "0" & strHour
    if len(strMinute) = 1 then strMinute = "0" & strMinute
    ConvertToWMIDate = strYear & strMonth & strDay & strHour & _
        strMinute & "00.000000+***"
end function
```

In Example 3-11, the date function `Now()` is used to obtain the current date and time on the system. Any valid date and time can be used. For example, `12/02/2007 2:43 AM` will be converted to `20071202024300.000000+***`.

See Also

- Recipe 3-11, "Updating Collection Membership," demonstrates how to manually update collection membership.
- The SMS 2003 SDK provides additional information about `SMS_ST_RecurInterval`.
- Recipe 6-8, "Configuring a Recurring Advertisement," provides a solution that uses `SMS_ST_RecurWeekly` so you can configure a weekly update schedule.

3-13. Limiting Collection Membership to Another Collection

Problem

You want to limit a query-based collection's membership to only members of another collection.

Solution: Using a Graphical Interface

1. Open the SMS Administrator console.
2. From the SMS Administrator console, expand Site Database (<*Site Code*>), expand Collections, and then select a query-based collection to limit the membership by another collection.
3. Right-click the collection, select Properties from the menu, and select the Membership Rules tab in the Collection Properties dialog box.
4. Double-click the query membership rule that you would like to limit to another collection.
5. On the Query Rule Properties dialog box, change Collection Limiting from Not Collection Limited to Limit to Collection and enter the collection name or use the Browse button to identify the collection for limiting the query.
6. Click OK and you will see the collection limit associated with your query rule in the "Limit to" column of the Membership Rules tab of the Collection Properties dialog box.
7. Click OK to apply the settings.

Solution: Using VBScript

Example 3-12 demonstrates how to limit collection membership to another collection.

Example 3-12. *LimitCollectionMembership.vbs*

```
strSMSServer = <SMSServer>

strCollID = "LAB0002B" 'this is the collection to modify
strCollLimit = "SMS000GS" 'this is the limiting collection

Set objLoc =  CreateObject("WbemScripting.SWbemLocator")
Set objSMS= objLoc.ConnectServer(strSMSServer, "root\sms")
Set Results = objSMS.ExecQuery _
    ("SELECT * From SMS_ProviderLocation WHERE ProviderForLocalSite = true")
For each Loc in Results
    If Loc.ProviderForLocalSite = True Then
        Set objSMS = objLoc.ConnectServer(Loc.Machine, "root\sms\site_" & _
            Loc.SiteCode)
    end if
Next

Set objCollection=objSMS.Get("SMS_Collection='" & _
    strCollID & "'" )
'Get the array of embedded SMS_CollectionRule objects.
RuleSet = objCollection.CollectionRules
For Each Rule In RuleSet
    if Rule.Path_.Class = "SMS_CollectionRuleQuery" then
        Rule.LimitToCollectionID = strCollLimit
    end if
Next
objCollection.Put_
```

Discussion

Limiting collection membership to members of another collection is a great way to easily step into an application deployment. You don't have to create several different queries for each group of your deployment as long as you already have collections for them. For example, say you are going to deploy version 6.0 of an application to all systems with version 5.0 or earlier. First, create your base query of all systems that have version 5.0 or earlier. Import the query into your "Version 6.0" collection but limit it to your "Test systems" collection. Advertise the package to the "Version 6.0" collection. When the systems from the "Test systems" collection have successfully installed the application, you can change the "Version 6.0" collection to limit the query to a collection for a floor or a department, but less than the entire target group. Once you feel comfortable with the deployment, you can remove the collection limit and all systems that meet the query criteria will be targeted.

In Example 3-12, we enumerate all query-based rules for the collection and add the collection limit to each one. After we have modified all query rules, we use the `Put_` method to save the changes to the `SMS_Collection` object. This script could be modified to display current collection limiting information for auditing purposes also.

See Also

- Recipe 3-5, "Creating a Linked Collection," describes how to create a linked collection.
- Recipe 3-11, "Updating Collection Membership," describes how to update collection membership.

- The SMS 2003 SDK contains a section titled "Adding, Deleting, and Refreshing Members of a Collection."
- Appendix C of the *SMS 2003 Operations Guide* provides an additional example of creating a collection using VBScript.

3-14. Removing Direct Members from a Collection

Problem

You want to remove direct members from a collection.

Solution: Using a Graphical Interface

1. Open the SMS Administrator console.
2. From the SMS Administrator console, expand Site Database (<*Site Code*>), expand Collections, and then select a collection from which to delete a direct membership rule.
3. Right-click the collection, select Properties from the menu, and select the Membership Rules tab in the Collection Properties dialog box.
4. Select the direct membership rule that you would like to remove and click the Delete Direct Membership button (black "X") to remove it.
5. Click OK to apply the settings.

Solution: Using VBScript

Example 3-13 demonstrates how to delete all direct membership rules from a collection.

Example 3-13. *DelDirectMembership.vbs*

```
strSMSServer = <SMSServer>

strCollID = "LAB0002B"

Set objLoc =  CreateObject("WbemScripting.SWbemLocator")
Set objSMS= objLoc.ConnectServer(strSMSServer, "root\sms")
Set Results = objSMS.ExecQuery _
    ("SELECT * From SMS_ProviderLocation WHERE ProviderForLocalSite = true")
For each Loc in Results
    If Loc.ProviderForLocalSite = True Then
        Set objSMS = objLoc.ConnectServer(Loc.Machine, "root\sms\site_" & _
            Loc.SiteCode)
    end if
Next

Set objCollection=objSMS.Get("SMS_Collection='" & strCollID & "'" )
'Get the array of embedded SMS_CollectionRule objects.
RuleSet = objCollection.CollectionRules
For Each Rule In RuleSet
    if Rule.Path_.Class = "SMS_CollectionRuleDirect" then
        objCollection.DeleteMembershipRule Rule
    end if
Next
```

Discussion

A common mistake among SMS administrators is deleting a resource from SMS entirely when they actually wanted to only delete the membership of a resource from a collection. You will want to heed the warning that is displayed when you attempt to delete a resource, as it may be the only reminder that you are about to delete the wrong thing. This is one of the main reasons we try to avoid direct memberships in collections whenever possible; managing those direct memberships can become a difficult task. It's an especially difficult task if one of those systems is rebuilt or replaced, because then you will have to add another direct membership for the new system.

Each direct membership is a membership rule for the collection in question. So in Example 3-13, we enumerate and delete all direct membership rules. This script could be modified to remove a specific rule, but you would also be required to identify the `ResourceID` (or `RuleName`) for the specific rule to be deleted.

See Also

- Recipe 3-4, "Deleting a Collection," describes how to delete a collection.
- Recipe 3-15, "Deleting Resources from SMS," describes how to delete resources from SMS.
- Appendix C of the *SMS 2003 Operations Guide* provides an additional example of using `DeleteMembershipRule`.

3-15. Deleting Resources from SMS

Problem

You want to delete members of a collection. Unlike the case in Recipe 3-14, "Removing Direct Members from a Collection," this time you really do want to delete the resources (representing the members) from SMS.

■ **Caution** Be aware that the solutions in this recipe will delete resources from SMS. Deleting is different from removing a member from a collection—review the "Discussion" section of this recipe for more information. Be sure that you fully understand what you are doing before executing this recipe.

Solution: Using a Graphical Interface

Deleting One Resource from SMS

1. Open the SMS Administrator console.
2. From the SMS Administrator console, expand Site Database (<*Site Code*>), expand Collections, and then select a collection that contains the member to delete.
3. Right-click the appropriate resource and select Delete.
4. Click Yes to confirm deletion.

Deleting Multiple Resources from SMS

1. Open the SMS Administrator console.
2. From the SMS Administrator console, expand Site Database (<*Site Code*>), expand Collections, and then select a collection that contains all members to be deleted.
3. Right-click the collection and select Delete Special.
4. Click Yes to confirm deletion of *all* resources in a specified collection.

Solution: Using VBScript

Example 3-14 demonstrates how to delete a resource from SMS.

Example 3-14. *DelResource.vbs*

```
strSMSServer = <SMSServer>

strComputer = "Computer1"

Set objLoc =  CreateObject("WbemScripting.SWbemLocator")
Set objSMS= objLoc.ConnectServer(strSMSServer, "root\sms")
Set Results = objSMS.ExecQuery _
    ("SELECT * From SMS_ProviderLocation WHERE ProviderForLocalSite = true")
For each Loc in Results
    If Loc.ProviderForLocalSite = True Then
        Set objSMS = objLoc.ConnectServer(Loc.Machine, "root\sms\site_" & _
            Loc.SiteCode)
        strSMSSiteCode = Loc.SiteCode
    end if
Next

'get the resource ID of the computer
intResourceID = GetResourceID(strComputer)

'Remove ResourceID
Set objResource = GetObject( "WinMgmts:\\" & strSMSServer & _
    "\root\SMS\site_" & strSMSSiteCode & _
    ":SMS_R_System.ResourceID=" & cint(intResourceID))
objResource.Delete_
wscript.echo "Deleted " & strComputer & "(" & intResourceID & ")"

Function GetResourceID(strComputerName)
    Set colResourceIDs = objSMS.ExecQuery _
        ("select ResourceID from SMS_R_System where Name = '" & _
             strComputer & "'")
    for each objResID in colResourceIDs
        GetResourceID = objResID.ResourceID
    next
End Function
```

Discussion

When you delete a resource from SMS, that resource will disappear from all collections, web reports, and everywhere else within SMS. This recipe shows how to delete resources when you no longer need them. If you wish to simply remove a resource from a direct membership collection while retaining that resource for possible use in other collections, then refer to Recipe 3-14, "Removing Direct Members from a Collection." If you wish to remove a resource from a query-based collection, then you need to modify the query to exclude the resource that's no longer desired.

Rather than delete just one resource, you may find you need to delete multiple resources from SMS. Say, for example, you have replaced all systems in the human resources department, and you see the "old" PCs in SMS (these PCs will remain there by default for 90 days). If for some reason you want to remove these systems from SMS before SMS purges them automatically, you could create a collection of all systems in the HR organizational unit that have not processed a hardware inventory within the past 30 days. Once you have this collection, you could perform a "delete special" to permanently remove these resources from SMS.

Keep in mind that when you delete resources from SMS you lose all inventory information, including all inventory history information. For license audits, these "extra" systems in SMS can skew the license counts, so that may be a driver to remove the inventory information. On the other hand, by having the old inventory data available, you have an additional avenue to research when Joe User says, "Hey, I had Visio 2003 Professional on my old PC, but I don't have it installed on my new one. Please install ASAP." By keeping the inventory of the old systems, you may be able to determine whether Joe is being honest, or if he's trying to obtain software without paying for it.

If you happen to delete a resource by mistake, the good news is the resource will reappear within SMS the next time a discovery record is generated for the resource. The bad news is that if the resource was a member of any collections by direct membership, those memberships will not, of course, be restored (because you deleted them, implicitly, when you deleted the resource). Membership in query-based collections will be restored the next time those collection memberships are updated. And, when the resource is brought back into a query-based collection, any advertisements that the client has already run will not rerun (unless, of course, the advertisement is set to rerun on a schedule).

In Example 3-14, we obtain the `ResourceID` of the computer resource, and then delete it from `SMS_R_System`. To perform the same actions as a "delete special," you could enumerate all resources in a collection and call the `Delete_` method on each.

See Also

- Recipe 3-4, "Deleting a Collection," describes how to delete a collection.
- Recipe 3-6, "Deleting a Linked Collection," describes how to delete a linked collection.
- Recipe 3-14, "Removing Direct Members from a Collection," describes how to remove members from a collection.
- The *SMS 2003 Scripting Guide* demonstrates how to delete objects from SMS.

3-16. Modifying Permissions of a Collection

Problem

You want to modify permissions to a specific collection.

Solution: Using a Graphical Interface

1. Open the SMS Administrator console.
2. From the SMS Administrator console, expand Site Database (<*Site Code*>), expand Collections, and then select a collection to modify.
3. Right-click the collection, select Properties from the menu, and select the Permissions tab in the Collection Properties dialog box.
4. Click the New or Delete button to add or remove class or instance security rights. You can also double-click any existing classes or instance rights to modify them.
5. Click OK to apply the settings.

Solution: Using VBScript

Example 3-15 demonstrates how to modify access permissions to a collection.

Example 3-15. *ModCollPerms.vbs*

```
'This script will grant the group "SMSVPC\Help Desk" read and
    ' modify permissions to collection ID "LAB00159".
strSMSServer = <SMSServer>
strHelpDesk="LAB\SMSHelpDesk" 'Domain\Group or username
strCollID = "LAB00159"  'ID of the collection

Set objLoc =  CreateObject("WbemScripting.SWbemLocator")
Set objSMS= objLoc.ConnectServer(strSMSServer, "root\sms")
Set Results = objSMS.ExecQuery _
    ("SELECT * From SMS_ProviderLocation WHERE ProviderForLocalSite = true")
For each Loc in Results
    If Loc.ProviderForLocalSite = True Then
        Set objSMS = objLoc.ConnectServer(Loc.Machine, "root\sms\site_" & _
            Loc.SiteCode)
    end if
Next

Set objNewRight = objSMS.Get _
  ("SMS_UserInstancePermissions").SpawnInstance_()
objNewRight.UserName = strHelpDesk
objNewRight.ObjectKey = 1 '1=collection
objNewRight.InstanceKey = strCollID
objNewRight.InstancePermissions = 1+2'grant Read and Modify
objNewRight.Put_
```

Discussion

Proper collection permissions are essential if you will be delegating any SMS tasks to other users in your environment and want to make sure that they can only impact those systems they are responsible for. If you haven't looked into delegating any SMS tasks, you should try it. You will find that you can safely give access to people to perform functions (such as adding a computer to a collection), which will allow for faster service from SMS.

Be very careful when modifying permissions, or you may lock yourself out of a collection! Especially when you modify permissions programmatically, be sure to test your code in a test environment before running it in production.

See Also

- Recipe 4-9, "Modifying Permissions of a Package," describes how to modify the permissions of a package.
- Recipe 6-17, "Modifying the Permissions of an Advertisement," describes how to modify the permissions of an advertisement.
- The *SMS 2003 Scripting Guide* provides more information about setting security rights for an SMS object.
- The SMS 2003 SDK describes `SMS_UserInstancePermissions`.
- The *SMS 2003 Scripting Guide* also describes `SMS_UserInstancePermissions`.

3-17. Viewing Advertisements Assigned to a Collection

Problem

You want to show all advertisements that apply to a specific collection.

Solution: Using a Graphical Interface

1. Open the SMS Administrator console.
2. From the SMS Administrator console, expand Site Database (*<Site Code>*), expand Collections, and then select a collection to view the advertisements for that collection.
3. Right-click the collection, select Properties from the menu, and select the Advertisements tab in the Collection Properties dialog box to view the advertisements.
4. Click OK or Cancel to close the dialog box.

Solution: Using VBScript

Example 3-16 demonstrates how to view advertisements assigned to a collection.

Example 3-16. *ListAdvertsToColl.vbs*

```
strSMSServer = <SMSServer>
strCollID = "LAB000FE"

Set objLoc =  CreateObject("WbemScripting.SWbemLocator")
Set objSMS= objLoc.ConnectServer(strSMSServer, "root\sms")
Set Results = objSMS.ExecQuery _
    ("SELECT * From SMS_ProviderLocation WHERE ProviderForLocalSite = true")
For each Loc in Results
    If Loc.ProviderForLocalSite = True Then
        Set objSMS = objLoc.ConnectServer(Loc.Machine, "root\sms\site_" & _
            Loc.SiteCode)
    end if
Next

ListAdverts strCollID, False
CheckParent strCollID

Sub ListAdverts(strCollID,blnSubCollect)
    If blnSubCollect then
        'This is to check all parent questions of the
        'collection in question. We can only look at
        'parent collections that are set to "Include
        'members of subcollections"
        Set colAdverts = objSMS.ExecQuery _
        ("select * from SMS_Advertisement where " & _
        "CollectionID = '" & strCollID & "' and " & _
        "IncludeSubCollection = 1")
    Else
        'This is for the first check; it will look for
        'advertisements assigned directly to the collection
        Set colAdverts = objSMS.ExecQuery _
         ("select * from SMS_Advertisement where " & _
         "CollectionID = '" & strCollID & "'")
    end if
    for each objAdvert in colAdverts
        wscript.echo "Advertisement: " & _
           objAdvert.AdvertisementName
        wscript.echo "   Collection: " & _
           GetCollectionName(objAdvert.CollectionID) & _
           " (" & objAdvert.CollectionID  & ")" & VbCRLF
    next
End Sub
```

```
Function GetCollectionName(strCollID)
    Set instColl = objSMS.Get _
    ("SMS_Collection.CollectionID=""" & strCollID & """")
    GetCollectionName = instColl.Name
End Function

Sub CheckParent(strCollID)
    Set colParents = objSMS.ExecQuery _
    ("select * from SMS_CollectToSubCollect where subCollectionID = '" & _
    strCollID & "'")
    for each objParent in colParents
        if not objParent.ParentCollectionID = "COLLROOT" then
            ListAdverts objParent.ParentCollectionID , True
            CheckParent objParent.ParentCollectionID
        end if
    next
End Sub
```

Discussion

Using the techniques shown in this recipe is a great way of keeping track of current advertisements and planning the impact of adding new systems to current collections. The list of advertisements displayed by this recipe's solution include not only the advertisements targeted directly at the collection in question, but also the advertisements targeted to the collection through links. The collection value listed with each advertisement tells you which collection is the base collection, but it won't show you how the current collection is linked to it.

Example 3-16 is a good example of the use of subroutines, functions, and recursion. If SMS did not contain the functionality to "Include Members of SubCollections," the recipe solution would be much easier to code. We would simply need to look at all advertisements, and see if any of them are assigned to the collection in question. However, since the collection in question could be a subcollection to a collection that is receiving the advertisement, we need to dig a little deeper.

In Example 3-16, we start by taking the collection in question and seeing whether there are any advertisements directly assigned to that collection (`Sub ListAdverts` handles this task). We then check to see if the collection in question has a parent collection. If the collection does have a parent, we check that parent collection for advertisements that are set to include subcollections. This process continues until we run out of parent collections to check.

See Also

- Recipe 3-18, "Viewing Advertisements Assigned to a Computer," demonstrates how to display all advertisements for a computer.
- The SMS 2003 SDK describes `SMS_Advertisement`, `SMS_Collection`, and `SMS_CollectToSubCollect` in detail.

3-18. Viewing Advertisements Assigned to a Computer

Problem

You want to show all advertisements that apply to a specific client computer in a collection.

Solution: Using a Graphical Interface

1. Open the SMS Administrator console.
2. From the SMS Administrator console, expand Site Database (*<Site Code>*), expand Collections, and then select a collection to view the advertisements for a system in that collection.
3. Right-click a system in the right pane, select Properties from the menu, and select the Advertisements tab in the Resource Properties dialog box to view the advertisements for that system.
4. Click OK or Cancel to close the dialog box.

Solution: Using VBScript

Example 3-17 demonstrates how to view advertisements assigned to a computer.

Example 3-17. *ViewAdvertsAssignedToMachine.vbs*

```
strSMSServer = <SMSServer>

strComputer = "Computer1"

Set objLoc =  CreateObject("WbemScripting.SWbemLocator")
Set objSMS= objLoc.ConnectServer(strSMSServer, "root\sms")
Set Results = objSMS.ExecQuery _
    ("SELECT * From SMS_ProviderLocation WHERE ProviderForLocalSite = true")
For each Loc in Results
    If Loc.ProviderForLocalSite = True Then
        Set objSMS = objLoc.ConnectServer(Loc.Machine, "root\sms\site_" & _
            Loc.SiteCode)
    end if
Next

'first, get the resource ID of the computer
intResourceID = GetResourceID(strComputer)

Set colAdverts = objSMS.ExecQuery _
("select * from SMS_ClientAdvertisementStatus where ResourceID = " & _
     intResourceID)
for each strAdvert in colAdverts 'enumerate all adverts for client
    wscript.echo GetAdvertisementName(strAdvert.AdvertisementID) & _
        "(" & strAdvert.AdvertisementID & ")" & vbTAB  & _
    strAdvert.LastStateName & vbTAB & strAdvert.LastStatusTime & _
    vbTAB & GetCollectionName(strAdvert.AdvertisementID)
next

'used to obtain the Advertisement Name
Function GetAdvertisementName(strAdvertID)
    Set instAdvert = objSMS.Get _
        ("SMS_Advertisement.AdvertisementID='" & strAdvertID & "'")
    GetAdvertisementName = instAdvert.AdvertisementName
End Function
```

```
'used to obtain the Collection Name
Function GetCollectionName(strAdvertID)
    'first, get advert based on advert ID
    Set instAdvert = objSMS.Get _
        ("SMS_Advertisement.AdvertisementID='" & strAdvertID & "'")
    'then, get collection name based on collectionID from advert
    Set instCollection = objSMS.Get _
        ("SMS_Collection.CollectionID='" & instAdvert.CollectionID & "'")
    GetCollectionName = instCollection.Name
End Function

'used to obtain the SMS resource ID
Function GetResourceID(strComputerName)
    Set colResourceIDs = objSMS.ExecQuery _
        ("select ResourceID from SMS_R_System where Name = '" & _
             strComputer & "'")
    for each objResID in colResourceIDs
        GetResourceID = objResID.ResourceID
    next
End Function
```

Discussion

We would direct help desk personnel and department managers to the SMS web reports to get information on advertisements assigned to a specific computer. For the SMS administrator who is working in the SMS Administrator console, the solution in this recipe can be very helpful when taking inventory of which advertisements are currently being targeted at a specific SMS client. Maybe you have been asked to make sure that Suzy's computer has the same applications as Jack's PC. In such a case, you can easily discover which applications may need to be added and which collections to change.

In Example 3-17, we first obtain the `ResourceID` of the computer, and then check the class `SMS_ClientAdvertisementStatus` to obtain all advertisements assigned to that client. Finally, we display the advertisement name, the advertisement ID, and the last reported state for the advertisement for the client in question.

See Also

- The SMS web report titled "All advertisements for a specific computer" displays all advertisements assigned to a specific computer.
- Recipe 3-17, "Viewing Advertisements Assigned to a Collection," details how to obtain a list of all advertisements advertised to a specific collection.
- The SMS 2003 SDK describes `SMS_ClientAdvertisementStatus`, `SMS_Advertisement`, `SMS_Collection`, and `SMS_R_System` in detail.

3-19. Viewing Resources for a Member of a Collection

Problem

You want to view hardware and software resources for a specific client in a collection.

Solution: Using a Graphical Interface

1. Open the SMS Administrator console.
2. From the SMS Administrator console, expand Site Database (<*Site Code*>), expand Collections, and then select a collection that contains the computer you wish to analyze.
3. Locate the computer resource in the right pane, right-click it, and select All Tasks ➤ Start Resource Explorer.
4. From Resource Explorer, expand <*Computer Name*>.
5. Then expand and browse Hardware, Hardware History, and Software as desired.
6. Close Resource Explorer when you've finished.

Solution: From the Command Line

1. Create a batch file called `ResExplorer.bat` and save it in `%windir%\System32`.
2. Enter the following text into `ResExplorer.bat`:

```
set strSMSServer = <SMSServer>
set strSMSSiteCode = <SMSSiteCode>
rem ***The next three lines are actually one line!
start mmc C:\SMSAdmin\bin\i386\explore.msc -s -sms:ResExplrQuery=
"Select ResourceID From SMS_R_SYSTEM Where Name = ""%1""" –
sms:connection=\\%strSMSServer%\root\sms\site_%strSMSSiteCode%
```

3. Enter the appropriate site code and SMS server name, and verify the proper path to `explore.msc` for your environment.
4. Ensure the last three lines in the example are all on the same line.
5. Save `ResExplorer.bat`.
6. Click Start ➤ Run, enter **ResExplorer *Computer1*** (where *Computer1* is the name of the computer to observe), and click OK.
7. Resource Explorer will launch and display data for *Computer1*.
8. Close Resource Explorer when you've finished.

Discussion

Resource Explorer is a great tool for viewing hardware and software inventory information on a specific system. Hardware history is also available in Resource Explorer. The following is a brief explanation of each major node in Resource Explorer:

Hardware: This displays all data collected through hardware inventory. The majority of this information appears because of settings in the `SMS_DEF.MOF` file. Other sources for hardware inventory include `IDMIF` and `NOIDMIF` files (if enabled). Review Chapter 11 for more information about the sources for hardware inventory.

Hardware History: Think of Hardware History as a "paper trail" of changes that have occurred (as far as hardware inventory is concerned) over the past 90 days (by default). Modify the Delete Aged Inventory History task to reduce or extend the amount of time to keep this history. The history data can be helpful in determining when hardware was installed, when the amount of memory in a system has changed, and when a disk drive(s) free space is below a predetermined amount, among many other possibilities.

Software: Software inventory displays properties for inventoried files (by default, `*.exe`) and helps to associate product names to files. If you utilize the file collection feature of SMS, the file data is also available here.

Resource Explorer allows SMS administrators to see the data available for use when creating query-based collections and SMS queries.

WHY IS ADD/REMOVE PROGRAMS LISTED IN HARDWARE INVENTORY WHEN THE PROGRAMS ARE ACTUALLY SOFTWARE?

Good question. *Every* new SMS administrator asks that question. Think of Software inventory as querying only specific files (e.g., `msxml3.dll`) or files with wildcards (e.g., `*.exe`). Hardware inventory queries Windows Management Instrumentation (WMI) to obtain processor information, disk information, and more. Hardware inventory is also used to obtain data from the Windows registry (using WMI). Add/Remove Programs data is stored in `HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Uninstall`, which explains why this data is included in Hardware inventory. Review Chapter 11 for more information about inventorying registry data using SMS and WMI.

See Also

- Recipe 3-17, "Viewing Advertisements Assigned to a Collection," describes how to view advertisements assigned to a collection.
- Recipe 3-18, "Viewing Advertisements Assigned to a Computer," describes how to view advertisements assigned to a client.
- Chapter 9 of the *Microsoft Systems Management Server 2003 Administrator's Companion* by Steven Kaczmarek (Microsoft Press, 2004) provides some detail about Resource Explorer.
- Chapter 2 of the *SMS 2003 Operations Guide* provides additional information on SMS Resource Explorer.
- The *SMS 2003 Operations Guide*, Table 3-1 (titled "Inventory Data Type and Classification in SMS") provides more information about which inventory component is used for various system properties.

3-20. Adding a Right-click Option to Affect a Member of a Collection

Problem

You want to add an action to the right-click functionality in the SMS Administrator console that affects a specific client.

■ **Note** Two solutions are provided in this recipe. The first demonstrates the basic process using a batch file, and the second provides a more advanced solution using VBScript. You will likely find that both are valuable for your environment.

Solution: Displaying the Logs Directory of an SMS Advanced Client

The following example solution uses a batch file to add a right-click menu option allowing you to display the logs directory of an SMS Advanced Client. You can adapt the solution to other tasks by following the same process while using a different batch file name and script.

1. From the system that has the SMS Administrator console installed, browse to `C:\SMSAdmin\` (or `SMSAdmin` or `SMS`, depending on SMS components installed).
2. Create a new directory called `SMSTools` and open it for viewing.
3. Create a batch file called `ShowClientLogs.bat`.
4. Edit `ShowClientLogs.bat` in Notepad, and enter the following text:

```
if not exist \\%1\admin$\system32\ccm\logs goto notexist
explorer \\%1\admin$\system32\ccm\logs
exit

:notexist
echo "\\%1\admin$\system32\ccm\logs does not exist."
echo "  Verify admin$ and SMS Adv Client Installed"
pause
exit
```

5. Save `ShowClientLogs.bat`.
6. Open the Windows registry editor by selecting Start ➤ Run, entering **Regedit.exe**, and then clicking OK.
7. Browse to `HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\MMC\NodeTypes\{4D26C0D4-A8A8-11D1-9BD9-00C04FBBD480}\Extensions`.
8. From the `Extensions` key, create a new key named `SMS_Tools`.
9. From the `SMS_Tools` key, create a new key named `ShowClientLogs`. It's a good idea to have this registry key name match your batch file name; doing so can help you keep things straight.
10. From the `ShowClientLogs` key, create a new string value called `Name`, and for the data, enter **Show Client Logs Folder**.
11. From the `ShowClientLogs` key, create a new string value called `CommandLine`, and for the data, enter **C:\SMSAdmin\SMSTools\ShowClientLogs.bat ##SUB:Name##**.
12. To verify functionality, right-click a system in any collection and select SMSTools ➤ Show Client Logs Folder.

Solution: Requesting Machine Policy for an SMS Advanced Client

The following example solution uses a VBScript file to add a right-click menu option allowing you to refresh machine policy of an SMS Advanced Client. You can adapt the solution to other tasks by following the same process while using a different file name, script, and registry entries.

1. From the system that has the SMS Administrator console installed, browse to `C:\SMSAdmin\` (or `SMSAdmin` or `SMS`, depending on SMS components installed).
2. Create a new directory called `SMSTools` and open it for viewing.
3. Create a batch file called `RefreshMachinePolicy.vbs`.
4. Edit `RefreshMachinePolicy.vbs` in Notepad, and enter the following text:

```
Set objArgs = WScript.Arguments
strComputerName = objArgs(0)
Set smsClient = GetObject("winmgmts:\\" & strComputerName & _
    "\root\ccm:SMS_Client")
smsClient.RequestMachinePolicy(0)
wscript.echo "Machine Policy Refreshed on " & strComputerName
wscript.sleep 5000
```

5. Save `RefreshMachinePolicy.vbs`.
6. Open the Windows registry editor by selecting Start ➤ Run, entering **Regedit.exe**, and then clicking OK.
7. Browse to `HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\MMC\NodeTypes\{4D26C0D4-A8A8-11D1-9BD9-00C04FBBD480}\Extensions`.
8. From the `Extensions` key, create a new key named `SMS_Tools`.
9. From the `SMS_Tools` key, create a new key named `RequestMachinePolicy`.
10. From the `RequestMachinePolicy` key, create a new string value called `Name`, and for the data, enter **Request Machine Policy**.
11. From the `RequestMachinePolicy` key, create a new string value called `CommandLine`, and for the data, enter **cscript.exe C:\SMSAdmin\SMSTools\RefreshMachinePolicy.vbs ##SUB:Name##**.
12. To verify functionality, right-click a system in any collection and select SMSTools ➤ Request Machine Policy.

Discussion

The process demonstrated in this recipe is called *context menu integration*, and it is a very powerful tool to extend the SMS Administrator console. Any node in the SMS Administrator console can be extended in this fashion. The SMS 2003 SDK provides more information regarding how to determine the location in the registry required to use other nodes.

The basic structure of the two recipe solutions may be used for several tasks to make your job easier. Keep in mind that anything that can be executed from a command line (`.exe`, `.mdb`, etc.) can be used in right-click functionality in the SMS Administrator console. SMSView (`http://www.smsview.com`) is an example of an application that can be launched by performing a right-click of a client in a collection. Also, there is no requirement for the external application to execute an SMS function. For example, if you often find yourself viewing an internal (or external) web site when managing collections, you could change your command line to launch your favorite web browser, passing a command-line argument giving the desired web site.

Review step 11 from this recipe and note that `##SUB:Name##` is the variable used for the computer name when executing from a right-click tool in the Administrator console. Additional discovery data can be obtained by replacing `Name` with the data name (e.g., `ADSiteName`, `ClientVersion`, `ResourceDomainOrWorkgroup`, etc.). Data can also be passed to the external program by inserting multiples in the `CommandLine` field (e.g., `##SUB:Name## ##SUB:ADSITENAME## ##SUB:ClientVerision##`). Here are a few constants also available with context menu integration:

`##SUB:__SERVER##`: This is the name of the SMS server.

`##SUB:__NAMESpace##`: This is the namespace name for SMS. It's often used to connect to WMI on the server.

Arrays are also available for discovery data of type array (e.g., `IPAddresses`, `MACAddresses`, `SMSAssignedSites`, etc.). Refer to the SMS 2003 SDK for more information about using arrays with context menu integration.

Both solutions provided assume the user who initiates the actions has proper permissions to perform the actions. In both solutions, the remote system must be powered on and connected to the network. In the first solution, the user must have permissions to the `admin$` share. In the second solution, proper permissions to WMI are required. Notice the `strComputerName = objArgs(0)` in the preceding VBScript example. This is used to obtain the first argument (zero-indexed) on the command line (in this case, the computer name). To create and capture a second argument, simply modify your `CommandLine` in the registry (e.g., `##SUB:ADSiteName##`), and add a line to the VBScript to capture the second argument (`strADSiteName = objArgs(1)`).

This solution only scratches the surface of this functionality's possibilities. Other solutions throughout this book also leverage context menu integration.

CORY BECHT'S RIGHT-CLICK TOOL

One of the best right-click utilities you'll find for SMS is a tool by Cory Becht to initiate SMS Advanced Client actions remotely. You can find this tool at myITforum.com: `http://www.myitforum.com/articles/8/view.asp?id=7099`. Once you've downloaded and installed the tool, you can initiate the following actions by right-clicking a collection or a computer object in the SMS Administrator console:

- Reassign the SMS site code
- Restart the SMS Agent host service
- Regenerate the SMS client GUID
- Rerun advertisements without modifying the advertisement
- Discovery
- Perform software inventory: delta and full
- Perform hardware inventory: delta and full
- Perform file collection
- Check software metering usage
- Refresh machine policies
- Evaluate policies
- Update Windows Installer sources
- Change port number
- Change cache size

See Also

- Recipe 3-21, "Adding a Right-click Option to Affect All Members of a Collection," shows how to use these functions on all members of a collection.
- Cory Becht's right-click tool at `http://www.myitforum.com/articles/8/view.asp?id=7099` enables you to remotely manage one or multiple SMS clients, as described in this recipe.
- The SMS 2003 SDK details the context menu integration process in a section titled "Implementing Context Menu Integration."

3-21. Adding a Right-click Option to Affect All Members of a Collection

Problem

You want to add an action to the right-click functionality in the SMS Administrator console that affects *all* systems in a specific collection.

Solution

This recipe's solution is an improvement on the previous recipe's VBScript solution. This solution provides right-click functionality for both a single system and multiple systems in a collection. Here are the steps to follow:

1. From the system that has the SMS Administrator console installed, browse to `C:\SMSAdmin\` (or `SMSAdmin` or `SMS`, depending on SMS components installed).
2. Create a new directory called `SMSTools` and open it for viewing.
3. Create a batch file called `RefreshMachinePolicy.vbs`.
4. Edit `RefreshMachinePolicy.vbs` in Notepad, and enter the following text:

```
if wscript.arguments(0) = "S" then
    strComputerName = wscript.arguments(1)
    Set smsClient = GetObject("winmgmts:\\" & strComputerName & _
        "\root\ccm:SMS_Client")
    smsClient.RequestMachinePolicy(0)
    wscript.echo "Machine Policy Refreshed on " & strComputerName
    wscript.sleep 5000
else
    strSMSServer =  wscript.arguments(1)
    strSMSNameSpace = wscript.arguments(2)
    strSMSCollectionID = wscript.arguments(3)
    Set loc = CreateObject("WbemScripting.SWbemLocator")
    Set objSMS = loc.ConnectServer(strSMSServer, strSMSNameSpace)
    set ColMembers=objSMS.ExecQuery _
        ("select Name from SMS_CM_RES_COLL_" & strSMSCollectionID)
    for Each Member in ColMembers
        wscript.echo "Refreshing Machine Policy on " & Member.Name
        Set smsClient = GetObject("winmgmts:\\" & Member.Name & _
            "\root\ccm:SMS_Client")
        smsClient.RequestMachinePolicy(0)
    next
    wscript.sleep 5000
end if
```

5. Save `RefreshMachinePolicy.vbs`.
6. Open the Windows registry editor by selecting Start ➤ Run, entering **Regedit.exe**, and then clicking OK.
7. Browse to `HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\MMC\NodeTypes\{4D26C0D4-A8A8-11D1-9BD9-00C04FBBD480}\Extensions`.
8. From the `Extensions` key, create a new key named `SMS_Tools`.
9. From the `SMS_Tools` key, create a new key named `RequestMachinePolicy`.
10. From the `RequestMachinePolicy` key, create a new string value called `Name`, and for the data, enter **Request Machine Policy**.
11. From the `RequestMachinePolicy` key, create a new string value called `CommandLine`, and for the data, enter **cscript.exe C:\SMSAdmin\SMSTools\RefreshMachinePolicy.vbs S ##SUB:Name##**.
12. Browse to `HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\MMC\NodeTypes\{3AD39FF1-EFD6-11D0-BDCF-00A0C909FDD7}\Extensions`.
13. From the `Extensions` key, create a new key named `SMS_Tools`.
14. From the `SMS_Tools` key, create a new key named `RequestMachinePolicy`.

15. From the `RequestMachinePolicy` key, create a new string value called `Name`, and for the data, enter **Request Machine Policy**.
16. From the `RequestMachinePolicy` key, create a new string value called `CommandLine`, and for the data, enter **cscript.exe C:\SMSAdmin\SMSTools\RefreshMachinePolicy.vbs M ##SUB:__SERVER## ##SUB:__Namespace## ##SUB:COLLECTIONID##**.
17. To verify single system functionality, right-click a system in any collection and select SMSTools ➤ Request Machine Policy.
18. To verify multiple system functionality, right-click any collection and select SMSTools ➤ Request Machine Policy.

Discussion

The discussion in the previous recipe's solution explains most of what you see in this solution. Here we will describe the differences and provide additional required information.

First, an `if-else` statement is included to allow functionality from one script for initiating on both a single system and a collection. Notice that in step 11 an "S" is added for "single" computer, while in step 16 an "M" was added to signify "multiple" computers. By adding an "S" or "M" to the command line, we can determine which part of the VBScript to execute at runtime. The first section of the script (under the `if` statement) is identical to the previous solution. The second section of the script (under the `else` statement) is the code that pertains directly to performing functions on multiple systems.

If, for example, an "M" is passed to the script, we capture the arguments into variables for clarity, and then connect to the SMS namespace. After connecting to the SMS namespace on the SMS site, we execute a query to obtain all members of the desired collection. During the `for-each` loop, we perform the actual process of refreshing the machine policy on the system. Finally, the script sleeps for five seconds to allow the administrator a moment to view the results. Before implementing this code into production, be sure to add error handling (or at least an `On Error Resume Next` at the beginning of the script). If you attempt to connect to a system that is not currently on the network, the script will fail *very slowly*. Later in this book, we provide an example to demonstrate how to verify a system is on the network before attempting to connect to it via WMI.

Also in Recipe 3-20, "Adding a Right-click Option to Affect a Member of a Collection," we discuss Cory Becht's right-click tool. This tool can be used on both a collection and a specific member of a collection. Don't reinvent the wheel! Download and install Cory's application, and then review the VBScript to learn how to extend and customize to your needs.

See Also

- Recipe 3-21, "Adding a Right-click Option to Affect all Members of a Collection," describes how to use these functions on all members of a collection.
- Cory Becht's right-click tool at `http://www.myitforum.com/articles/8/view.asp?id=7099` enables you to remotely manage one or multiple SMS clients, as described in this recipe.
- The SMS 2003 SDK details the context menu integration process in a section titled "Implementing Context Menu Integration."