Using the VMware Mirage API to Develop Your Own Front-End Interfaces

VMware Mirage 5.1 and Later

TECHNICAL WHITE PAPER

vmware®

Table of Contents

VMware Mirage™ 5.1, released in September 2014, includes a new and exciting feature: the Mirage API, which lets you develop your own front-end interfaces to the Mirage infrastructure. This feature can make it easier to manage your endpoint devices. In addition, you can integrate your Mirage environment with third-party systems, such as scheduling and automation systems, to improve Mirage administrator efficiency and reduce the potential for manual errors.

Following are some guidelines and best practices for developing applications that use the Mirage API.

Setting Up Your Environment for the Mirage API

The Mirage API is hosted on Microsoft Internet Information Services (IIS). To install the API, use the Mirage Web Manager. You must have the following Mirage components installed before continuing:

- Mirage Management Server 5.1 or later
- Mirage Server 5.1 or later
- Mirage Web Manager 5.1 or later

For more information on these components, see the VMware Mirage Installation Guide.

The Mirage API is delivered as a Simple Object Access Protocol (SOAP) Web service, which requires support from Windows Communication Foundation (WCF) HTTP Activation. The Mirage Web Manager supports Windows Server 2008 R2, Windows Server 2012, and Windows Server 2012 R2. Installation instructions for the WCF HTTP Activation feature differ for the various operating system versions.

Windows Server 2008 R2

Use the following steps to install WCF HTTP Activation on Windows Server 2008 R2:

1. Log in as the administrator.
2. Click **Start > Administrative Tools** and select **Server Manager**.
3. In the Server Manager Window, click **Features** and select **Add Features**.
4. Expand .NET Framework 3.5.1 Features and select **WCF Activation**.
5. Select **HTTP Activation**.
6. Click **Install**.

There is one more step. Mirage Web Manager uses .NET Framework 4.0, and you must have it installed on the server to successfully use the Mirage API. As a result, you need to use the ASP.NET IIS Registration Tool (`Aspnet_regiis.exe`) to update the ASP.NET ISAPI version. Run the following command from the command prompt to update the ASP.NET ISAPI version:

```
%WINDIR%\Microsoft.NET\Framework\v4.0.30319\aspnet_regiis.exe –iru
```

Windows Server 2012 and 2012 R2

Use the following steps to install WCF HTTP Activation on Windows Server 2012 and 2012 R2:

1. Log in as the administrator.
2. Click **Start > Control Panel**, and select **Turn Windows features on or off**.
3. Click **Next** until the Select Features window appears.
4. Expand **.NET Framework 4.5 Features** and select **WCF Services**.
5. Select **HTTP Activation**.
6. Click **Install**.

After you install WCF HTTP Activation in the Mirage Web Manager, you can use your browser to check the status of the Mirage API service. The URL of the Mirage API service is `https://server-address:744d/mirageapi/MitService.svc`, where `server-address` is the IP address or FQDN of the Mirage Web Manager.

Note: Do *not* use "`localhost`" or "`127.0.0.1`" as the server address, Doing so might cause problems in later steps.

Figure 1 shows the results of a successful installation.

Figure 1: Mirage API Web Service Confirmation

From the second link in the MitService Service Web page in Figure 1, download and save the single WSDL (Web Services Description Language) file. This file is used to generate the libraries your program uses to consume the Mirage API Web service.

Configuring Your Development Environment

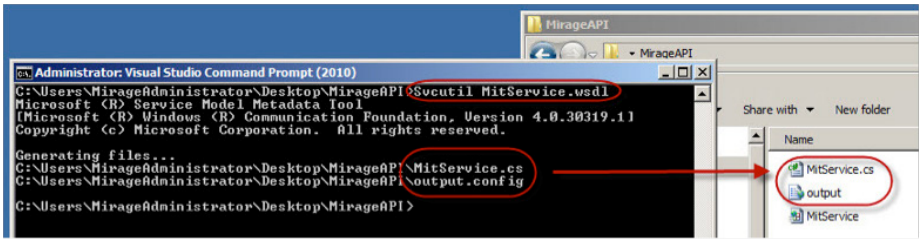
After the back-end infrastructure is configured and operational, you can focus on the application development environment. You can use C# (in the Microsoft .NET environment), Java, or Python to develop applications based on the Mirage API. The configurations for these languages are described in the following sections.

Windows C# Development Environment

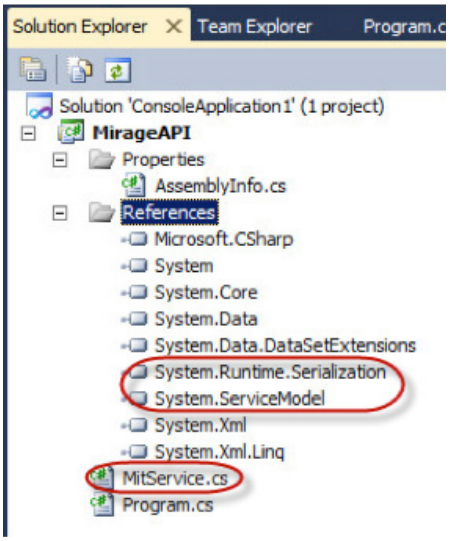
Use the following steps to set up the Windows C# development environment:

1. Locate the single WSDL file you downloaded from the Mirage API Web service.
2. Open the Visual Studio command prompt and navigate to the directory where the single WSDL file is stored.
3. Run the `Svcutil MitService.wsdl` command.

   `Svcutil.exe` is the ServiceModel Metadata Utility Tool, which generates the Mirage API client source code.



4. Copy the generated source code file to the folder of your C# project.
5. Add the source code to your C# project.
6. Add `System.Runtime.Serialization` and `System.ServiceModel` references to your C# project.



After you complete these tasks, you can develop C# programs with the Mirage API.

Java Development Environment

Use the following steps to set up the Java development environment:

1. Install Java and set the environment variable `%JAVA_HOME%`.
2. Download the Java SOAP library `Axis2` from Apache Software Foundation, for example, `axis2-1.6.2.zip`.
3. Unzip the library to a folder and set the environment variable `%AXIS2_HOME%` to the path of this folder. For example, this folder might be `C:\axis2-1.6.2`, so that the variable is `AXIS2_HOME = C:\axis2-1.6.2`.
4. Generate Java classes using the following command from command prompt:

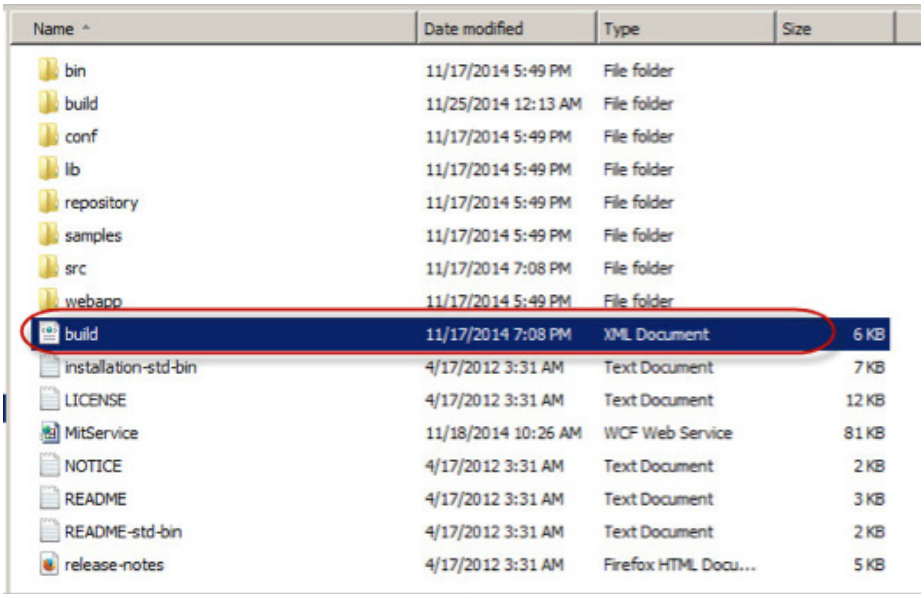
   ```
   %AXIS2_HOME%\bin\wsdl2java -uri MitService.wsdl -p
   com.vmware.mirage.mit
   ```

   where `MitService.wsdl` is the single WSDL file you downloaded from the Mirage API Web service.

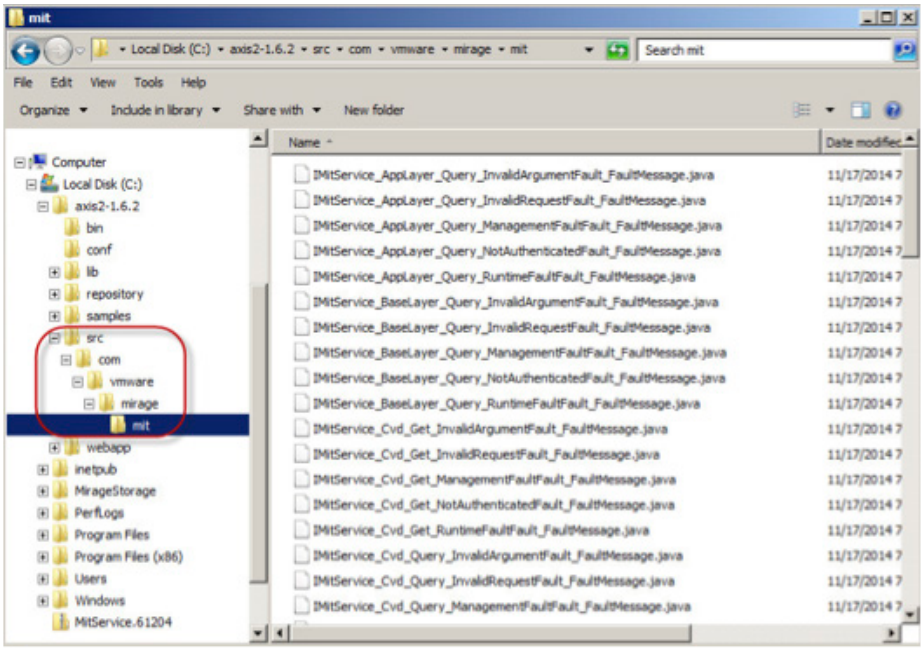
   When this command succeeds, a new folder is generated in `%AXIS2_HOME%:`

   ```
   src\come\vmware\mirage\mit
   ```

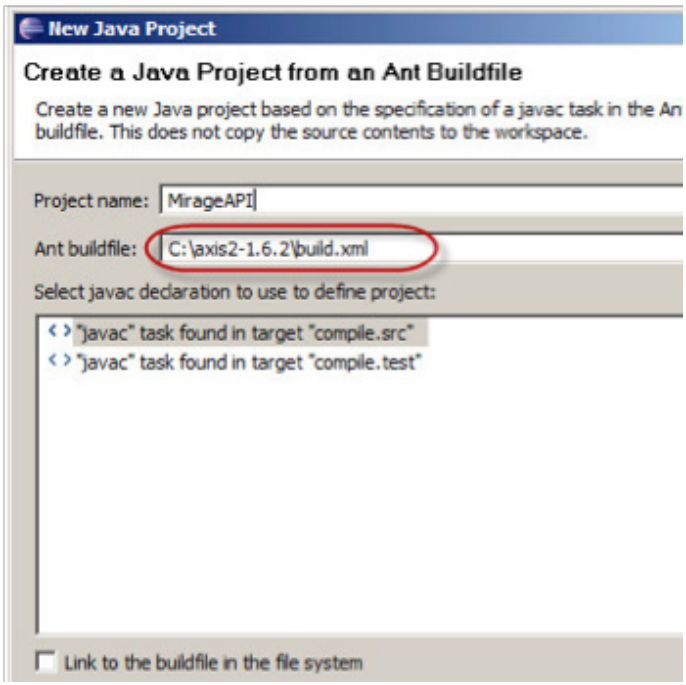
   An Ant `build` file is created in this folder:



New Java classes are also generated in this folder:



5. Create a project from the created Ant `build` file:



After you complete these tasks, you can develop Java programs with the Mirage API.

Python Development Environment

To set up the Python development environment:

1. Install Easy Install.
2. From the command prompt, use the following command to install suds:

   ```
   easy_install.exe suds
   ```

   Your program will now use suds to consume the Mirage API Web Service.

```
C:\Python27\Scripts>easy_install.exe suds
Searching for suds
Reading https://pypi.python.org/simple/suds/
Best match: suds 0.4
Downloading https://pypi.python.org/packages/source/s/suds/suds-0.4.tar.gz#md5=b
7502de662341ed7275b673e6bd73191
Processing suds-0.4.tar.gz
Writing c:\users\wuj\appdata\local\temp\easy_install-lgdo4k\suds-0.4\setup.cfg
Running suds-0.4\setup.py -q bdist_egg --dist-dir c:\users\wuj\appdata\local\tem
p\easy_install-lgdo4k\suds-0.4\egg-dist-tmp-_2qokl
zip_safe flag not set; analyzing archive contents...
Adding suds 0.4 to easy-install.pth file

Installed c:\python27\lib\site-packages\suds-0.4-py2.7.egg
Processing dependencies for suds
Finished processing dependencies for suds
```

3. Import suds into your program and define plug-ins for the client.

   Following is sample code in Python IDLE:

```
File  Edit  Format  Run  Options  Windows  Help
##Import library from suds
from suds.client import Client
from suds.plugin import MessagePlugin
from suds.sax.attribute import Attribute

##Define log plugin
class LogPlugin(MessagePlugin):
  def sending(self, context):
    pass
  def received(self, context):
    pass

##Define soap fixer
class SoapFixer(MessagePlugin):
    pass

##Create client
client = Client('https://<Mirage_server_IP>:7443/MirageApi/MitService.svc?wsdl', plugins=[LogPlugin(),SoapFixer()])

username, password, domain = ('username', 'password', 'domain')

##Log in
client.service.Login(username, password)

## Perform your tasks

##Log out
client.service.Logout()
```

For more instructions about suds, see the suds documentation.

After you complete these tasks, you can develop Python programs with the Mirage API.

Use Cases and Workflows

Access to the Mirage API service requires authentication. Figure 2 shows a workflow for all use cases:

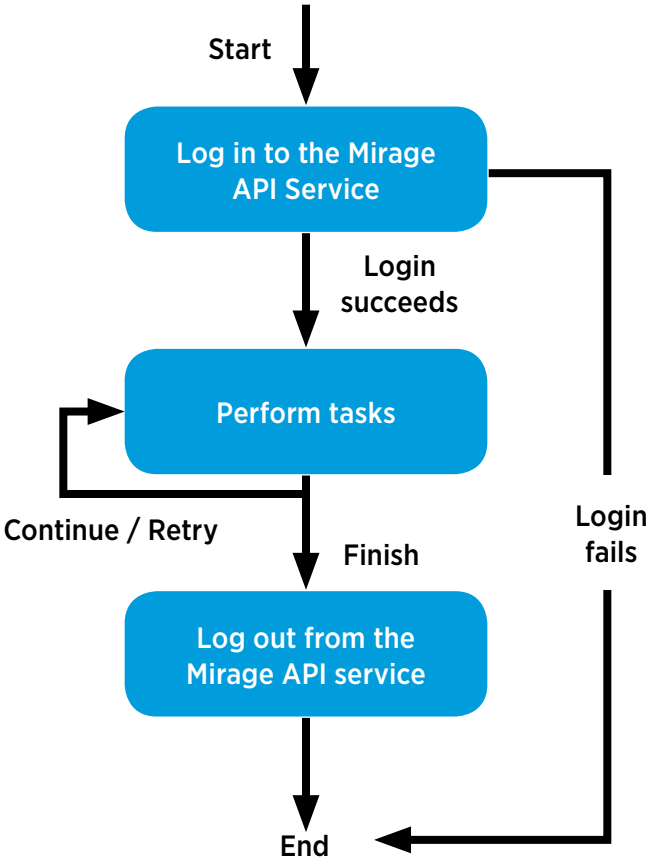


Figure 2: Workflow for All Use Cases

Two major use cases are supported in Mirage 5.1: centralization, and OS migration. Figure 3 shows a possible workflow for centralization. For a similar workflow for OS migration, see Figure 4.

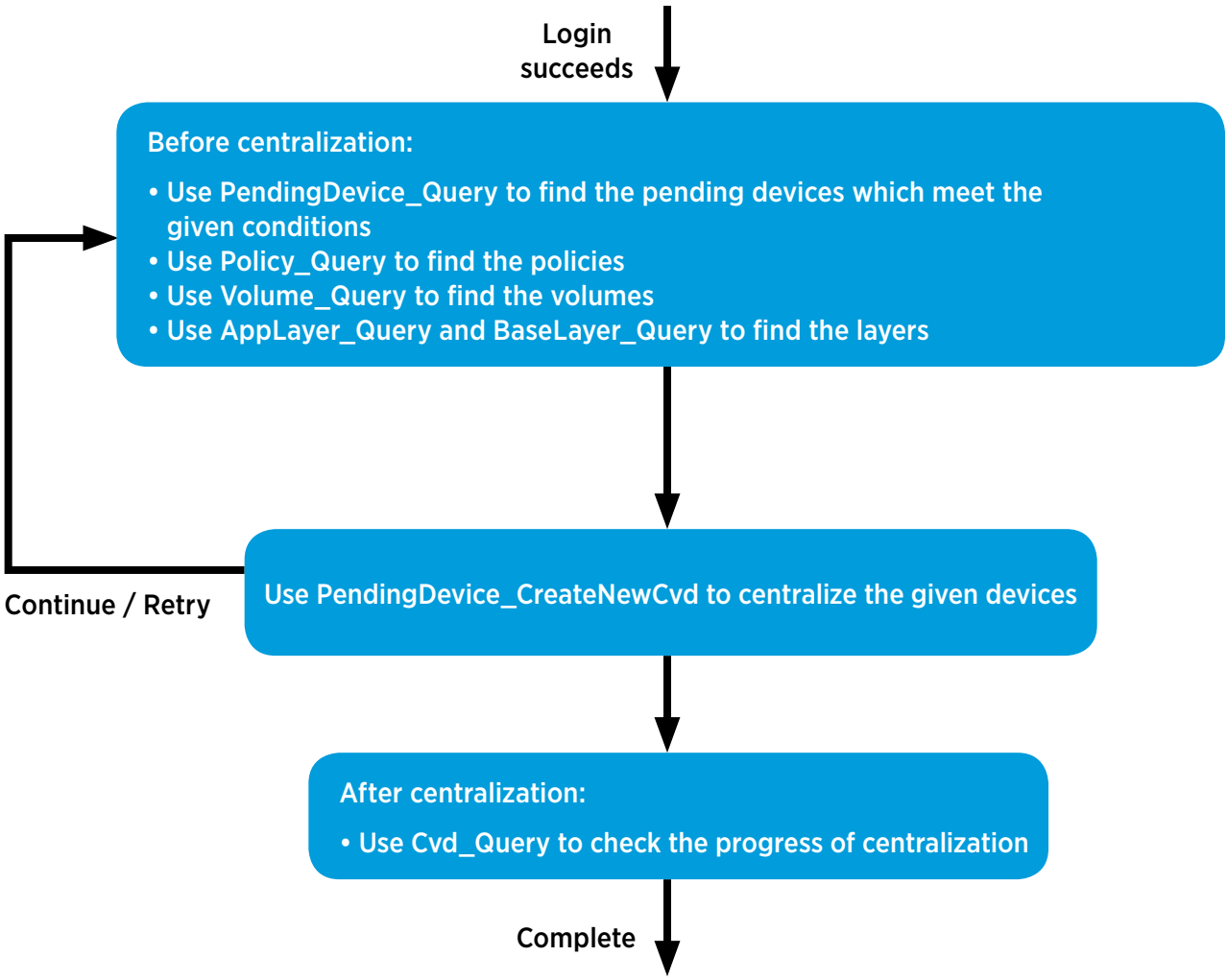


Figure 3: Workflow for Centralization

It is not *necessary* to perform any particular task in this workflow before centralization or after centralization, but it is *recommended* that you perform some of them so that you gather important information, such as the names of the policies and volumes during the centralization process.

Note: This workflow omits steps to log in or log out because both are required in every workflow.

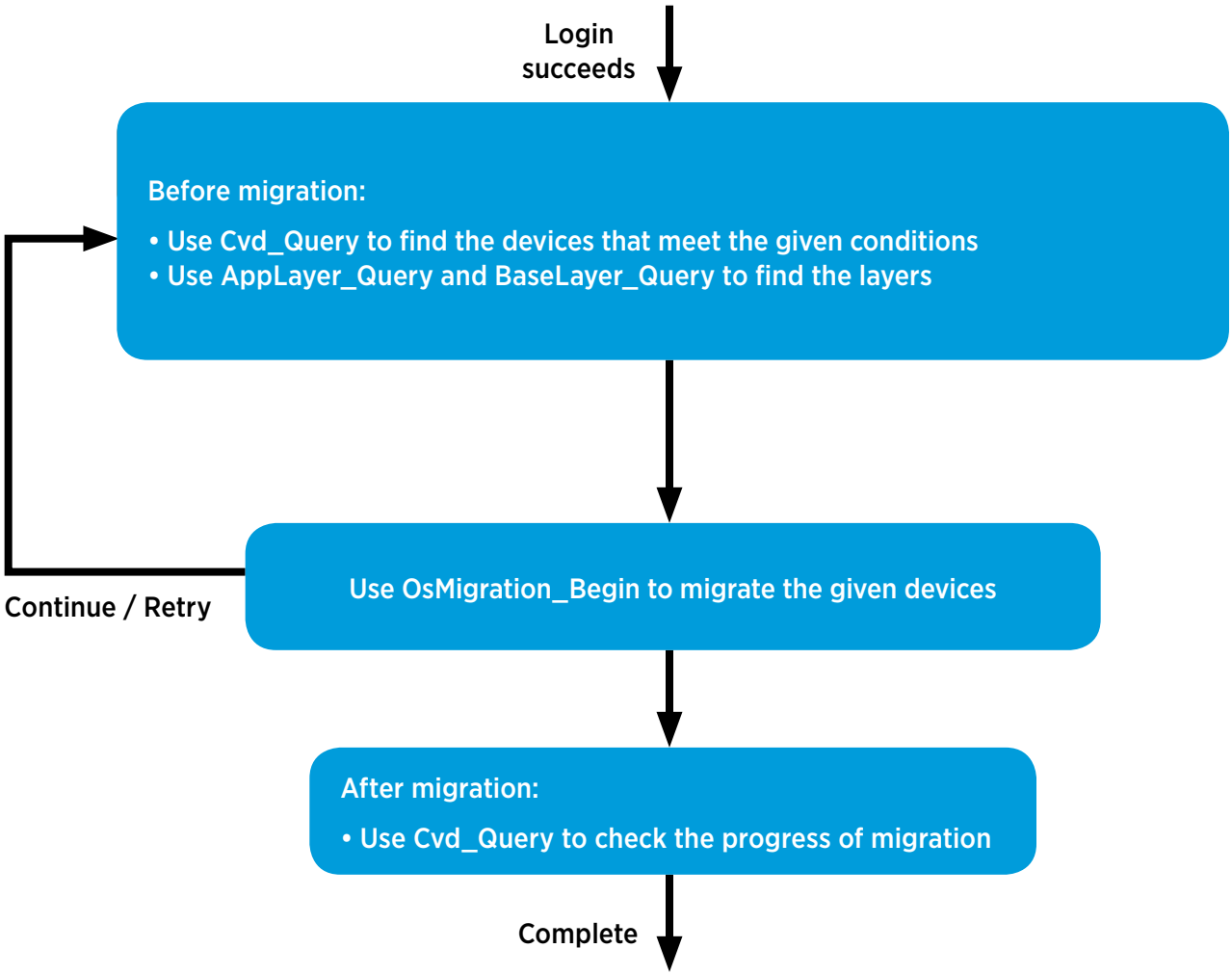


Figure 4: Workflow for OS Migration

Other use cases are supported, such as for inventory. For example, use `Cvd_Query` to find all the devices with names containing `api`; and use `PendingDevice_Query` to find the pending devices, for example, to find all the pending devices that are Windows 7.

For more detailed information, such as object types, methods, properties, and sample applications, see the VMware Mirage API Reference.

The Mirage API is an ever-evolving part of the Mirage solution, and we will continue to develop it in future releases. To comment on this paper, contact the VMware End-User Computing Solutions Management and Technical Marketing team at twitter.com/vmwarehorizon.

Additional Resources

VMware Mirage Documentation

- VMware Mirage Installation Guide
- VMware Mirage API Reference

Download VMware Mirage

Mirage Community Forum

VMware Developer Forum

About the Author and Contributors

Judy Wu, End-User-Computing Solution Engineer, VMware, wrote this document.

The following people contributed content and review:

- Zhibin He, Senior Member of the Technical Staff, End-User Computing, VMware
- Stephane Asselin, End-User-Computing Architect, VMware

vmware®

VMware, Inc. 3401 Hillview Avenue Palo Alto CA 94304 USA Tel 877-486-9273 Fax 650-427-5001 www.vmware.com