Professional Expertise Distilled

VMware vRealize Orchestrator Essentials

Get hands-on experience with vRealize Orchestrator and automate your VMware environment

Daniel Langenhan

[PACKT] PUBLISHING enterprise professional expertise distilled

VMware vRealize Orchestrator Essentials

Get hands-on experience with vRealize Orchestrator and automate your VMware environment

Daniel Langenhan

[PACKT] PUBLISHING enterprise
professional expertise distilled

BIRMINGHAM - MUMBAI

VMware vRealize Orchestrator Essentials

Every effort has been made in the preparation of this book to ensure the accuracy of the information presented. However, the information contained in this book is sold without warranty, either express or implied. Neither the author, nor Packt Publishing, and its dealers and distributors will be held liable for any damages caused or alleged to be caused directly or indirectly by this book.

Packt Publishing has endeavored to provide trademark information about all of the companies and products mentioned in this book by the appropriate use of capitals. However, Packt Publishing cannot guarantee the accuracy of this information.

First published: September 2015

Production reference: 1210915

Published by Packt Publishing Ltd.
Livery Place
35 Livery Street
Birmingham B3 2PB, UK.

ISBN 978-1-78588-424-5

www.packtpub.com

Credits

Author
Daniel Langenhan

Reviewers
Burke Azbill
Christophe Decanini
Stefan Rodenstein

Commissioning Editor
Dipika Gaonkar

Acquisition Editor
Ruchita Bhansali

Content Development Editors
Pooja Nair
Amey Varangaonkar

Technical Editor
Pramod Kumavat

Copy Editor
Vedangi Narvekar

Project Coordinator
Francina Pinto

Proofreader
Safis Editing

Indexer
Priya Sane

Graphics
Sheetal Aute

Production Coordinator
Nitesh Thakur

Cover Work
Nitesh Thakur

Foreword

When I began working with Orchestrator in 2007, shortly before VMware's acquisition of the Swiss-based company, it was a Dunes product named Virtual Service Orchestrator that had very little in the way of actual documentation. This presented me with quite a challenge in learning the ins and outs of such a powerful tool.

Fast-forward eight years, and we see numerous blogs focused on VMware's vRealize Orchestrator as well as multiple products that are built on top of it and which rely on its powerful orchestration engine. Additionally, we see a thriving community that has continued to grow around the product as the Orchestrator sheds its old nickname of being VMware's best-kept secret. It gives me great pleasure to see Daniel's work on *vRealize Orchestrator Essentials* come together to help Orchestrator developers quickly become more productive. I only wish that such a book, or even a fraction of this level of detailed documentation, was available back when I started working with Orchestrator.

I wish you all the best for your integration and automation projects powered by Orchestrator. This book is sure to help you get a good start with the product!

Burke Azbill

Consulting Architect, LiVefire Solutions and Services
VMware, Inc.
@TechnicalValues / `www.vcoteam.info`

Foreword

If there is something that I did right from the beginning of my IT career, it was automation and integration. Coming from a development background, it was natural for me to script or develop small tools to automate most of the manual tasks that our team had to accomplish. It was the mid-1990s and this required using a combination of scripting and programming languages.

Later, the IT field evolved by providing Application Programming Interfaces to get information and perform remote operations. Simpler scripting languages were released to make these easily consumable, and virtualization provided the agility that was missing in the automation of a lot of datacenter operations.

Streamlining these operations as modular, reusable, and highly available workflows is what VMware vRealize Orchestrator brought to the equation.

After being prominently used not only by service providers to build their public cloud offerings, but also by various enterprises for their private cloud, Orchestrator is fast becoming a core component of several VMware Software-Defined Data Center capabilities such as self-service provisioning, custom services authoring, DevOps, and automatic remediation. The Orchestrator workflows that power these can be leveraged to design your own custom automation and integrations.

In 2014, Daniel Langenhan wrote *VMware vRealize Orchestrator Cookbook*. This is the most comprehensive book to date that covers Orchestrator. With Orchestrator being used more broadly within different IT job roles, a need for an entry-level book arose. This is now being covered by *VMware vRealize Orchestrator Essentials*. It explains the Orchestrator architecture and steps through its installation, configuration, workflow design, and packaging from a newcomer's perspective. By leveraging it, you will get to grips with the basics of Orchestrator and be ready to implement your own automation and integrations.

I have been working with Daniel Langenhan for the past six years, and I am delighted that he managed to share the great amount of knowledge and experience that he acquired by delivering orchestration solutions to our customers so that you can unleash the power of Orchestrator.

I wish you all the best for your workflow designs and deployments.

Christophe Decanini

Consulting Architect

VMware Global Technical and Professional Services

Office of the CTO Ambassador

@vCOTeam / vCOTeam.info

About the Author

Daniel Langenhan is a virtualization expert with formidable skills in architecture, design, and implementation for large multi-tier systems. His experience and knowledge of process management, enterprise-level storage, and Linux and Windows operating systems has made him and his business a highly sought-after international consultancy in the Asia-Pacific and European regions for multinational clientele in the areas of finance, communication, education, and government. Daniel has been working with VMware products since 2002 and has been directly associated with VMware since 2008. He has a proven track record of successfully integrating virtualization into different business areas while minimizing the costs and maximizing the reliability and effectiveness of the solution for his clients.

His expertise and practical approach towards VMware has resulted in the publication of the following books by Packt Publishing:

- *Instant VMware vCloud Starter*
- *VMware View Security Essentials*
- *VMware vCloud Director Cookbook*
- *VMware vRealize Orchestrator Cookbook*

He has also lent his expertise to many other publishing projects as a technical editor.

> I would like to thank my wife, Renata, for her endless effort and patience as well as for helping me create this book.
>
> I would like to thank Christophe Decanini and Burke Azbill for their technical expertise. I would especially like to mention Stefan Rodenstein, who helped me make this book accessible for beginners.

About the Reviewers

Burke Azbill has been a technology professional since 1996 and holds certifications from Cisco, Citrix, ITIL, Linux Professional Institute, Microsoft, Novell, and VMware. He joined VMware in 2007 as a part of the acquisition of Dunes Technologies from Lausanne, Switzerland, where he began working with Orchestrator. Burke is the founder of, and a contributor to, the blog `http://www.vcoteam.info` as well as a leading contributor to the VMTN communities for Orchestrator. During his tenure at VMware, Burke has trained hundreds of employees on Orchestrator, built many integrations for customers and partners, and worked in various roles in VMworld Hands-on Labs. He is the contributing author for *VMware vCloud Architecture Toolkit (vCAT)* (VMware Press, 2013) and a technical resource for *Automating vSphere with VMware vCenter Orchestrator* (VMware Press, 2012), *VMware vSphere for Dummies* (For Dummies, 2011), and *VMware vRealize Orchestrator Cookbook* (Packt Publishing, 2015).

Christophe Decanini is a consulting architect at VMware, Inc. He joined VMware in 2007 as a part of the acquisition of Dunes Technologies.

Based in Gland, Switzerland, Christophe is an Orchestrator expert who provides support to VMware customers, partners, and field resources globally.

He has worked on several important contributions to Orchestrator, including product features that were vital in making Orchestrator a successful product.

He has presented orchestration solutions at conferences such as VMworld and is a chief contributor to the blog `www.vcoteam.info` and the official VMware Orchestrator community.

He has reviewed and contributed books that cover Orchestrator, including *VMware vCloud Architecture Toolkit* and *Automating vSphere with vCenter Orchestrator*, both by VMware press, and *VMware vRealize Orchestrator Cookbook* by Packt Publishing. He has been awarded the vExpert designation for several years and is a VMware CTO Ambassador.

He has 20 years of experience in IT automation and holds a VMware Certified Professional 5 – Data Center Virtualization (VCP5-DCV) certification and a bachelor's degree in computer science. You can follow him on Twitter. His Twitter handle is `@vCOTeam`.

Stefan Rodenstein is a storage, server, and virtualization expert with experience in the design, implementation, and operation of cloud infrastructure.

With his outstanding knowledge of enterprise-level storage, Linux and other Unix-like operating systems, virtualization, system programming, monitoring, databases, and performance tuning, he has planned and installed data center and cloud solutions in Europe and North America for several DAX and DOW Jones companies.

He has a diploma (the equivalent of an MSc) in physics and is currently working for Atos.

Table of Contents

Preface

This is the second book that I have written on the subject of vRealize Orchestrator, the first being the *vRealize Orchestrator Cookbook*. The cookbook covers a lot of ground and is pretty full on. So we came to the conclusion to write a book that covers more of the essentials of Orchestrator and helps novices get an easy start.

This book focuses on the basic skills that you need to learn to use Orchestrator. When you've worked through it, you will know enough to create workflows and use them productively as well as share them with the rest of the community. In this book, I tried to keep the focus on the essential skills, and I will explain about backgrounds as much as possible. The book can be regarded as a training class where we build up our skills from one chapter to the next. We will develop a new workflow and then constantly improve it to understand and implement new topics.

Please note that VMware vRealize Orchestrator was renamed from vCenter Orchestrator in late 2014 and is not a new product. In this book, we will just use the name Orchestrator.

This book has been written with vRealize Orchestrator 6.02, but almost everything discussed in this book is applicable to the older versions as well. The following table gives you an overview of what won't work in which version:

vCenter Orchestrator (vCO)	4.x	No Web Client integration No REST
	5.1.x	No debug mode General improvements in usage
	5.5.x	General improvements in usage vCO Appliance Cluster support

vRealize Orchestrator (vRO)	5.5.2	Switch workflow element Default Error
	6.0.x	The book is based on this version No Webviews SOAP deprecated

What this book covers

The best approach towards this book is to start at the beginning and work your way towards the end. I also recommend that you try out each example, as this will not only increase your skill, but also your understanding.

Chapter 1, Architectural Overview, covers all the questions regarding the sizing, connection ability, and plug-ins of Orchestrator.

Chapter 2, Deploying and Configuring the Orchestrator Appliance, shows you how to install and configure the vRO Appliance.

Chapter 3, Integrating Orchestrator with vSphere, deals with connecting and using Orchestrator with vSphere Web Client.

Chapter 4, Working with Workflows, is an introduction to how to start and schedule a workflow as well as deal with the outcomes.

Chapter 5, Combining and Modifying Workflows, shows you how to build your own workflows by using the existing ones from the huge library that is a part of the product.

Chapter 6, Advanced vRO Scripting with JavaScript, will go through ways to improve your scripts by using JavaScript as well as arrays and actions.

Chapter 7, Improving Workflows with Presentation, shows you how to make workflows not only more user-friendly, but also less prone to error entries.

Chapter 8, Errors, Logs, and Debug Mode, introduces you to the usage of logs and error handling. We also discuss the Orchestrator debug mode.

Chapter 9, Packing It All Up, looks into the creation, export, and import of packages and workflows.

What you need for this book

This book has been written with vRealize Orchestrator 6.02 Appliance. However, as mentioned earlier, you can also use the vCenter 5.5 Appliance.

If you have a valid VMware vSphere or vCenter standard license, you automatically own Orchestrator. You just have to download the appliance.

To use this book, you will need a VMware vSphere vCenter. You can use the trial version that is available for download at `www.vmware.com`. Please note that at the time of writing this book, the vSphere 6 trial version gives you no access to vRealize Orchestrator.

Who this book is for

This book is aimed especially at Orchestrator novices. You should be a reasonably experienced vSphere administrator. A basic understanding of the VMware vSphere terms and concepts will be helpful.

We made sure that the book is especially suitable for beginners. The team of technical reviewers consisted of not only the two most respected people in the world of Orchestrator, but also a total novice in Orchestrator.

Conventions

In this book, you will find a number of styles of text that distinguish between different kinds of information. Here are some examples of these styles, and an explanation of their meaning.

Code words in text, database table names, folder names, filenames, file extensions, pathnames, dummy URLs, user input, and Twitter handles are shown as follows: "Click here to start the Orchestrator Java Client. You can also access the Client directly by browsing to `https://[IP or FQDN]:8281/vco/client/client.jnlp`."

A block of code is set as follows:

```
var myOutput = new Number();
myOutput = 5;
```

New terms and **important words** are shown in bold. Words that you see on the screen, in menus or dialog boxes for example, appear in the text like this: "It's always a good idea to read the **Release Notes** to understand changes to the current version."

Warnings or important notes appear in a box like this.

Tips and tricks appear like this.

Reader feedback

Feedback from our readers is always welcome. Let us know what you think about this book—what you liked or may have disliked. Reader feedback is important for us to develop titles that you really get the most out of.

To send us general feedback, simply send an e-mail to `feedback@packtpub.com`, and mention the book title via the subject of your message.

If there is a topic that you have expertise in and you are interested in either writing or contributing to a book, see our author guide on `www.packtpub.com/authors`.

Customer support

Now that you are the proud owner of a Packt book, we have a number of things to help you to get the most from your purchase.

Errata

Although we have taken every care to ensure the accuracy of our content, mistakes do happen. If you find a mistake in one of our books—maybe a mistake in the text or the code—we would be grateful if you could report this to us. By doing so, you can save other readers from frustration and help us improve subsequent versions of this book. If you find any errata, please report them by visiting `http://www.packtpub.com/submit-errata`, selecting your book, clicking on the **Errata Submission Form** link, and entering the details of your errata. Once your errata are verified, your submission will be accepted and the errata will be uploaded to our website or added to any list of existing errata under the Errata section of that title.

To view the previously submitted errata, go to `https://www.packtpub.com/books/content/support` and enter the name of the book in the search field. The required information will appear under the **Errata** section.

Piracy

Piracy of copyrighted material on the Internet is an ongoing problem across all media. At Packt, we take the protection of our copyright and licenses very seriously. If you come across any illegal copies of our works in any form on the Internet, please provide us with the location address or website name immediately so that we can pursue a remedy.

Please contact us at `copyright@packtpub.com` with a link to the suspected pirated material.

We appreciate your help in protecting our authors and our ability to bring you valuable content.

Questions

If you have a problem with any aspect of this book, you can contact us at `questions@packtpub.com`, and we will do our best to address the problem.

1
Architectural Overview

In this first chapter, we will have a look at what Orchestrator actually does, where it fits into the VMware world, and how it can be used. We will cover the following topics in this chapter:

- What is VMware Orchestrator?
- Orchestrator plug-ins
- Licensing
- Compatibility and binaries
- Additional sources for information

What is VMware Orchestrator?

VMware vRealize Orchestrator is the latest iteration of the Orchestrator product that VMware started shipping with vSphere 4.

Most users new to VMware have problems understanding how Orchestrator fits into the VMware landscape and what it can be used for. So, let's have a closer look at this. We will start with the history of Orchestrator, then look into its features, and finally, how to expand its possibilities using plug-ins.

A short history of Orchestrator

Orchestrator started its life as Virtual Service Orchestrator (VS-O) with a small company named Dunes in Lausanne, Switzerland. In 2007, VMware bought Dunes, renaming the product as VMware Orchestrator (VMO), and then introduced Orchestrator into vSphere 4.0 as vCenter Orchestrator (vCO). Orchestrator's first stage debut was with VMware Lifecycle Manager, which used Orchestrator to automate the virtual infrastructure life cycle.

Orchestrator itself never really received the spotlight until the recent launch of VMware vCloud Automation Center (vCAC). In the beginning, vCAC used Orchestrator only as an extension, but with version 6.1, it became the central tool for automation.

In October 2014, VMware renamed vCenter Orchestrator (vCO) to vRealize Orchestrator (vRO) to align with their new strategies. vRO is not a new product; it is just the new name of vCO. VMware also renamed vCAC with version 6.2 to vRealize Automation.

> Due to the renaming, we will only refer to vRO as Orchestrator in this book.

In vSphere 4.x and 5.x, Orchestrator was automatically installed when one installed vCenter. Orchestrator began its life as a Windows install, which was finally phased out with the introduction of vSphere 6. With the start of vSphere 5, a Linux-based Orchestrator appliance was introduced, which now remains the only Orchestrator installation that is officially available.

Since the introduction of vRealize Automation, Orchestrator has become the focus of a lot of development and attention from VMware and others. More vendors have released plug-ins for Orchestrator and many have been discovering Orchestrator as an innovative solution for their automation needs.

Features of Orchestrator

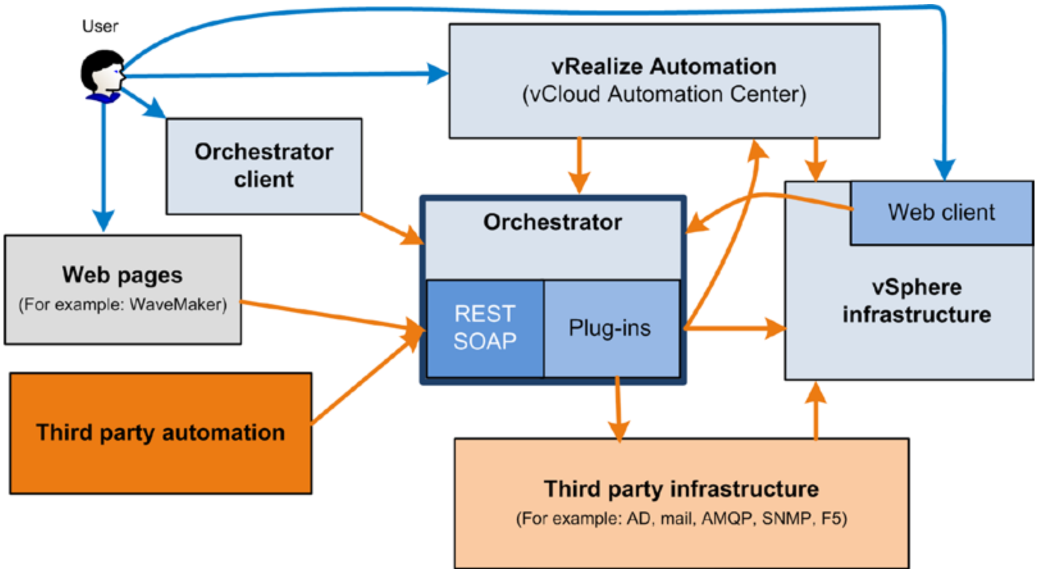
Most vSphere administrators don't know about Orchestrator, or underestimate its possibilities.

In essence, Orchestrator is a visual scripting tool using JavaScript. The programs/scripts that one creates with Orchestrator are called **workflows**. Each workflow consists of little building blocks called workflow elements. A workflow element can be a workflow, an action (a JavaScript script), or one of many other predefined commands. We will work with workflows in *Chapter 4*, *Working with Workflows*, and start creating our own workflows from *Chapter 5*, *Combining and Modifying Workflows*, onwards. Orchestrator comes with a large library of precreated workflows that cover a lot of the typical administrator work, such as creating VMs or connecting CD-ROMs.

A user can interact with Orchestrator either using the Orchestrator Client, a Java executable, via the vCenter vSphere Web Client or through websites that use the Orchestrator API. Orchestrator uses a full REST API (the SOAP API was removed in 6.x) so tools such as **vRealize Automation** (**vRA**) can easily communicate with Orchestrator.

Orchestrator uses a plug-in architecture to expand its base usability, enabling it to not only interact with various VMware products but also integrate with a range of other products. There are plug-ins to interact with hardware from vendors such as HP, Cisco, EMC, and NetApp as well as with other automation solutions such as Puppet and Microsoft SCOM or Microsoft SCVMM. We will discuss plug-ins a bit further down the track.

The following figure shows how one can interact with Orchestrator and how Orchestrator interacts with other solutions:



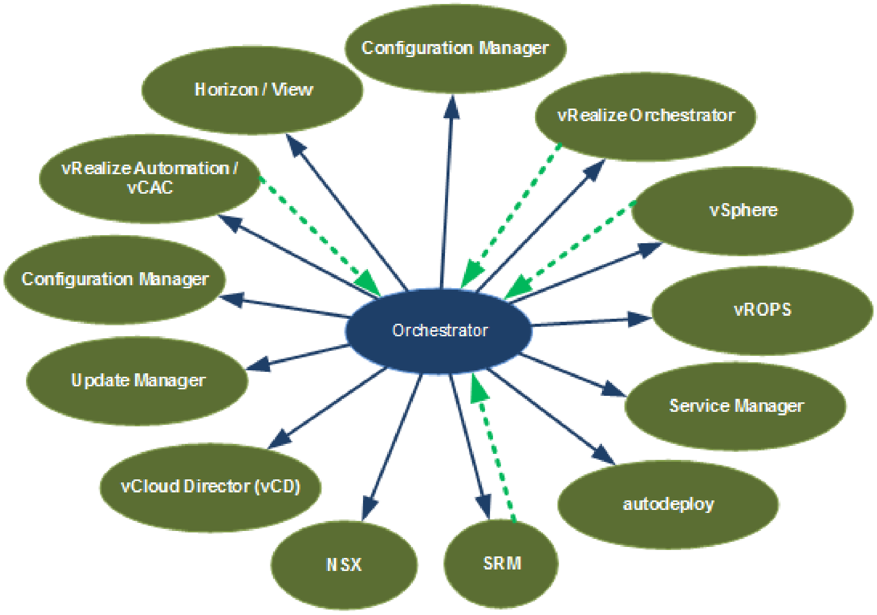
By executing Orchestrator workflows, one can interact with any Orchestrator technology integrated into Orchestrator. As this still sounds pretty complex, here are some examples:

- Using the vCenter vSphere Web Client, right-click on a cluster to schedule the deployment of a VM using an Orchestrator workflow
- Use an Orchestrator workflow to configure the CISCO UCS hardware of an ESXi server, and then automatically roll out an ESXi installation using VMware Autodeploy
- Connect to your EMC or NetApp storage and use an Orchestrator workflow to configure a new datastore and mount it directly on all ESXi hosts in a cluster

- Schedule an Orchestrator workflow that runs a PowerShell script against all Microsoft servers to collect information and then send that information via e-mail
- Intercept an SNMP trap and automatically trigger the execution of an Orchestrator workflow

VMware has been releasing Orchestrator plug-ins for most of its products. This allows you to create workflows that tie all the VMware products together making it possible to easily create interactions between VMware products using a single tool.

The following figure shows all currently existing VMware plug-ins that Orchestrator can use. The green dotted line also shows which VMware products can start Orchestrator workflows.



An example of plug-in interaction could be to create a VM in vSphere and configure it with a dedicated network using NSX, and then configure this VM automatically for SRM. Or another way around would be to use an SRM recovery step to execute an Orchestrator workflow via the API to launch some network configurations in NSX.

So what it comes down to is that Orchestrator is a central tool from which one can control, trigger, and drive separate products in your environment.

More about plug-ins

The real strength of Orchestrator lies in its ability to use plug-ins. Plug-ins can be written by anyone, as VMware publishes a guide on how to write them along with an SDK to help you get started. Plug-ins are mostly written by vendors to make it possible for Orchestrator to access their product. An unofficial full list of all existing plug-ins can be found at `http://www.vcoteam.info/links/plug-ins.html`.

Additional to this, there is the official plug-in site, the VMware solution exchange, where you can find more plug-ins that are vendor-specific or may require licensing:

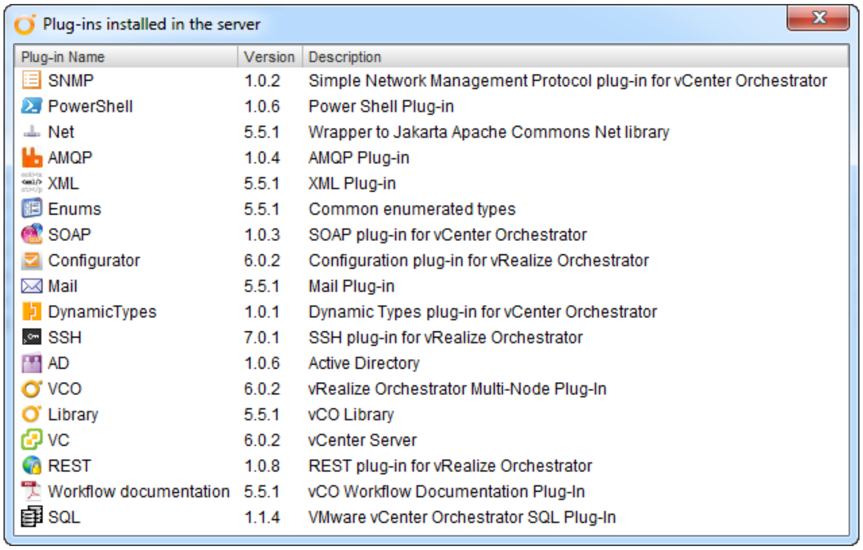
`https://solutionexchange.vmware.com/store/category_groups/cloud-management`

Examples of vendors that have plug-ins are BMC, Chef, CISCO, Docker, EMC, F5, Hitachi, HP, IBM, Infoblox, NetApp, Puppet, and Riverbed.

If there is no plug-in for a program you use, you could always use SOAP, REST, SSH, Mail, or PowerShell to access this program and integrate it into Orchestrator. Another possibility is to use Dynamic Types. Dynamic Types help you construct types and Orchestrator inventory entries enabling you to build XaaS (Anything as a Service). Have a look at Christophe Decanini's posts and examples:

`http://www.vcoteam.info/articles/learn-vco/281-enabling-vcloud-automation-center-xaas-with-vco-dynamic-types.html`

Orchestrator itself comes with quite a lot of plugins preinstalled. The following screenshot shows all the plug-ins that vRealize Orchestrator 6.0.2 comes with:



Plug-ins installed in the server

Plug-in Name	Version	Description
SNMP	1.0.2	Simple Network Management Protocol plug-in for vCenter Orchestrator
PowerShell	1.0.6	Power Shell Plug-in
Net	5.5.1	Wrapper to Jakarta Apache Commons Net library
AMQP	1.0.4	AMQP Plug-in
XML	5.5.1	XML Plug-in
Enums	5.5.1	Common enumerated types
SOAP	1.0.3	SOAP plug-in for vCenter Orchestrator
Configurator	6.0.2	Configuration plug-in for vRealize Orchestrator
Mail	5.5.1	Mail Plug-in
DynamicTypes	1.0.1	Dynamic Types plug-in for vCenter Orchestrator
SSH	7.0.1	SSH plug-in for vRealize Orchestrator
AD	1.0.6	Active Directory
VCO	6.0.2	vRealize Orchestrator Multi-Node Plug-In
Library	5.5.1	vCO Library
VC	6.0.2	vCenter Server
REST	1.0.8	REST plug-in for vRealize Orchestrator
Workflow documentation	5.5.1	vCO Workflow Documentation Plug-In
SQL	1.1.4	VMware vCenter Orchestrator SQL Plug-In

In this book, we will work with the vCenter plug-in. All other plug-ins are discussed in detail and with example workflows in the *VMware vRealize Orchestrator Cookbook*.

Obtaining Orchestrator

Now that we know what Orchestrator does, we should have a look at licensing and how to actually get a copy of Orchestrator.

Licensing

Orchestrator is licensed with vCenter. You need at least a vSphere Standard licensing to use Orchestrator. Although Orchestrator is not available with the Essentials and Essentials Plus licensing, it can be operated in Player mode only. This limits your usage to executing existing workflows and prevents you from editing or creating them. All other licensing models allow you to use Orchestrator with all its features.

Downloading

As already stated, Orchestrator was available as a Windows install in versions 4.x and 5.x. With the introduction of Version 6.x, the Windows installable has been retired. In Version 5.x, a Linux-based appliance was introduced, and it is now the only available version of Orchestrator.

> Please remember that Orchestrator 5.x is called vCenter Orchestrator (vCO) and Orchestrator 6.x is called vRealize Orchestrator (vRO).

If you are using vSphere 4.x and 5.x and you have installed vCenter, you will already have had Orchestrator installed as well. To enable it, you just need to start the required service in Windows. Due to the retirement of the Windows install, this book will not cover this method; however, it is described in the *VMware vRealize Orchestrator Cookbook*.

The Orchestrator appliance can be downloaded from the VMware website. You will find it in the vSphere downloads.

Compatibility

The VMware compatibility matrix shows that vRealize Orchestrator 6.0.2 is compatible with vCenter Server 5.1, 5.5, and 6. That said, you need to be a bit more careful; SSO has been changed quite a lot between vSphere 5.1, 5.5, and 6, so there still are some problems. Best practice dictates that you should stay with the same versions. You can use vRO 6 with vCenter 5.5 or 5.1, but you should replace the vCenter plug-in and that's not that easy.

For a beginner, I would suggest you stick to any of the following combinations of vSphere, vCenter, and Orchestrator:

vSphere	**vCenter**	**Orchestrator**
6.0	6.0	vRO 6.0.2
5.5	5.5 Update 2e	vCO 5.5.2.1
5.1	5.1 Update 3a	vCO 5.1.3.1

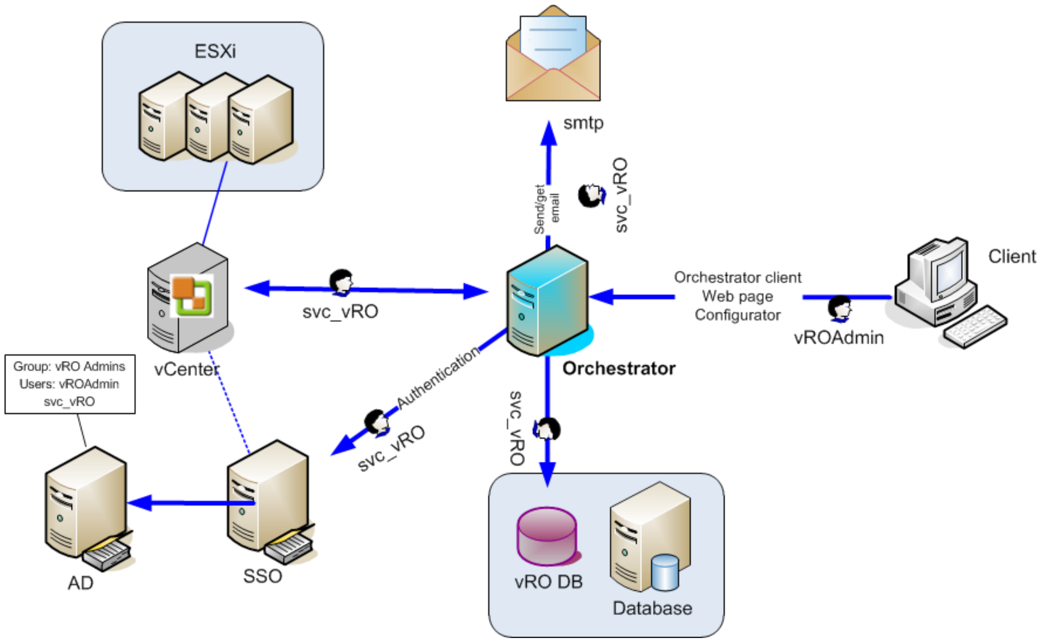
In this book, we focus on vSphere 6 with Orchestrator 6.

The Orchestrator architecture

By "architecture", IT people mean the framework, the items that make up the piece and its surroundings; basically, the same way as a "normal" architect who builds houses. The architecture of a house decides how it is built, how it fits together, and how it works in the context of its surroundings.

When talking about an IT product, we normally look at things like its limitations, its properties, what it depends on, and what it needs. The following figure shows Orchestrator and its dependent components.

The figure also contains the dedicated technical user account svc_vRO that is used to connect Orchestrator to other components.



Authentication

Another part of the architectural view is how Orchestrator authenticates users.

Authentication is the topic that describes how Orchestrator knows that a certain user can access it and what he is allowed to do. Since VMware introduced **SSO** (**Single Sign on**) in vSphere 5.1, it should be considered the preferred method for Orchestrator authentication. However, Orchestrator is also able to use AD and LDAP for authentication. We will configure Orchestrator with SSO in the next chapter.

Dedicated service account

After we have talked about authentication, we should also talk about a dedicated service account. What this means is that when Orchestrator itself needs to communicate with another service, for example the vCenter server, it should use a user account only used by Orchestrator. This is a pretty common practice in Enterprise systems and allows for easier reading of log files if there is a problem.

You can use one dedicated user for all services, as shown in the figure earlier, where the same technical user svc_vRO is used for vCenter, database, and e-mail. Alternatively, you can have one user for each service, a different user for e-mail, and one for the database connection.

Database

Orchestrator needs a database. The Appliance comes with an internal database (PostgreSQL); however, for production use, you should consider using a dedicated MS SQL, PostgreSQL, or Oracle database. If you are using Orchestrator in a production environment, you want to make sure that the database is backed up the same way your other important systems such as vCenter are.

VMware infrastructure

In theory, you can use Orchestrator without any VMware infrastructure around it; however, that's not fun. You should have at least a complete VMware vSphere environment consisting of a vCenter server and ESXi hosts. Orchestrator itself is a core component for VMware vRealize Automation (formally known as vCenter Automation Center, vCAC).

Other services

Other services that help make Orchestrator more productive are E-mail (SMTP system) or SNMP. This allows you to send e-mails from Orchestrator to users, notifying them about workflow executions and errors.

You can also access a POP or IMAP account via Orchestrator using the mail plug-in. This allows you to send e-mails to the Orchestrator server in order to execute workflows.

Limitations

Orchestrator is not almighty; there are still some limitations you should know about. vRealize Orchestrator version 6.0.2 has the following limitations when left unconfigured:

Maximal concurrent connected vCenters	20
Maximal concurrent connected ESXi hosts	1280
Maximal concurrent connected VMs	3500
Maximal concurrent running workflows	300

Steps may be taken to increase these values, but this topic is beyond the scope of this book and is covered in the product documentation.

So what to do if you need more than this? Easy; use the multinode plug-in to use more than one Orchestrator instance at the same time.

If you are using Orchestrator as your main tool for production, you might want to consider clustering it. Clustering provides high availability (HA) and load balancing of the number of workflows but does not increase the maximum connected vCenters / ESXi hosts / VMs to each Orchestrator. Clustering is not part of this book; however, you can find it in the second chapter of *VMware vRealize Orchestrator Cookbook*.

Additional sources for Orchestrator

Working with Orchestrator became much simpler in the last year. As more and more people and companies adopt Orchestrator for their automation, many more publications and posts are created. The following listing is essentially a snapshot of the most used resources.

Documentation

The official documentation for Orchestrator can be found on the VMware website. Go to `https://www.vmware.com/support/pubs/orchestrator_pubs.html`.

Please note that you can select the version you want to have the documentation for. It's always a good idea to read the **Release Notes** to understand changes to the current version.

Books

There are not that many books on Orchestrator. Actually, there are only three books that I know of, counting the one you are reading right now.

The first one that was widely recognized was Cody Bunch's book *Automating vSphere with VMware vCenter Orchestrator* published by VMware Press Technology. The book is based on version 4 of Orchestrator, but is still valid for most of its content. Cody has some really good explanations and examples; however, a lot of the topics discussed and what is captured in that book will look different or might not work in the way described in version 6.

The other book about Orchestrator is *VMware vRealize Orchestrator Cookbook* that I authored, and was published by Packt Publishing in February 2015. It covers a lot of ground and if you are a total beginner, it's better to start with the book you are currently reading. In that book, we talk a lot more about specific plug-ins and how to use them. It comes with more than 100 workflows that can be directly used. The book was based on version 5.5.x and 6 of vRealize Orchestrator.

There are many more books on JavaScript. Packt has published many JavaScript books. A book for beginners is *Thinking in JavaScript*, which also has the added bonus of being free. Another one is *Learning JavaScript Data Structures and Algorithms*.

In this book, we will introduce you to JavaScript in *Chapter 6*, *Advanced vRO Scripting with JavaScript*.

VMware community

The VMware Orchestrator community is quite big and very active. You will find it under:

`http://communities.vmware.com/community/vmtn/vcenter/orchestrator`

Here you can ask questions, find answers, and in general, obtain a lot of help. I recommend anyone looking for VMware related queries to start here.

Websites

There are quite a lot of websites and blogs that cover Orchestrator; we can't mention everyone, so here are the author's and tech reviewers' choices:

- `http://www.vcoteam.info/`: This is Christophe Decanini and Burke Azbill's website. It contains a lot of topics starting from very basics up to some specialized plug-in handling. If you like to learn new things this is where you will probably find it.
- `http://kaloferov.com/blog/`: This blog contains extremely useful code examples especially for PowerShell.
- `http://www.vcoportal.de/`: Joerg Lew has been working with Orchestrator since 2004 and has collected an amazing amount of useful tips.
- Websites that help you learn JavaScript are plenty. I personally find `http://www.w3schools.com/js/` a really useful page.

Google searches

If you are looking for help on Orchestrator, one of the first things to do is search for the problem in Google. To help you find better results, you can use the following tricks (replace `[problem]` with what you are looking for):

- `(vCO OR vRO OR Orchestrator) [problem]`: Due to the renaming of Orchestrator, a lot of posts and blogs still use the old naming—vCO (vCenter Orchestrator). Using Google and the uppercase `OR`, you can search for the old as well as the new name.
- `site:communities.vmware.com [problem]`: The `site:` tag will make sure that Google only looks inside the VMware communities.

Summary

In this chapter, we had a quick look at all the basic information that you need to know about Orchestrator. We covered what it does, how it does it, and where to download it from. We also looked at how to get additional information and help for Orchestrator related questions.

In the next chapter, we will deploy and configure the vRealize Orchestrator appliance.

2

Deploying and Configuring the Orchestrator Appliance

The Orchestrator Appliance is a Linux-based appliance that comes completely preconfigured and ready to run. An appliance is basically a fully installed and configured VM with an operating system, also known in the industry as shrink-wrapped. In this chapter, you will learn how to deploy the appliance as well as base configure it. Along the way, we will look at the following topics:

- Downloading vRO
- Installing vRO
- Accessing Orchestrator
- Base configure the Orchestrator Appliance
- The Orchestrator configuration tools

Preparations

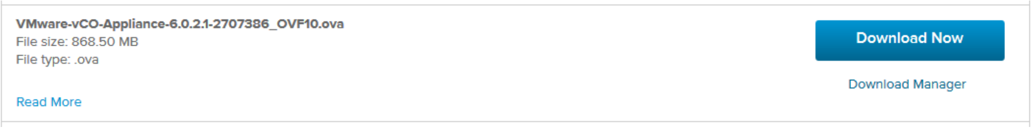
Before we start, we should get our heads around a few things.

Downloading the appliance

First of all, we need to download the appliance in the OVA format from the VMware website:

1. Open a browser and visit `www.vmware.com`.
2. Click on **Downloads**.
3. Select **Download Product** next to the **VMware vSphere** section.

4. Scroll down to `Standard` and then click on **Go to Downloads** next to **VMware vRealize Orchestrator Appliance**
5. Scroll down until you find the OVA download, as shown in the following screenshot:



VMware-vCO-Appliance-6.0.2.1-2707386_OVF10.ova
File size: 868.50 MB
File type: .ova

6. Now, click on **Download Now**.
7. You will be asked for your VMware credentials. Enter them and click on **Log In**.
8. **Agree** and click on **Accept** for the EULA.
9. Download the OVA to a location from where you can access vCenter (or on your desktop if you are using VMware Workstation).

If you don't have a valid VMware license, you can acquire an evaluation license for vSphere. At the time of writing, there is no specific Orchestrator trial available, but Orchestrator can be used with the free vSphere trial version.

To do so, perform the following steps:

1. Visit `www.vmware.com`, the VMware webpage.
2. Browse **Products** | **vSphere**.
3. Select **Try for Free**.
4. If you have a VMware account, log in now. If you don't have one, create one.
5. You should receive an email with your evaluation license key.
6. Now, follow the previous set of instructions.

Appliance's size requirements

Orchestrator Appliance 6.0.2 requires the following virtual resources:

CPU	2 vCPU with at least 2.0 GHz
Memory	3 GB
Disk space	12 GB
Network	1x Network 1x IP (DHCP possible)

Preparing the IP and DNS settings

The next important thing is to have an IP as well as a hostname for the appliance that is registered in the DNS. You can use DHCP, but for best results you should have a dedicated IP and FQDN for Orchestrator.

Make sure that you can resolve the hostname full, short, and reverse. Lets look at the following example: The hostname of a VM is `vRO.mylab.local` and its IP is `192.168.220.128`. As the VM doesn't exist yet, the pings will fail. However, the DNS reply should still be shown. If this works correctly, we can go on. The following table shows the commands that you should run as well as the results that you should see:

Resolve type	Command	Result
Full	`ping vro.mylab.local`	`192.168.220.128`
Short	`ping vro`	`192.168.220.128`
Reverse	`ping -a 192.168.220.128`	`vro.mylab.local`

Open ports

The Orchestrator Appliance requires the following open ports:

Source	Target	TCP port(s)	Description
Web browser	Orchestrator VM	80 8281	The Orchestrator home page
Web browser	Orchestrator VM	8283	The Orchestrator configuration
Orchestrator Client	Orchestrator VM	8286 8287	Orchestrator Client's Java communication
Orchestrator VM	MS SQL	1433	Needed only for MS SQL DB access
Orchestrator VM	Oracle	1521	Needed only for Oracle DB access

Creating users and groups

Orchestrator requires one group with at least one user. The group is used as the Orchestrator Administrator group, and the user should be a member of this group.

You can either create a dedicated group and at least one user for Orchestrator, or use an existing group and users. The group and users can be created either in SSO or in the Active Directory that is attached to your SSO.

If you are using new users and groups in AD, perform the following steps:

1. Add a new global security group to your AD, such as `vROAdmins`.
2. Create a new user in your AD, such as `vROAdmin`, and make it a member of the new group that you created.
3. Using the vSphere Web Client, add the new group to your vCenter with the Administrator role.
4. Log out of the Web Client, try to log in as the new user, and check whether it works.

Database

Orchestrator needs a database. The appliance comes with a preconfigured internal PostgreSQL database that is ready for use. However, for a production environment, the internal database is not good enough as there are no means to backup the DB or run maintenance. A production Orchestrator installation should use an external PostgreSQL, MS SQL, or Oracle database. It is *not* a good idea to use the vCenter database.

VMware recommends that the database should have at least a size of 1 GB. For a test installation, a size of 100 MB is fine.

If you want to create a new MS SQL database, perform the following steps:

1. Open MS SQL.
2. Create a new database and give it a name, such as `vRODB`.
3. Create a new login. You can use either the new AD account that you previously created or a SQL user.
4. Make sure that the DB user has the `sysadmin` role and is a `db_Owner` of the new database.

After installing and configuring Orchestrator, the DB user only needs the read and write permissions. The SysAdmin role is only needed for installation or update purposes.

Deploying the Orchestrator Appliance

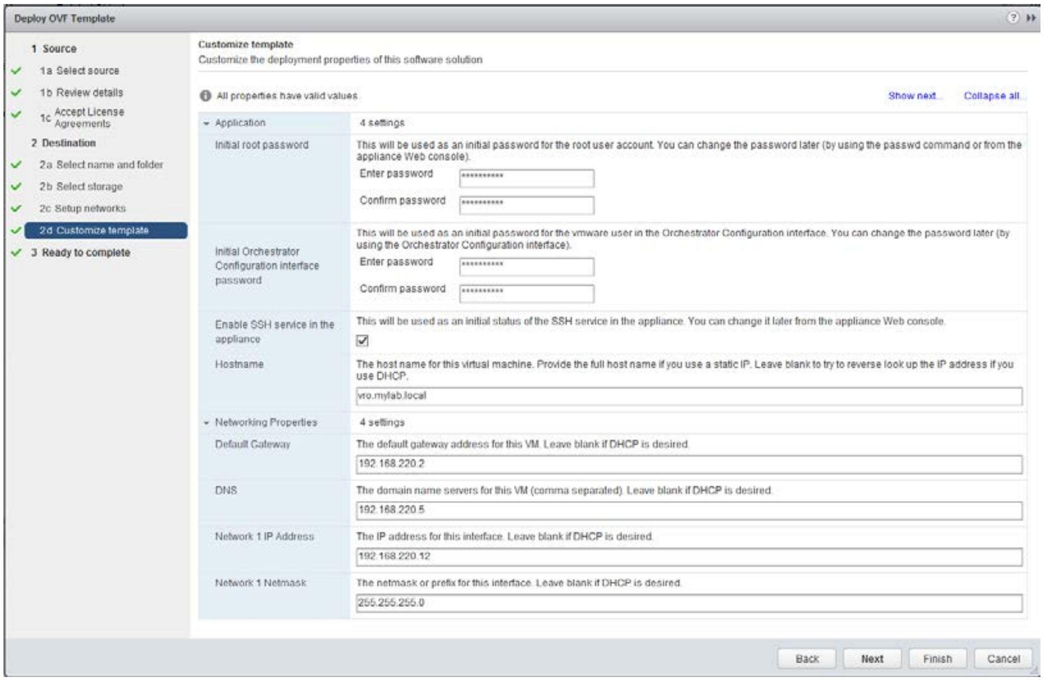
After downloading the Orchestrator Appliance, we need to deploy it.
In the following steps, we will show you how to do this in vCenter and with VMware Workstation.

Deploying the Appliance with vCenter

To make the best use of Orchestrator, its best to deploy it into your vSphere infrastructure. For this, we deploy it with vCenter.

1. Open your vSphere Web Client and log in.
2. Select a host or cluster that should host the Orchestrator Appliance.
3. Right-click the **Host or Cluster** and select **Deploy OVF Template**.
4. The deploy wizard will start and ask you the typical OVF questions:
    - Accept the EULA
    - Choose the VM name and the VM folder where it will be stored
    - Select the storage and network it should connect to. Make sure that you select a static IP
5. The **Customize template** step will now ask you about some more Orchestrator-specific details. You will be asked to provide a new password for the root user. The root user is used to connect to the vRO appliance operating system or the web console.
6. The other password that is needed is for the vRO Configurator interface. We will have a quick look at that in *Chapter 3*, *Integrating Orchestrator with vSphere*.

7. The last piece of information needed is the network information for the new VM. The following screenshot shows an example of the **Customize template** step:



Customize template

Customize the deployment properties of this software solution

All properties have valid values.

Application: 4 settings

Initial root password: This will be used as an initial password for the root user account. You can change the password later (by using the passwd command or from the appliance Web console).

Enter password: ***********

Confirm password: ***********

Initial Orchestrator Configuration interface password: This will be used as an initial password for the vmware user in the Orchestrator Configuration interface. You can change the password later (by using the Orchestrator Configuration interface).

Enter password: ***********

Confirm password: ***********

Enable SSH service in the appliance: This will be used as an initial status of the SSH service in the appliance. You can change it later from the appliance Web console. ☑

Hostname: The host name for this virtual machine. Provide the full host name if you use a static IP. Leave blank to try to reverse look up the IP address if you use DHCP.

vro.mylab.local

Networking Properties: 4 settings

Default Gateway: The default gateway address for this VM. Leave blank if DHCP is desired.

192.168.220.2

DNS: The domain name servers for this VM (comma separated). Leave blank if DHCP is desired.

192.168.220.5

Network 1 IP Address: The IP address for this interface. Leave blank if DHCP is desired.

192.168.220.12

Network 1 Netmask: The netmask or prefix for this interface. Leave blank if DHCP is desired.

255.255.255.0

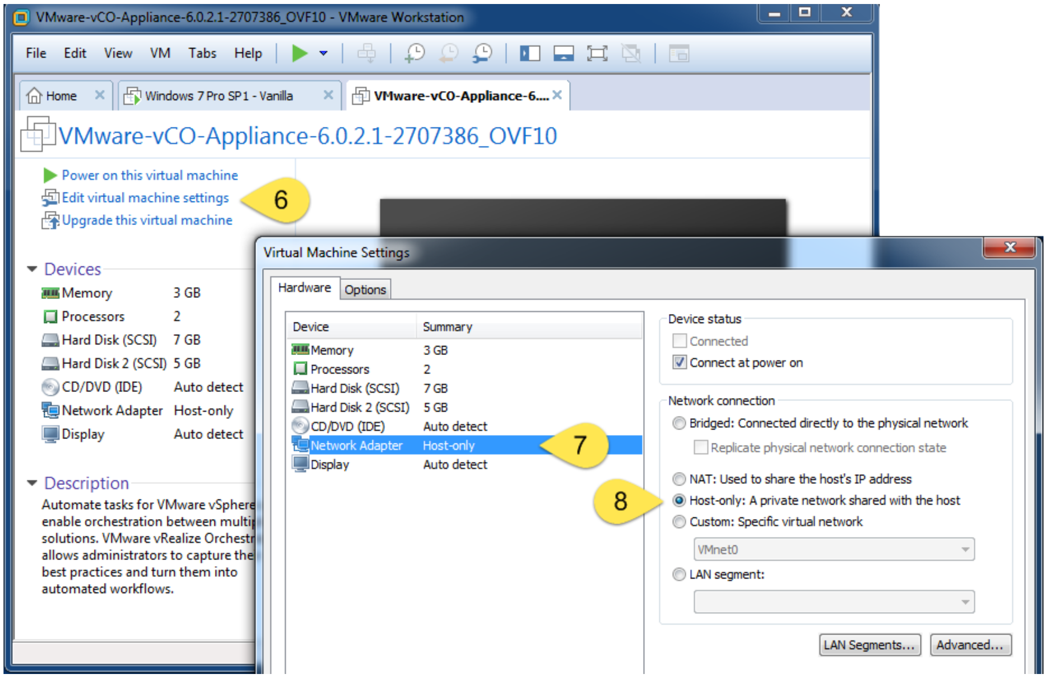
8. The last step summarizes all the settings and lets you power on the VM after creation. Click on **Finish** and wait until the VM is deployed and powered on.

Deploying the appliance into VMware Workstation

For learning how to use Orchestrator, or for testing purposes, you can deploy Orchestrator using VMware Workstation (Fusion for MAC users). The process is pretty simple:

1. Download the Orchestrator Appliance on to your desktop.
2. Double-click on the OVA file.
3. The import wizard now asks you for a name and location of your local file structure for this VM. Chose a location and click on **Import**.
4. **Accept** the EULA.

5. Wait until the import has finished.
6. Click on **Edit virtual machine settings**.
7. Select **Network Adapter**.
8. Chose the correct network (**Bridged**, **NAT**, or **Host only**) for this VM. I typically use **Host Only**.


VMware-vCO-Appliance-6.0.2.1-2707386_OVF10 - VMware Workstation
File
Edit
View
VM
Tabs
Help
Home
Windows 7 Pro SP1 - Vanilla
VMware-vCO-Appliance-6....
6
7
8
Virtual Machine Settings
Hardware
Options
LAN Segments...
Advanced...

VMware-vCO-Appliance-6.0.2.1-2707386_OVF10

Power on this virtual machine
Edit virtual machine settings
Upgrade this virtual machine

Devices

Memory	3 GB
Processors	2
Hard Disk (SCSI)	7 GB
Hard Disk 2 (SCSI)	5 GB
CD/DVD (IDE)	Auto detect
Network Adapter	Host-only
Display	Auto detect

Description

Automate tasks for VMware vSphere enable orchestration between multi solutions. VMware vRealize Orchestr allows administrators to capture the best practices and turn them into automated workflows.

Device	Summary
Memory	3 GB
Processors	2
Hard Disk (SCSI)	7 GB
Hard Disk 2 (SCSI)	5 GB
CD/DVD (IDE)	Auto detect
Network Adapter	Host-only
Display	Auto detect

Device status

□ Connected
☑ Connect at power on

Network connection

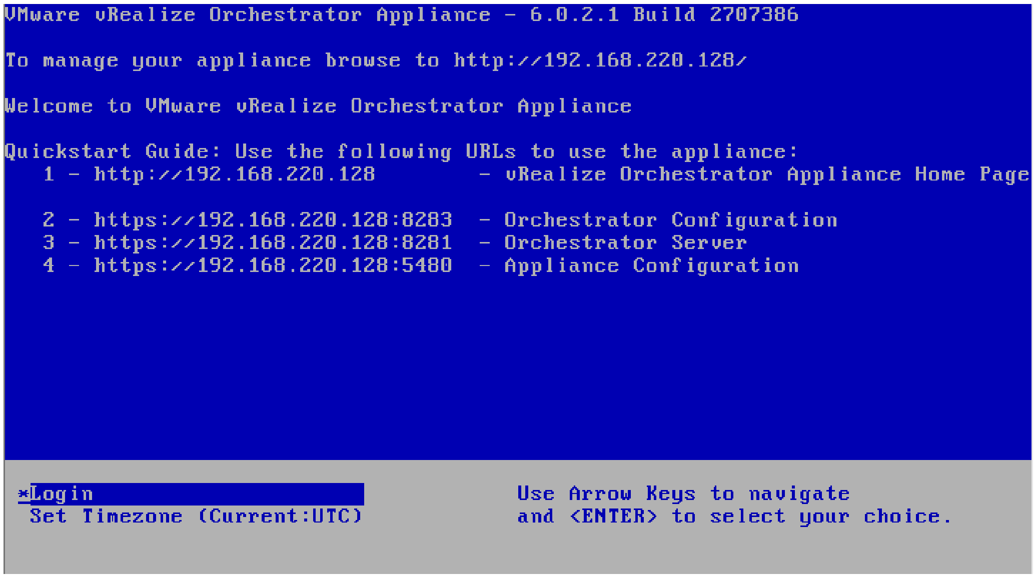
○ Bridged: Connected directly to the physical network
□ Replicate physical network connection state
○ NAT: Used to share the host's IP address
● Host-only: A private network shared with the host
○ Custom: Specific virtual network
VMnet0
○ LAN segment:

9. Click on **OK** to exit the settings.
10. Power on the VM.
11. Watch the boot screen. At some stage, the boot will stop and you will be prompted for the `root` password. Enter a new password and confirm it.
12. After a moment, you will be asked for the password for the Orchestrator Configurator. Enter a new password and confirm it.
13. After this, the boot process should finish, and you should see the Orchestrator Appliance DHCP IP.

If you would like to configure the VM with a fixed IP, access the appliance configuration, as shown on the console screen (see the next section).

After the deployment

If the deployment is successful, the console of the VM should show a screen that looks like the following screenshot:



```
VMware vRealize Orchestrator Appliance - 6.0.2.1 Build 2707386

To manage your appliance browse to http://192.168.220.128/

Welcome to VMware vRealize Orchestrator Appliance

Quickstart Guide: Use the following URLs to use the appliance:
   1 - http://192.168.220.128         - vRealize Orchestrator Appliance Home Page

   2 - https://192.168.220.128:8283   - Orchestrator Configuration
   3 - https://192.168.220.128:8281   - Orchestrator Server
   4 - https://192.168.220.128:5480   - Appliance Configuration

 *Login                                        Use Arrow Keys to navigate
  Set Timezone (Current:UTC)                   and <ENTER> to select your choice.
```

You can now access the Orchestrator Appliance, as shown in the next section.

Accessing Orchestrator

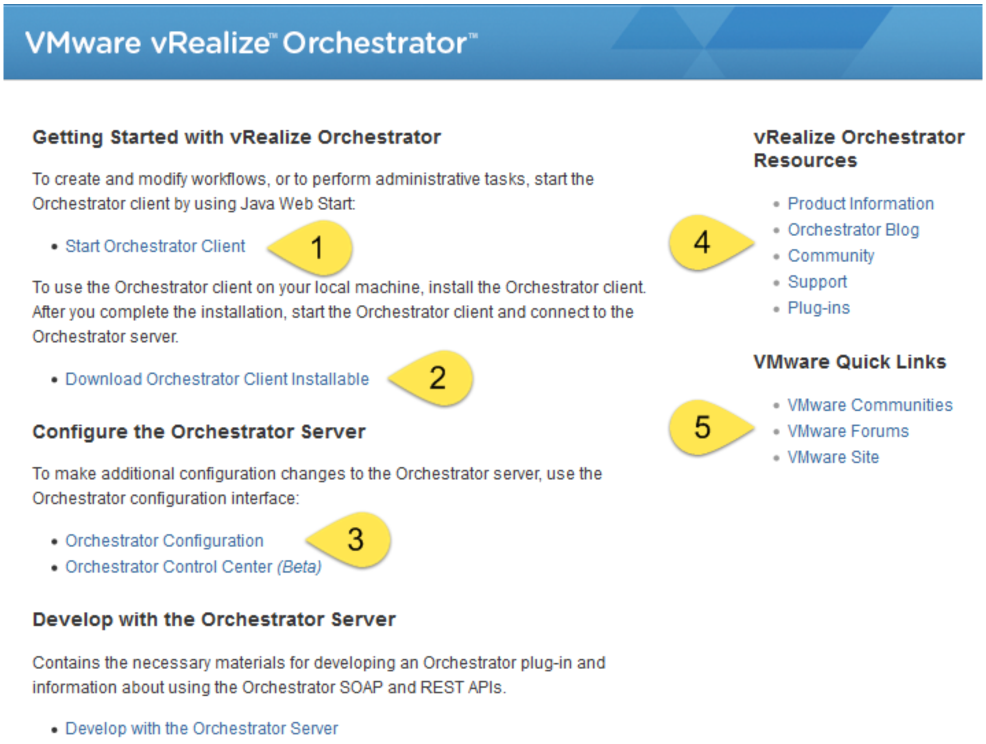
Orchestrator has its own little webserver that can be accessed by any web browser.

Accessing the Orchestrator home page

We will now access the Orchestrator home page:

1. Open a web browser such as Mozilla Firefox, IE, or Google Chrome.
2. Enter the IP or FQDN of the Orchestrator Appliance.

3. The Orchestrator home page will open. It looks like the following screenshot:



VMware vRealize™ Orchestrator™

Getting Started with vRealize Orchestrator

To create and modify workflows, or to perform administrative tasks, start the Orchestrator client by using Java Web Start:

- Start Orchestrator Client

To use the Orchestrator client on your local machine, install the Orchestrator client. After you complete the installation, start the Orchestrator client and connect to the Orchestrator server.

- Download Orchestrator Client Installable

Configure the Orchestrator Server

To make additional configuration changes to the Orchestrator server, use the Orchestrator configuration interface:

- Orchestrator Configuration
- Orchestrator Control Center *(Beta)*

Develop with the Orchestrator Server

Contains the necessary materials for developing an Orchestrator plug-in and information about using the Orchestrator SOAP and REST APIs.

- Develop with the Orchestrator Server

vRealize Orchestrator Resources

- Product Information
- Orchestrator Blog
- Community
- Support
- Plug-ins

VMware Quick Links

- VMware Communities
- VMware Forums
- VMware Site

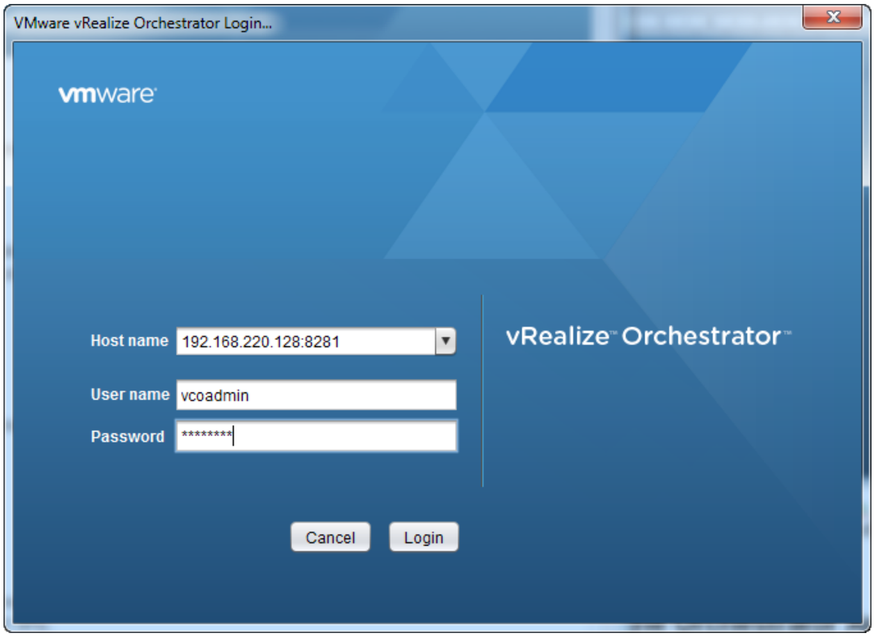
The home page contains some very useful links, as shown in the above screenshot. Here is an explanation of some of them:

Number	Description
1	Click here to start the Orchestrator Java Client. You can also access the Client directly by visiting `https://[IP or FQDN]:8281/vco/client/client.jnlp`.
2	Click here to download and install the Orchestrator Java Client locally.
3	Click here to access the Orchestrator Configurator, which is scheduled to disappear soon. The way forward will be Orchestrator Control Center.
4	This is a selection of links that can be used to find helpful information and download plugins.
5	These are some additional links to VMware sites.

Starting the Orchestrator Client

Let's open the Orchestrator Client. We will use an internal user to log in until we have hooked up Orchestrator to SSO. For the Orchestrator Client to work, you need to run at least Java 7.

1. From the Orchestrator home page, click on **Start Orchestrator Client**.
2. Your Java environment will start. You may be required to acknowledge that you really want to start this application.
3. You will now be greeted with the login screen to Orchestrator:



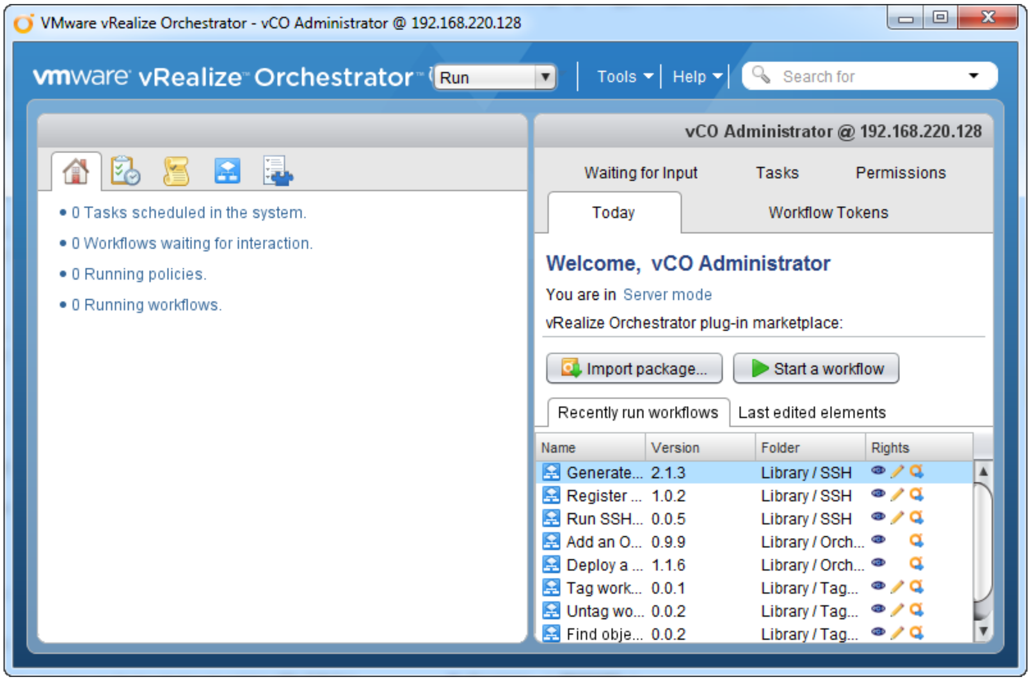
Host name 192.168.220.128:8281

User name vcoadmin

Password ********

4. Enter `vcoadmin` as the username and `vcoadmin` as the password. This is a preconfigured user that allows you to log in and use Orchestrator directly. Click on **Login**.

5. Now, the Orchestrator Client will load. After a moment, you will see something that looks like the following screenshot:



VMware vRealize Orchestrator - vCO Administrator @ 192.168.220.128

- 0 Tasks scheduled in the system.
- 0 Workflows waiting for interaction.
- 0 Running policies.
- 0 Running workflows.

Welcome, vCO Administrator

You are in Server mode

vRealize Orchestrator plug-in marketplace:

Name	Version	Folder	Rights
Generate...	2.1.3	Library / SSH	
Register ...	1.0.2	Library / SSH	
Run SSH...	0.0.5	Library / SSH	
Add an O...	0.9.9	Library / Orch...	
Deploy a ...	1.1.6	Library / Orch...	
Tag work...	0.0.1	Library / Tag...	
Untag wo...	0.0.2	Library / Tag...	
Find obje...	0.0.2	Library / Tag...	

You are now logged in to the Orchestrator Client.

Base configure the Orchestrator Appliance

Now that Orchestrator is up and running, we can start configuring it. We will configure Orchestrator to use SSO for authentication as well as an external database.

What is preconfigured?

Wait! We did say that Orchestrator is preconfigured. So, what is already actually there ready to be used?

The Orchestrator Appliance comes with an embedded PostgreSQL database as well as a configured LDAP directory service. The LDAP contains the `vcoadmin` user that we just used to log in. In addition to this, Orchestrator has a self-signed packaged certificate and a 90-day trial license. Both of these items are stored in the embedded PostgreSQL database, which also means that these items have to be recreated when you use a new and empty external database.

What is not configured is any connection to the vSphere environment. We will do this in the next chapter. In this chapter, we will have a look at how one can replace the preconfigured database and authentication with an external database and SSO authentication.

> If you want to use the internal DB or LDAP, just skip the following steps.

How to run a workflow

We will use the Orchestrator workflows to configure Orchestrator. In *Chapter 4, Working with Workflows*, we will take a closer look at how one can run workflows and deal with errors. If you experience any problems with the following instructions, skip forward to *Chapter 4, Working with Workflows*, and have a look at how to deal with errors in workflows.

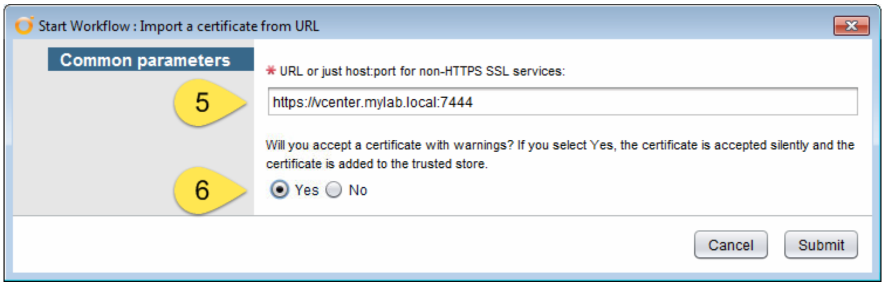
Configuring Orchestrator with SSO

The first thing that we should do is configure Orchestrator to use SSO. You don't necessarily need do this. You can operate Orchestrator by using its internal LDAP authentication. However, if you want to use Orchestrator in production, the best option is SSO authentication. If you prefer to keep the local authentication with the `vcoadmin` users, then just skip this step.

First, let's add the SSL certificate of the SSO component to Orchestrator, as follows:

1. Open the Orchestrator Client.
2. Click on Workflows, .

3. Drill down to **Library** | **Configuration** | **SSL Trust Manager**.
4. Right-click on **Import a certificate from URL** and select **Start workflow**.
5. Enter `https://` and then the FQDN of your vCenter Server (if you are using the embedded SSO/PSC) or your Platform Services Controller (PSC) followed by `:7444`.
6. Then, select **Yes** in order to accept a self-signed SSL certificate:



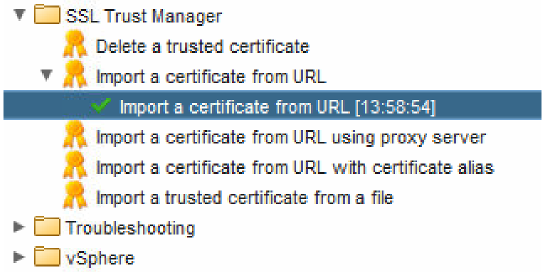
* URL or just host:port for non-HTTPS SSL services:

https://vcenter.mylab.local:7444

Will you accept a certificate with warnings? If you select Yes, the certificate is accepted silently and the certificate is added to the trusted store.

● Yes ○ No

7. Click on **Submit** and wait until the workflow has successfully finished.
   If the workflow execution shows a green tick icon as shown in the following screenshot, the workflow was successful:

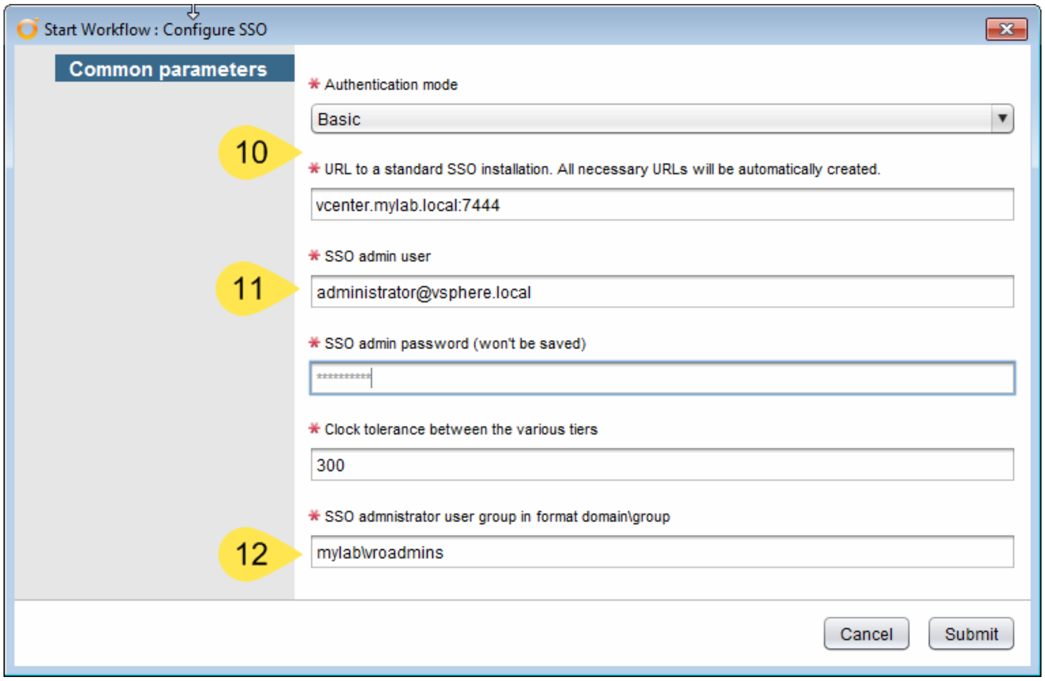


```
▼ SSL Trust Manager
    Delete a trusted certificate
  ▼ Import a certificate from URL
      ✓ Import a certificate from URL [13:58:54]
    Import a certificate from URL using proxy server
    Import a certificate from URL with certificate alias
    Import a trusted certificate from a file
▶ Troubleshooting
▶ vSphere
```

Now, we add Orchestrator to SSO.

8. Drill down to **Library** | **Configuration** | **Authentication** | **SSO**.
9. Right-click on **Configure SSO** and select **Start workflow**.
10. Select **Basic** and enter the FQDN of your vCenter, followed by `:7444`.
11. Enter `administrator@vsphere.local` as well as the corresponding password.

12. The SSO administrator user group is the AD group that you created in the `[Short domain name]\[usergroup]` form. Please note that this entry is case-sensitive.



* Authentication mode: Basic

* URL to a standard SSO installation. All necessary URLs will be automatically created.: vcenter.mylab.local:7444

* SSO admin user: administrator@vsphere.local

* SSO admin password (won't be saved): **********

* Clock tolerance between the various tiers: 300

* SSO admnistrator user group in format domain\group: mylab\vroadmins

13. Click on **Submit** and wait until the workflow has finished successfully.

Please note:

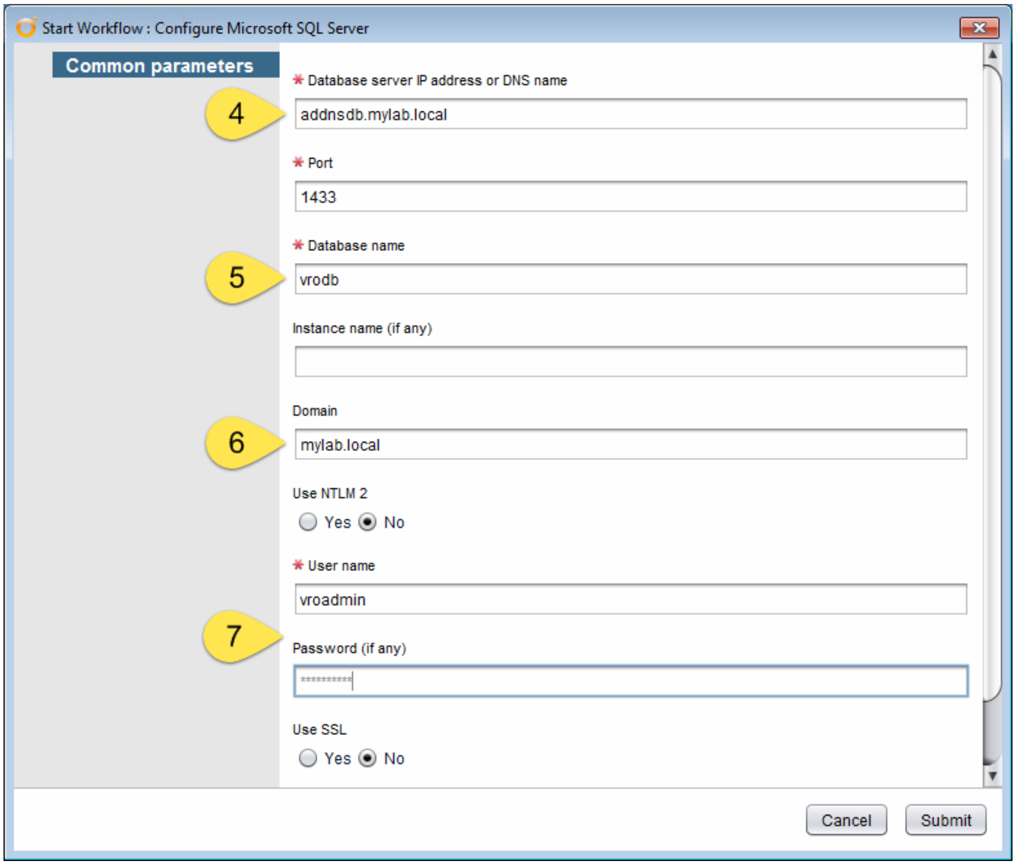
After you have enabled SSO and restarted or rebooted the Orchestrator service/server, the vcoadmin login won't work any more. Only a user who is a member of the Orchestrator admin group that you defined in step 11 will be able to log in to Orchestrator.

Configuring Orchestrator with an external database

The internal PostgreSQL database can be used. However, as already mentioned, an external database is better. We will now connect a MS SQL database to Orchestrator.

If you would like to connect an external PostgreSQL or Oracle database to Orchestrator, the same process is used.

1. Open the Orchestrator Client.
2. Drill down to **Library** | **Configuration** | **Database**.
3. Right-click on **Configure Microsoft SQL Server** and select **Start workflow**.
4. Enter the FQDN of your database server.
5. Enter the name of your database.
6. Enter the name of your windows domain if you are a Windows user. If you are an MS SQL user, leave this field empty.
7. Enter the username and the corresponding password:



* Database server IP address or DNS name: addnsdb.mylab.local

* Port: 1433

* Database name: vrodb

Instance name (if any):

Domain: mylab.local

Use NTLM 2: ○ Yes ● No

* User name: vroadmin

Password (if any): **********

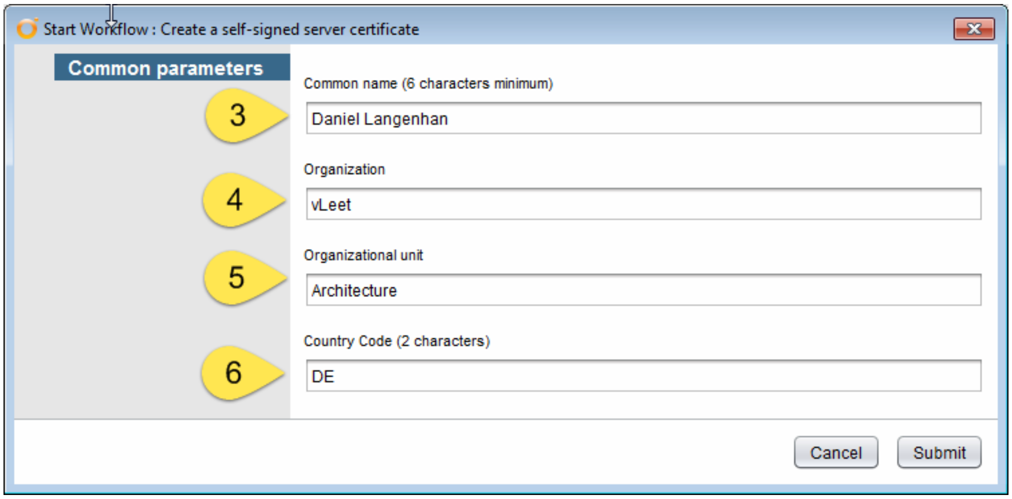
Use SSL: ○ Yes ● No

8. Click on **Submit** and wait until the workflow has finished successfully.

Creating a Package Signing Certificate

The package certificate that we will now create will make sure that all the workflows that you create are stored with your name or the company's name. Later, when you export and import workflows (see *Chapter 9*, *Packing It All Up*), you will see that the package certificate will always stay with them.

1. Navigate to **Library** | **Configuration** | **Package Signing Certificate**.
2. Right-click on **Create a Self-signed server certificate** and select **Start workflow**.
3. In **Common name**, enter your name. You can also use the FQDN of Orchestrator or the name of your business. The purpose is to create a certificate that identifies the origin of the exported items.
4. In **Organization**, enter the name of the business that you are working for.
5. In **Organization Unit**, enter the name of the unit that you are working with.
6. Last but not least, put in the two-letter code for your country:



Common name (6 characters minimum): Daniel Langenhan

Organization: vLeet

Organizational unit: Architecture

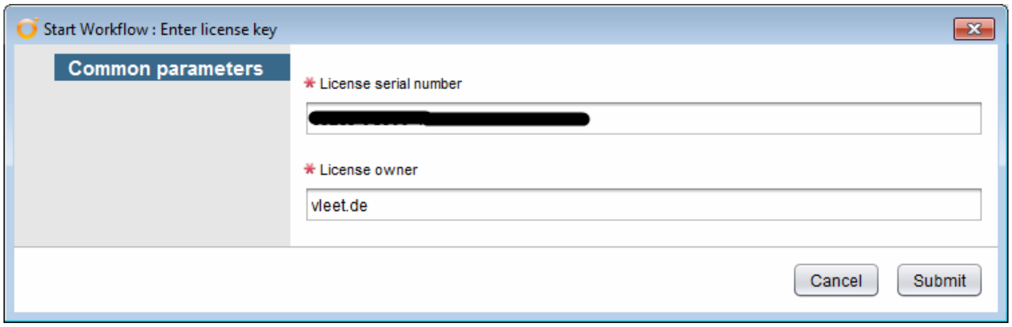
Country Code (2 characters): DE

7. Click on **Submit** and wait until the workflow has finished successfully.

Entering a license

Now that everything is configured, we will need to license Orchestrator. This is done by using the vCenter license. Also, there is a workflow that lets you use the vCenter license directly. However, there are some differences between vSphere 5.x and vSphere 6. So, for the sake of simplicity, we will just enter the 25-letter vCenter license key. You can skip this step if you would like to use the 90-day trial license that Orchestrator is automatically configured with.

1. Navigate to **Library** | **Configuration** | **License**.
2. Right-click on **Enter license key** and select **Start workflow**.
3. Enter your vCenter license key and the name that you would like this license to be associated with:



Start Workflow : Enter license key

Common parameters

* License serial number: [redacted]

* License owner: vleet.de

4. Click on **Submit** and wait until the workflow has finished successfully.

Last steps

It might be a bit weird to run a troubleshooting workflow now, but due to an easy and fast configuration, we are going to clean up a few things. Please make sure that these steps are carried out before we can go on and enjoy Orchestrator.

1. Navigate to **Library** | **Configuration** | **Troubleshooting**.
2. Right-click on **Reinstall the plug-ins when the sever starts** and select **Start workflow**.
3. If there is nothing to enter, just wait until the workflow has finished successfully.
4. Now, close the Orchestrator Client and go to the vSphere Web Client.

5. Restart the guest system of the Orchestrator VM.
6. Wait until the VM has rebooted. After this, you should be able to log in as the AD user that you created in one of the following forms:

Form	Example
[Short AD domain name]\[username]	`mylab\vroadmin`
[username]@[[FQDN AD domain name]	`vroadmin@mylab.local`

The Orchestrator configuration tools

We configured Orchestrator by using the workflow. However, there are two more methods to do the same.

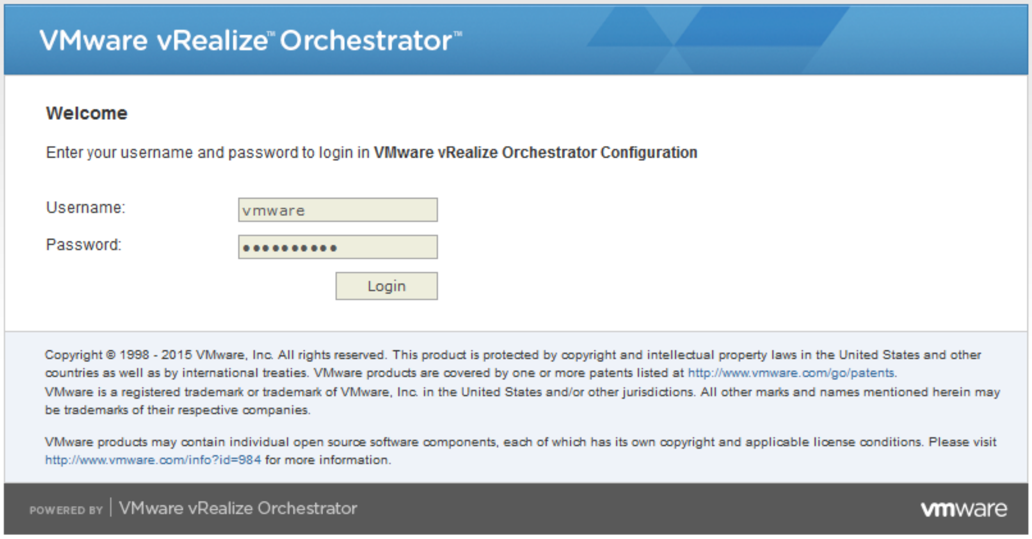
Orchestrator Configurator

The Orchestrator Configurator is considered the "old" way of configuring Orchestrator. It has been around since the first iteration in vSphere 4.1. It is a useful tool that can be used to configure and troubleshoot Orchestrator. The Configurator is deprecated and will not be available anymore in the next major release of Orchestrator, as it will be replaced by workflow configuration and the Control Center. However, both the workflow and Control Center methods don't do everything yet (at the time of writing) that the configurator can do. We will encounter the Configurator in the next chapter, where we will use it to install additional plugins. You can (and should) look into it. Have a look at how the items that we have configured with the workflows are reflected in the Configurator.

To access the Orchestrator Configurator, perform the following steps:

1. Open a browser (IE9 has issues here) and type in the FQDN or IP of Orchestrator.
2. The Orchestrator home page opens. Select **Orchestrator Configuration**.

3. A second page opens up and asks you for your credentials. Enter `vmware` as the username, and for the password, enter the password that you defined for the Configurator while deploying the Orchestrator Appliance:



VMware vRealize™ Orchestrator™

Welcome

Enter your username and password to login in **VMware vRealize Orchestrator Configuration**

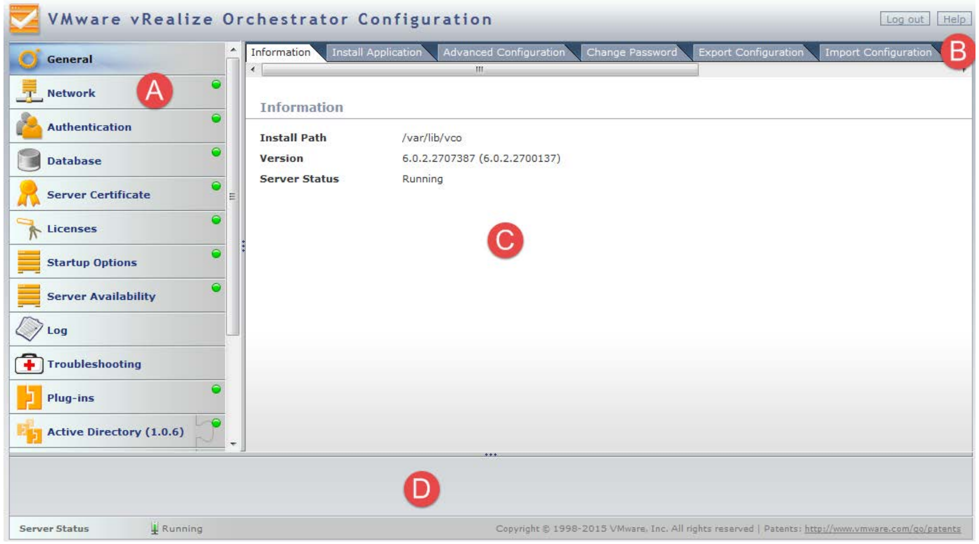
Username: vmware

Password: ●●●●●●●●●●

Copyright © 1998 - 2015 VMware, Inc. All rights reserved. This product is protected by copyright and intellectual property laws in the United States and other countries as well as by international treaties. VMware products are covered by one or more patents listed at http://www.vmware.com/go/patents.
VMware is a registered trademark or trademark of VMware, Inc. in the United States and/or other jurisdictions. All other marks and names mentioned herein may be trademarks of their respective companies.

VMware products may contain individual open source software components, each of which has its own copyright and applicable license conditions. Please visit http://www.vmware.com/info?id=984 for more information.

After logging in, you will be presented with the following window:



Information

Install Path /var/lib/vco

Version 6.0.2.2707387 (6.0.2.2700137)

Server Status Running

Server Status Running

Copyright © 1998-2015 VMware, Inc. All rights reserved | Patents: http://www.vmware.com/go/patents

On the left-hand side, you have various tabs (area **A**) that let you choose what you would like to configure. Note the green ball in each tab. These green balls show that the item is configured correctly. If a red triangle is shown, the item is not configured correctly. On the other hand, a blue ball means that the Orchestrator Configurator is still checking whether the configuration is okay.

For each tab, there is a menu (area **B**). From here, you can select from some specific options to configure for the selected item.

In area **C**, the configuration item that can be configured will be displayed.

At the bottom, you will find a status and log bar. The status bar is the lowest section. It shows the status of your Orchestrator sever. The log bar (area **D**) is reserved for responses from your configuration. Errors will be displayed here.

Go through the items. As long as you don't click on **Apply**, you won't affect anything.

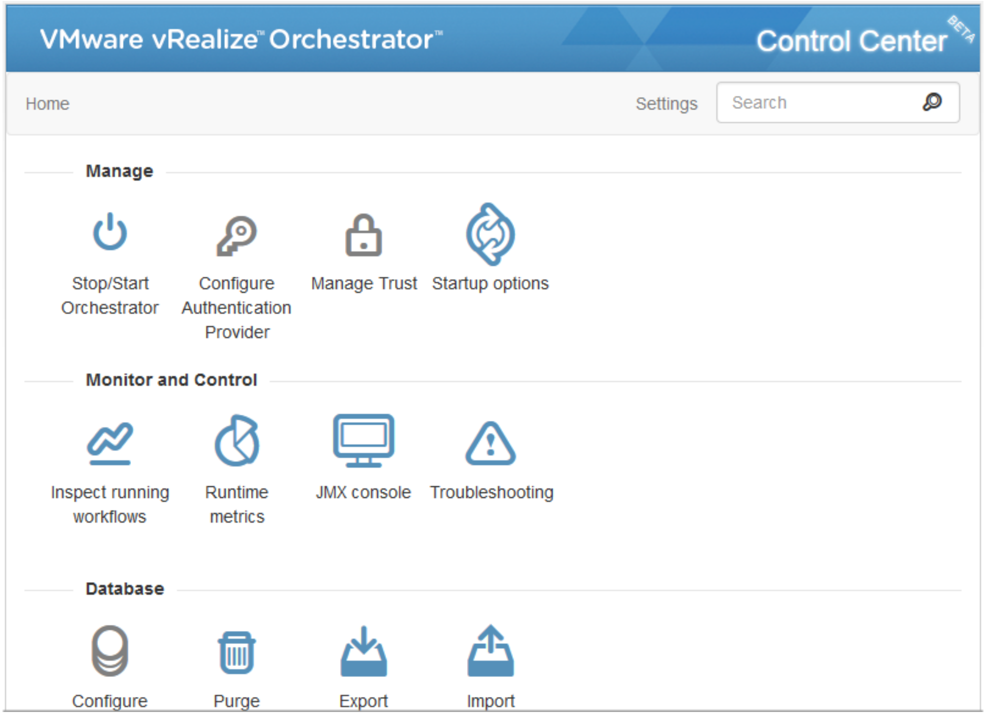
Orchestrator Control Center (Beta)

The Control Center is a new feature of vRO 6 and is currently in beta, which means that it's not fully functional yet. Many icons will still appear grey and inactive. This means that their functionality is planned but has not been implemented yet. You haven't done anything wrong; they just don't work yet. Have a look around.

To access the Control Center, perform the following steps:

1. Open a browser and type in the FQDN or IP of Orchestrator.
2. The Orchestrator home page will open. Select **Orchestrator Control Center (Beta)**.
3. A second page will open and ask you to log in. Enter `root` as the username. For the password, enter the password of the root user, as specified during the installation.

4. The Control Center will open up. It looks like the following screenshot:



One thing that may be of interest to you is the ability to start and stop the Orchestrator service. When the Control Center is finished, we won't need to perform the entire configuration that we did using the workflows any more. Check out the possibilities.

Summary

This chapter guided you through the process, from downloading the Orchestrator Appliance to configuring it with an external database and SSO authentication. This is the quick and easy way of configuring Orchestrator. In *VMware vRealize Orchestrator Cookbook*, we discuss several options in detail, as well as learn some very interesting tuning tips for the Orchestrator Appliance.

In the next chapter, we will go ahead and configure the connection to vSphere and the other VMware components.

3

Integrating Orchestrator with vSphere

In the last chapter, we installed and configured Orchestrator with SSO authentication and an external database. Now, we will integrate Orchestrator into vCenter and ultimately with other VMware or third-party tools. This will place Orchestrator in a central position for all your automation purposes and needs.

You can add a vCenter 5.5 or 5.1 to Orchestrator 6 without any problems. However, please be aware that, if you construct workflows in Orchestrator 6 intended for a vCenter 5.x, you might experience problems. A typical problem is SSO, which underwent some major changes in 5.1, 5.5, and 6.0, and it will make a vSphere Web Client integration hard, if not impossible. If you need to stay compatible, you should use Orchestrator 5.5.x with vSphere 5.5.x.

In this chapter, we will look at the following topics:

- Integrating Orchestrator with vCenter
- Installing additional plug-ins
- Configuring the vCenter plug-in

Integrating Orchestrator with vCenter

Your first task is to link Orchestrator to vCenter. This will allow you to not only automate vCenter tasks with Orchestrator, but also run, schedule, and monitor workflows by using the Web Client.

Open TCP ports

For Orchestrator to access vCenter, we need these ports open between Orchestrator and vCenter: TCP 443 and 7444.

If you would like to integrate Orchestrator into the vSphere Web Client, you need to have port TCP 443 open between vCenter and Orchestrator.

Allowing user access

To get started, we need to make sure that we have access to vCenter. Therefore, we will now assign the new AD group (vroAdmins) that we created in the last chapter to the vCenter Administrators role. You can use your existing Administrator user, such as Administrator@vsphere.local, for this. However, for tracking and logging purposes, it's recommended that you use a dedicated user.

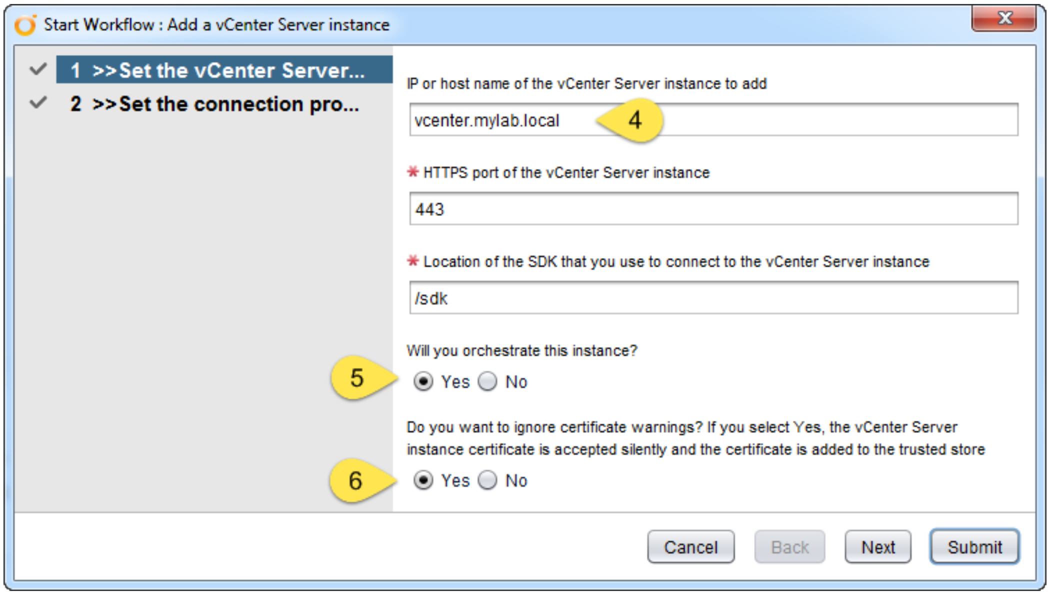
1. Open the vSphere Web Client and log in with the vCenter administrator rights.
2. Add the group (vroAdmin) that you created in the last chapter to your vCenter with the Administrator role.
3. Log out of the vSphere Web Client.

Adding a vCenter to Orchestrator

Now, we will add vCenter to Orchestrator, as follows:

1. Open the Orchestrator Client and log in.
2. Browse to **Library** | **vCenter** | **Configuration**.
3. Right-click on the workflow, click on **Add a vCenter Server instance**, and select **Start workflow**.
4. Enter the FQDN of your vCenter.

5. Select that you would like to orchestrate this vCenter instance. This means that Orchestrator will save all the vCenter login data.



Start Workflow : Add a vCenter Server instance

✓ 1 >>Set the vCenter Server...
✓ 2 >>Set the connection pro...

IP or host name of the vCenter Server instance to add

vcenter.mylab.local

* HTTPS port of the vCenter Server instance

443

* Location of the SDK that you use to connect to the vCenter Server instance

/sdk

Will you orchestrate this instance?

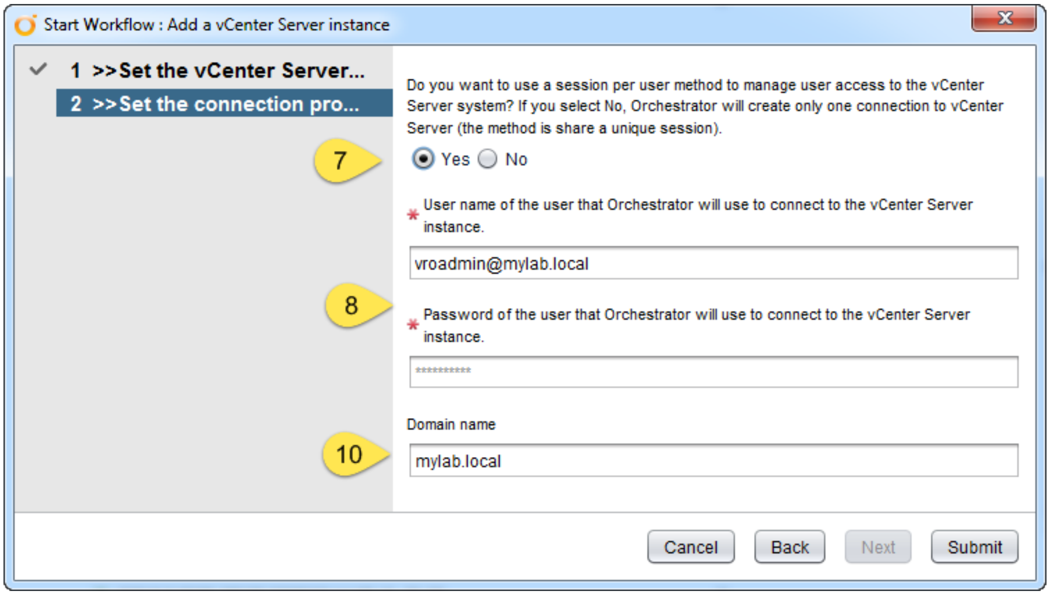
● Yes ○ No

Do you want to ignore certificate warnings? If you select Yes, the vCenter Server instance certificate is accepted silently and the certificate is added to the trusted store

● Yes ○ No

6. If you are using a self-signed certificate in your vSphere environment, you should make sure that you ignore the certificate warnings. Click on **Next**.
7. The question about a session per user is quite important. If you select **Yes**, Orchestrator will use the logged-in user's credentials to execute tasks on vCenter. If you select **No**, all the vCenter tasks will be started using the credentials entered in step 8.
8. Enter a vCenter Administrator user so that Orchestrator can add access to the vCenter API.

9. If you chose **Yes** in step 7, this step defines the AD domain that users need to be a part of in order to execute workflows on vCenter.



Start Workflow : Add a vCenter Server instance

1 >>Set the vCenter Server...

2 >>Set the connection pro...

Do you want to use a session per user method to manage user access to the vCenter Server system? If you select No, Orchestrator will create only one connection to vCenter Server (the method is share a unique session).

● Yes ○ No

* User name of the user that Orchestrator will use to connect to the vCenter Server instance.

vroadmin@mylab.local

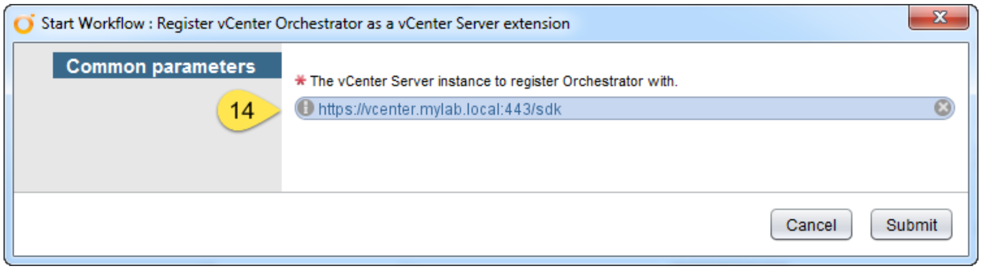
* Password of the user that Orchestrator will use to connect to the vCenter Server instance.

****

Domain name

mylab.local

10. Click on **Submit** and wait until the workflow has successfully completed.
11. The next steps are only needed if you wish to run the Orchestrator workflows from the vSphere Web Client.
12. Browse to **Library** | **vCenter** | **Configuration** and start the workflow **Register vCenter Orchestrator as a vCenter Server extension**.
13. Select the vCenter instance that you added and click on **Submit**.



Start Workflow : Register vCenter Orchestrator as a vCenter Server extension

Common parameters

* The vCenter Server instance to register Orchestrator with.

https://vcenter.mylab.local:443/sdk

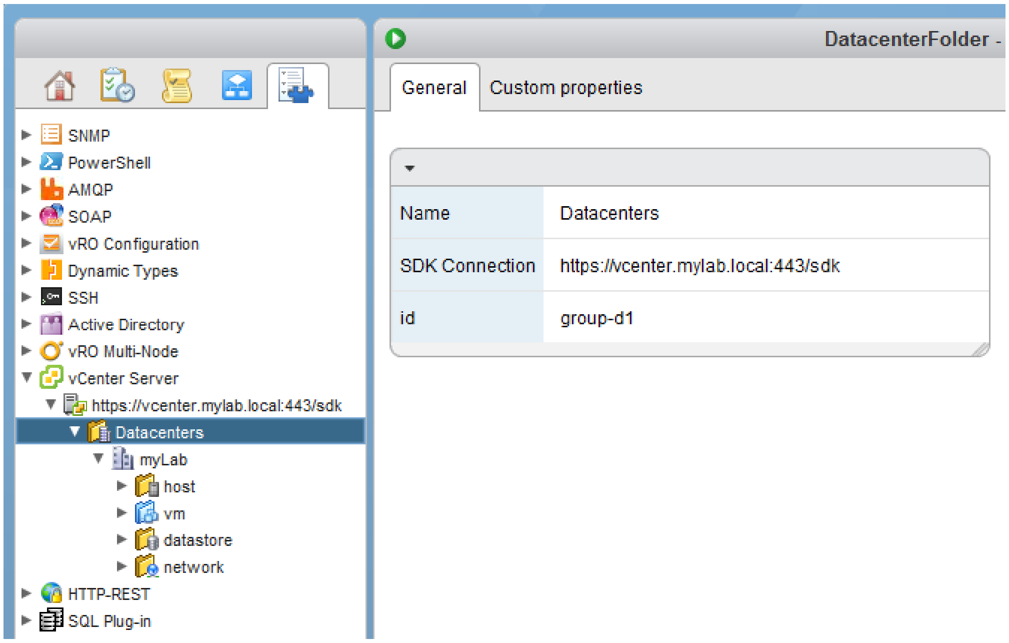
14. Wait until the workflow has completed successfully.

So, that's it. We added a vCenter to Orchestrator. You can add multiple vCenters to one Orchestrator (don't forget to add their SSL certificate first). This allows you to have a central Orchestrator instance that controls vCenters in several environments, such as development and production.

The Orchestrator inventory

Now that vCenter has been added to Orchestrator, we can see its contents in Orchestrator. Let's explore this:

1. In the Orchestrator Client, click on the Inventory icon .
2. Find the **vCenter Server** section and expand it. It might require a bit for it to open, as Orchestrator is now querying vCenter for its inventory.
3. Dig deeper and explore what you can see.



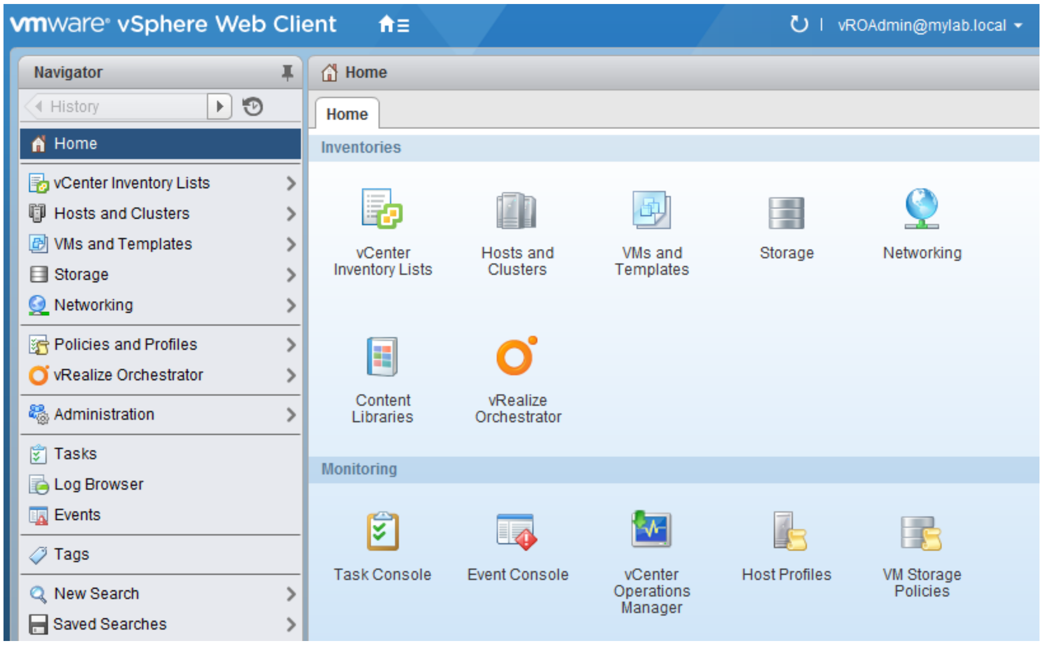
Name	Datacenters
SDK Connection	https://vcenter.mylab.local:443/sdk
id	group-d1

If you see the vCenter Server inventory, you know that vCenter has been successfully added to Orchestrator. We will run and work with vCenter workflows in the next chapter.

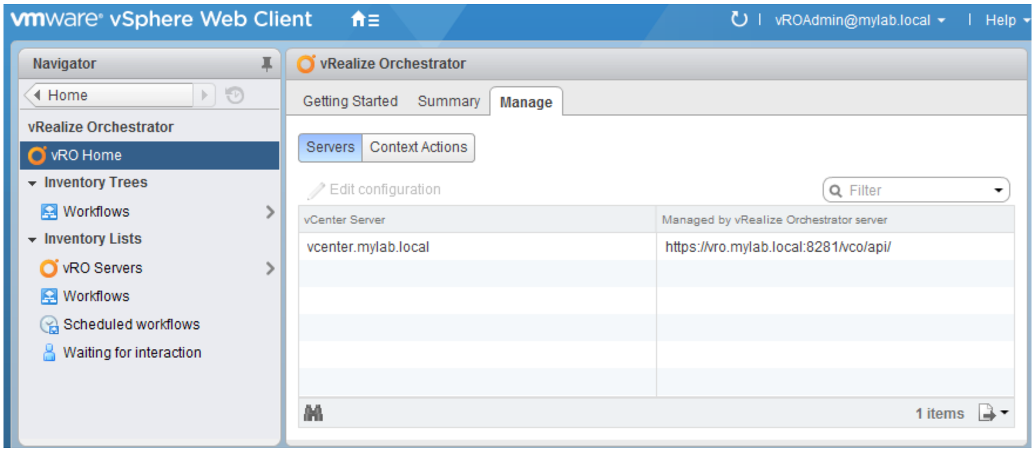
vSphere Web Client

After seeing that vCenter appears in Orchestrator, let's move on to see how we can make Orchestrator work in the vSphere Web Client:

1. Open the vSphere Web Client with a member of the Orchestrator Administrator group, such as vroAdmin.
2. Select the **vRealize Orchestrator** entry or the icon from the menu, as seen in the following screenshot:



3. Click on **vRO Home**. This will show you the vCenter instance as well as the Orchestrator that is registered with it. Remember that you can have multiple vCenters in the Web Client.



vCenter Server	Managed by vRealize Orchestrator server
vcenter.mylab.local	https://vro.mylab.local:8281/vco/api/

When you see the Orchestrator instance in the vSphere Web Client, you know that Orchestrator was successfully registered as an extension to vCenter. This now allows you to use the vSphere Web Client to run the Orchestrator workflow directly.

In the next chapter, we will use the vSphere Web Client to run and schedule a workflow.

Installing additional plugins

As discussed in the first chapter with regard to the Orchestrator architecture, Orchestrator uses a plugin infrastructure to expend its possibility. We also said in the last chapter that plugins have to be installed at the moment (at the time of writing this book) by using the vRealize Orchestrator Configuration, as the Control Center isn't ready to perform this operation yet.

Downloading the VMware plug-in

There are many plug-ins for Orchestrator. Check *Chapter 1*, *Architectural Overview*, for an overview and links to the web pages where you can get them. In this chapter, I will install the VMware NSX (also called vCNS) plug-in. The method used is the same for all the other plug-ins that exist. Feel free to download and install a different plug-in.

1. Open a web browser and visit `www.vmware.com`.
2. Click on **Downloads**.
3. Search for **VMware vSphere** and click on **Drivers & Tools**.
4. Expand **VMware vRealize Orchestrator Plugin**.
5. Select **Go to downloads** next to the plug-in that you would like to download.
6. Enter your VMware credentials and proceed to download.
7. The file that you are downloading has the `.vmoapp` ending.

If you would like to download any other plug-in, have a look at the *More about plug-ins* section of the first chapter.

Installing the plug-in

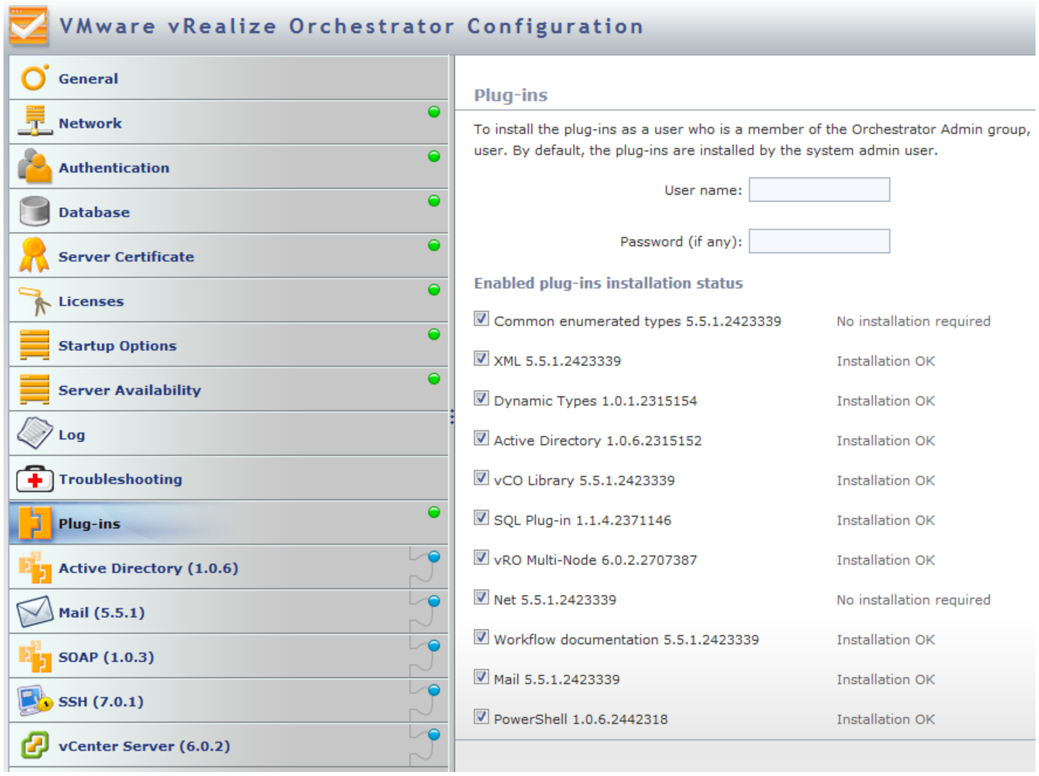
Now that we have downloaded the plug-in, we should install it in Orchestrator. As already mentioned, we will have to use the Orchestrator Configuration. If you are using a newer version (I am using version 6.0.2), check out the Control Center to check whether the plug-in install works.

Opening the Orchestrator configuration

We will now open the Orchestrator configuration and check for all the installed plug-ins:

1. Open a web browser and open the Orchestrator home page by entering the FQDN or IP of your Orchestrator, as shown in *Chapter 2*, *Deploying and Configuring the Orchestrator Appliance*.
2. From the Orchestrator home page, select the link to the **Orchestrator Configuration**.

3. A second web page opens up and asks you to log in. Use `vmware` as the username, and use the password that you assigned during the deployment.
4. Click on **Plug-ins**. This page shows you all the plug-ins that are already installed. Have a look:



Plug-ins

To install the plug-ins as a user who is a member of the Orchestrator Admin group, user. By default, the plug-ins are installed by the system admin user.

User name:

Password (if any):

Enabled plug-ins installation status

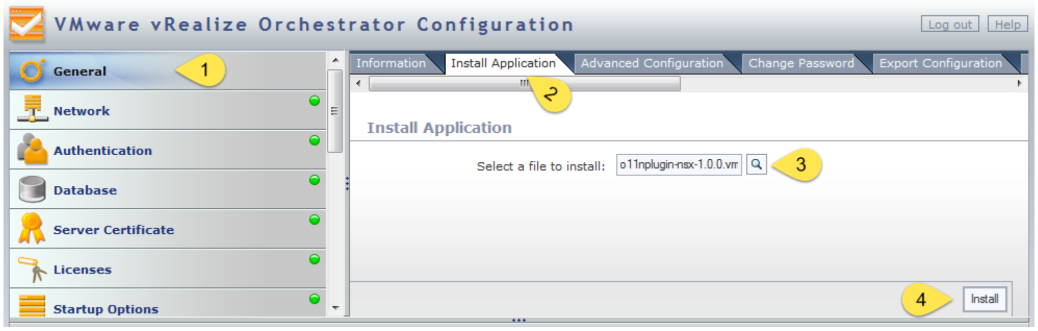
☑ Common enumerated types 5.5.1.2423339	No installation required
☑ XML 5.5.1.2423339	Installation OK
☑ Dynamic Types 1.0.1.2315154	Installation OK
☑ Active Directory 1.0.6.2315152	Installation OK
☑ vCO Library 5.5.1.2423339	Installation OK
☑ SQL Plug-in 1.1.4.2371146	Installation OK
☑ vRO Multi-Node 6.0.2.2707387	Installation OK
☑ Net 5.5.1.2423339	No installation required
☑ Workflow documentation 5.5.1.2423339	Installation OK
☑ Mail 5.5.1.2423339	Installation OK
☑ PowerShell 1.0.6.2442318	Installation OK

If you uncheck the tick box in front of a plug-in, you can disable it. There isn't really a method as such to uninstall a plugin. If you install a newer version of a plugin, the old version will be overwritten. However, you cannot replace a newer version with an older version so easily.

Installing a new plug-in

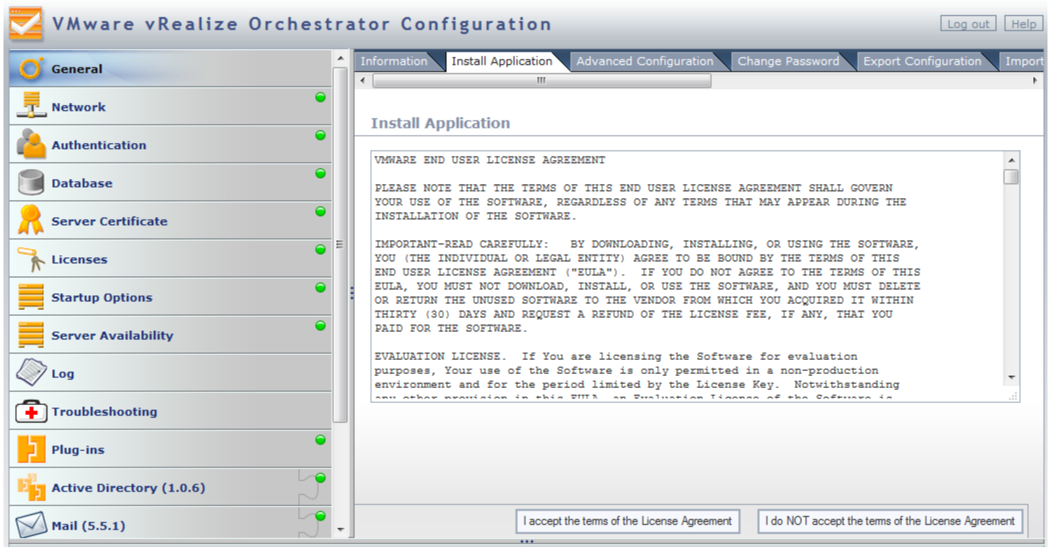
Let's install a new plug-in in Orchestrator:

1. In the Orchestrator Configuration, click on **General**.
2. Select **Install Application**.
3. Click on the magnifying glass icon. Now, select the plug-in file that you would like to install.
4. Click on **Install**.



Select a file to install: o11nplugin-nsx-1.0.0.vm

5. Accept the EULA by clicking on **I accept the terms of the License Agreement**.



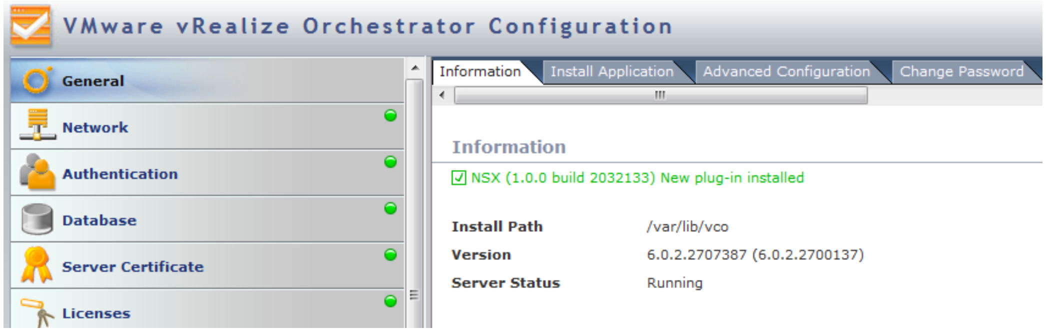
```
VMWARE END USER LICENSE AGREEMENT

PLEASE NOTE THAT THE TERMS OF THIS END USER LICENSE AGREEMENT SHALL GOVERN
YOUR USE OF THE SOFTWARE, REGARDLESS OF ANY TERMS THAT MAY APPEAR DURING THE
INSTALLATION OF THE SOFTWARE.

IMPORTANT-READ CAREFULLY:   BY DOWNLOADING, INSTALLING, OR USING THE SOFTWARE,
YOU (THE INDIVIDUAL OR LEGAL ENTITY) AGREE TO BE BOUND BY THE TERMS OF THIS
END USER LICENSE AGREEMENT ("EULA").  IF YOU DO NOT AGREE TO THE TERMS OF THIS
EULA, YOU MUST NOT DOWNLOAD, INSTALL, OR USE THE SOFTWARE, AND YOU MUST DELETE
OR RETURN THE UNUSED SOFTWARE TO THE VENDOR FROM WHICH YOU ACQUIRED IT WITHIN
THIRTY (30) DAYS AND REQUEST A REFUND OF THE LICENSE FEE, IF ANY, THAT YOU
PAID FOR THE SOFTWARE.

EVALUATION LICENSE.  If You are licensing the Software for evaluation
purposes, Your use of the Software is only permitted in a non-production
environment and for the period limited by the License Key.  Notwithstanding
any other provision in this EULA, an Evaluation License of the Software is
```

6. If the plug-in has installed successfully, you should see a green message at the top of the window. Also, you can check the **Plug-ins** tab.



Information

☑ NSX (1.0.0 build 2032133) New plug-in installed

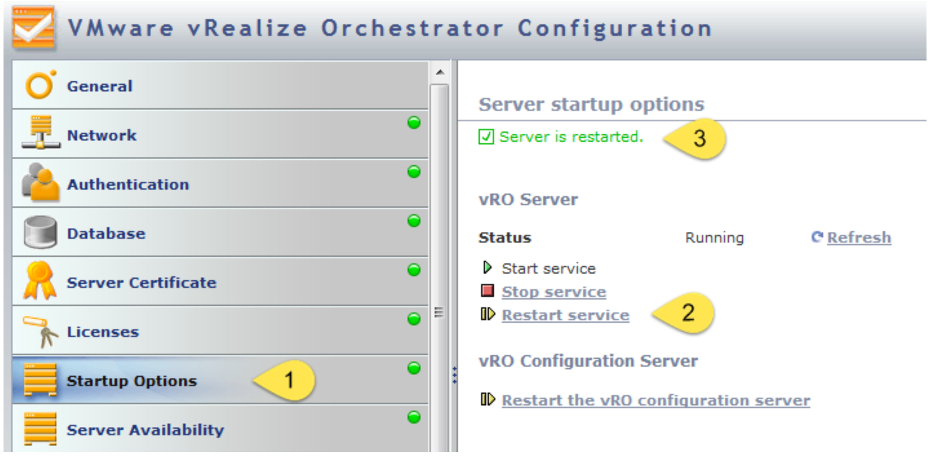
Install Path	/var/lib/vco
Version	6.0.2.2707387 (6.0.2.2700137)
Server Status	Running

We have installed the plug-in. However, to make it work, we need to restart the Orchestrator service.

Restarting the Orchestrator service

After you have installed a plug-in, you need to restart the Orchestrator service. You can do this in two ways. The first is to restart the Orchestrator VM. This is a bit drastic and takes longer time. However, it also works. The easier and smarter way is to just restart the Orchestrator service.

1. In the Orchestrator Configuration, click on **Startup Options**.
2. Then, click on **Restart service**.
3. After the server has restarted, you should see a green message that tells you that the restart process has successfully finished.



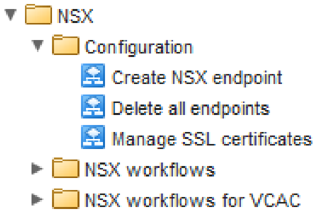
☑ Server is restarted.

Status Running

Checking out the new plug-in

After installing the plug-in, we should have a look what it provides us with. Each plug-in comes with predefined workflows and actions. Let's have a look at the pre-defined workflows:

1. Open the Orchestrator client and log in as a user of the Orchestrator Admin group.
2. Go to the workflow library and expand it. Look for the plug-in. As I chose to install the VMware NSX plug-in, I look for NSX.
3. Expand the subfolders and browse around. Have a look at what is there.



Most plug-ins come with a **Configuration** folder that contains workflows to bind Orchestrator to a specific instance of the plug-in; in my case, an NSX endpoint.

If you haven't heard about VMware NSX, it is a network virtualization tool that allows you to create isolated networks, routers, NATs, gateways, load balancers, and VPNs, without the need to have additional hardware.

Plugin problems

It happens from time to time that, after you install a plugin, and you check the Orchestrator client, you cannot find any additional workflows or actions. If this is the case, you need to run the workflow **Library** | **Configuration** | **Troubleshooting** | **Reinstall the plug-ins when the sever starts**. We have used this in the last step in *Chapter 2*, *Deploying and Configuring the Orchestrator Appliance*. After running this workflow, restart either the Orchestrator service or the whole appliance.

Summary

In this chapter, we had a closer look at the vCenter integration, as well as installing additional plug-ins. We saw how easy it is to integrate the central VMware vCenter infrastructure into Orchestrator. Also, we were introduced to the additional possibilities that the Web Client has to offer to us.

In the next chapter, we will have a proper go at workflows and learn how to run them in the Orchestrator Client as well as the vSphere Web Client.

4

Working with Workflows

Now that we have installed, configured, and integrated Orchestrator in our VMware vSphere infrastructure, we can finally have a look at how to use the product. We will learn how to use pre-created workflows that exist in the Orchestrator library. This includes running and stopping them, checking their outcome, and dealing with errors.

We will use the Orchestrator Client as well as the vSphere Web Client.

For this chapter, we need to have Orchestrator configured to use the vSphere vCenter. If you need help with this, have a look at *Chapter 3*, *Integrating Orchestrator with vSphere*. We will cover the following topics here:

- Workflow basics
- Starting workflows
- Workflow – renaming the virtual machine
- Handling errors
- Starting workflows from the vSphere Web Client
- Import and export workflows
- Scheduling workflows

Using the Orchestrator Client

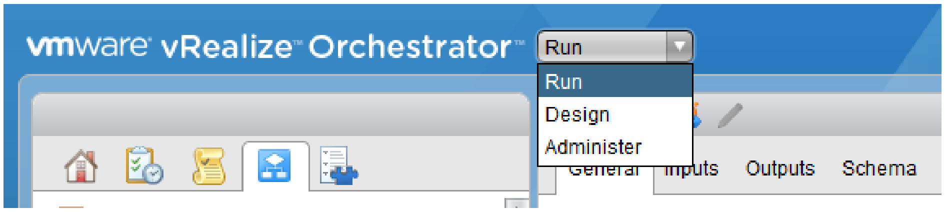
Before we actually start working with the Orchestrator workflows, we should first have a look at how the Orchestrator Client works.

Introducing the Orchestrator Client

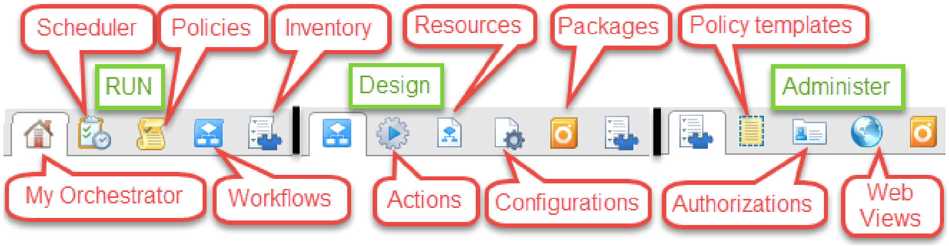
To open the Orchestrator Client, perform the following steps:

1. Open a web browser such as Firefox, IE, or Chrome.
2. Enter the IP or FQDN of the Orchestrator Appliance. The Orchestrator homepage opens.
3. Click on **Start Orchestrator Client**.
4. Your Java environment will now start. You may be required to acknowledge that you really want to start this application.
5. You will now be greeted with the login to Orchestrator. If you have configured Orchestrator with SSO (see *Chapter 2*, *Deploying and Configuring the Orchestrator Appliance*), enter a user who is a member of the Orchestrator Admin group. If you are using an unconfigured Orchestrator, just enter `vcoadmin` as the username and `vcoadmin` as the password. This is a preconfigured user that allows you to login and use Orchestrator directly.
6. Click on **Login**. The Orchestrator Client will load after a moment.

When you open the Orchestrator Client, it will be in the **Run** mode. There are three modes in total—**Run**, **Design**, and **Administer**.



Each of these modes is used for different phases of the workflow lifecycle. In the following image, you will see a collage of all the Orchestrator icons, which is followed by a table with a short description about what each icon is used for:



Icon		What it does
	My Orchestrator	This gives an overview of all the active tasks and configures non-admin access.
	Scheduler	This manages the scheduled workflows. We will schedule workflows later in this chapter.
	Policies	A policy is a trigger that can be configured to trigger when certain events occur.
	Workflows	This manages everything that has to do with workflows.
	Inventory	This shows all the objects that each plugin has access to.
	Actions	This manages everything that has to do with actions. Actions are like mini workflows. We will discuss Actions in detail in *Chapter 6, Advanced vRO Scripting with JavaScript*.
	Resources	A resource is a file that can be used from workflows.
	Configurations	Configurations are centrally defined attributes that are available to all workflows.
	Packages	A package contains workflows, actions, as well as all the other elements that export and import the Orchestrator solutions. We will work with packages in *Chapter 9, Packing It All Up*.
	Policy templates	These are the templates for policies.
	Authorizations	This is deprecated and not used anymore.
	Web Views	Web Views are HTTP-based web pages that interact with Orchestrator (deprecated and removed from Version 6).

We do not have the page count to discuss all the other items in this list. Please refer to the *vRealize Orchestrator Cookbook* for an extended discussion on resources, non-admin access to the Orchestrator, configurations, and policies.

Workflow properties

Let's get started with some very basic things about workflows. We will see what a workflow consists of as well as how we can handle it.

A workflow is made up of different elements. We will now look into each of these elements one after another. But first, we have to navigate to a workflow and have a look at it:

1. Start the Orchestrator Client, as shown at the beginning of the chapter.
2. Click on the Workflow icon, .
3. Now, you are in the workflow library. Expand the **Library** tree and then scroll down to **vCenter**.
4. Expand **vCenter** and then select **Virtual Machine management**.
5. Now, expand **Basic** and then click on the workflow **Create simple virtual machine**.

> From now on, we will use the shorter path description: **Library** | **vCenter** | **Virtual Machine management** | **Basic** | **Create simple virtual machine**.

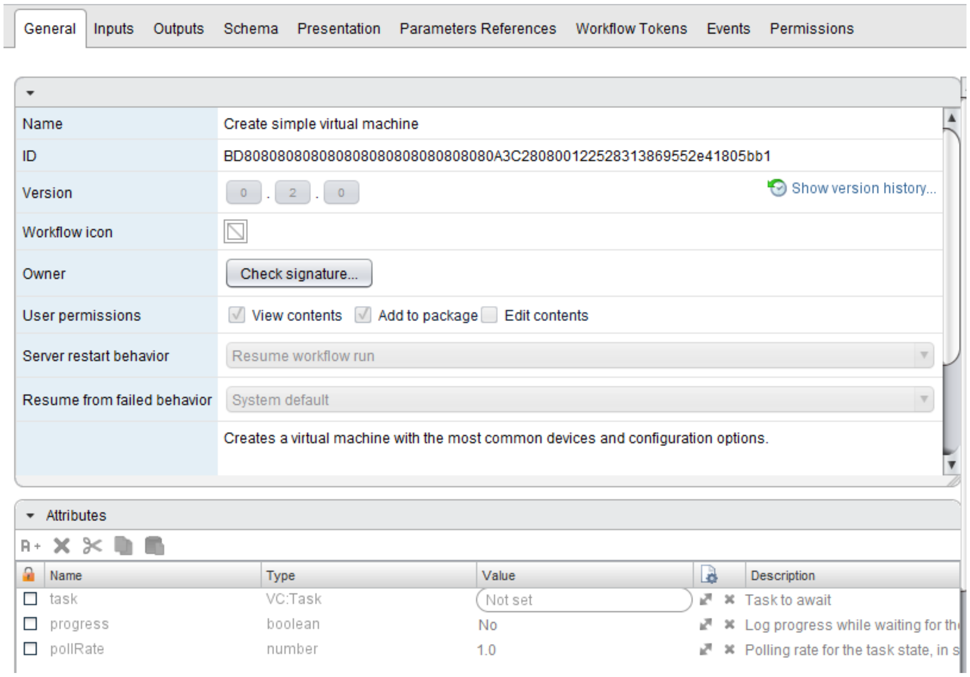
We will now go through all the properties of a workflow, as this is the central part of Orchestrator.

The General tab

The **General** tab consists of two main areas. The upper area shows all the properties of the workflow such as its name, ID, and so on. The lower area is where its attributes are shown (see the following figure). We will first have a closer look at the upper area:

- **Name**: The upper area contains the name of the workflow. The name can be changed, but it has to be unique in a folder.
- **ID**: The next item is the workflow ID. The workflow ID is automatically generated by Orchestrator and cannot be changed. It is unique for all Orchestrator installations. The ID is the overall identifier of a workflow. If you want to start a workflow via REST you would, use its ID, not its name.
- **Version**: This shows the current version of the workflow and also presents you with a link to see all the previous versions of a workflow. We will look at workflow versioning later in more detail in *Chapter 5*, *Combining and Modifying Workflows*.

- **Workflow icon**: You can change the workflow icon of a workflow to any icon that has been imported into Orchestrator. There are some predefined ones.
- **Owner**: The one who created this workflow is the owner. You can click on **Check signature** (the SSL certificate) to check whether the owner of the workflow is the current Orchestrator. The signature is the SSL certificate that we created in *Chapter 2*, *Deploying and Configuring the Orchestrator Appliance*.
- **User permissions**: This attribute defines the permissions that a given user has for this workflow. These settings are defined when exporting a workflow to determine if and how a workflow can be reused. We will have a look at this in *Chapter 9*, *Packing It All Up*.
- **Server restart behavior**: This defines what should happen if the Orchestrator server becomes unavailable and then available again (basically, Orchestrator is powered off) while the workflow is running.
- **Resume from failed behavior**: This determines what should happen if a workflow fails. The **System Default** option stops the workflow. The alternative is to have the workflow resume enabled. This means that in case the workflow fails, the workflow will stop and ask the user for alternative inputs for its attributes and inputs.
- **Description**: This attribute allows an editor to leave more information about a workflow.



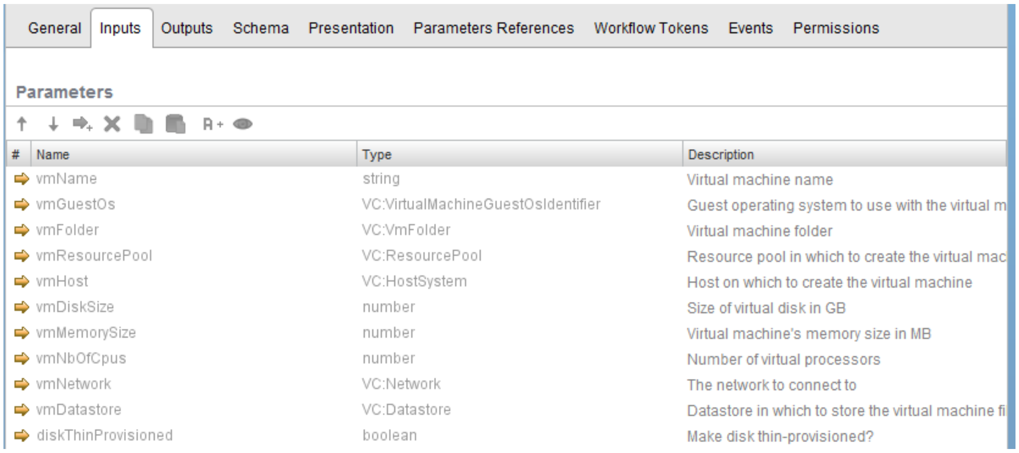
Name	Create simple virtual machine
ID	BD80808080808080808080808080808080A3C280800122528313869552e41805bb1
Version	0 . 2 . 0
Workflow icon	
Owner	Check signature...
User permissions	☑ View contents ☑ Add to package ☐ Edit contents
Server restart behavior	Resume workflow run
Resume from failed behavior	System default
	Creates a virtual machine with the most common devices and configuration options.

	Name	Type	Value		Description
☐	task	VC:Task	Not set		Task to await
☐	progress	boolean	No		Log progress while waiting for th
☐	pollRate	number	1.0		Polling rate for the task state, in s

The lower section contains all the attributes of this workflow. An attribute is like a global variable that is defined for all elements of this workflow. We will have a closer look at them in *Chapter 5, Combining and Modifying Workflows*.

The Inputs tab

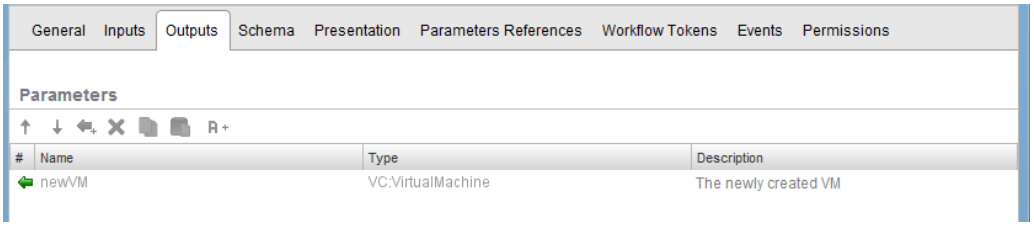
The **Inputs** tab shows all the input parameters that exist for this workflow. An input parameter is a variable that needs to be filled with a value when the workflow is started. Have a look at the first input parameter. It will ask for the name of the new VM. We will have a closer look at these parameters in *Chapter 5, Combining and Modifying Workflows*.



#	Name	Type	Description
➪	vmName	string	Virtual machine name
➪	vmGuestOs	VC:VirtualMachineGuestOsIdentifier	Guest operating system to use with the virtual m
➪	vmFolder	VC:VmFolder	Virtual machine folder
➪	vmResourcePool	VC:ResourcePool	Resource pool in which to create the virtual mac
➪	vmHost	VC:HostSystem	Host on which to create the virtual machine
➪	vmDiskSize	number	Size of virtual disk in GB
➪	vmMemorySize	number	Virtual machine's memory size in MB
➪	vmNbOfCpus	number	Number of virtual processors
➪	vmNetwork	VC:Network	The network to connect to
➪	vmDatastore	VC:Datastore	Datastore in which to store the virtual machine fi
➪	diskThinProvisioned	boolean	Make disk thin-provisioned?

The Outputs tab

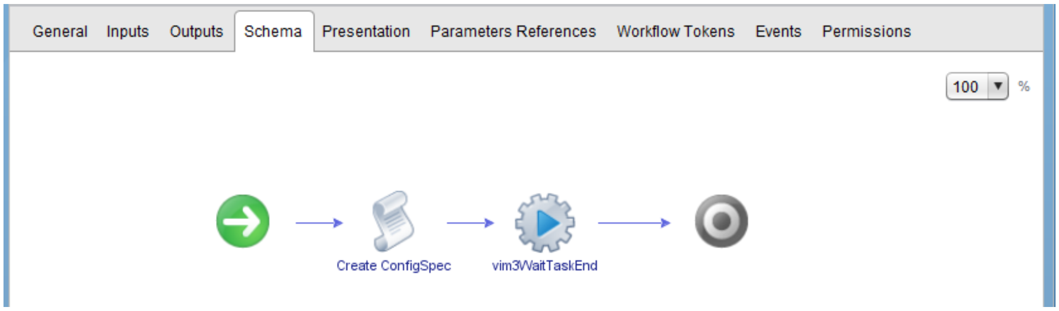
The **Outputs** tab shows all the output parameters of the workflow. These values are filled when the workflow is finished. The values can then be used by another workflow as the input parameters. The only output parameter of this workflow is the internal vSphere ID of the newly created VM. We will have a closer look at these parameters in *Chapter 5, Combining and Modifying Workflows*.



#	Name	Type	Description
⇦	newVM	VC:VirtualMachine	The newly created VM

The Schema tab

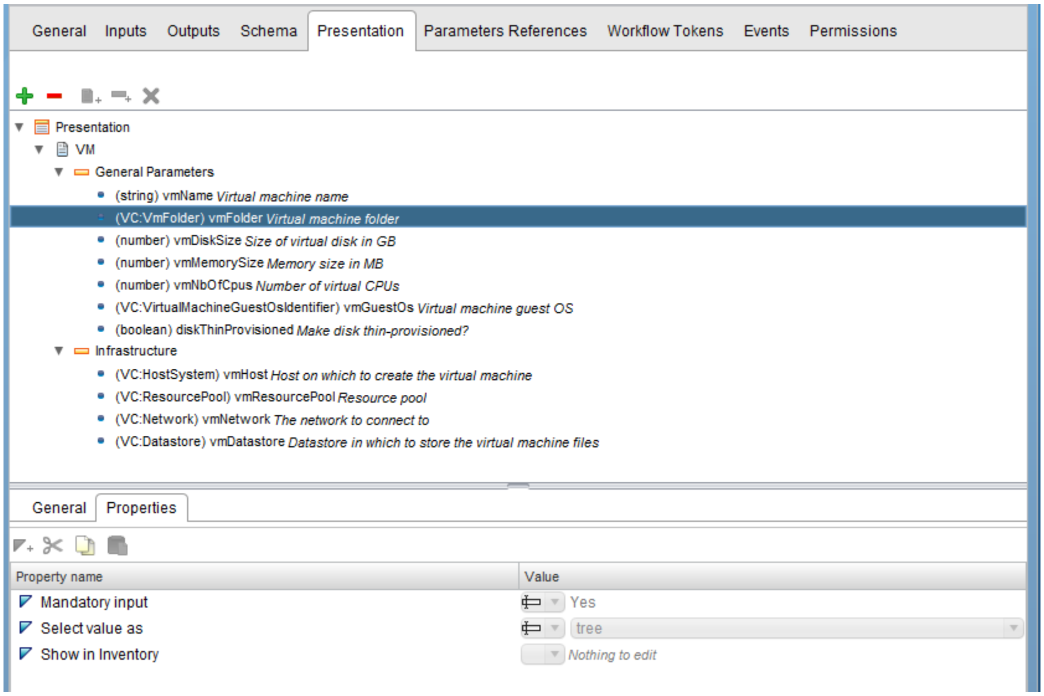
The **Schema** tab shows how the workflow is constructed. It shows all the separate elements that the workflow is made up of. Every workflow will consist of one Start element (the green arrow icon) and at one or more End elements (the grey shield icon). Between these two elements, you can see two other elements—a scripting element (the scroll icon) and an action element (the gear icon). On the upper right-hand side, you have an option to zoom in and out. We will have a closer look at this in *Chapter 5*, *Combining and Modifying Workflows*.



The Presentation tab

The **Presentation** tab is where you define how the input parameters are presented and formatted for the user. You can create pages and sections and sort the input parameters. You can also define a set of rules for each input parameter. We will work with the Presentation tab more in *Chapter 7*, *Improving Workflows with Presentation*. Have a look at the highlighted **vmFolder** input parameter in the following figure.

You will see that it has two properties, the first one being **Mandatory input**, which means that this input parameter must have a value, or else the workflow will not start:



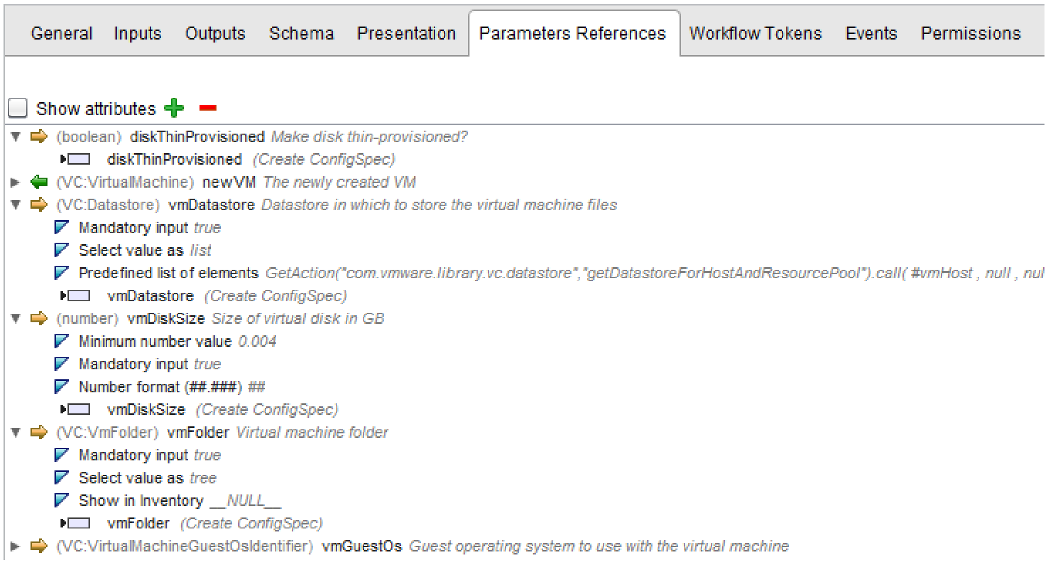
- Presentation
  - VM
    - General Parameters
      - (string) vmName *Virtual machine name*
      - (VC:VmFolder) vmFolder *Virtual machine folder*
      - (number) vmDiskSize *Size of virtual disk in GB*
      - (number) vmMemorySize *Memory size in MB*
      - (number) vmNbOfCpus *Number of virtual CPUs*
      - (VC:VirtualMachineGuestOsIdentifier) vmGuestOs *Virtual machine guest OS*
      - (boolean) diskThinProvisioned *Make disk thin-provisioned?*
    - Infrastructure
      - (VC:HostSystem) vmHost *Host on which to create the virtual machine*
      - (VC:ResourcePool) vmResourcePool *Resource pool*
      - (VC:Network) vmNetwork *The network to connect to*
      - (VC:Datastore) vmDatastore *Datastore in which to store the virtual machine files*

Property name	Value
Mandatory input	Yes
Select value as	tree
Show in Inventory	*Nothing to edit*

The Parameters References tab

The **Parameters References** view shows all the parameters of the workflow and its properties in an alphabetical order. For example, you can see that the **vmDiskSize** input parameter is of type number (it expects a number to be entered) and has multiple properties, such as **Minimum number value** of 0.004. This is an information page; you cannot change anything here.

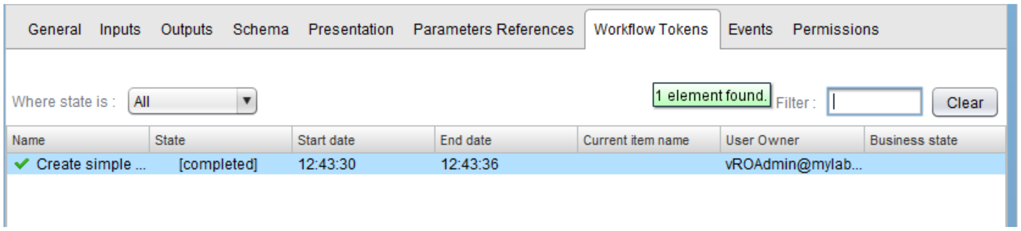
We will work with presentation properties in *Chapter 7, Improving Workflows with Presentation*.



```
▼ (boolean) diskThinProvisioned Make disk thin-provisioned?
      diskThinProvisioned (Create ConfigSpec)
▶ (VC:VirtualMachine) newVM The newly created VM
▼ (VC:Datastore) vmDatastore Datastore in which to store the virtual machine files
      Mandatory input true
      Select value as list
      Predefined list of elements GetAction("com.vmware.library.vc.datastore","getDatastoreForHostAndResourcePool").call( #vmHost , null , nul
      vmDatastore (Create ConfigSpec)
▼ (number) vmDiskSize Size of virtual disk in GB
      Minimum number value 0.004
      Mandatory input true
      Number format (##.###) ##
      vmDiskSize (Create ConfigSpec)
▼ (VC:VmFolder) vmFolder Virtual machine folder
      Mandatory input true
      Select value as tree
      Show in Inventory __NULL__
      vmFolder (Create ConfigSpec)
▶ (VC:VirtualMachineGuestOsIdentifier) vmGuestOs Guest operating system to use with the virtual machine
```

The Workflow Tokens tab

A workflow token is created when a workflow is started. As a workflow can be started multiple times at the same time, each execution of the workflow creates a unique workflow token. This is an information page that shows all the workflow tokens and their status. Each workflow token contains its status as well as its start and end time. Sometimes, it is more important to know which user actually started a workflow.



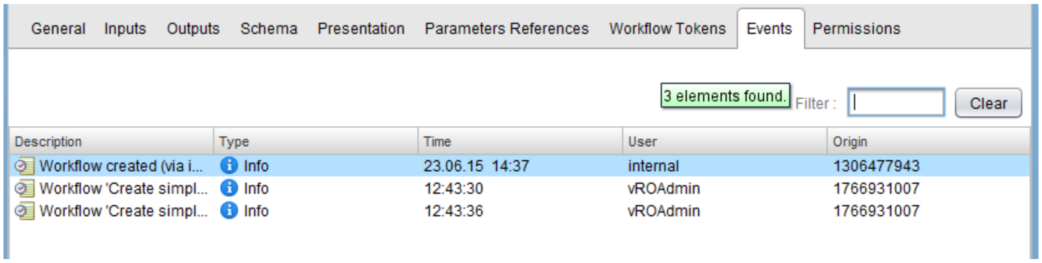
Where state is : All

1 element found.

Name	State	Start date	End date	Current item name	User Owner	Business state
✓ Create simple ...	[completed]	12:43:30	12:43:36		vROAdmin@mylab...	

The Events tab

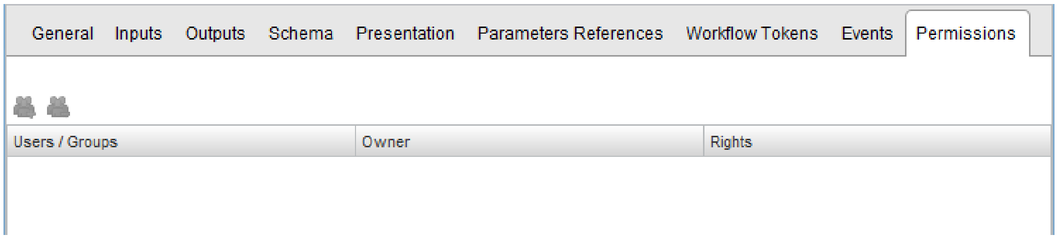
The **Events** tab is also an information page. It shows all the events that belong to this workflow. For instance, we have an event where this workflow was created, and the time when this happened is also shown (when you installed Orchestrator):



Description	Type	Time	User	Origin
Workflow created (via i...	Info	23.06.15 14:37	internal	1306477943
Workflow 'Create simpl...	Info	12:43:30	vROAdmin	1766931007
Workflow 'Create simpl...	Info	12:43:36	vROAdmin	1766931007

The Permissions tab

In the **Permissions** tab, one can regulate the access level of any nonadministrative user. We will not be able to go into detail here, so please refer to the *VMware vRealize Orchestrator Cookbook*.



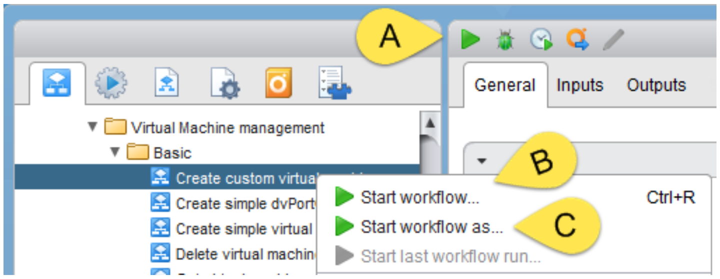
Users / Groups	Owner	Rights

Starting a workflow

Now that we have discussed what the different elements of a workflow are, let's run a workflow.

> We will now create a new VM. The workflow will not power the VM on.

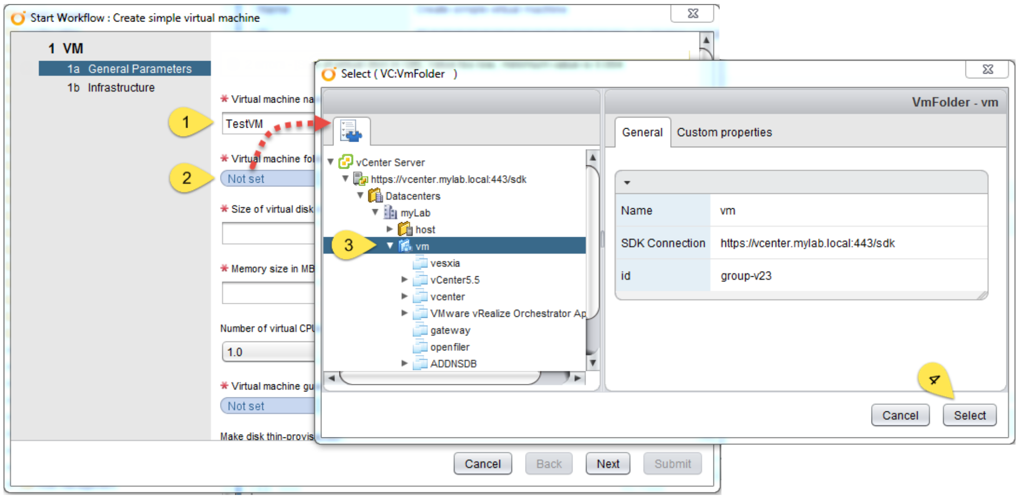
There are multiple methods that can be used to start a workflow. You can use the play button (the green arrow) that is located at the top of the workflow (Marker **A**), or you can right-click on the workflow in the library and then select **Start workflow** (Marker **B**). If you would like to start the workflow as a different user and not the one that you are currently logged on as, right-click on the workflow and select **Start workflow as** (Marker **C**).



After starting the workflow, a window will appear, displaying the input parameters that have to be provided in order to execute it. We will now perform the following steps to execute this workflow:

1. Enter the name of the new VM. This is a text field and can contain any ASCII code. However, if you are using any language specific characters such as ä, ö, and ü, you might get into trouble.
2. The virtual machine's folder is a vCenter type, therefore it requires the ID of an existing vCenter object. To assign a vCenter Object, click on **Not Set** to open the vCenter inventory.
3. A second window opens and lets you browse thought the vCenter inventory. Look for a folder called `VM`. This is the top VM folder when no other VM folder exists.

4. Try out the following: browse through the inventory and click on an object. Have a look at the **Select** button. Only if you click on an object that is compatible with the type that the input needs can you click on the button. This helps a lot if you are not sure where the object is located or what it should be.



Start Workflow : Create simple virtual machine

1 VM
1a General Parameters
1b Infrastructure

* Virtual machine na: TestVM
* Virtual machine fol: Not set
* Size of virtual disk:
* Memory size in MB:
Number of virtual CPU: 1.0
* Virtual machine gu: Not set
Make disk thin-provis

Select (VC:VmFolder)

VmFolder - vm

General | Custom properties

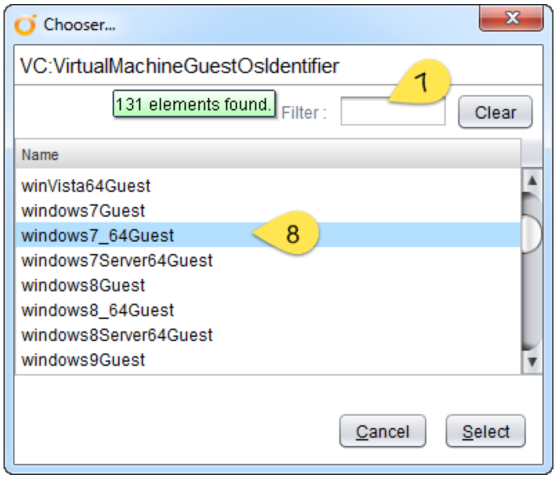
Name	vm
SDK Connection	https://vcenter.mylab.local:443/sdk
id	group-v23

5. The next three fields are number fields where you can only fill in numbers. The third field (vCPU) is also a number. However, using the presentation, it is a drop-down menu with only a preselected value. We will have a quick look at presentation in *Chapter 7, Improving Workflows with Presentation*.

6. The Virtual machine **guest OS** again has a vCenter type, but in this case, not one where you select something from the inventory, but from a list of predefined values. Click on **Not set**, and a new window will open up.

7. The window is empty when it opens, and this is a confusing thing for novices. Simply click into **Filter** and then press *Enter*. This will activate an empty filter that displays all options. You can also enter text, such as `64`, to filter for specific options. This kind of window is pretty common in Orchestrator.

8. Select one of the options and then click on **Select**.



Chooser...

VC:VirtualMachineGuestOsIdentifier

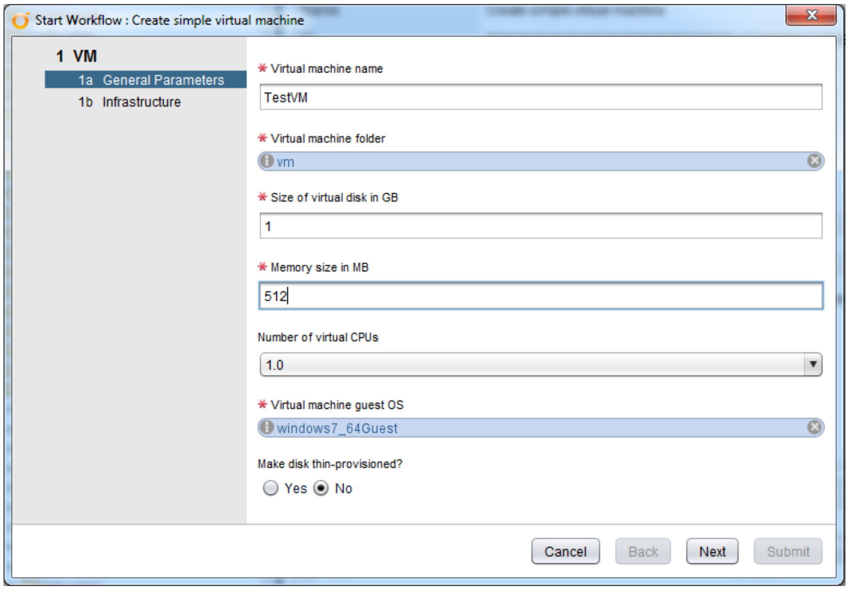
131 elements found.

Name

winVista64Guest
windows7Guest
windows7_64Guest
windows7Server64Guest
windows8Guest
windows8_64Guest
windows8Server64Guest
windows9Guest

9. You can now choose to make the VM disk thin provisioned. This is a Boolean type, and Orchestrator displays **Yes** or **No**. However, in JavaScript, Orchestrator uses the correct Boolean values, `true` and `false`.

10. After filling everything out, you will see that you can select **Next**. Have a look at the red * symbol before each text. It indicates that this parameter is a mandatory input.



Start Workflow : Create simple virtual machine

1 VM
1a General Parameters
1b Infrastructure

* Virtual machine name
TestVM

* Virtual machine folder
vm

* Size of virtual disk in GB
1

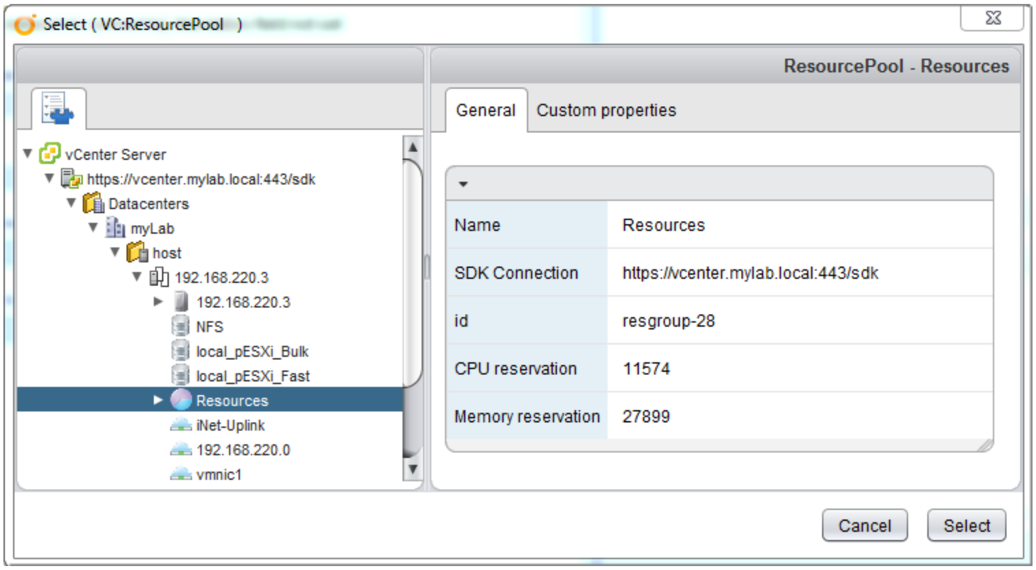
* Memory size in MB
512

Number of virtual CPUs
1.0

* Virtual machine guest OS
windows7_64Guest

Make disk thin-provisioned?
○ Yes ● No

11. On the second page, you are asked for the host. However, there is no red * symbol. So, it's not a mandatory input. We will leave it alone.
12. The ResourcePool again is a vCenter object. Click on **Not Set** and then select **Resources** under the cluster or host where you want to create the VM. The **Resources** resource pool is not a real pool, it represents the resources of the cluster or the host as such.

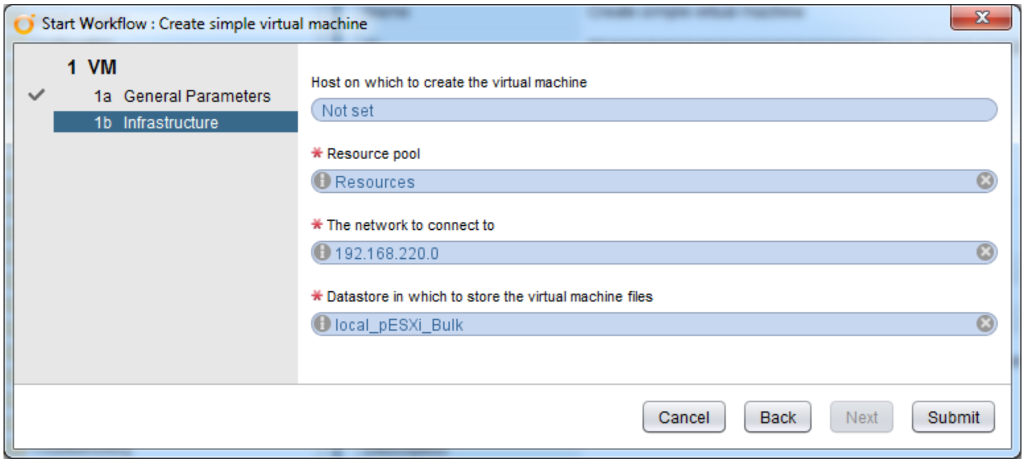


ResourcePool - Resources

Name	Resources
SDK Connection	https://vcenter.mylab.local:443/sdk
id	resgroup-28
CPU reservation	11574
Memory reservation	27899

13. The next item is a vCenter network object. Browse and find your correct network.
14. The last input is the vCenter datastore where you want to store the VM.

15. After you are finished, click on **Submit** to start the workflow.


Start Workflow : Create simple virtual machine

Host on which to create the virtual machine: Not set

* Resource pool: Resources

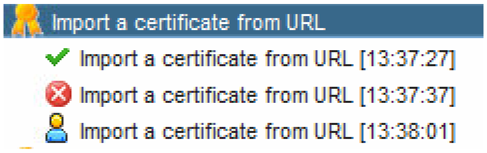
* The network to connect to: 192.168.220.0

* Datastore in which to store the virtual machine files: local_pESXi_Bulk

Workflow run and results

By clicking on **Submit**, you have started the workflow. Orchestrator now switches to the workflow token (execution). You will see how it runs through each step and the variables fill up.

After the workflow completes its run, it can be in three states—successful (a green tick), failed (a red bullet with a white cross), or it will ask for more information (the person icon). The following figure shows all the executions with their respective start time next to them. Each of these executions is a workflow token, as shown in the **Workflow Tokens** section of the workflow's properties.


Import a certificate from URL

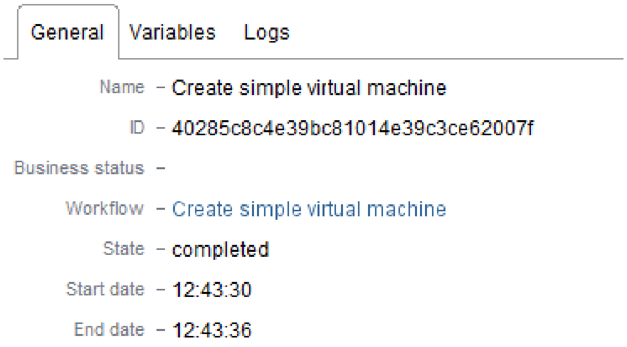
✓ Import a certificate from URL [13:37:27]

☒ Import a certificate from URL [13:37:37]

👤 Import a certificate from URL [13:38:01]

When you click on a workflow execution, you get additional information about the workflow's run. There are three sections—the **General**, the **Variables**, and the **Logs** tab.

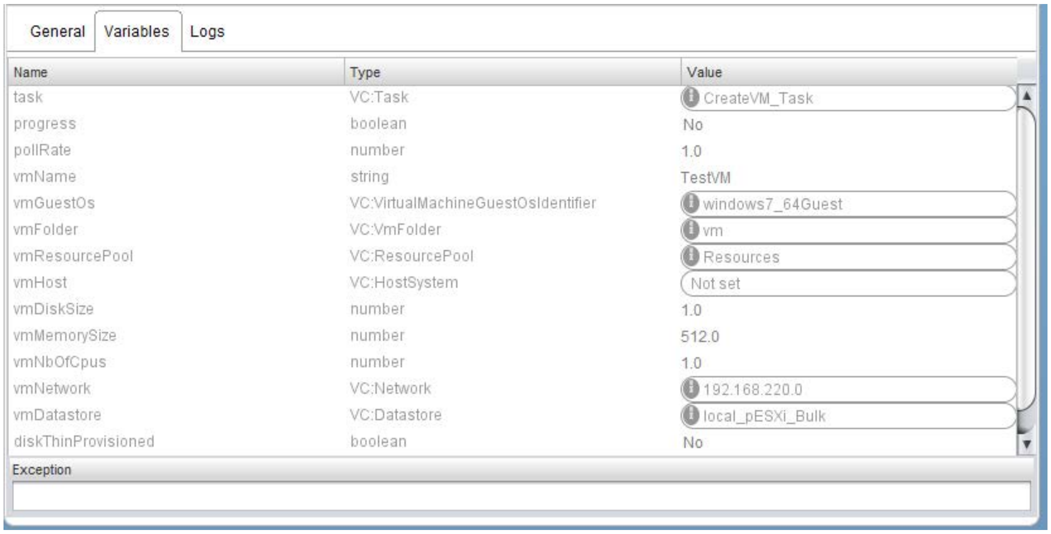
The **General** tab shows the name of the workflow and its token ID, which is a unique ID inside Orchestrator. Also, the status and execution time are shown:



Name – Create simple virtual machine
ID – 40285c8c4e39bc81014e39c3ce62007f
Business status –
Workflow – Create simple virtual machine
State – completed
Start date – 12:43:30
End date – 12:43:36

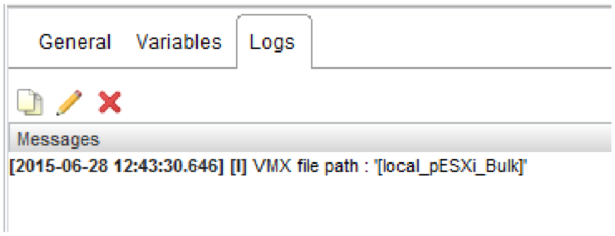
The **Variables** tab shows all the parameters (the in- and out-parameters and attributes), each with its type and the value it has when the workflow is finished. Some of the values have a grey **i** icon before it. If you hover the mouse on it, you will be shown additional information.

Underneath the parameters is a window titled **Exception**. If an error occurred during the workflow run, the error will be displayed here in red text.



Name	Type	Value
task	VC:Task	CreateVM_Task
progress	boolean	No
pollRate	number	1.0
vmName	string	TestVM
vmGuestOs	VC:VirtualMachineGuestOsIdentifier	windows7_64Guest
vmFolder	VC:VmFolder	vm
vmResourcePool	VC:ResourcePool	Resources
vmHost	VC:HostSystem	Not set
vmDiskSize	number	1.0
vmMemorySize	number	512.0
vmNbOfCpus	number	1.0
vmNetwork	VC:Network	192.168.220.0
vmDatastore	VC:Datastore	local_pESXi_Bulk
diskThinProvisioned	boolean	No

The **Logs** tab shows the logs that the workflow created while it was running in the Orchestrator Client. We will have a look at creating logs in *Chapter 8*, *Errors, Logs, and Debug Mode*.

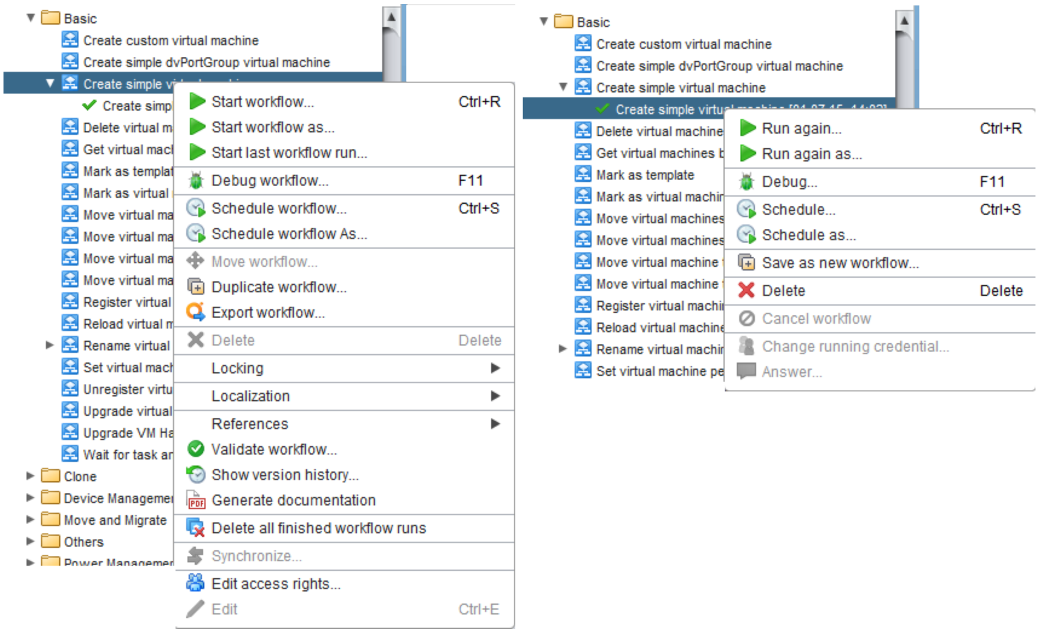


```
[2015-06-28 12:43:30.646] [I] VMX file path : '[local_pESXi_Bulk]'
```

Rerunning workflows

When an error occurs in the workflow run, most of the time you can fix the underlying problem and then you can rerun the workflow. Entering all the information again is a bit annoying.

There are two ways to do this, but basically, all result in the same thing. Right-click either on the workflow and select **Start last workflow run** or on the workflow execution (workflow token) and select **Run again**.



When you choose to rerun a workflow, the workflow input window will open and all the values, except for the passwords, will already be filled with the values that you had selected the last time you ran it.

The difference between the two methods is that the first method, by using the workflow, will result in rerunning the last execution of the workflow, whereas in the second, you can select the execution of the workflow that you are rerunning.

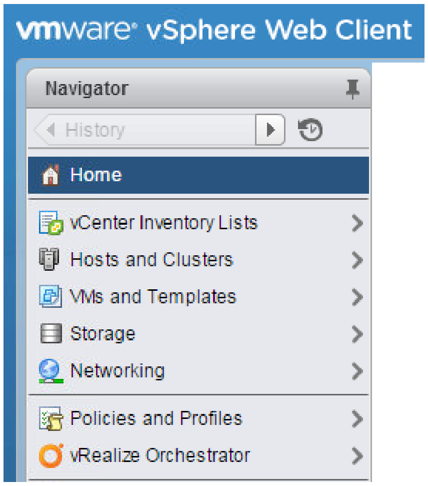
Using the vSphere Web Client

After exploring how to use the Orchestrator client to run a workflow, let's do the same with the vSphere Web Client. The advantage in using the Web Client is that you do not need to start and log into a separate client, and you can directly make use of the existing objects.

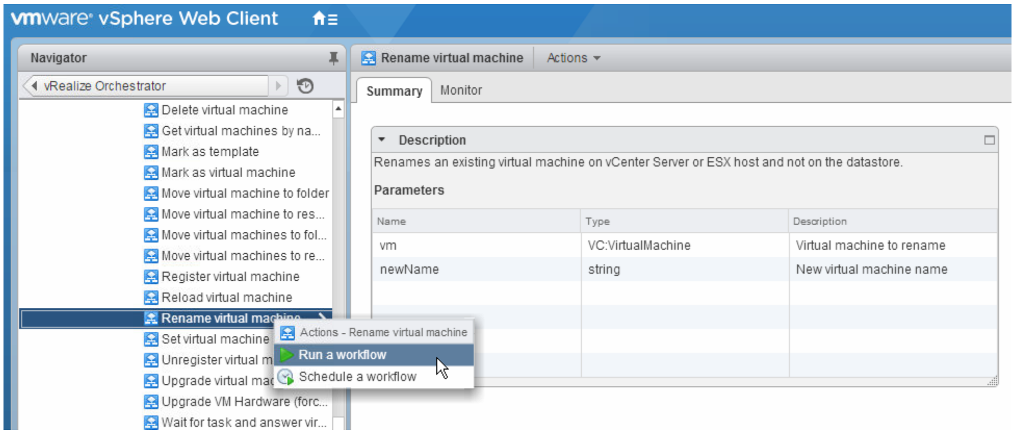
Starting a workflow from the Orchestrator plugin

We will now run a workflow using the vSphere Web Client. This method is good to run any workflow. Later, we will run the workflows directly from the vCenter inventory by simply right-clicking on them.

1. Open the vSphere Web Client and log in as a user who is a member of the Orchestrator administration group.
2. Click on the Orchestrator icon.



3. Click on **Inventory Trees** | **Workflows**.
4. The Orchestrator workflow library will be shown.
5. Browse **Library** | **vCenter** | **Virtual Machine Management** | **Basic**.
6. Right-click on **Rename virtual machine** and select **Run a workflow**.
7. You will be asked to allow the execution of the Orchestrator workflow.



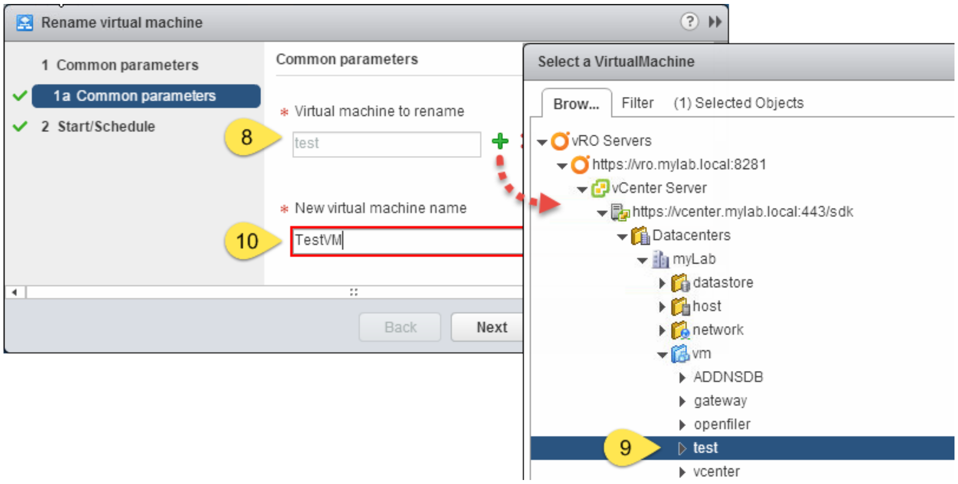
Description

Renames an existing virtual machine on vCenter Server or ESX host and not on the datastore.

Parameters

Name	Type	Description
vm	VC:VirtualMachine	Virtual machine to rename
newName	string	New virtual machine name

8. The workflow input window is displayed. Click on the green **+** sign to open the Orchestrators vCenter inventory.
9. In the inventory, drill down to the VM that we created in the last section and click on **Select**.
10. Enter a new name for the VM and click on **Finish**.

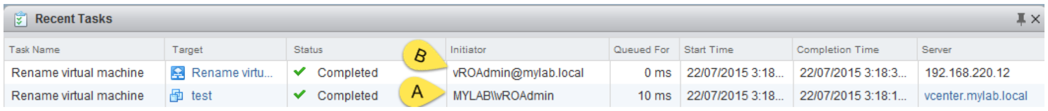


Common parameters

* Virtual machine to rename: test

* New virtual machine name: TestVM

After clicking on **Finish**, the workflow will be executed. You can see a new task pop up in the vCenter task bar. As you can see in the following screenshot there are two tasks, were one would have expected to see only one task. Have a closer look. The first one (marked **A**) is run as the user who is logged into vCenter, and the second (marked **B**) is the one that is run by Orchestrator against vCenter. This is the case if you have selected one session instead of per-user session when configuring Orchestrator with vCenter (see *Chapter 2*, *Deploying and Configuring the Orchestrator Appliance*).

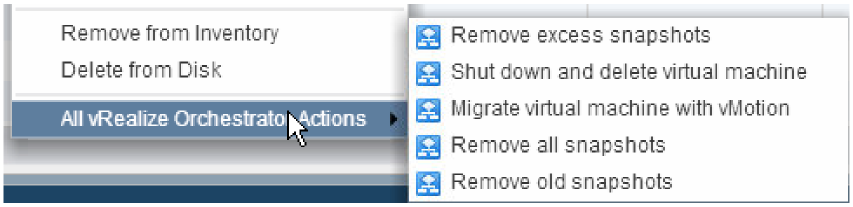


Task Name	Target	Status	Initiator	Queued For	Start Time	Completion Time	Server
Rename virtual machine	Rename virtu...	✓ Completed	vROAdmin@mylab.local	0 ms	22/07/2015 3:18...	22/07/2015 3:18:3...	192.168.220.12
Rename virtual machine	test	✓ Completed	MYLAB\\vROAdmin	10 ms	22/07/2015 3:18...	22/07/2015 3:18:1...	vcenter.mylab.local

Configuring workflows for the vCenter inventory

Not all workflows are integrated directly into the vSphere Web Client. We will have a look at how to configure them in a second.

All available workflows for a given vCenter type are shown when you right-click on a vCenter object and then select **All vRealize Orchestrator Actions**, as shown in the following screenshot:

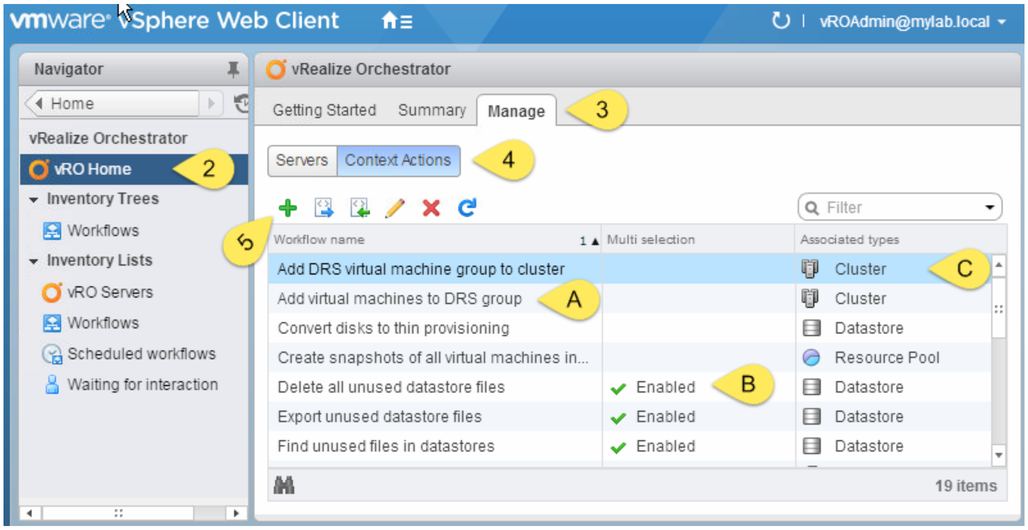

Remove from Inventory
Delete from Disk
All vRealize Orchestrator Actions
Remove excess snapshots
Shut down and delete virtual machine
Migrate virtual machine with vMotion
Remove all snapshots
Remove old snapshots

Before we run a workflow from the vCenter inventory, we will configure a workflow, which will be used in the next chapter, by performing the following steps:

1. In the vSphere Web Client, go to the Orchestrator section.
2. Click on **vRO Home**.
3. Click on the **Manage** tab.

4. Select **Context Actions**.

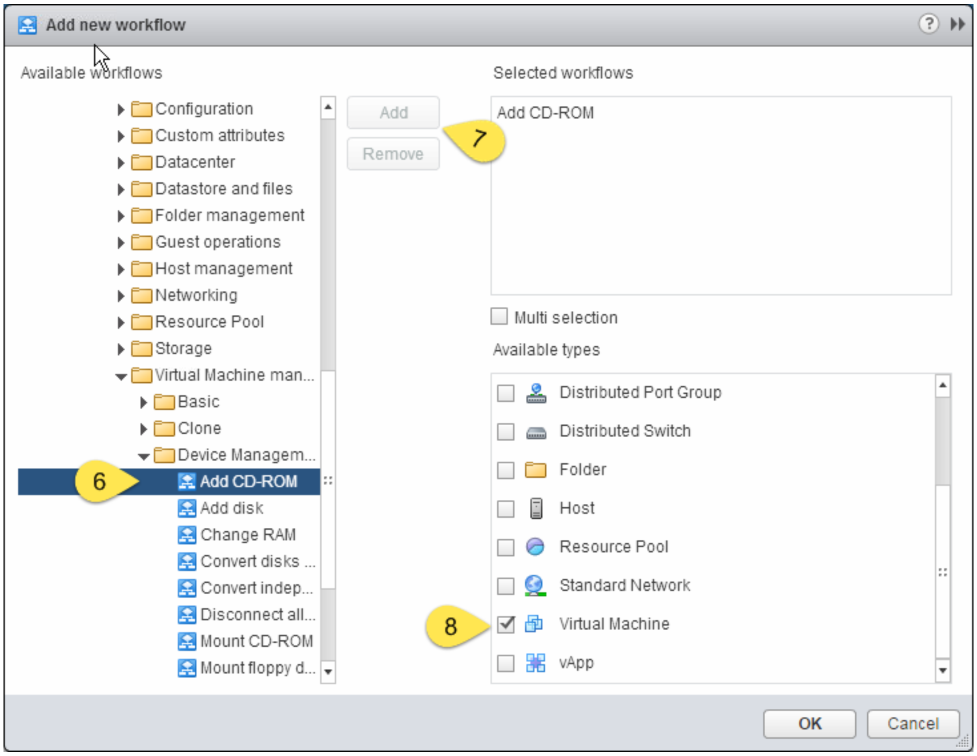
   Here you find all the workflows configured for use within the vCenter inventory. In section **A**, you will find the name of the workflow. Section **B** shows whether this plug-in is enabled for **Multi selection**. Multi selection means that you can run this workflow with more than one vCenter object at once. Section **C** shows for which vCenter object type this workflow will be accessible.



Workflow name	Multi selection	Associated types
Add DRS virtual machine group to cluster		Cluster
Add virtual machines to DRS group		Cluster
Convert disks to thin provisioning		Datastore
Create snapshots of all virtual machines in...		Resource Pool
Delete all unused datastore files	✓ Enabled	Datastore
Export unused datastore files	✓ Enabled	Datastore
Find unused files in datastores	✓ Enabled	Datastore

5. To add a workflow to the vSphere Web Client, click on the green **+** sign.
6. The Orchestrator workflow library will now open. Browse **Library** | **vCenter** | **Virtual machine management** | **Device Management** and select **Add CD-ROM**.
7. Click on **Add**. The workflow will now be added to the selection on the right.

8. Search for and click on **Virtual Machine** to assign this workflow.



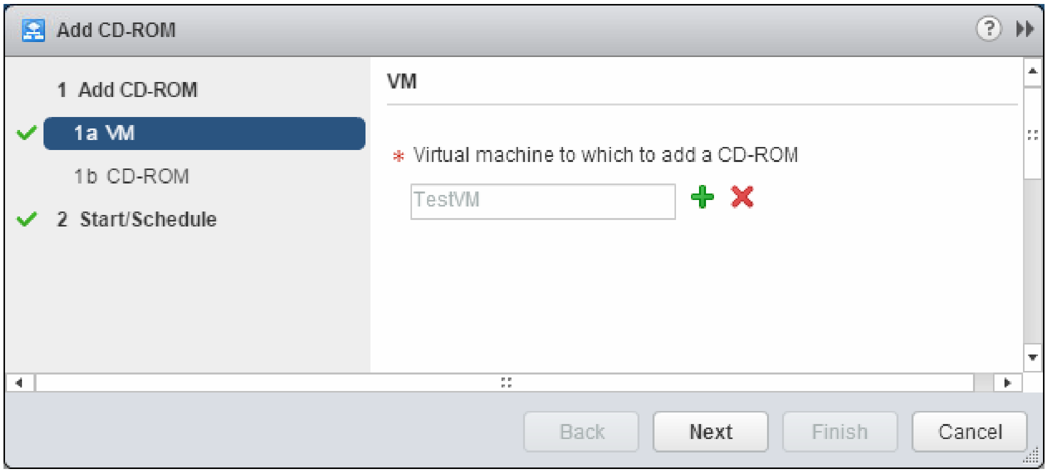
The workflow has been added and assigned to a vCenter VM type. You may need to log out of the vSphere Web Client and then log in again to make the change visible.

Starting a workflow from the vCenter inventory

We will now execute the workflow form the vCenter inventory by performing the following steps:

1. In the vSphere Web Client, click on **Hosts and clusters**.
2. Select the VM that we created and then right-click on it.
3. Select from **All vRealize Orchestrator Actions** the **Add CD-ROM** workflow.

4. The workflow input window shows up. The name of the VM that we just right-clicked on is already in it.

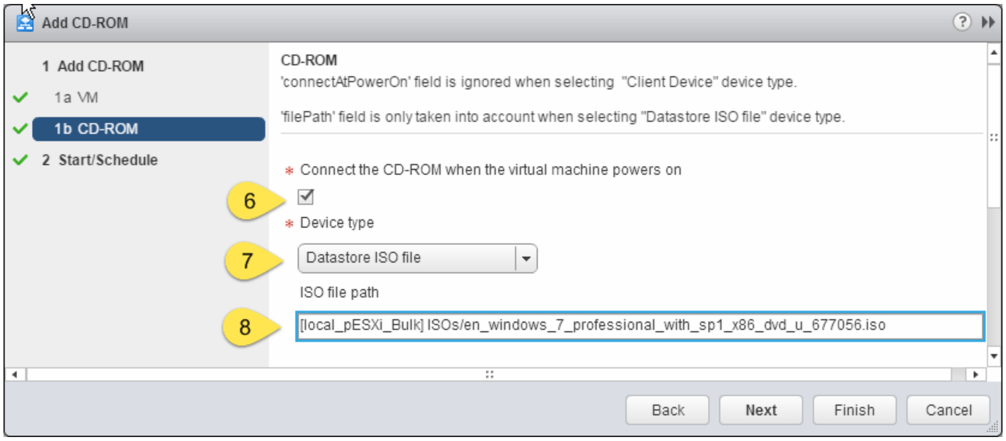


VM

* Virtual machine to which to add a CD-ROM

TestVM

5. Click on **Next**.
6. Select **Connect the CD-ROM when the virtual machine powers on**.
7. Select from the dropdown **Device type**, the option **Datastore ISO file**.
8. Enter the path to an ISO file in the format `[datastore] folder/file`.



CD-ROM

'connectAtPowerOn' field is ignored when selecting "Client Device" device type.

'filePath' field is only taken into account when selecting "Datastore ISO file" device type.

* Connect the CD-ROM when the virtual machine powers on

☑

* Device type

Datastore ISO file

ISO file path

[local_pESXi_Bulk] ISOs/en_windows_7_professional_with_sp1_x86_dvd_u_677056.iso

After the workflow has finished, the VM should have a CD-ROM that it didn't have before, and the CD-ROM should be connected to an ISO.

Scheduling workflows

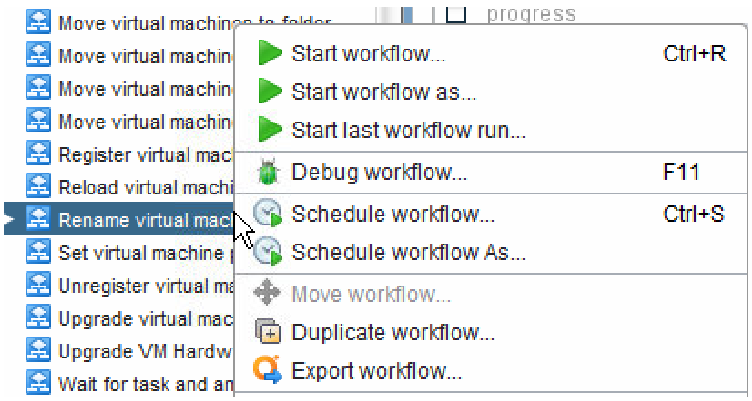
One of the cool functions of Orchestrator that helps a lot of admins is the possibility to schedule workflows for a later run. Just imagine the following scenario. You need to power off a VM at midnight for a scheduled outage time and then make sure that it's up again by 8:00 a.m. You can remotely log in if you have the possibility, or you have to stay in the office until then. Using Orchestrator, you can simply schedule these jobs. After you learn to build your own workflows, you can not only do simple jobs of running a library workflow, but also combining them. We will delve into this in the next chapter.

You can schedule workflows using the Orchestrator client or the vSphere Web Client. We will show how to do it both ways. I will use the rename VM workflow to showcase this as it has a low impact on the vSphere environment. The main difference between the Orchestrator Client and the vSphere Web Client is that by using the Orchestrator Client, you have more granularity in the selection of reoccurrences.

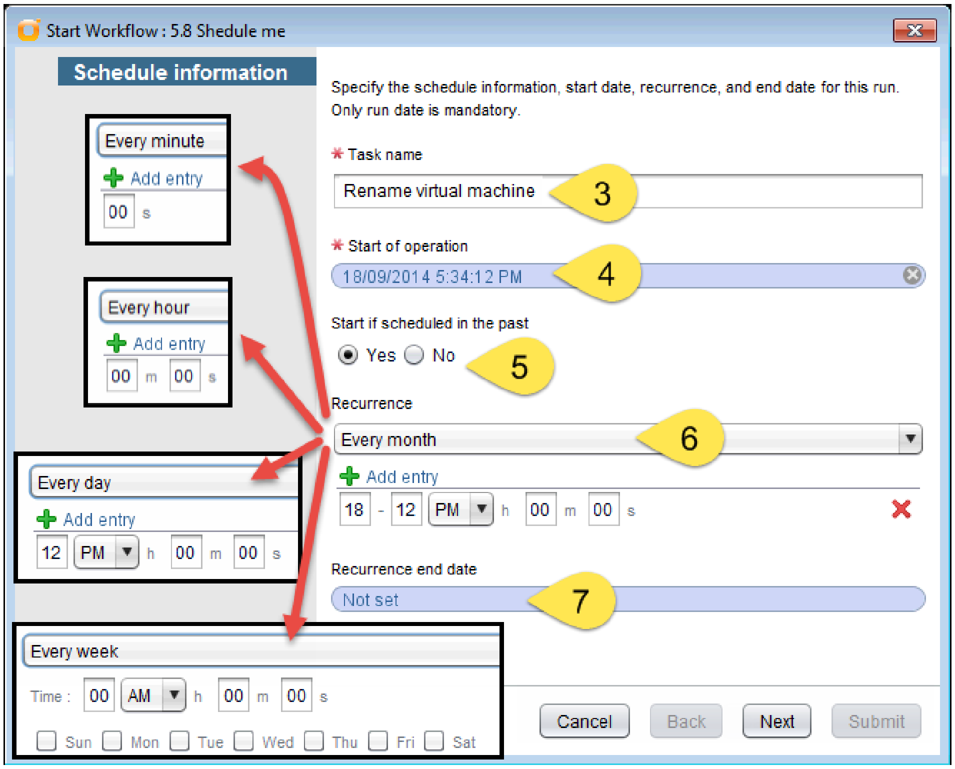
The Orchestrator Client

Scheduling a workflow by using the Orchestrator Client works in the following way:

1. Open the Orchestrator client and go to the workflow library.
2. Right-click on the workflow in **Library** | **vCenter** | **Virtual Machine management** | **Basic** | **Rename virtual machine** and select **Schedule workflow**. The **Schedule workflow as** option works exactly like the **Schedule workflow** with the added benefit that you can change the credentials of the executing user.



3. In the input window, select a **Task name**; the default is the name of the workflow. Sometimes, it's a good idea to make it a bit more descriptive.
4. Select a start date and time for the workflow to execute.
5. If the Orchestrator server was not available when the workflow should have been started, would you like to start it as soon as it is available?
6. Select the reoccurrence of the workflow execution. You can choose from **No Reoccurrence** to **Monthly.** Please note the **Add entry** link where you can select multiple occurrences in the same interval.
7. You can select an end date, which is when the last reoccurrence should take place. If you don't set it (leave it at **Not set**), the task will be executed indefinitely.



Start Workflow : 5.8 Shedule me

Schedule information

Specify the schedule information, start date, recurrence, and end date for this run. Only run date is mandatory.

* Task name: Rename virtual machine

* Start of operation: 18/09/2014 5:34:12 PM

Start if scheduled in the past: ● Yes ○ No

Recurrence: Every month

Add entry

18 - 12 PM h 00 m 00 s

Recurrence end date: Not set

Every minute

Add entry

00 s

Every hour

Add entry

00 m 00 s

Every day

Add entry

12 PM h 00 m 00 s

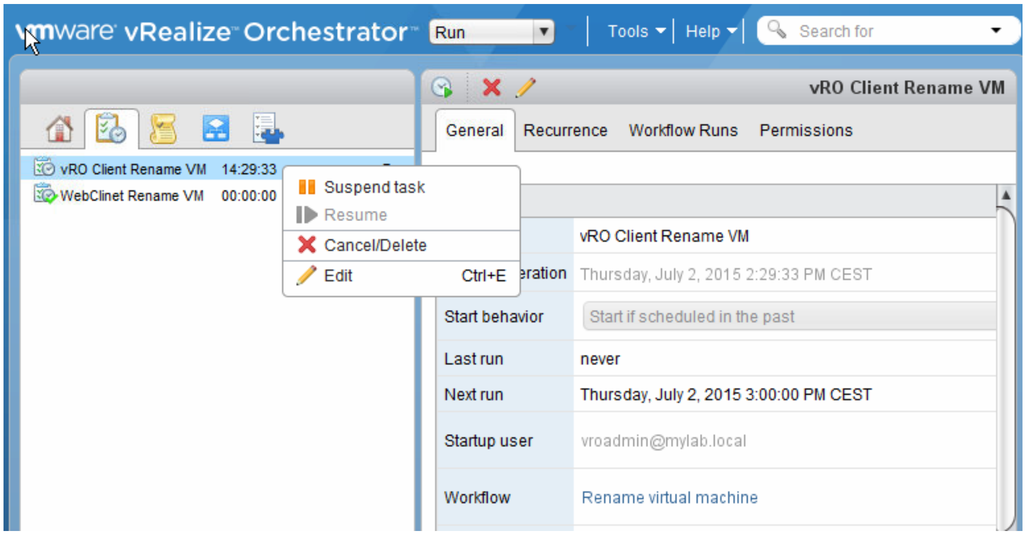
Every week

Time : 00 AM h 00 m 00 s

□ Sun □ Mon □ Tue □ Wed □ Thu □ Fri □ Sat

8. After you click on **Next**, you can continue with the input window for the workflow parameters.

The workflow is now scheduled, and you can look at it by clicking on the Scheduler icon that is displayed in the **Run** module.



vRO Client Rename VM 14:29:33
WebClinet Rename VM 00:00:00

	vRO Client Rename VM
eration	Thursday, July 2, 2015 2:29:33 PM CEST
Start behavior	Start if scheduled in the past
Last run	never
Next run	Thursday, July 2, 2015 3:00:00 PM CEST
Startup user	vroadmin@mylab.local
Workflow	Rename virtual machine

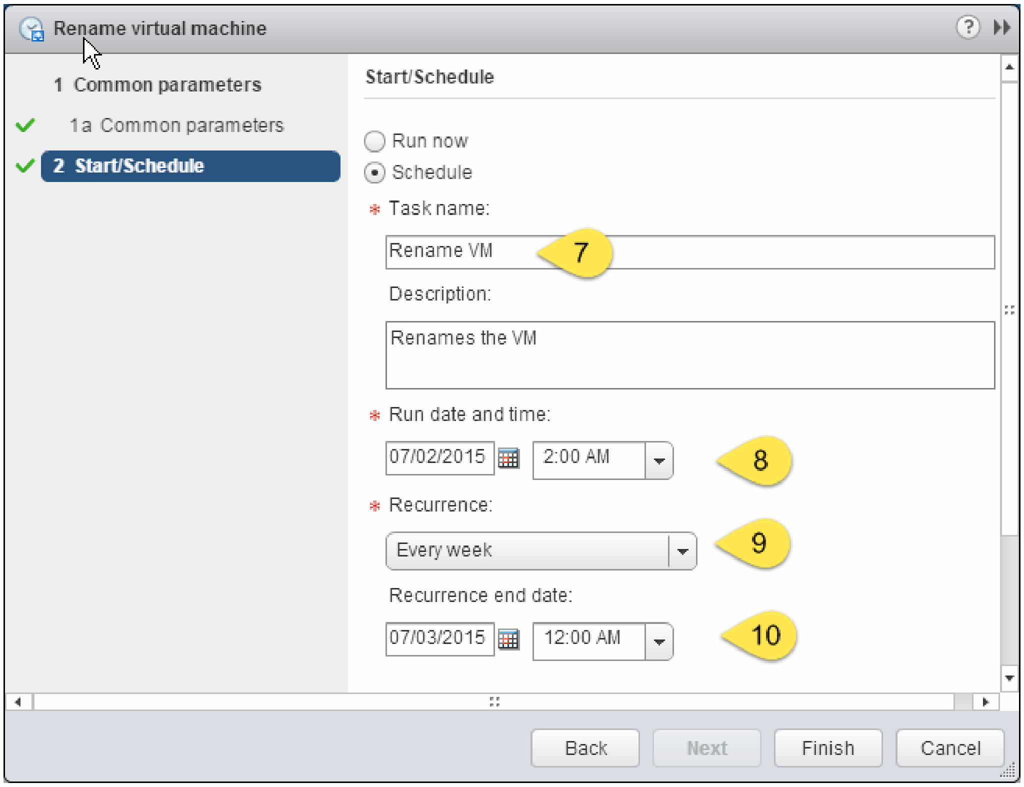
In the scheduler, you can suspend, resume, edit and cancel/delete the task. Editing allows you to edit the reoccurrences, but not the workflow parameters. To change the parameters, you need to cancel the workflow scheduling and schedule it anew.

The vSphere Web Client

We will now schedule a workflow using the vSphere Web Client:

1. Open the vSphere Web Client and log in as a user who is a member of the Orchestrator administration group.
2. Click on the Orchestrator icon.
3. Navigate to **Inventory Trees** | **Workflows**. The Orchestrator workflow library will be shown.
4. Right-click on the workflow **Library** | **vCenter** | **Virtual machine management** | **Basic** | **Rename virtual machine** and select **Schedule a workflow**.
5. Select the input values of the workflow, as we already did in the previous section.

6. Click on **Next**.
7. Give the task a name and maybe a description.
8. Select a start date and time.
9. Choose a reoccurrence from the following options: **Run Once**, **Every Minute**, **Every Hour**, **Every Day**, **Every Week**, and **Every Month**. Whatever you select, there isn't an additional detailed selection possible as with the Orchestrator Client.
10. Select **Reoccurrence end date**.



Rename virtual machine

Start/Schedule

○ Run now
● Schedule

* Task name: Rename VM

Description: Renames the VM

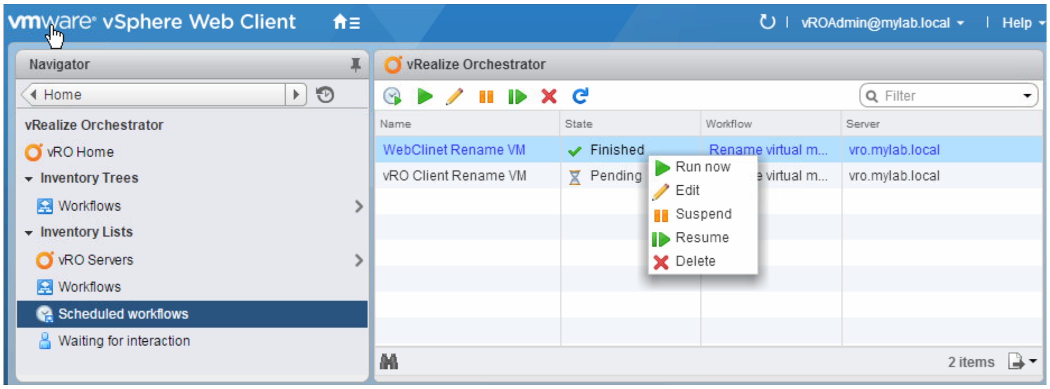
* Run date and time: 07/02/2015 2:00 AM

* Recurrence: Every week

Recurrence end date: 07/03/2015 12:00 AM

11. After you click on **Finish**, the workflow is scheduled.

You can check which workflows are scheduled by clicking on **Scheduled Workflows** in the main Orchestrator site in the vSphere Web Client. As with the Orchestrator client, you can suspend, resume, edit, and delete. Additionally you can run the workflow now, which is not an option available in the Orchestrator Client.



Name	State	Workflow	Server
WebClinet Rename VM	Finished	Rename virtual m...	vro.mylab.local
vRO Client Rename VM	Pending	e virtual m...	vro.mylab.local

Summary

In this chapter, we had a close look at workflows. We learned how to run them using the Orchestrator Client and the vSphere Web Client. We also saw how we can configure and run a workflow directly from the vCenter inventory. This feature will come in handy frequently, as this is what makes Orchestrator stand out.

The possibility to schedule workflow is pretty cool for admin tasks, but it has a much wider range of possibilities. In advanced programming, you can create scheduled tasks by running a workflow, making it possible to automate, for instance, the creation of a VM at a given time.

In the next chapter, we will learn how to combine the existing workflows together.

5
Combining and Modifying Workflows

Now that we understand how to work with a workflow, we will take the next step of combining workflows. As there are a lot of previously created workflows in the library, we can simply play Lego and build a new workflow from the existing elements. This is a common practice. What is also pretty common is to modify the existing workflows to make them do what you would like them to do. In this chapter, we will do all that and a bit more.

For this chapter, I assume that you have gone through the previous chapter or are familiar with its concepts. Here are the major concepts that we'll look at in this chapter:

- Combining the existing workflows
- Workflow validation and history
- Useful workflow operations such as copying, deleting, and undeleting a workflow
- Introduction to parameter types
- Introduction to the workflow decision element
- Modifying the existing workflows

Combining existing workflows

A very common task for workflow creators is to use the existing workflows and put them together. In this section, we will create a new workflow that consists of other workflows. We will build a workflow that will create a VM, add and mount a CD-ROM, and then start it. We have already used the workflows involved in the last chapter. During the next chapters we will enhance this workflow further.

> It's always a good idea to try out an existing workflow before using it so that you know what it is really doing and what inputs it needs.

Creating a new workflow

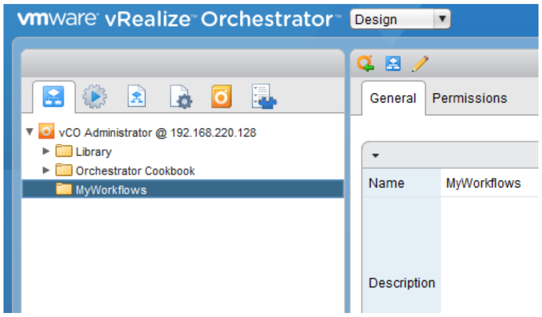
In order to build a workflow that combines two or more existing workflows, we first need to create a new empty workflow. Just placing a workflow somewhere in the library isn't really that professional. So, we will create a new folder in the library that we will use to store our workflows in. This will also help us later when we create a new package for export (see *Chapter 9*, *Packing It All Up*).

Creating a new library folder

To create a new library folder, perform the following steps:

1. In the Orchestrator client, select the **Design** mode (the drop-down menu at the top) and then select workflows, (the blue icon with white elements).
2. Right-click on the top library element, (the orange, square icon with white cycle).
3. Select **Add folder**.
4. Enter the new folder's name, such as **MyWorkflows**.

A new folder has been created.



- vCO Administrator @ 192.168.220.128
  - Library
  - Orchestrator Cookbook
  - MyWorkflows

Name: MyWorkflows

Description:

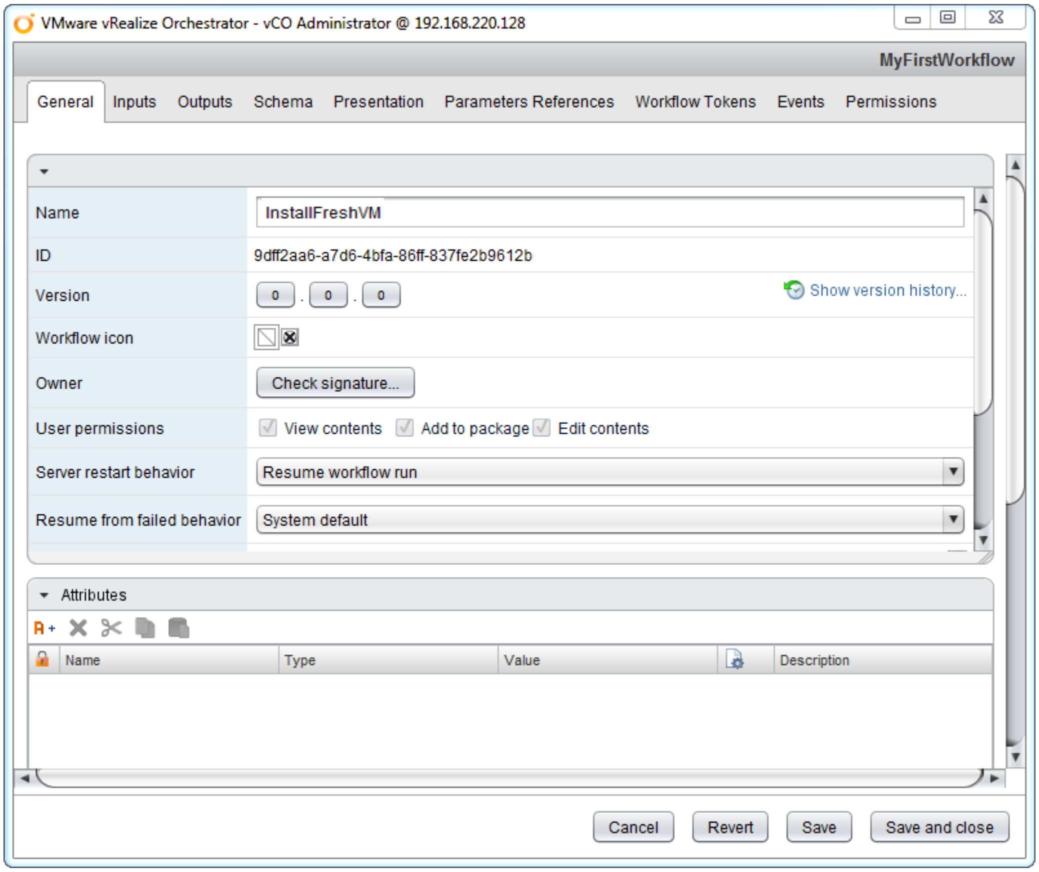
You can change the name of the folder by right-clicking on it and then selecting **Edit**.

Opening a new workflow for editing

Now that we have a folder, we can create a new empty workflow in it, simply follow these steps:

1. Right-click on the new library folder that you created and select **New workflow**.
2. You are now asked to give the workflow a name, such as **InstallFreshVM**.

The new workflow opens up and is now ready for editing:



VMware vRealize Orchestrator - vCO Administrator @ 192.168.220.128

MyFirstWorkflow

Name: InstallFreshVM

ID: 9dff2aa6-a7d6-4bfa-86ff-837fe2b9612b

Version: 0 . 0 . 0

Workflow icon

Owner: Check signature...

User permissions: ☑ View contents ☑ Add to package ☑ Edit contents

Server restart behavior: Resume workflow run

Resume from failed behavior: System default

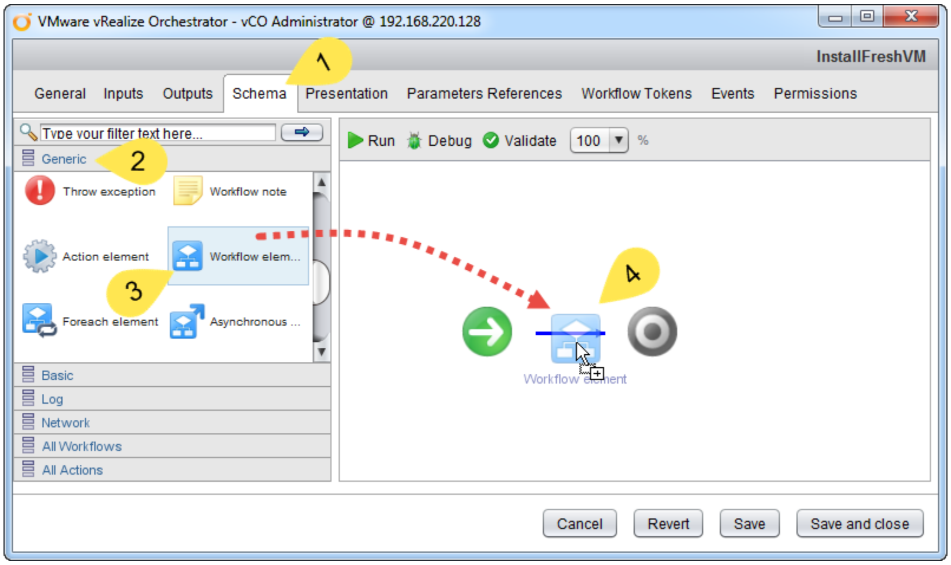
You have seen the preceding window before in *Chapter 4*, *Working with Workflows*, where we discussed what each item in this window means.

Adding workflows

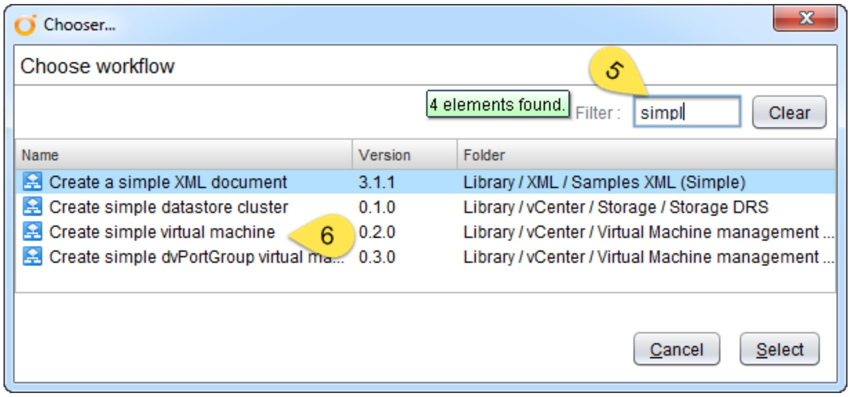
Let's add elements to our new workflow.

There are two methods that can be used to add workflows to a new workflow. We will first add the **Create simple virtual machine** workflow by searching for it and then the **Add CD-ROM** workflow by selecting it from the library.

1. In the new workflow that you opened for editing, click on **Schema**.
2. In the tool box to the left, you will find the **Generic** box already opened.
3. Scroll down and find **Workflow element**.
4. Drag the **Workflow element** icon between the start icon and the end icon, as shown in the following screenshot:



5. As soon as you release the mouse, a window will appear, where you can search for the workflow to be added. In the filter, start typing `simple`. The more you type, the fewer results are shown. Now type [space] and `vi`.



Chooser...

Choose workflow

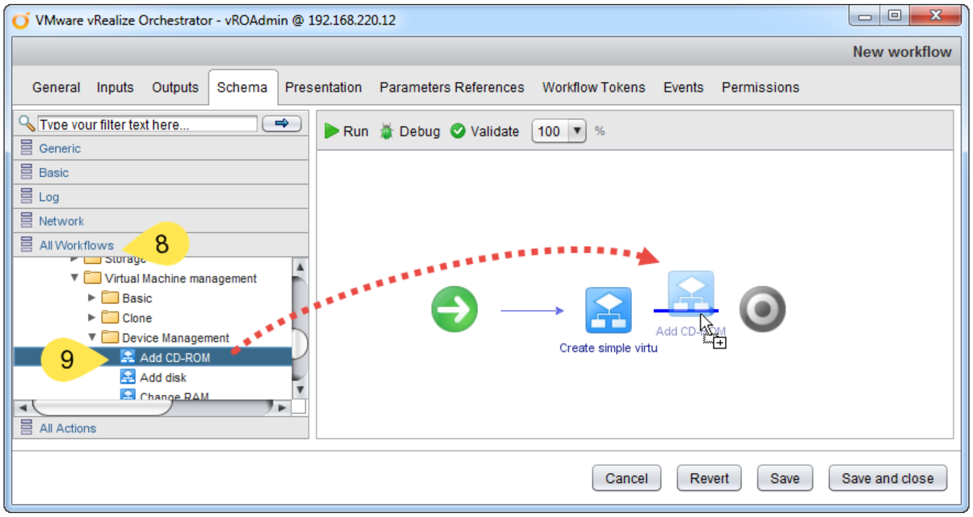
4 elements found. Filter : simpl Clear

Name	Version	Folder
Create a simple XML document	3.1.1	Library / XML / Samples XML (Simple)
Create simple datastore cluster	0.1.0	Library / vCenter / Storage / Storage DRS
Create simple virtual machine	0.2.0	Library / vCenter / Virtual Machine management ...
Create simple dvPortGroup virtual ma...	0.3.0	Library / vCenter / Virtual Machine management ...

6. This should result in only one workflow—**Create simple virtual machine**.
7. Click on **Select**.

   The workflow element has been added to our new workflow. Let's add the **Add CD-ROM** workflow a different way.

8. In the toolbox to the right, select **All Workflows**.
9. As you can see, the workflow tree library is shown. Dig down to **Library** | **vCenter** | **Virtual Machine management** | **Device Management** and then drag the **Add CD-ROM** workflow onto the schema just after the **Create simple virtual machine** workflow.

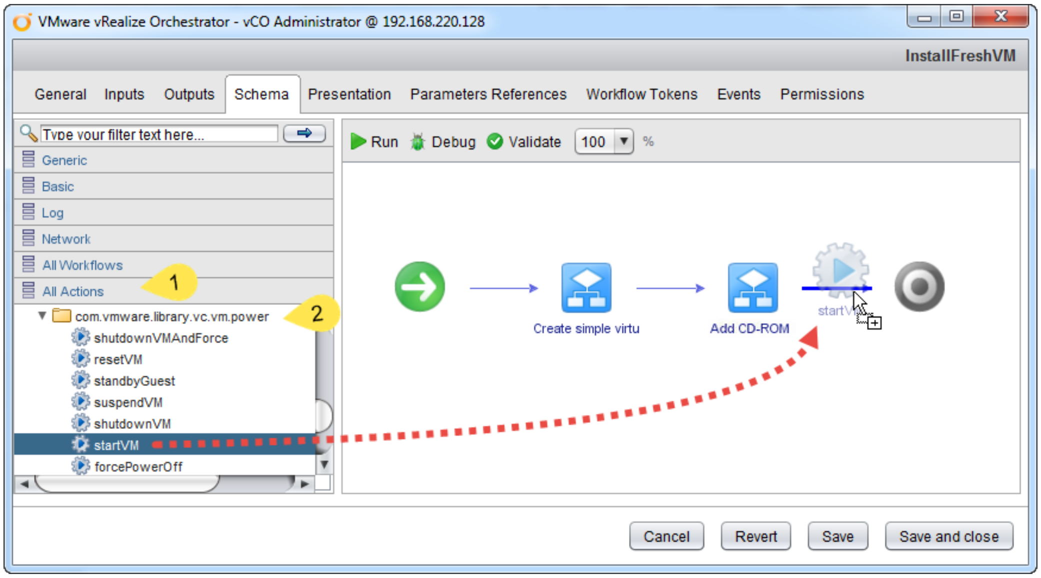


Adding actions to a workflow

Actions are a different type of workflow. They are what programmers call functions. In Orchestrator, an action is written in JavaScript and can only have one out-parameter, whereas a workflow can have more than one. We will have a closer look at actions in the next chapter when we also look at JavaScript.

Adding an action also works via two methods, as is the case with workflows. Now, we will add an action to our workflow that starts up (powers on) a VM:

1. In **Schema**, click on the **All Actions** toolbox tab.
2. Look for an action module (the folder symbol) called **com.vmware.libary.vc.vm.power**. Actions are not sorted in a library view like workflows, but come in modules. As you can see from the name and by browsing through the actions, the modules are sorted and named to simulate a treelike structure.
3. Drag the **StartVM** action into the schema and place it after the **Add CD-ROM** workflow element:



We can also use the **Action element** from the **Generic** toolbox, which works by searching for the name of an action, just like we did with the workflows.

The big question is, why should we add an action? Surely there is a workflow that powers on a VM? Well, there is one such workflow, and we will work with it at the end of this chapter when we talk about altering existing workflows. The problem with the **Start virtual machine and wait** workflow is that it waits for the VMware tools to start up inside the VM. However, we are creating a brand new, empty VM. So, there are no VMware tools yet. Therefore, we have to use the action.

> There are lots of actions that come with Orchestrator. It's a good idea to browse the library actions.

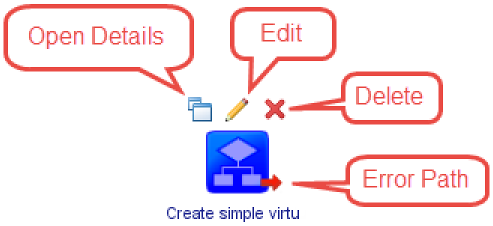
Now that we have added two workflows and one action, we have all the elements of our new workflow together, and we can now go ahead and assign all of their parameters.

Assigning parameters

As you know from *Chapter 4*, *Working with Workflows*, each workflow has attributes as well as input and output parameters. To make our new workflow actually work, we need to assign the input and output parameters to our newly created workflow.

Just be mindful of the fact that this process involves a pretty long set of steps, but persistence definitely pays off.

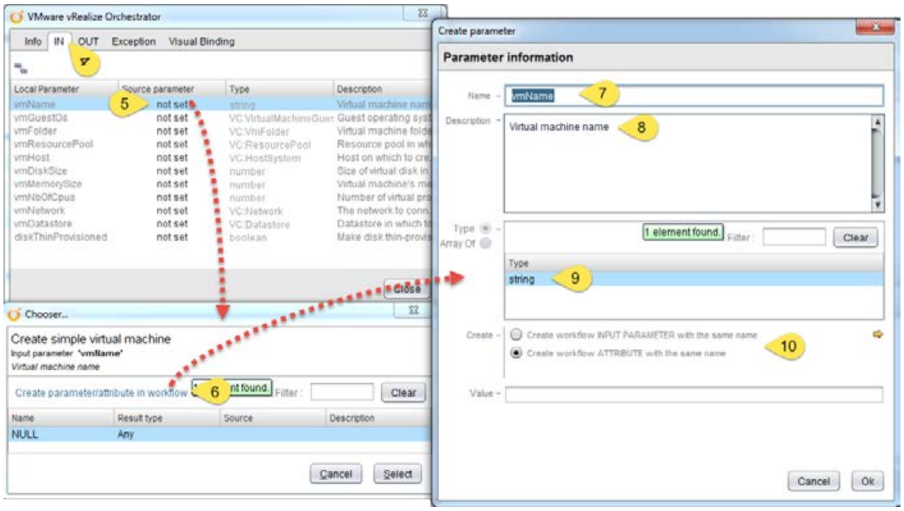
1. We start where we left off.
2. In the workflow, hover your mouse over the first workflow element. You will see that the following four icons are displayed:



The **Delete** icon deletes this element from the schema. The **Error Path** icon will be discussed later in *Chapter 8*, *Errors, Logs, and Debug Mode*. The **Open Details** icon displays the workflow in a separate window so that you can check what it actually does. The **Edit** icon is the one that we will be using.

> In Orchestrator 6, you can also use the "bulk import" parameters as soon as you click on a workflow element. We will have a look at that at the end of this section.

3. Click on the **Edit** icon.
4. A new window will open. Click on the **IN** tab to select the input parameter.
5. Each input parameter needs to be assigned. If a parameter is not assigned, **not set** is written next to its name. Now, click on **not set** next to the **vmName** parameter.
6. A new window will open up and show all the possible values that already exist. As this workflow is totally empty, only **NULL** is shown. Click on **Create parameter/attribute in workflow**.
7. Another window opens and lets you create a new attribute inside your new workflow. First, you can (if you want) change the name. However, I normally go with the default name.
8. A description is a good idea as it explains what a parameter is for.
9. Now we come to the type of the parameter. We will have a closer look at all the types a bit later in this chapter. Nothing to do here, we will use the default.
10. You now have two choices. You can make this parameter an input parameter for our new workflow, or you can keep the value fixed. As an input parameter, the user will be asked for it when you start the new workflow. As an attribute, the value is fixed. Choose **Create workflow INPUT PARAMETER with the same name**.



Local Parameter	Source parameter	Type	Description
vmName	not set	string	Virtual machine nam
vmGuestOs	not set	VC:VirtualMachineGue	Guest operating syst
vmFolder	not set	VC:VmFolder	Virtual machine folde
vmResourcePool	not set	VC:ResourcePool	Resource pool in wh
vmHost	not set	VC:HostSystem	Host on which to cre
vmDiskSize	not set	number	Size of virtual disk in
vmMemorySize	not set	number	Virtual machine's me
vmNbOfCpus	not set	number	Number of virtual pro
vmNetwork	not set	VC:Network	The network to conn
vmDatastore	not set	VC:Datastore	Datastore in which to
diskThinProvisioned	not set	boolean	Make disk thin-provis

Create simple virtual machine
Input parameter 'vmName'
Virtual machine name

Create parameter/attribute in workflow
nt found. Filter : Clear

Name	Result type	Source	Description
NULL	Any		

Parameter information

Name - vmName

Description - Virtual machine name

Type ● - Array Of ○

1 element found. Filter : Clear

Type
string

Create - ○ Create workflow INPUT PARAMETER with the same name
● Create workflow ATTRIBUTE with the same name

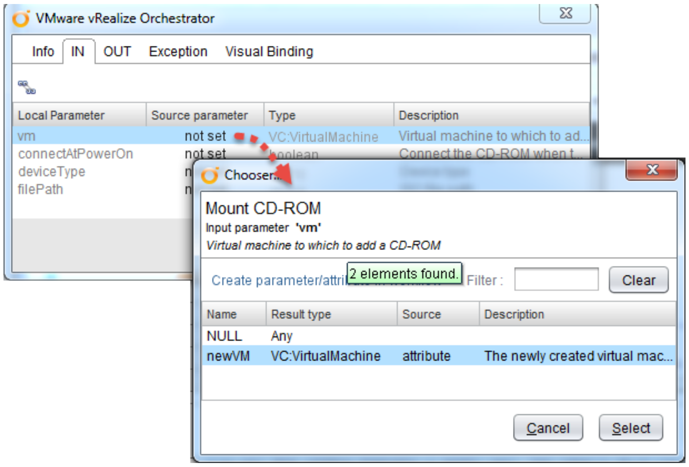
Value -

11. Click on **Ok** to return to the list of all the parameters.
12. Now, click on the **not set** that is next to **vmGuestOs**.
13. Again, we will create a new parameter like we did before. But this time, we will choose to keep this one fixed. Choose **Create workflow ATTRIBUTE with the same name**.
14. Select an operating system for your workflow by clicking on **not set** next to **Value** and choosing one from the list, such as **windows7_64Guest**. Remember to press *Enter* in the filter. Otherwise, you won't see anything.
15. Click on **Ok** to return to the list of all the parameters.
16. Now, click on **not set** next to **vmHost**.
17. As we saw in the last chapter, we don't need this parameter. So, just click on the existing **NULL**. Then, click on **Select**.
18. Now that we learned the basics lets speed this up a bit. For the first workflow element, we will define the parameters in the following way:

Parameter	Type	Value
vmName	INPUT	
vmGuestOs	Value	windows7_64Guest
vmFolder	Value	Select a folder in your infrastructure
vmResourcePool	Value	Select a resource pool in your infrastructure
vmHost	NULL	We don't need to fill anything into this parameter
vmDiskSize	Value	Select some number of GBs for this disk
vmMemorySize	INPUT	
vmNbOfCpus	INPUT	
vmNetwork	Value	Select a VM network from your inventory
vmDatastore	Value	Select a datastore in the your infrastructure
diskThinProvisioned	Value	Select **Yes** to allow for thin provisioning

19. Now that all the input parameters are defined, let's look at the out parameter. Click on **OUT**.
20. There is only one out parameter, **newVM**. This parameter will hold the ID of the new VM, and we will reuse it in the workflow.
21. Click on **not set** next to **newVM** and then create a new parameter. Select a value, but leave the value as **not set**. Click on **OK**. This will create an attribute that is empty, so that the workflow can fill it in later.

22. Now that we have finished the first workflow element, let's go to the second one, **Add CD-ROM**. Edit the workflow element and click on **IN** to see all the input parameters.
23. The first parameter is called **vm**, whose type is the same as that of the **newVM** output parameter. Click on **not set**, select the already existing **newVM** parameter, and click on **Select**.



Local Parameter	Source parameter	Type	Description
vm	not set	VC:VirtualMachine	Virtual machine to which to ad...
connectAtPowerOn	not set	boolean	Connect the CD-ROM when t...
deviceType	n		
filePath	n		

Mount CD-ROM
Input parameter **'vm'**
Virtual machine to which to add a CD-ROM

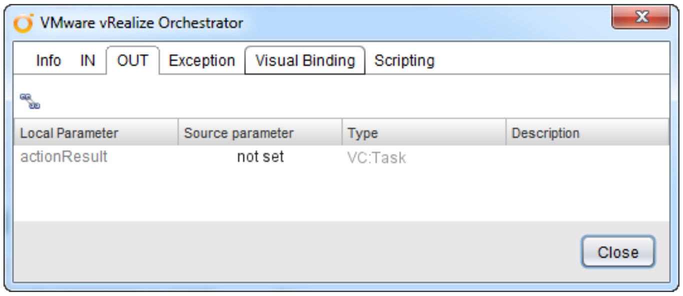
Name	Result type	Source	Description
NULL	Any		
newVM	VC:VirtualMachine	attribute	The newly created virtual mac...

24. Fill the rest of the parameters, as follows:

Parameter	Type	Value
vm	existing	newVM
ConnectAtPowerOn	Value	Yes
deviceType	Value	Datastore ISO file
filePath	Value	[Datastore name] folder/file

25. Have a look at the out-parameter, as there isn't one, we are done with this element and can move on.
26. Edit the action element and select **IN**.
27. There are only two input parameters. **vm** will take the **newVM** attribute. The other is **host**, and the description says that it's optional. So, set it to **NULL**.

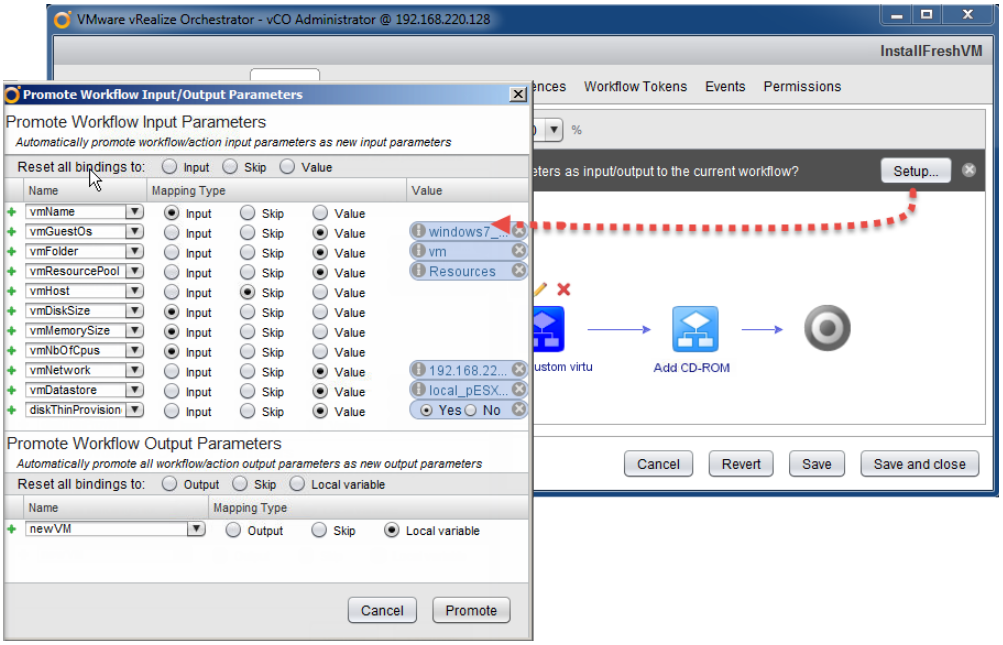
28. Click on **OUT**. Every action has the same out-parameter called **actionResult**. Only the type changes. We are not interested in the return value. So, we will just set it to **NULL**.



Local Parameter	Source parameter	Type	Description
actionResult	not set	VC:Task	

We are done. We have assigned all the parameters of all the elements in our schema. We learned how to assign a parameter as an input and set a value for it or ignore it by setting it to NULL. We also learned about the special output parameter that actions have.

Since Orchestrator 5.5, things are easier. When you click on the **Edit** icon, note the grey information bar with a **Setup** button. By clicking on it, you will now be able to quickly assign all the parameters in one window. Selecting **Skip** will set the parameter to NULL.


VMware vRealize Orchestrator - vCO Administrator @ 192.168.220.128
InstallFreshVM
ences
Workflow Tokens
Events
Permissions
%
eters as input/output to the current workflow?
Setup...
ustom virtu
Add CD-ROM
Cancel
Revert
Save
Save and close
Promote Workflow Input/Output Parameters

Promote Workflow Input Parameters

Automatically promote workflow/action input parameters as new input parameters

Reset all bindings to: ○ Input ○ Skip ○ Value

Name	Mapping Type			Value
vmName	● Input	○ Skip	○ Value	
vmGuestOs	○ Input	○ Skip	● Value	windows7_...
vmFolder	○ Input	○ Skip	● Value	vm
vmResourcePool	○ Input	○ Skip	● Value	Resources
vmHost	○ Input	● Skip	○ Value	
vmDiskSize	● Input	○ Skip	○ Value	
vmMemorySize	● Input	○ Skip	○ Value	
vmNbOfCpus	● Input	○ Skip	○ Value	
vmNetwork	○ Input	○ Skip	● Value	192.168.22...
vmDatastore	○ Input	○ Skip	● Value	local_pESX...
diskThinProvision	○ Input	○ Skip	● Value	● Yes ○ No

Promote Workflow Output Parameters

Automatically promote all workflow/action output parameters as new output parameters

Reset all bindings to: ○ Output ○ Skip ○ Local variable

Name	Mapping Type		
newVM	○ Output	○ Skip	● Local variable

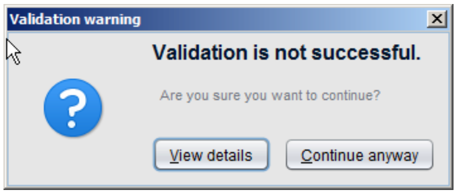
Cancel Promote

Workflow verification

Now that we have assigned all the parameters of the workflow element to the new workflow, we can finally save it and give it a go.

To save the workflow, click on **Save and close**. If you have created the workflow correctly, Orchestrator will ask whether you want to create a new version of this workflow. In this case, skip to the next section.

If you did something wrong (and you may want to try this), you will be presented with a window that looks like this:



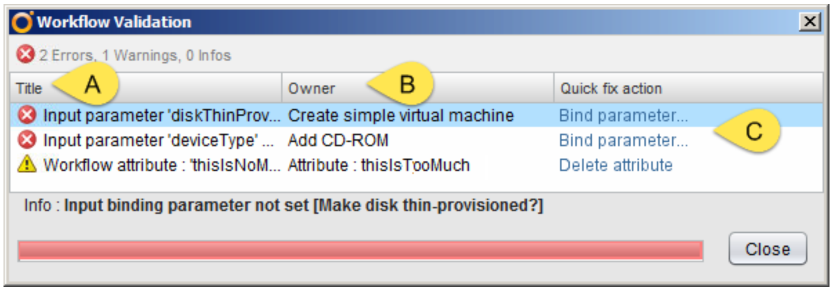
Validation warning

Validation is not successful.

Are you sure you want to continue?

When Orchestrator saves a workflow, it checks whether all the parameters are assigned, whether there are any empty scripts, and so on. If this is the case, you will get a warning. You can ignore this warning by clicking on **Continue anyway**. This will save the workflow in the current state. However, you will not be able to execute it.

If you click on **View details**, Orchestrator will present you with a window that shows its findings.



Workflow Validation

2 Errors, 1 Warnings, 0 Infos

Title	Owner	Quick fix action
Input parameter 'diskThinProv...	Create simple virtual machine	Bind parameter...
Input parameter 'deviceType' ...	Add CD-ROM	Bind parameter...
Workflow attribute : 'thisIsNoM...	Attribute : thisIsTooMuch	Delete attribute

Info : **Input binding parameter not set [Make disk thin-provisioned?]**

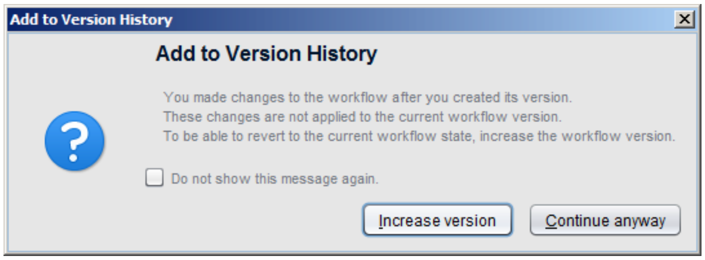
In the preceding example, I didn't assign the **deviceType** and **diskThinProvisioned** parameters. I also added a **thisIsTooMuch** attribute, which isn't used. In section **A**, we find the error that Orchestrator found. In section **B**, we see the workflow element in which this error occurred. In section **C**, you are presented with a possibility of fixing the problem. Clicking on **Bind parameter** will bring you directly to the window where you can create a new parameter or choose from an existing one.

The **Delete attribute** option should be used with caution. You may not want to delete the parameter. You may just have forgotten to assign it to something. So, check it out first.

You can also run the validation at any point by clicking on **Validation Validate** at the top of the schema.

Workflow history

When you save and close a workflow, you are asked whether you would like to **Add to Version History**. What this means is that Orchestrator can keep different histories of a workflow.

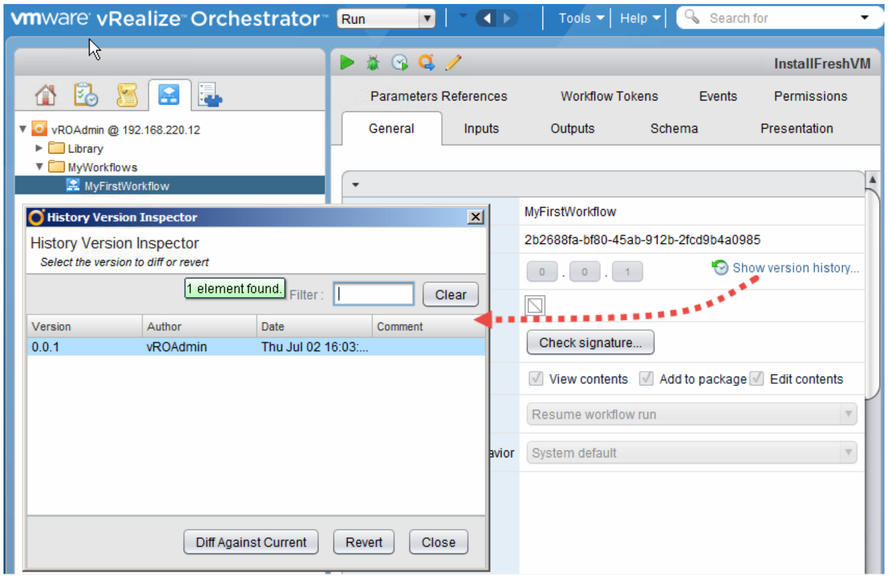

Add to Version History

Add to Version History

You made changes to the workflow after you created its version.
These changes are not applied to the current workflow version.
To be able to revert to the current workflow state, increase the workflow version.

□ Do not show this message again.

Selecting **Increase version** will create a new version of the workflow. You can see the current versions of a workflow in the **General** section. By clicking on **Show version history**, you can see all the existing versions of this workflow.



History Version Inspector

History Version Inspector

Select the version to diff or revert

1 element found. Filter :

Version	Author	Date	Comment
0.0.1	vROAdmin	Thu Jul 02 16:03:...	

MyFirstWorkflow

2b2688fa-bf80-45ab-912b-2fcd9b4a0985

0 . 0 . 1

☑ View contents ☑ Add to package ☑ Edit contents

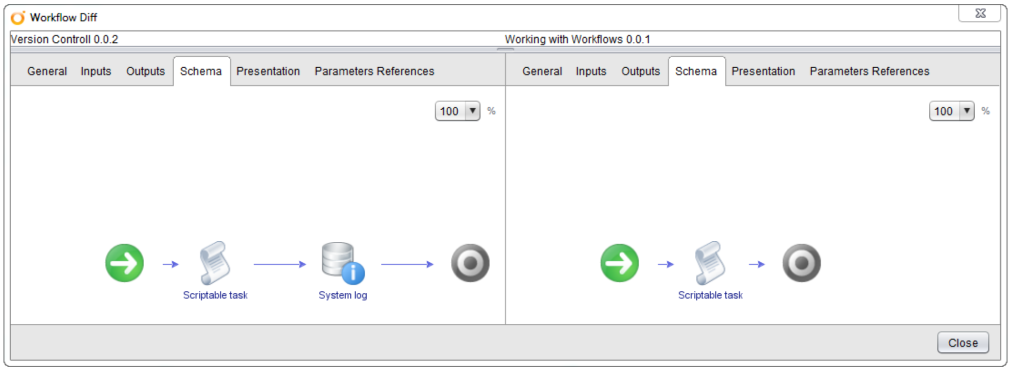
Resume workflow run

System default

Comparing the versions

You can use this function to compare the versions of a workflow.

1. On the workflow's **General** tab, click on **Show version history**.
2. Select the version that you would like to compare the present one against and click on **Diff Against Current**.
3. A window will pop up and show both the versions next to each other. Resize the window as required.
4. When you are finished, click on **Close**.



Reverting to an older version

Perform the following steps to revert to an older version of the workflow:

1. Click on **Show version history**.
2. Select the version that you'd like to revert to and click on **Revert**.

Your workflow is now of an earlier version. However, the newer version is still available. You can also *revert* to a newer version if you wish.

Useful workflow operations

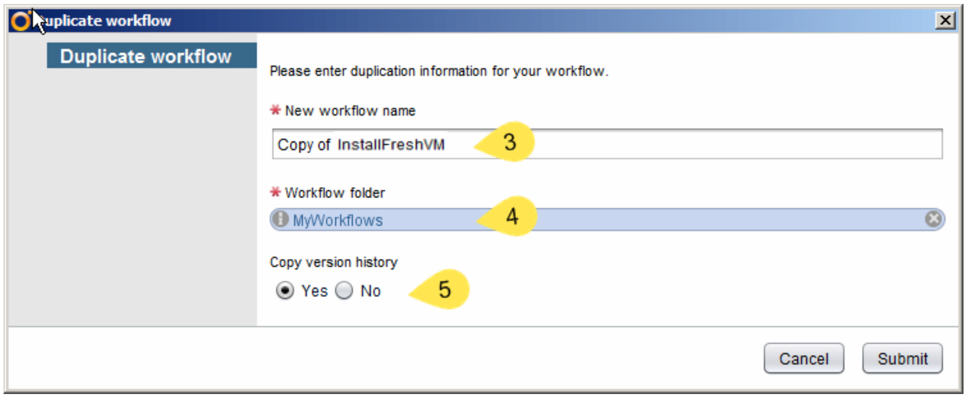
Here are a couple of useful operations that you should know when creating or editing workflows. We will go through them via an example.

Copying a workflow

Creating a copy of an existing workflow is something that you will do quite a lot. For example, you may want to create a copy of an existing workflow and modify it according to your needs, which is what we will do in a later section in this chapter.

Let's copy our first workflow by performing the following steps:

1. In the Orchestrator Client, select the workflow that we just created.
2. Right-click on the workflow and select **Duplicate workflow**.
3. Give it a new name or keep the default one, which will always be **Copy of [Original workflow name]**.
4. Select the folder in which you want to place the copy of your workflow. To do so, just click on the name that is displayed and then search for the folder that you want.
5. If you also decide to copy the history of this workflow, the new workflow will contain all the old workflow's events as well as its history (see above).



Duplicate workflow

Please enter duplication information for your workflow.

* New workflow name: Copy of InstallFreshVM

* Workflow folder: MyWorkflows

Copy version history: ● Yes ○ No

6. Click on **Submit** to create the copy.

The new copy will now be created and placed in the target folder that you specified.

Renaming a workflow element

Each workflow element can be renamed in the schema. This isn't about renaming a workflow or an action. It's just about the text that is shown below the element itself.

1. Click on the workflow that we just duplicated.
2. Right-click and select **Edit**. Alternatively, you can click on the **edit Icon** at the top of the workflow.
3. Change to the schema.
4. Double click on a workflow element.
5. Now, you can change the workflow element's name.
6. When you're finished, just press *Enter*.
7. Don't exit and save yet. Go to the next section.

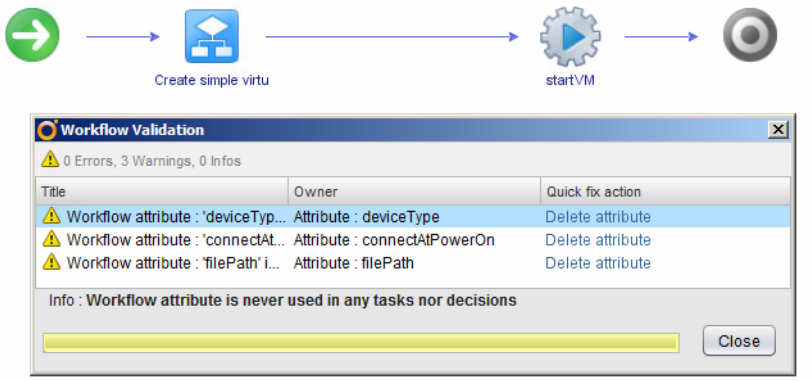
The renaming of the elements is mostly done when you use other workflow elements from the toolbox, such as decisions or scriptable tasks.

Deleting an element

If you want to delete a workflow element from the schema, things are slightly more complicated. Deleting an element is easy, but don't underestimate the cleanup.

To understand this, let's go through an example where we delete an element:

1. We will continue from the point where we are editing the workflow.
2. We are now going to delete the **Add CD-ROM** workflow element. Click on it and then select the red **X**.
3. The element is deleted. However, if you now click on **Validate**, you will find that the parameters of the workflow still exist.
4. Don't exit and save yet. Go to the next section.



⚠ 0 Errors, 3 Warnings, 0 Infos

Title	Owner	Quick fix action
⚠ Workflow attribute : 'deviceTyp...	Attribute : deviceType	Delete attribute
⚠ Workflow attribute : 'connectAt...	Attribute : connectAtPowerOn	Delete attribute
⚠ Workflow attribute : 'filePath' i...	Attribute : filePath	Delete attribute

Info : **Workflow attribute is never used in any tasks nor decisions**

The important thing when deleting workflow elements is to understand what parameters you can delete as well the elements that you have to keep.

Exit without saving and reverting

If you worked on a workflow and you found out that you did something that you didn't intend to do and want to go back to the last version, there are two ways to do so. First, you can exit the edit mode without saving the changes. The other way involves clicking on **Revert**.

As we have just deleted an element, we can try this out:

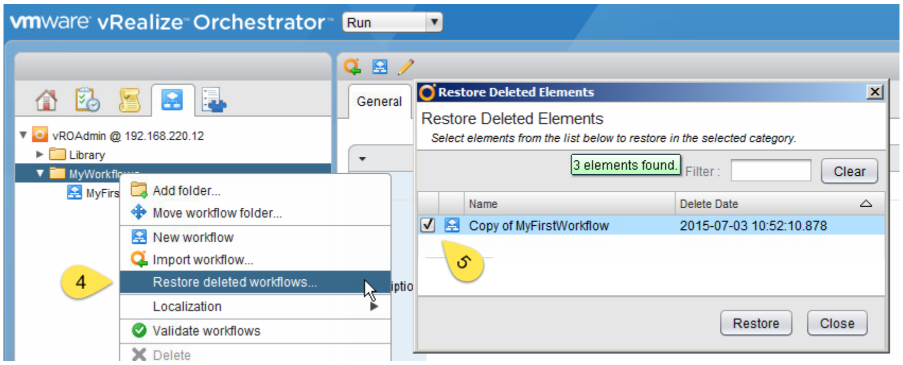
1. While editing the workflow, make sure that you have deleted an element.
2. Click on **Revert**. You will see that the element you deleted has reappeared. Note that this happened without you being asked to confirm it. So, you have to be a bit careful.
3. Let's dig deeper. Rename a workflow element.
4. After renaming it, click on **Save**.
5. Now, delete a workflow element and click on **Revert**. Note that the deletion has been remedied, but not the renaming. You probably have already noted that you haven't been asked about saving either. So again, be careful what you click on.
6. Delete an element again, but this time, click on **Cancel** and then **Close anyway**.
7. You will see that you have exited the editing of the workflow and you're now looking at all the changes since the last save have been discarded.

Deleting and undeleting a workflow

Last but not least, we will clean up again by deleting the workflow, as follows:

1. Make sure that you exit the workflow editing mode.
2. Right-click on the workflow that we copied, select **Delete**, and confirm the delete operation.
3. The workflow will then be deleted. Check your library.
4. Now, click on the folder that contained the workflow that you deleted and select **Restore deleted workflows**.

5. A new window will open, which will show all the workflows that you can restore. Select the one that you want and click on **Restore**.



Restore Deleted Elements

Select elements from the list below to restore in the selected category.

3 elements found.

	Name	Delete Date
☑	Copy of MyFirstWorkflow	2015-07-03 10:52:10.878

This concludes our little exercise.

Workflow parameters

Now that we understand some of the basics of how Orchestrator programming works, we will look at the different types of parameters that exist.

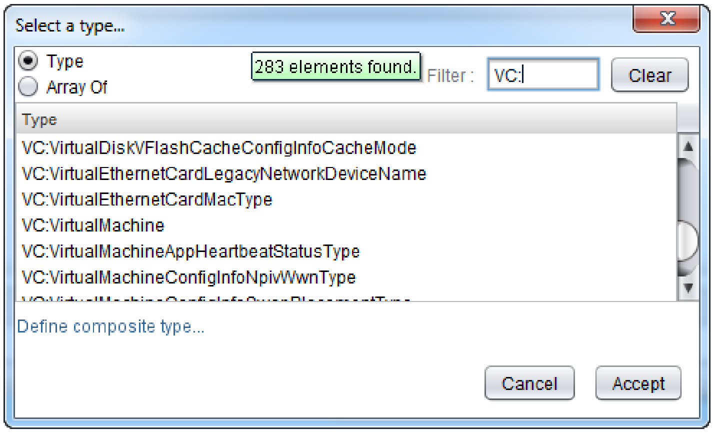
Parameter types

There are many variable types that are already implemented in Orchestrator out of the box, but the basic variable types are as follows:

Variable type	Description
any	This can contain *any* content. It is used to carry variables to the other elements that are not defined in the Orchestrator GUI, such as XML. Note that **Any** should only be used if nothing else will do, as it has been known to mishandle some content such as complex variables.
boolean	This has only two values, either `true` or `false`. However, Orchestrator uses **Yes** and **No** in the GUI.
credential	This contains a username and password. The password is encrypted.
date	This is used to store the date or time in the JavaScript format.
number	This contains only numbers, which can be integers or real numbers. Everything is stored as floats in Orchestrator.

Variable type	Description
secure string	When entering values, *s will be shown instead of characters. The value is plain text and visible to the workflow developer, but it is encrypted when the workflow runtime information is stored in the database.
encrypted string	This is like the secure string. However, the value is always encrypted.
string	This can contain any characters.
NULL	This is not really a type, but defining a variable as NULL means that anything that is put into it will be discarded.

In addition to the base types, each plug-in will install its own type. These types are identified by their prefixes. For example, types that come with the vCenter plug-in have the **VC:** prefix, and types from the SSH plug-in have the **SSH:** prefix. When you click on the type of a parameter, you can see all the existing types, as follows:



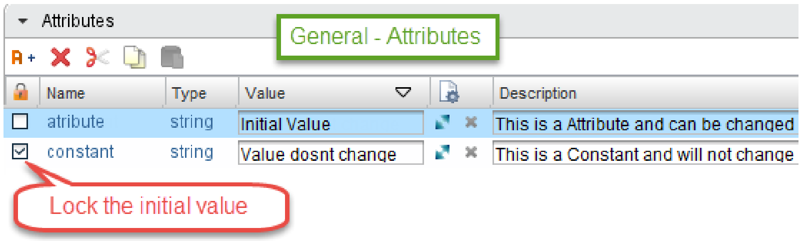
VC:VirtualDiskVFlashCacheConfigInfoCacheMode
VC:VirtualEthernetCardLegacyNetworkDeviceName
VC:VirtualEthernetCardMacType
VC:VirtualMachine
VC:VirtualMachineAppHeartbeatStatusType
VC:VirtualMachineConfigInfoNpivWwnType

Each workflow can have a parameter in three different areas. Depending on where they are located, they are called slightly differently.

Parameters in the General section

A parameter in the **General** section is called an **attribute**. An attribute is accessible throughout the whole workflow. However, it is not accessible outside the workflow. An attribute can have an initial value at the start of a workflow, but it can also be changed at any stage.

Attributes are mostly used for two things—as a constant (defined once and not changed), or as a way to exchange a value between two workflow elements. You can lock an attribute (see the following screenshot) to make sure that the initial value can't be changed:



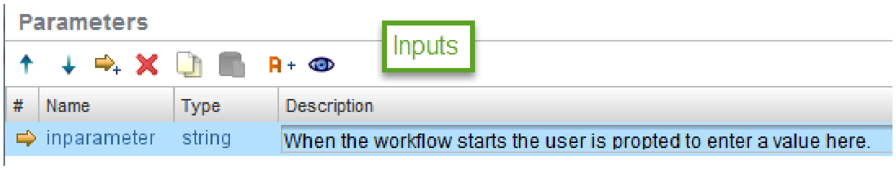
General - Attributes

🔒	Name	Type	Value		Description
□	atribute	string	Initial Value		This is a Attribute and can be changed
☑	constant	string	Value dosnt change		This is a Constant and will not change

Lock the initial value

Variables in the input section

A variable that is present in the input section is called an **INPUT parameter**. The content of an input variable is defined at runtime and entered by the user. Input variables cannot be changed during the workflow execution directly, as you cannot assign an in-parameter as the output of a workflow element.

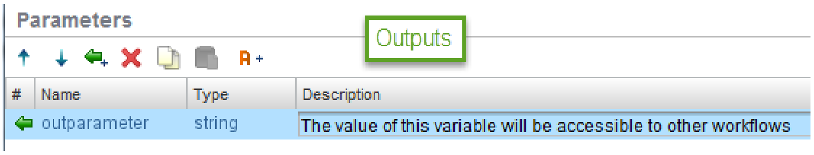


Inputs

#	Name	Type	Description
➪	inparameter	string	When the workflow starts the user is propted to enter a value here.

Variables in the output section

A variable in the output section is called an **OUTPUT parameter**. The content of an output variable can be defined within the workflow and is available to the other elements when the workflow has finished.



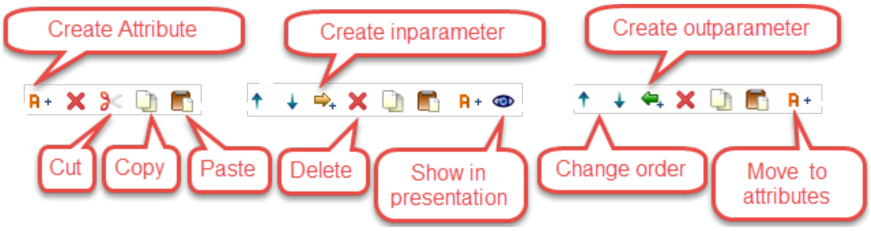
Outputs

#	Name	Type	Description
⇦	outparameter	string	The value of this variable will be accessible to other workflows

Working with parameters and attributes

For each parameter section, there are several action icons. Most of them are repetitive and clear to understand, such as **Delete**, **Cut**, **Copy**, and **Paste**.

Each section has an icon that will create a new parameter. In the input and output section, you also have a possibility to change the order of the parameters. The order will influence the appearance in the presentation.



Moving a parameter or attribute

If you created a parameter in the wrong section, you can move it to a different section. The input and output sections have an action icon that will move a parameter to the attributes section.

In the attributes section, you need to right-click on the attribute. Then, you can move the parameter either to the input or the output section.

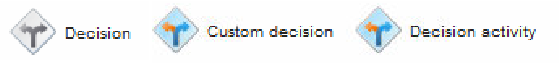
> When you move a parameter around, all of its existing bindings are lost. For example, you may have created an attribute and assigned it to a workflow element. You may then move the attribute to be an INPUT parameter. If you go back to the workflow element, you find that the assignment is set to **Not set**.
>
> This is a bug that has been reported in 5.5 and will be fixed in time.

Some more advanced programming

Now that we have gone over the basics of programming, let's go a little further. We will use one of the items of the toolbox, the decision element. We will work with some other elements later on in this book, but we won't be able to cover all of them. All the elements are explained with examples in the *vRealize Orchestrator Cookbook*.

Adding a decision element

A decision element is what programmers call an **If-clause**. It checks for a condition and then points to one or the other way. A decision uses a test to check whether a certain condition has been met. The result is always `true` or `false`. The following three types of decision elements exist in Orchestrator:



The differences between the three decision elements are the following:

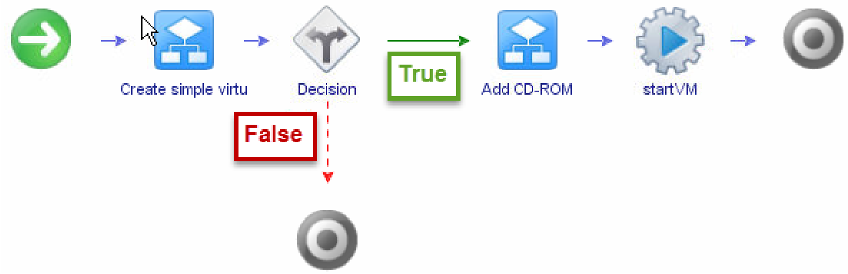
- A (basic) **Decision** can only check whether a single parameter has fulfilled a certain condition. There are simple checks that depend on the type of the parameter. For instance, it can check whether a number is greater than a set value or whether a string contains a certain value or the power state of a VM.
- A **Custom decision** enables the programmer to write a JavaScript to determine whether a condition has been met. This check can be as complex as the programmer likes.
- The **Decision activity** checks the outcome (the OUTPUT parameter) of a workflow against a certain simple condition like the (basic) **Decision**.

In the next chapter about JavaScript, we will dive a bit deeper into conditions and these checks.

We will now use the (basic) **Decision** element to modify our first workflow by performing the following steps:

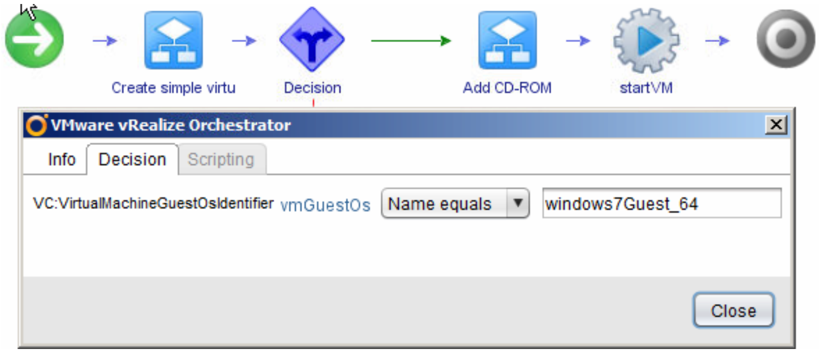
1. By using the Orchestrator Client, copy our first workflow or create a new version as shown earlier.
2. Now, edit the copied workflow.
3. Drag the **Decision** element from the **Generic** toolbox into the workflow between the **Create simple virtual machine** and **Add CD-ROM** workflows.

4. You will see that after inserting this element, a new end point has been created, and there is a green and a red line. The green line represents the flow of the program if the result of decision is **True**, whilst the red is the **False** result.



5. We will now implement a check in the **Decision** element. The aim is to insert a CD-ROM image depending on the chosen operation system. To do this, we first need to move the **vmGuestOs** parameter from its current place in the attributes to the INPUT parameters so that we can choose a different OS.
6. Go to the attributes, right-click on the **vmGuestOs** parameter, and select **Move as INPUT parameter**.
7. Now, go back to the schema and edit the **Create simple virtual machine workflow** element. You will see that the **vmGuestOs** is not set anymore. We discussed this behavior before.
8. Set the **vmGuestOs** as the INPUT parameter again.
9. Now, we will set up the **Decision** element. Edit the **Decision** element and then click on **Decision**.
10. The parameter that should be checked is currently not defined and shows **Not set (NULL)**. Just click on it.
11. A window will open up and display all the attributes and INPUT parameters of the workflow. Choose **vmGuestOs**.

12. Then, choose the **Name equals** from the drop-down menu and enter `windows7Guest_64` in the text box next to it, as follows:



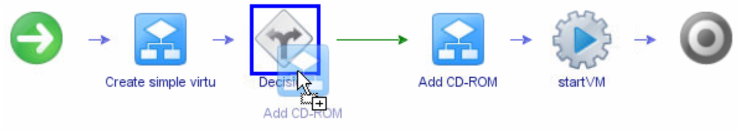
VC:VirtualMachineGuestOsIdentifier vmGuestOs Name equals windows7Guest_64

We now have a decision. When the workflow runs and someone selects the Windows 7 64-bit operating system, the workflow will follow the green line. If someone selects anything else, the workflow will end. This is not really ideal, but we are not finished yet.

Working with connection lines

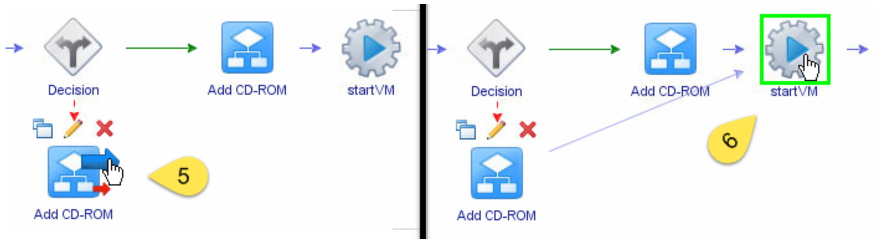
Each element in a workflow is connected by a line to another. We will now modify our workflow to make it a bit more useful and learn how to connect workflow elements as well as break them.

1. When editing the workflow, select the line between the **Decision** element and the end point by simply clicking on it. It should get bolder.
2. Now, press *Delete*, and the red line as well as the end element will disappear.
3. Drag another **Add CD-ROM** workflow directly onto the **Decision** element, as follows:



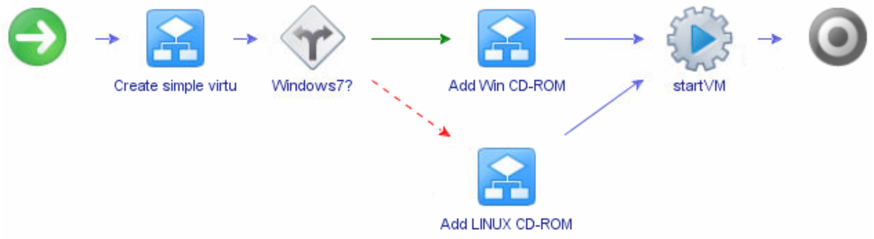
4. The new **Add CD-ROM** workflow element will now be automatically connected to the **Decision** element by using a red (false) arrow. This is because the green (true) arrow is already connected.

5. Hover your mouse over the new **Add CD-ROM** element. You will see that a blue arrow and a red arrow appear. Hover over the blue one until it becomes bigger.
6. Drag the blue line onto the **startVM** action, as follows:



We have now connected all the elements of this workflow as well as created a more meaningful false branch. Let's clean up and make it not only prettier, but also more readable.

7. Drag the second **Add CD-ROM** workflow element underneath the other one.
8. Rename the **Decision** element to something that resembles the check that you perform, such as **Windows7?**.
9. Rename the **Add CD-ROM** workflows to clarify what the difference is, such as **Add Win CD-ROM** and **Add LINUX CD-ROM**.



10. Now, we need to assign parameters to the new **Add CD-ROM** workflow element. Edit the workflow and assign the parameters, as follows:

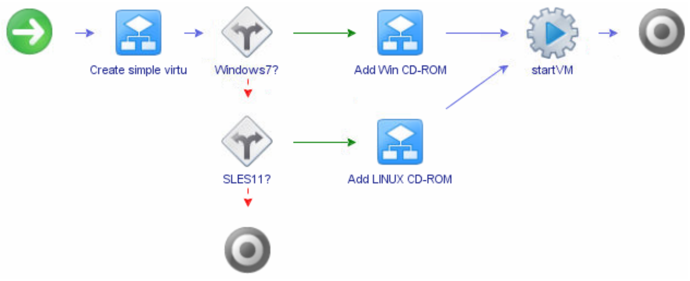
Parameter	**Type**	**Value**
vm	existing	newVM
ConnectAtPowerOn	existing	ConnectAtPowerOn
deviceType	existing	deviceType
filePath	Value	[Datastore name] folder/file of a different ISO image

This concludes our little example for the moment. We will come back to it later and improve it further.

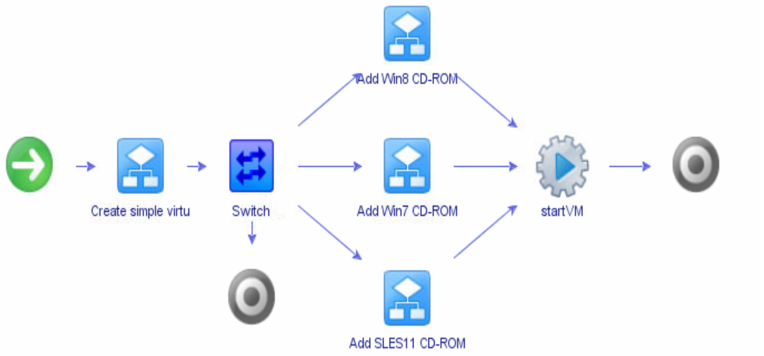
Things you might like to try out

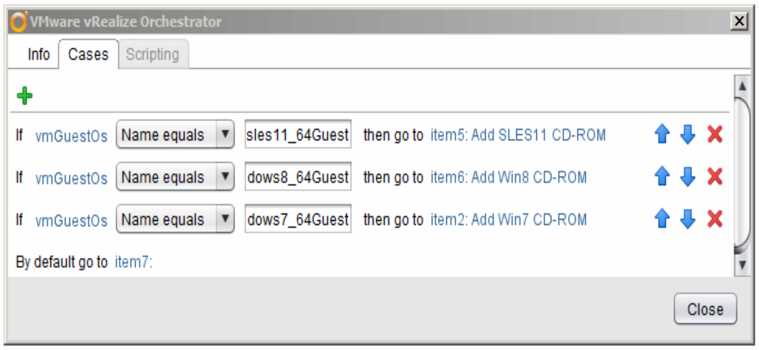
Here are some ideas that you can try out with this workflow:

- Add another **Decision** element to check whether it's a SUSE Linux.



- Explore the **Switch** element from the toolbox (needs vRO 6).





If vmGuestOs Name equals sles11_64Guest then go to item5: Add SLES11 CD-ROM

If vmGuestOs Name equals dows8_64Guest then go to item6: Add Win8 CD-ROM

If vmGuestOs Name equals dows7_64Guest then go to item2: Add Win7 CD-ROM

By default go to item7:

- Check out the different decisions that you can perform depending on the variable type. To do this, create or use parameters of the type: number, string, VM, and datastore in the **Decision** element and check the content of the drop-down menu.

Modifying an existing workflow

Copying and then modifying an existing workflow is something that is pretty common. We will go through this by using an example. We will copy and modify the **Start virtual machine and wait** workflow.

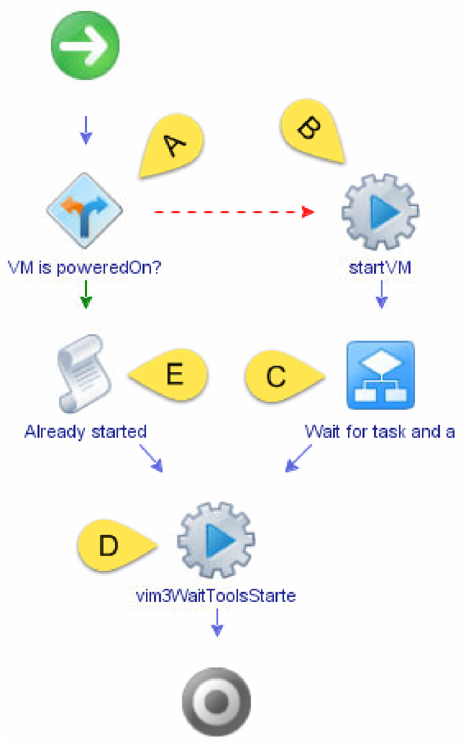
1. In the Orchestrator Client, browse to the workflow **Library | vCenter | Virtual machine management | Power Management | Start virtual machine and wait**.
2. Have a look at the **Edit** option. It's greyed out. This is because it's a Vendor supplied library workflow.
3. If we want to modify the workflow, we have to duplicate it. Go ahead and do this, as shown in the last section.

After duplicating the workflow, you can edit it.

The trick with modifying existing workflows is to understand what they are doing. This can be done simply by running the workflow and having a closer look at each element.

The decision element (**A**) checks the power status of the VM. If it is powered on, it will move to the scripting element (**E**). If the VM is not powered on, it will go to the **startVM** action (**B**). We have used this action before, and you may remember that its output is of the type VC:Task. The workflow (**C**) checks whether the VM is powered on and automatically answers vCenter questions, if there are any.

The last action (**D**) will wait for the VMware tools inside the VM to start.



This workflow is very good. However, if the VM has no operating system or VMware tools are not installed, it will show an error. So, we will modify it, as follows:

1. Click on the last action (**D**) and then delete it.
2. You will see that this automatically reconnects the lines to the end element.
3. Run validation and delete all the parameters that are left over from the action that we deleted.
4. Save and close the workflow.

If you want to, you can now replace the **startVM** action in the workflow that you created.

Summary

This was a big chapter, and we went through many topics. We started off by learning how to place workflow elements into a new workflow and thus combine them to form a new workflow. While doing this, we discussed parameters and ways to assign them. We also learned how they are built and used.

We not only discussed how to deal with rearranging, renaming, and removing elements, but also the ability to revert or exit the editing of a workflow without saving.

We had a short look at the workflow history function and improved our programming skills by using a Decision element. Last but not least, we learned how to modify the existing workflows.

In the next chapter, you will be introduced to the JavaScript language. We will learn some simple commands, deal with actions, and explore the Array variable type.

6

Advanced vRO Scripting with JavaScript

In the previous chapter, we learned how to combine existing workflows to create new workflows. Before we can go ahead and delve into the advanced aspects of programming, we need to learn some JavaScript. Orchestrator is actually a JavaScript engine that has a massive visual component.

In this chapter, we will not only use an example workflow to learn JavaScript, but also expand the **InstallFreshVM** workflow that we created in the previous chapter.

This chapter will explore the following topics:

- JavaScript – the very basics
- How to use JavaScript in Orchestrator
- The if-clause in JavaScript as well as operators and conditions
- Creating and using actions

JavaScript – the very basics

JavaScript is a script language, which means that it doesn't need to be compiled before runtime. The Java language is a "real" language, and it needs to be compiled before it can be executed. Compiling is defined as the creation of an executable (`.exe`) file from the Java-sourced code. For our purposes, Java and JavaScript have nothing to do with each other. JavaScript is what we will learn, and it's used to create the Orchestrator workflows and action.

To learn JavaScript, you can have a look at `http://www.w3schools.com/js/`.

Basic rules

Let's have a look at the rules that govern how JavaScript is written. In the next section, we will try some stuff out.

Every line ends with a semicolon

Each line of JavaScript that you write will be interpreted by Orchestrator. For Orchestrator to know that a command has ended, you need to place a semicolon (;) at its end. It's just like a full stop in a normal sentence that marks the end of a thought.

It's like the old "Let's eat, Grandpa" joke, where a missing comma leads to cannibalism ("Let's eat Grandpa").

Variables

Variables (or parameters) in JavaScript are just text and must start with a letter. After the first letter, numbers can be used. There is a convention that you should follow when creating variable names. The first letter should be in lower case. Then, capitalize the first letter of the words that follow. Examples of good variable names are `newVM`, `myNewCoolVariable`, and `vmConnection`.

Variables in JavaScript don't have to be declared, but it's good practice to do so. A declaration means that you have to define the content type of a variable type before using it. To tell Orchestrator that you want to define a variable called output that should contain numbers, you will write the following code:

```
var myOutput = new Number();
myOutput = 5;
```

As noted earlier, it's not really needed. You can also write `var myOutput = 5`, or just `myOutput = 5`. However, in some cases such as properties or arrays, you need to declare the variable properly.

We have already discussed the different types of variables in the previous chapter. Have a look at it. We will also discuss in a bit more detail some number and string operations via a few examples.

Case sensitivity

Every command or variable is case sensitive, which means that the `System.log` command is correct, but `system.log` or `System.Log` are not.

The same is true for variable names. The `myTest` variable is not the same as `MyTest` or `mytest`.

Orchestrator actually helps you a bit here. We will explore this a bit later.

Comments

Comments are text that are placed in the code and which are not interpreted for execution. Comments are extremely useful when writing your program. You can write in them what a specific line will do or where a variable comes from.

There are two kinds of comments—single-line comments and multi-line comments. You use single-line comments mostly directly after a command line, and they finish automatically with the *Enter* key. On the other hand, you use multi-line comments either to write longer text, or to comment out multiple script lines. Commenting out script lines means that you put the comment signs around a script line so that it is ignored when the program is run. This is mostly done when testing a program. Make sure that the `;` end marker is placed outside the comment directly after the command.

- Single-line comments are made by using `//`, as follows:

  ```
  System.log(test); //put the content of test into the logs
  ```

- Multiline comments begin with `/*` and end with `*/`, as follows:

  ```
  /* This is a big comment
  that goes
  over multiple lines */
  ```

Formatting

JavaScript has some agreed-upon formatting rules that should be followed. Not following them will still create a working script, but others will find it harder to read. Have a look at `http://javascript.crockford.com/code.html` to understand how JavaScript should be formatted.

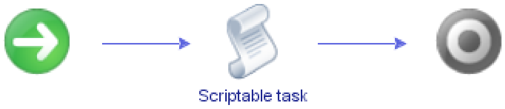
Running though some examples

Let's leave the dry theory behind and have a look at some practical examples.

Creating an example workflow

First, we will create a space where we can write some JavaScript and make Orchestrator use it, as follows:

1. Open the Orchestrator Client.
2. Go to the workflow folder that you created in the previous chapter.
3. Right-click on it and create a new workflow.
4. From the **Generic** toolbox, drop a **Scriptable task** onto the Schema.



The scriptable task element will contain our JavaScript, and it is the only element that we will need in this whole chapter.

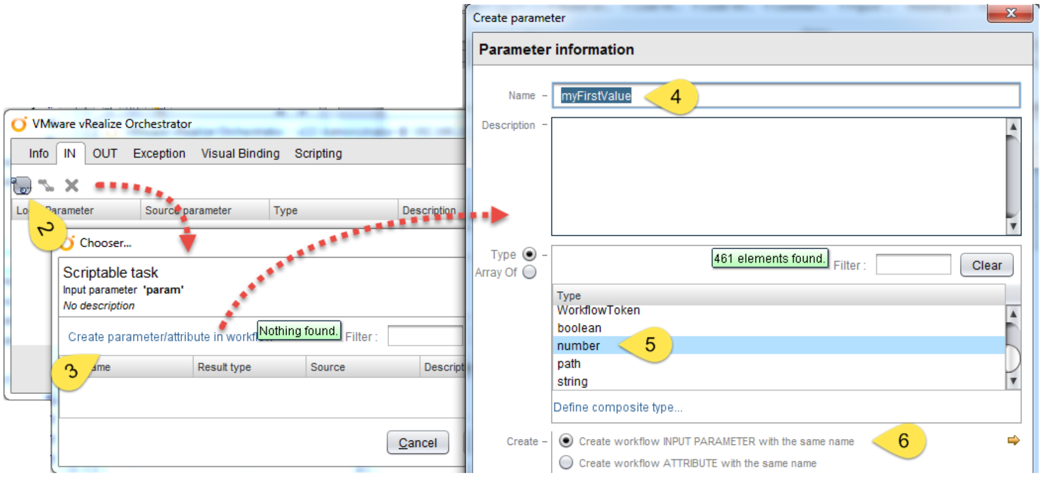
Creating new parameters

Let's start the show with some simple number operations. We will use the basic mathematical operations: `+`, `-`, `*`, and `/`.

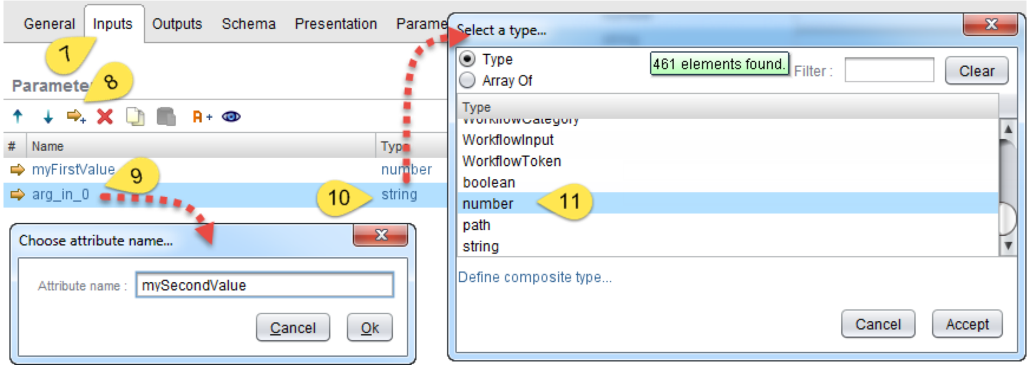
Starting from where we left off, we will now create two input parameters and one output parameter and assign all the parameters to the scriptable task. We will do this in two different ways.

1. Click on the **Scriptable task** and select **Edit**.
2. Select the **IN** tab. There are no parameters in it yet because this is a brand-new element. Click on **Bind to workflow parameter/attribute** .
3. The **Chooser** window will open, with nothing to choose from. Click on **Create parameter/attribute in workflow**.
4. Now we can give the new parameter a name.
5. We choose this parameter to be of type **Number** as we want to do some math.

6. Create the parameter as an IN-parameter and then click on **Ok**.



7. Let's add an IN-parameter another way. Click on **Inputs**.
8. Click on **Add parameter**. A new line is added.
9. Click on the new parameter named **arg_in_0**. A new window will open up. Enter a new name for this parameter, such as **mySecondValue**, and click on **OK**.
10. Now, we need to change the parameter type. Click on the default **String** type.
11. In the new window, select the **Number** type and click on **Accept**.



12. Click on **Schema** and then on the **Scriptable task** and select **Edit**.
13. Only the first parameter has been created. Click on the **Bind to workflow parameter** attribute .
14. The **Chooser** window will open and show the second parameter that we have created. Choose it and click on **Ok**.
15. Use one of the described methods to create an OUT-parameter called **myOutput** of the **Number** type.

In this chapter and the one after this, you will be asked to create parameters and attributes. To do this, you should use one of the two methods that we just demonstrated.

Numerical operations

Now that we have two input parameters and one output parameter, we can do some calculations, as follows:

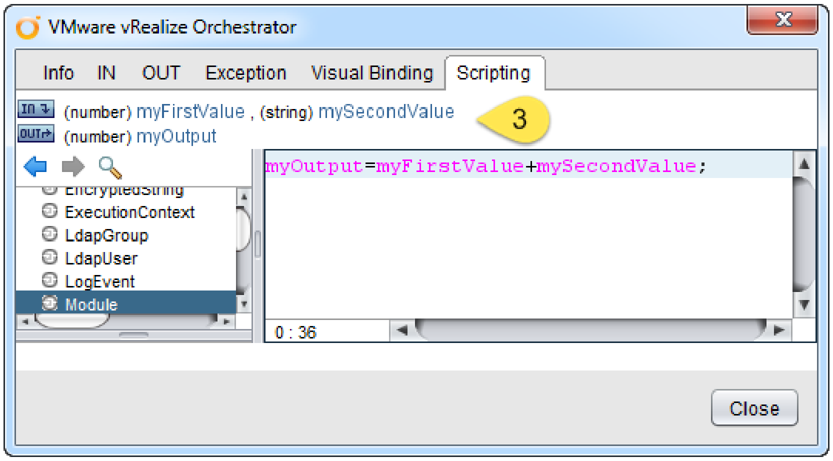
1. Starting from where we left off, edit the **Scriptable task** and click on **Scripting**.
2. Let me draw your attention to something. In the empty field, start typing `myOutput`. Did you see that as soon as you finished the word, it turned purple? Cool. Delete the word and type `myoutput` (no capital O). See how this doesn't turn purple? This is because of the case sensitivity. The parameters `myoutput` and `myOutput` are two different variables.

> Purple-colored variables indicate that the variable is bound as an input or output of the scriptable task, custom decision, or the action that is currently being edited.

3. You can also just click on the parameter names to paste the variable into the scripting.
4. Create the following script:

```
myOutput = myFirstValue + mySecondValue;
```

Please note that I inserted an extra space to make this line more readable. This extra space is not needed, but it's fine if you keep it.



IN (number) myFirstValue , (string) mySecondValue
OUT (number) myOutput

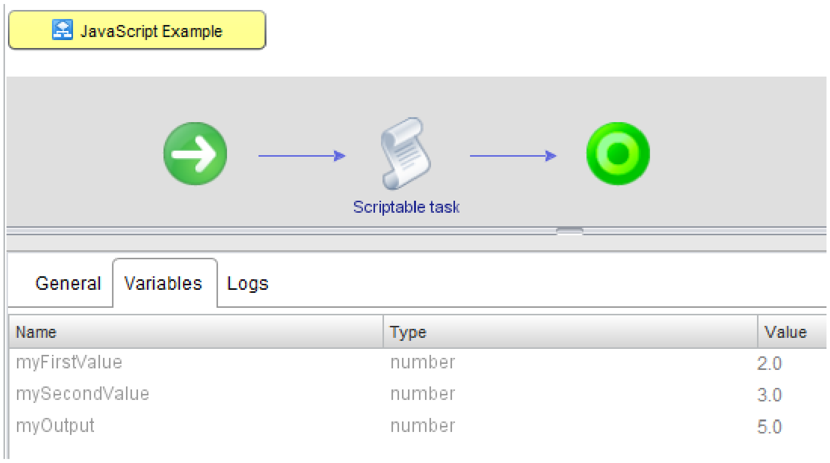
```
myOutput=myFirstValue+mySecondValue;
```

0 : 36

5. Save and close the workflow and then run it.

> If you are getting tired of saving, closing, running, and opening, have a look at *Chapter 8*, *Errors, Logs, and Debug Mode*, where we explain the debug modus.

6. Check the variable output for the results:



Name	Type	Value
myFirstValue	number	2.0
mySecondValue	number	3.0
myOutput	number	5.0

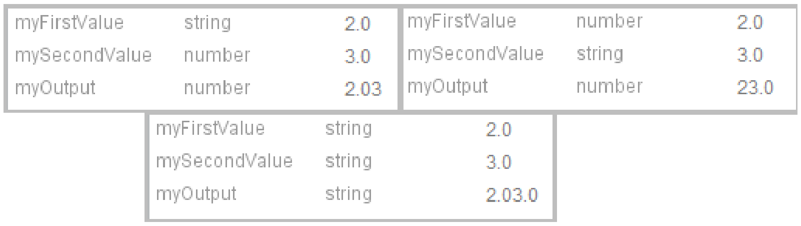
7. Go back and edit the workflow. Then try out the mathematical operations: minus (-), multiply (*), and divide (/).

String operations

We just played around with numbers. Let's do the same with strings, as follows:

1. Starting from where we left off, edit the workflow and click on **Inputs**.
2. Change the type of **myFirstValue** from `Number` to `String`.
3. Save and run the workflow. Can you see the funny result?
4. Now, set **myFirstValue** back to `Number` and **mySecondValue** to `String`. Run it.
5. Now, change all the input and output types to `String` and run it again.

The results should look like this:



myFirstValue	string	2.0
mySecondValue	number	3.0
myOutput	number	2.03

myFirstValue	number	2.0
mySecondValue	string	3.0
myOutput	number	23.0

myFirstValue	string	2.0
mySecondValue	string	3.0
myOutput	string	2.03.0

The results are different because Orchestrator does a bit of autocasting here. What this means is that it adds the String "2.0" to number 3, but because the first variable type is of the type String, Orchestrator decides that the second has to be of the same type too and converts 3 to "3". It then glues both "2.0" and "3" together to form the string "2.03". The output is again of the Number type. So, Orchestrator goes ahead and converts "2.03" into a number as 2.03. Repeat this thought experiment with the second and the last example. This example shows that you need to be a bit careful when combining variable types.

The + sign is used to put two strings together. Let's do this with just strings, as you would do in reality:

1. Make sure that all the input and output parameters are of the `String` type.
2. Run the workflow again and this time, enter `Hello` and `World`. The result should be **HelloWorld**.
3. Let's improve this. Change the script to the following:

   ```
   myOutput = myFirstValue +" "+ mySecondValue ;
   ```

4. Run it. This does look better now, right?

Integrating JavaScript into our program

Let's use some of the stuff that we just learned to improve our workflow that creates a VM. This is for you to try and learn, as it will repeat topics from this and the last chapter.

Note that when we select the amount of memory for the VM, it's in MB. Let's change this to the easier, usable GB by working through the following steps:

1. You should consider making a duplicate or a new version of the original workflow before we get started.
2. Move the **vmMemorySize** input-parameter to be an attribute, which is similar to what we did in the *Creating new parameters* section of this chapter.
3. Create a new **vmMemGB** input parameter of type number.
4. Reassign the **vmMemorySize** parameter to the **Create simple virtual machine** workflow. You can use validation to make it easier to find them all.
5. Add a scriptable task before the **Create simple virtual machine** workflow element.
6. Add the **vmMemGB** input parameter as an IN-parameter of the scriptable task.
7. Add the **vmMemorySize** attribute as an OUT-parameter of the scriptable task.
8. Create the script `vmMemorySize = vmMemGB * 1024;`.
9. Rename the scriptable task to `Convert MB2GB`.
10. Give it a go.

The if-clause in JavaScript

The if-clause is equivalent to the `Decision` element that we used in the last chapter. It checks whether a condition has been met.

The JavaScript code for an `if` statement isn't that difficult. It is made up of a condition and the if-clause itself. The form looks like this:

Statement	Example

`if (condition) {`
`    code block`
`} else if (condition) {`
`    code block`
`} else {`
`    code block`
`}`
	`if (varString == "test") {`
	`    varResult=true;`
	`} else {`
	`    varResult=false;`
	`}`

In the preceding example, the `varString` string is tested if it contains the word `"test"`. If it does, the Boolean `varResult` is set to `true`. If it is not (the `else` part), it is set to `false`. The `else if` block can be added multiple times.

Conditions and operators

A condition is made up of a statement that is either true or false. This statement is built by using operators. The following are the operators that JavaScript knows:

and	or	Not	Equal	Not equal	Smaller	Bigger
`&&`	`||`	`!`	`==`	`!=`	`< <=`	`>= >`

As an example, if you want to know whether the number that is stored in the `a` variable is bigger than 5, the conditional statement is `(a > 5)`.If you want to know whether the string stored in the `text` variable equals `"Hello"`, the conditional statement will be `(text == "Hello")`.

You can glue conditions together with the `&&` (and) and `||` (or) operators as well as (), the normal brackets. For example, if a is bigger than 5 and the text is "Hello", the conditional statement will be `((a>5) && (text == "Hello"))`.

In the example part of this section, we will give all of this a try.

An example if-clause

Now, let's have a look at an example. We will use our JavaScript example workflow again:

1. Using our JavaScript example workflow, make sure the following parameter types are set:

Name	**Where**	**Type**
myFirstValue	INPUT	Number
mySecondValue	INPUT	String
myOutput	OUTPUT	String

2. Now, edit the scriptable task and replace the script with the following:

```
if (myFirstValue>5) {
  myOutput="bigger than 5";
} else {
  myOutput="smaller or equal than 5";
}
```

3. Run the script and observe the results.
4. Now, change the script to check whether **mySecondValue** is equal to `"Hello"` (the correct answer is given at the end of the chapter).
5. After you've done this, try to create a condition that checks whether **myFirstValue** is greater than 5 and **mySecondValue** equals `"Hello"` at the same time (the correct answer is provided at the end of the chapter).

We will integrate a more complex if-Clause into our deploy program in the section where we learn about actions.

Common string problems and solutions

I would like to showcase some typical string problems that you will encounter when working with VMs. The String type comes in JavaScript with some methods (functions) that you can use. You can find all the JavaScript String methods at `http://www.w3schools.com/jsref/jsref_obj_string.asp`.

Is a string part of another string?

This is a typical situation that you will come across. You have a parameter that contains a string, and you would like to know whether this string contains another string.

For example, you have the **myVM** parameter that contains the `"testVM.mylab.local"` string, and you want to know whether this VM is in the mylab domain. So, if the `"myLab"` string is a part of it, you can use the `indexOf()` method of the String type. The index method returns the position of the first occurrence. If the string is not a part of the other string, the return value will be -1. Here's an example:

```
Var myVM = "testVM.mylab.local";
If ( myVM.indexOf("myLab") !=  -1) {
  //is part of mylab
} else {
  //is NOT part of mylab
}
```

Case sensitivity

Another common problem is that a user might enter `MyLab`, `myLab`, or `MYLAB` instead of the `myLab` that you are expecting and checking for. Here is a simple way to solve this one. You can just keep everything in uppercase and then check. So, don't use the following code:

```
var myEntry = "MyLab";
var myTest = "myLab":
if ( myEntry == myTest ) {
  // both are the same
}
```

Instead of the preceding code, use the `.toUpperCase()` method of the String type, as follows:

```
If ( myEntry.toUpperCase() == myTest.toUpperCase() ) {
  // both are the same
}
```

The same method applies to hostnames or VM names. VM names or hostnames are mostly in lowercase. Use the `.toLowerCase()`method to do this.

Getting rid of space

Consider an instance where users enter `"myLab "` or `" myLab"` by mistake (an extra space at the beginning or the end), and you can imagine how this can wreak havoc with an if-clause. Use the `.trim()` method to solve this, as follows:

```
var dirtyString = "    Hello World!     ";
var cleanString = dirtyString.trim();
```

Creating an action

Actions are what programmers call functions. It is a piece of code that you intend to reuse often in different programs.

There are multiple differences between a workflow and an action. The main difference is that an action can return only *one* variable, whereas a workflow can return multiple variables. Another difference is that actions are purely JavaScript-based and do not contain any visual programming.

In an action, the in-parameters are defined the same way as in a workflow. However, the return type is a bit different. The return code is always one variable and its value is assigned by using the JavaScript `return` command. If you don't want or need any return code, define the return code as `void`.

A good naming convention for actions is to start the name of the action with a verb, such as get, set, create, delete, and so forth. Then, describe what the action is doing. A good way to make the name more readable is to capitalize each word (except the first word). Examples of good naming are startVM, removeAllSnapshots, and getAllVMsOfVApps. If you need more information on this, check the JavaScript style guide at `http://javascript.crockford.com/code.html`.

While exploring the existing action library, you will find a lot of useful pre-created actions that can be used in your own workflows.

We will now create a new action that will be used in our **InstallFreshVM** workflow. This action will enable us to use a more user-friendly operating system.

Creating a new action module

An action module is like a workflow folder; except that you can't create submodules.

1. In the Orchestrator Client, make sure that you are using the **Design** mode.
2. Click on **Actions**, ⚙.
3. Right-click on the top level , and select **New module**.
4. Give the module a name that is based on either your URL or the type of work that you intend to do with it. For example, I chose `com.packtpub.Orchestrator-essentials`.

A new action module has been created.

Creating a new action

Now that we have an action module, we will create an action in it by performing the following steps:

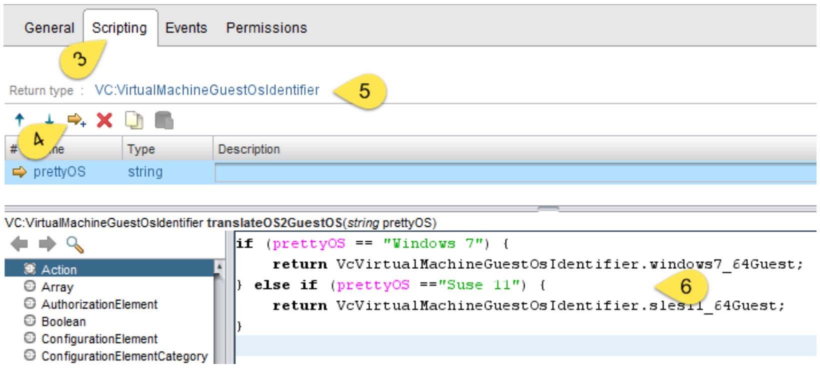
1. Right-click on the action module that you created and select **Add-action**.
2. The name should be descriptive and tell a user directly what it does. For this example, I chose the name as **translateOS2GuestOS**, as this action will translate for us a more descriptive OS name to the more cryptic Orchestrator type that we have been using so far.
3. Click on **Scripting**.
4. Now, click on **Add Parameter** and add the following variables:

Name	Type	Description
`prettyOS`	`String`	A "pretty" name for an operating system, such as "Windows 7"

5. Now, click on **void**, which is directly to the right of **Return type**, and select `VC:VirtualMachineGuestOsIdentifier`.
6. In the scripting field, enter the following code:

```
if (prettyOS == "Windows 7") {
  return VcVirtualMachineGuestOsIdentifier.windows7_64Guest;
} else if (prettyOS =="SLES 11") {
  return VcVirtualMachineGuestOsIdentifier.sles11_64Guest;
}
```

7. Click on **Save and close**.



Return type : VC:VirtualMachineGuestOsIdentifier

# Name	Type	Description
prettyOS	string	

VC:VirtualMachineGuestOsIdentifier **translateOS2GuestOS**(*string* prettyOS)

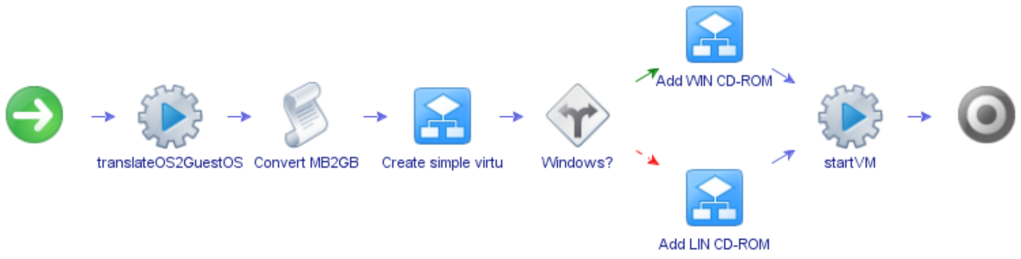
```
if (prettyOS == "Windows 7") {
    return VcVirtualMachineGuestOsIdentifier.windows7_64Guest;
} else if (prettyOS =="Suse 11") {
    return VcVirtualMachineGuestOsIdentifier.sles11_64Guest;
}
```

Implementing the action into a workflow

After creating our first action, we're going to use it in our deploy workflow, as follows:

1. By using the Orchestrator Client, make a backup copy or increase the version of your **InstallFreshVM** workflow.
2. Open the workflow for editing.
3. Move the **vmGuestOS** INPUT-parameter to be an attribute.
4. Create a new **vmGuestOSname** INPUT-parameter.
5. Reassign all the **vmGuestOS** parameters in all the workflow elements (you can use validation to make it easier to find them all).
6. Drag the **Action element** from the **Generic** toolbox into the workflow somewhere before the **Create simple virtual machine** workflow element.
7. In the Choose action menu, search and select the action that we have just created.

8. Assign the IN-parameter to **vmGuestOsName** and **prettyOsName** and the **actionResult** OUT-parameter to the **vmGuestOS** attribute.



9. Run the workflow, but now type in the "pretty" name that we defined in the input mask.

In the next chapter, we will improve this workflow further.

Things you might like to try

Here is a short list of things that you may like to try on your own:

Switch case

Instead of using the if-clause in the action, check out the JavaScript `switch` statement.

The `switch` statement in JavaScript looks like this:

Statement	Example

`Switch (expression) {`
`    case condition:`
`        code block`
`        break;`
`    default:`
`        default code block`
`}`
	`Switch (prettyOS) {`
	`    case "Windows 7":`
	`        return`
	`VcVirtualMachineGuestOsIdentifier.`
	`windows7_64Guest;`
	`        break;`
	`    case "SLES 11":`
	`        return`
	`VcVirtualMachineGuestOsIdentifier.`
	`sles11_64Guest;`
	`        break;`
	`    default:`
	`        throw "Unknown OS";`
	`}`

In `expression`, fill in the variable that you would like to check, and in the `case condition:` part, fill in the condition that you want to check against. Please note that you can only check the equals (==) condition with the `switch` statement. The `default:` part is used if all the other tests fail.

String clean-up

Use the methods that were showcased in the *Common string problems and solutions* section and implement them in the action to create a better user experience.

Summary

This chapter introduced you to JavaScript and how you can integrate it into Orchestrator. You learned how to use the scriptable task and the action element to directly write JavaScript code into your workflow.

We learned how to use the if-clause and what conditions are and how to form them. We also had a look at some of the common string problems that one may encounter.

In the next chapter, we will use presentation to improve user experience and reduce input errors.

Correct answers

Check whether `mySecondValue` equals `Hello`:

```
if (mySecondValue=="Hello") {
  myOutput="Well Helllo";
} else {
  myOutput="no hello";
}
```

Check both the conditions:

```
if ((myFirstValue>5)&&(mySecondValue=="Hello")) {
  myOutput="Well Helllo, its bigger than 5";
} else {
  myOutput="no hello and smaler then 5";
}
```

7
Improving Workflows with Presentation

In the previous chapters, we developed a workflow that creates a new VM and mounts a CD-ROM so that a new OS can be installed on it. Our first attempts were extremely manual. Over the last chapters we have now improved it quite a bit and made it much more user-friendly. Now, we will make it not only look and feel better, but also much less error-prone. We will use workflow presentation to do this.

In this chapter, we will learn the following:

- Basic presentation properties
- Working with arrays
- Linking presentation properties to other elements

Presentation basics

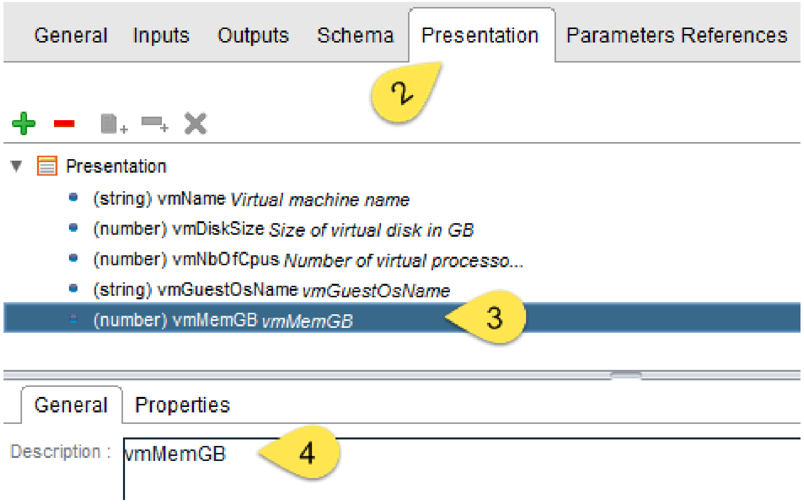
In the previous chapters, we concentrated on the workflow content and how to improve our scripting. In the chapters that we have covered so far, we also discussed the problems that wrong user input, such as the OS name, may cause and how we can intercept it by using JavaScript. Now, we will go a step further and make sure that a user cannot enter values that are wrong or not allowed.

Descriptions

We will start with something simple that has actually quite a massive effect, and that is, the description. A good description helps the user of a workflow better understand what they should enter in the fields.

1. By using the Orchestrator Client, edit the **InstallFreshVM** workflow.

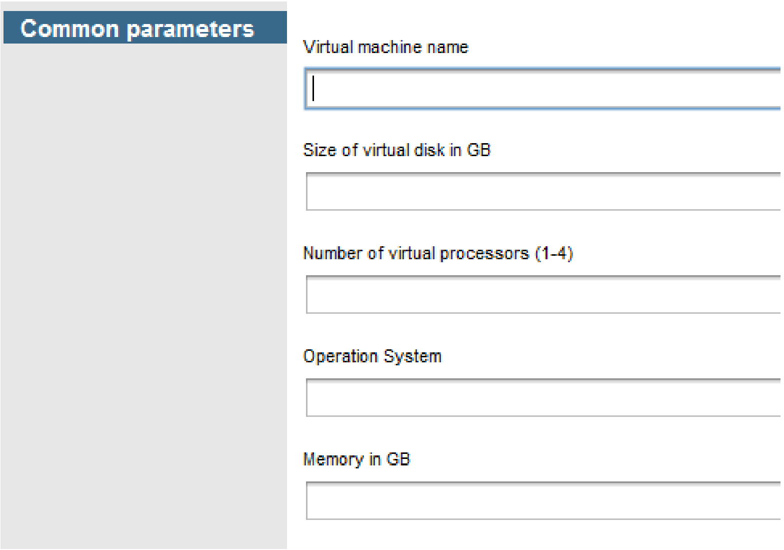
2. Click on **Presentation**. You will see all the input parameters displayed.
3. Click on **vmMemGB**. The **General** area will now show the content of the description of the **vmMemGB** parameter. At this moment, only the name of the parameter is shown, as we have just added this parameter.
4. Now, replace **vmMemGB** with **Memory in GB** to tell the user what they should enter:



Presentation
- (string) vmName *Virtual machine name*
- (number) vmDiskSize *Size of virtual disk in GB*
- (number) vmNbOfCpus *Number of virtual processo...*
- (string) vmGuestOsName *vmGuestOsName*
- (number) vmMemGB *vmMemGB*

Description : vmMemGB

Check out the other input parameters and decide on better descriptions. The result of your efforts becomes visible as soon as you run the workflow.



Virtual machine name

Size of virtual disk in GB

Number of virtual processors (1-4)

Operation System

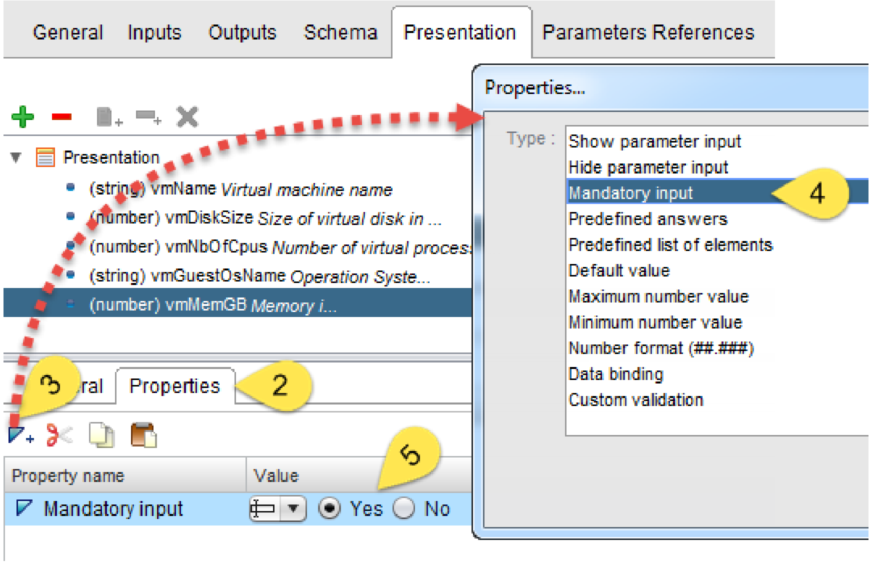
Memory in GB

Mandatory

There are more options to look into, and we will now start discussing them. First, we will make sure that a user has to provide a value. Currently, a user can just submit the workflow without entering anything, which will lead to an error.

We will now make a value mandatory by performing the following steps:

1. In **Presentation**, select the **vmMemGB** input parameter.
2. Click on **Properties**.
3. Now, click on **Add Property**.
4. Select **Mandatory input** and click on **OK**.
5. The property is added to the list, and you can decide whether this Input-parameter should be mandatory.



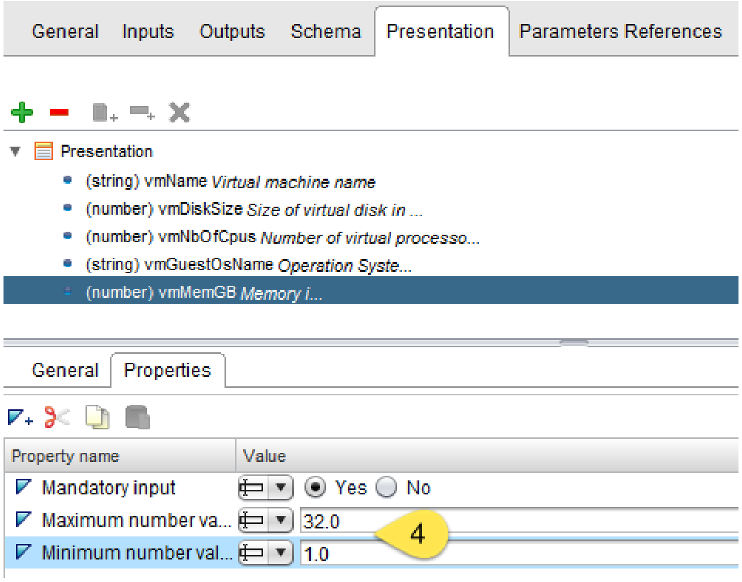
Try it out and think about which other parameters should be made mandatory.

Min and max values

We are now sure that the user has to enter a value, but he still could go ahead and enter, for instance, -1 CPUs or 1.000 GB memory, which isn't really that helpful. So, let's fix this by using min and max values, as follows:

1. Select the presentation properties of the **vmMemGB** input parameter.

2. Click on **Add Property** and select **Minimum number value**.
3. Now, add the **Maximum number value**.
4. In the **Properties** list, you can now enter the values that you want. For example, you can limit the amount of memory to between 1 and 32 GB.
5. Run the workflow and try entering a value less and more than the limits that you set.



Presentation
- (string) vmName *Virtual machine name*
- (number) vmDiskSize *Size of virtual disk in ...*
- (number) vmNbOfCpus *Number of virtual processo...*
- (string) vmGuestOsName *Operation Syste...*
- (number) vmMemGB *Memory i...*

Property name	Value
Mandatory input	● Yes ○ No
Maximum number va...	32.0
Minimum number val...	1.0

After locking in these values, think about which other values could benefit from a min and max value.

Have a look at the properties for the **vmName**. Check whether you can fix the length of the string entered. This can be quite useful, for instance, a hostname (not the FQDN but the name of the host) should have at least 2 characters, and it should not be longer than 15 characters. Have a go at it!

Default value

We have fixed the range of the values that can be entered, but when the user starts the workflow, it is still empty. It is a good idea to provide the user with some ideas or settings, such as a default value.

We can add a default value by performing the following steps:

1. Select the presentation properties of the **vmMemGB** input parameter.
2. Click on **Add Property** and select **Default Value**.
3. Select a default value of 4 and run the workflow.

Have a look at the other input parameters and think about which could benefit from a default value.

There are several more useful properties, but before we can go on, we need to learn about arrays.

Working with arrays

We have already taken a look at different variable types, but there is a special type that we should explore a bit more—the array. Any variable type can be made into an **array**. An array is a container that holds multiple values of the same variable type. For example, an array of the String variable type may contain values such as "Mum", "Dad", "Sister", and "Brother", whereas an array of VC:VirtualMachine may contain multiple VMs.

Arrays in JavaScript

In JavaScript, you can use several methods to start, define, and initialize an array. The following are some of these methods:

- By defining its content:

  ```
  var family = new Array("Mum", "Dad", "Sister", "Brother");
  ```

- Using a more compact style:

  ```
  var family=["Mum", "Dad", "Sister", "Brother"];
  ```

- Alternatively, if you want to initialize an empty array, just use the following:

  ```
  var family = new Array();
  ```

- If you want to add an element to an array, you just push it in:

  ```
  family.push("Aunt");
  ```

Typically, we access the content of an array via JavaScript. To access an array, we use square brackets (`[ ]`) and write in it the number of the elements that you would like to access. Please note that the content count of an array starts from 0 (zero). For example, `family[2]` will return `"Sister"`, and `family[0]` will return `"Mum"`.

> Please note that before adding a value to an array, it must be defined. If you create an attribute array, you will need to either initialize it in JavaScript, or set it from the Orchestrator Client with no value.

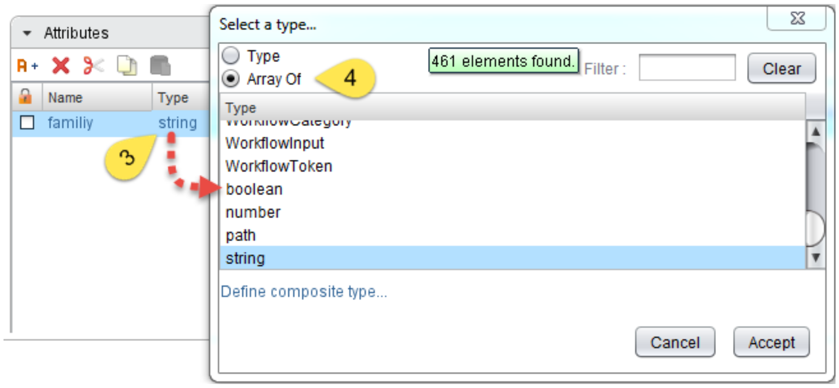
We will work on an example in a second.

Defining and filling arrays in the workflow

Arrays are used a lot in Orchestrator, and you need to know how one can define them in the workflow as a parameter.

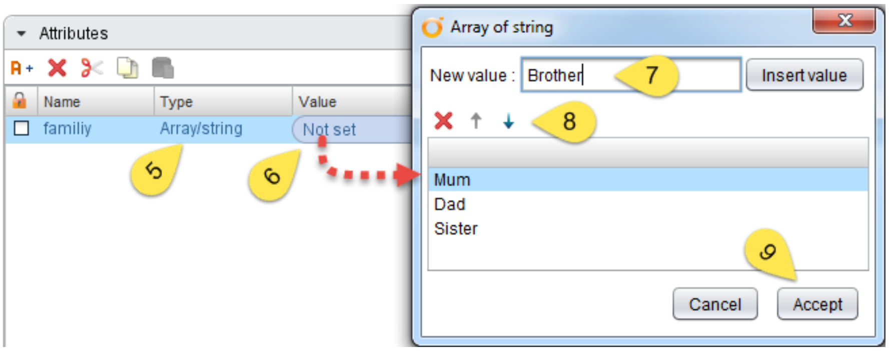
We will now create an array in the JavaScript example workflow that we created in the last section. We will first define an array and then fill it with content, as follows:

1. Edit the JavaScript example workflow that you created in the last chapter.
2. Go to the **Attributes** and create a new attribute called **family**.
3. Click on **string**.
4. Select **Array Of** and click on **Accept**.



5. We have generated an empty array that will contain strings. Have a look at the displayed variable type.
6. Now, we will fill it. Click on **Not set** under **Value**.

7. Enter a **New value** and click on **Insert value** to add it to the array. Do the same with the values: `Mum`, `Dad`, `Sister`, and `Brother`.
8. You can delete elements by clicking on the red **X** or change their order by using the arrows. Try it!
9. You need to click on **Accept**, otherwise all the content that you put in the array will be lost.



Name	Type	Value
familiy	Array/string	Not set

New value : Brother

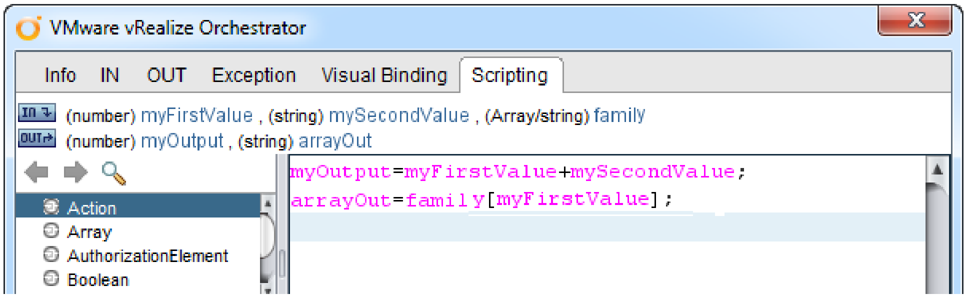
Mum
Dad
Sister

JavaScript example

We created an array in the workflow and filled it with values. We will now use JavaScript to read it:

1. From where we left off in the JavaScript example workflow, go to the schema.
2. Edit the scriptable task and add the **family** array to it as an IN-parameter.
3. Make sure that the **myFirstValue** parameter is of the type **Number**.
4. Create a new OUT-parameter called **arrayOut** of the type **String** in the scriptable task.
5. Add the following script to your **Scripting** section:

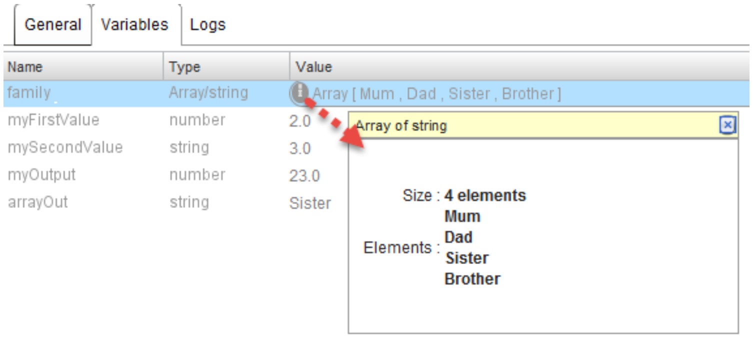
```
arrayOut = family [myFirstValue];
```



IN (number) myFirstValue , (string) mySecondValue , (Array/string) family
OUT (number) myOutput , (string) arrayOut

```
myOutput=myFirstValue+mySecondValue;
arrayOut=family[myFirstValue];
```

6. Run the script and check the results. Click on the little **i** that precedes the array to see the content of the array:



Name	Type	Value
family	Array/string	Array [Mum , Dad , Sister , Brother]
myFirstValue	number	2.0
mySecondValue	string	3.0
myOutput	number	23.0
arrayOut	string	Sister

Array of string

Size : **4 elements**
Elements : **Mum**
Dad
Sister
Brother

Now, we will add a value to the family array:

1. Add a second scriptable task to the workflow directly after the existing one.
2. Edit the scriptable task and add the **family** array to it as an IN- and OUT-parameter.
3. Add the **myFirstValue** input parameter as an IN-parameter to the scriptable task.
4. Add the following script:

```
family = family.push(myFirstValue);
```

5. Run the workflow again, and have a look at the result.

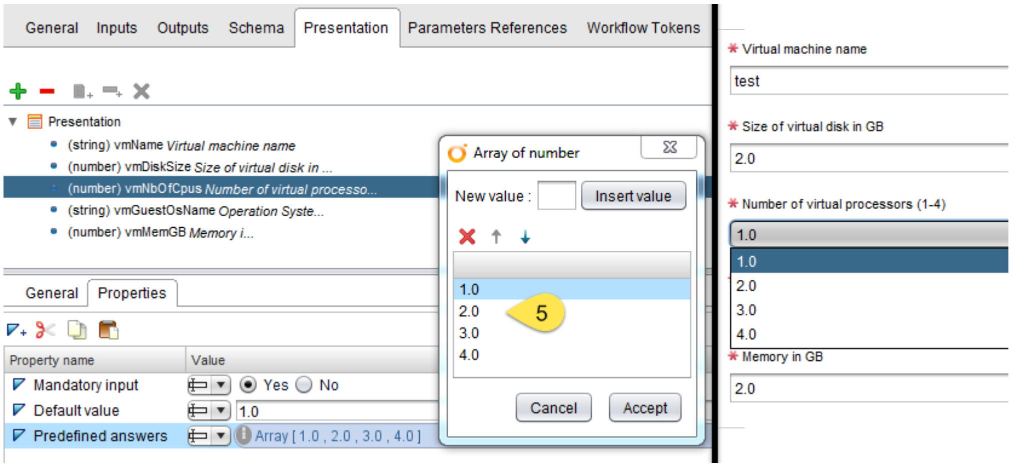
Predefined answers

So, now that we have learned about arrays, we can proceed to use an array in our workflow.

We are doing quite well in regards to the improvements to our workflow, but we still have some issues. For instance, a user can still enter 1.5 CPUs, which isn't going to work. So, we will offer the user only a list of values that they can choose from by performing the following steps:

1. Edit the **InstallFreshVM** workflow.
2. Click on **Presentation** and then select the **Properties** of the **vmNbOfCpus** input parameter.

3. Click on **Add Property** and select **Predefined answers**.
4. After the property has been added to the list, click on **Not set**.
5. An array field will open up. Fill in the `1`, `2`, `3`, and `4` values and click on **Accept**.
6. Save and run the workflow.



Presentation
- (string) vmName *Virtual machine name*
- (number) vmDiskSize *Size of virtual disk in ...*
- (number) vmNbOfCpus *Number of virtual processo...*
- (string) vmGuestOsName *Operation Syste...*
- (number) vmMemGB *Memory i...*

Property name	Value
Mandatory input	● Yes ○ No
Default value	1.0
Predefined answers	Array [1.0 , 2.0 , 3.0 , 4.0]

Array of number

New value :

1.0
2.0
3.0
4.0

* Virtual machine name
test

* Size of virtual disk in GB
2.0

* Number of virtual processors (1-4)
1.0
1.0
2.0
3.0
4.0

* Memory in GB
2.0

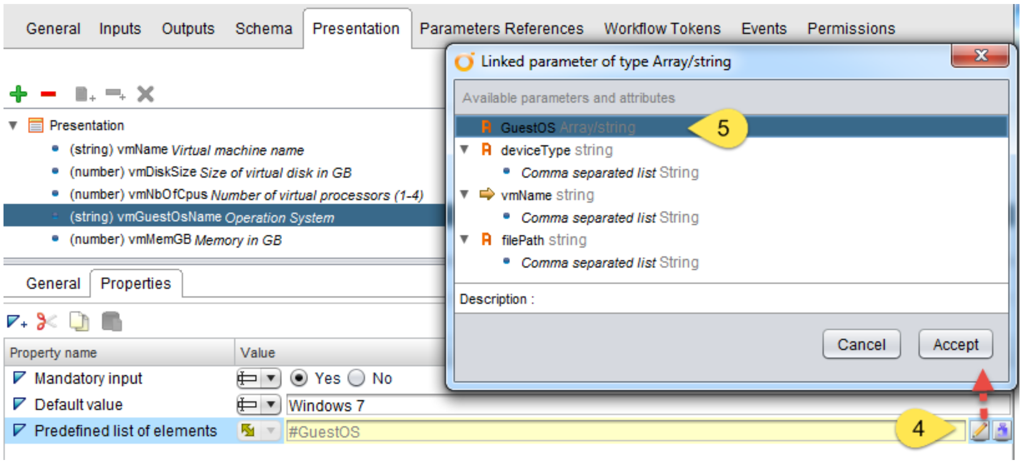
Linking presentations

Until now, we used fixed values in the presentation properties. We will now link properties to parameters.

To do this, we will create an array of strings that is filled with the operating system names and link them as a list for selection, as follows:

1. Edit the **InstallFreshVM** workflow.
2. Add a new array of strings in the attributes and name it **GuestOS**. Enter `Windows 7` and `SLES 11` as values.
3. Now, click on **Presentation**, select the **vmGuestOsName**, and add the **Predefined list of elements** property.
4. The property looks a bit different. For starters, it's yellow, which indicates that it's a linked property. Click on **Help editing OGNL**, ✎.
5. From all the values that fit the input parameter type (string), select the array that you have just created and click on **Accept**.

6. Now, save and run the workflow.



- Presentation
  - (string) vmName *Virtual machine name*
  - (number) vmDiskSize *Size of virtual disk in GB*
  - (number) vmNbOfCpus *Number of virtual processors (1-4)*
  - (string) vmGuestOsName *Operation System*
  - (number) vmMemGB *Memory in GB*

Property name	Value
Mandatory input	● Yes ○ No
Default value	Windows 7
Predefined list of elements	#GuestOS

Linked parameter of type Array/string

Available parameters and attributes

- GuestOS Array/string
- deviceType string
  - *Comma separated list* String
- vmName string
  - *Comma separated list* String
- filePath string
  - *Comma separated list* String

Description :

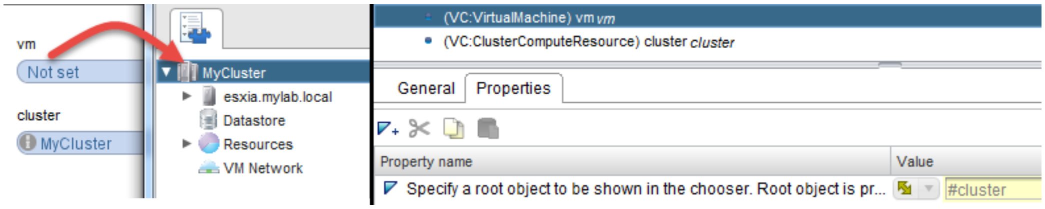
The linking of properties is quite useful. Just think about the fact that you can use inputs to alter the **Properties** settings.

VMware plug-in specific properties

There are some special properties of the VMware Server plug-in (the plug-in that starts with VC:) We will have a quick look at them now.

Specify a root object to be shown in the chooser

This setting lets you define a certain start point for your searches. For example, if you choose an ESXi cluster as the root element, then a user can only select objects under this cluster. To use the root element, you need to link an action or a variable to this property. The following is an example:



vm: Not set

cluster: MyCluster

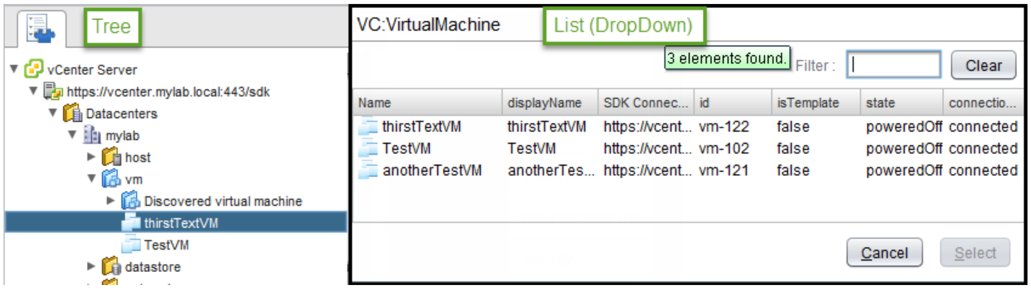
- MyCluster
  - esxia.mylab.local
  - Datastore
  - Resources
  - VM Network

- (VC:VirtualMachine) vm *vm*
- (VC:ClusterComputeResource) cluster *cluster*

Property name	Value
Specify a root object to be shown in the chooser. Root object is pr...	#cluster

Select value as

The `select value as` property has three choices—**Tree**, **List**, and **Dropdown**. This property makes it easier to manage what the input of an object looks like. The **Select value as** behaves differently, depending on the object type and client implementation. In the following screenshot, the dropdown for the VC:VirtualMachine selection is shown as a list:



Name	displayName	SDK Connec...	id	isTemplate	state	connectio...
thirstTextVM	thirstTextVM	https://vcent...	vm-122	false	poweredOff	connected
TestVM	TestVM	https://vcent...	vm-102	false	poweredOff	connected
anotherTestVM	anotherTes...	https://vcent...	vm-121	false	poweredOff	connected

Show in inventory

The `show in inventory` property is a powerful property. When you start a workflow from vSphere Web Client, the object you started the workflow on will be passed to the workflow as an IN-parameter. If this property is assigned to more than one IN-parameter, only one parameter is used at a time. For example, if you have an IN-parameter for a VM and a cluster, both have the **Show in Inventory** property assigned to them. When starting the workflow from a VM, the VM ID will be used, and the cluster ID will not be transferred to the workflow. When starting the workflow from a cluster, the cluster ID will be used, and the VM ID will be ignored.

Example

In order to get this showcased, we will use the **InstallFreshVM** deploy workflow. We will alter it a bit and then use it in the Web Client, as follows:

1. Make a copy or increase the version of the **InstallFreshVM** workflow, just to be on the safe side.
2. Edit the workflow and go to **Attributes**.
3. Move the parameter **vmResourcePool** and **vmFolder** to be an input parameter.
4. Reassign both the parameters to the **Create simple virtual machine** workflow (use the validation).

5. Create a new attribute called **rootVmFolder** of the **VC:VmFolder** type and assign it the **VM** root folder in your chooser.
6. Go to **Presentation** and select the **vmFolder** input parameter.
7. Add the **Specify root object to be shown in the chooser** property, click on **Help editing OGNL**, ✎, and assign it to the **rootVmFolder** attribute.
8. Now, add the **Show in Inventory** property to the **vmResourcePool** input parameter. Just add it. No extra configuration is needed.
9. Open the vSphere Web Client in a browser and add the **InstallFreshVM** workflow. Make sure that you select the **Host**, **Cluster**, and **Folder** types with it. We did this in *Chapter 4*, *Working with Workflows*. You can have a look at it if you need to refresh your memory.
10. Now, right-click on a VM folder and select the new workflow. Check how the properties are working.

Summary

In this chapter, we learned how to improve our workflows by using presentation. We explored some basic possibilities and showed you the way to explore more about them.

We also had a look at arrays and what they can do for us.

In the next chapter, we will learn how to deal with errors and the debug mode.

8
Errors, Logs, and Debug Mode

In the last few chapters, we learned how to build new workflows and improve them. What we haven't covered yet is how we can catch errors and deal with them. By using presentation properties, we have already learned how to reduce the amount of errors by introducing a method that can be used to limit the user's possibilities to enter wrong values. Now, we need to have a look at how we deal with "real" errors, such as what happens if the new VM can't be built because there isn't enough disk space on the datastore.

Logs help us keep track of what's happening in the workflow as well as the settings.

We will also have a look at the debug modus of Orchestrator and learn how we can use it to test our workflows.

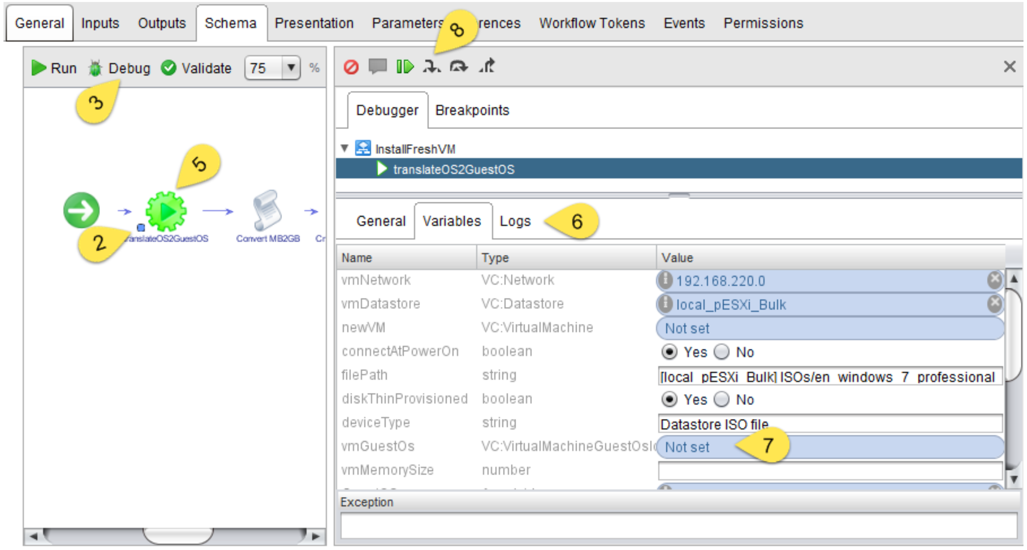
The debug mode

Lets' start with the Orchestrator debug mode; you may have noticed that the schema has some interesting extra buttons next to the validation. There is a play button and one that looks like a bug. We will now focus our attention on this second icon.

The Orchestrator debug mode allows you to stop and resume a workflow execution, and inspect the parameters to see the values that they currently have. You can already imagine what a great help this is.

1. In the Orchestrator Client, open the **InstallFreshVM** workflow for editing.

2. Right-click on the first element in the workflow (it should be the **translateOS2GuestOS**) and select **Toggle Breakpoint**. A blue dot should now be seen to the left of the element.
3. Now, click on **Debug**, Debug.
4. The normal workflow input window will open. Select the values and submit it.
5. As soon as you submit it, the workflow will start and pause on the first element. The element is now greenish in color.
6. To the right, an extra window opens and shows all the parameters in the **Variables** tab.
7. Please note that **vmGuestOS** is currently empty, as it shows **Not set**. This indicates that the workflow execution stopped before the action was executed.
8. Now, we want to continue the execution of the workflow step by step. Click on **Step into**, or press *F5*.



Name	Type	Value
vmNetwork	VC:Network	192.168.220.0
vmDatastore	VC:Datastore	local_pESXi_Bulk
newVM	VC:VirtualMachine	Not set
connectAtPowerOn	boolean	● Yes ○ No
filePath	string	[local_pESXi_Bulk] ISOs/en_windows_7_professional
diskThinProvisioned	boolean	● Yes ○ No
deviceType	string	Datastore ISO file
vmGuestOs	VC:VirtualMachineGuestOsI	Not set
vmMemorySize	number	

9. The workflow executes the action and stops on the scriptable task, but it does not execute it. Check the **vmGuestOS** parameter.
10. You can now go ahead and click on **Step into** or press *F5* until the workflow has finished. Alternatively, you can just click on **Resume**, ▶, to finish it.

The following control functions are used to debug:

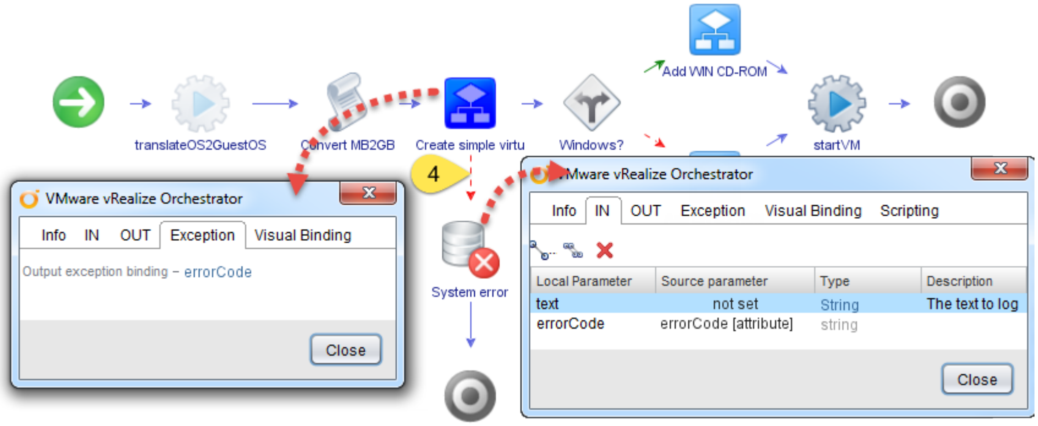
Icon	Name	Key	Description
⊘	**Cancel**		This stops a workflow execution.
💬	**Answer**		This answers a user interaction which the workflow issued.
▶	**Resume**		This resumes the workflow until the next breakpoint.
↴	**Step into**	F5	This steps into an element and starts debugging inside the child element. This can also be used to go to the next element.
↷	**Step over**	F6	This will step over an element. The debugging will not enter the child element.
↱	**Step return**	F7	This steps out of a child element. The debugging will continue with the parent element.

Logs and errors

As logs and errors go well together, let's use them. We will improve the **InstallFreshVM** workflow to catch errors, as follows:

1. Open the **InstallFreshVM** workflow for editing.
2. Click on **Schema** and open the **Log** toolbox.
3. Drag a **System Error** element onto the **Create simple virtual machine** element.
4. You will see that **System Error** connects itself to the **Create simple virtual machine** element with a dashed red line. This is the error line.

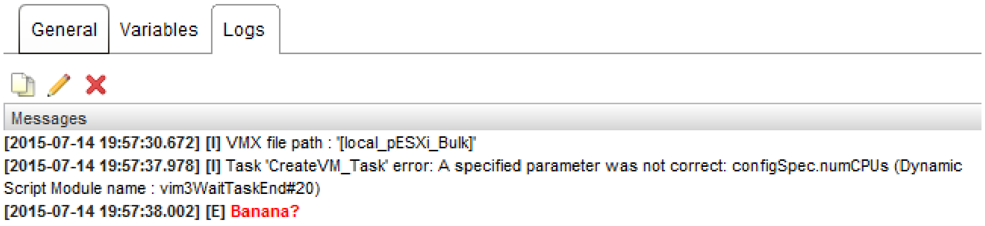
5. Click on the edit symbol of the **Create simple virtual machine** element and have a look at the **Exception** tab. You will find that a new attribute called **errorCode** is linked to the element. Check out the IN-parameters of the **System Error** element. You will also find the **errorCode** attribute here. These were automatically created and linked when you pulled the **Log** element onto it.



Output exception binding – errorCode

Local Parameter	Source parameter	Type	Description
text	not set	String	The text to log
errorCode	errorCode [attribute]	string	

6. All we now need to do is assign a string value to the IN-parameter text, and we are ready. Do this, and fill its value with something meaningful, such as `"Banana?"`.
7. Now, we need to create an error. You could lift some of the restrictions that you entered in the presentation and create a VM with 0.5 processors. This will create an error when the virtual machine is built.
8. Run the workflow with the debugger and have a look at the resulting logs:

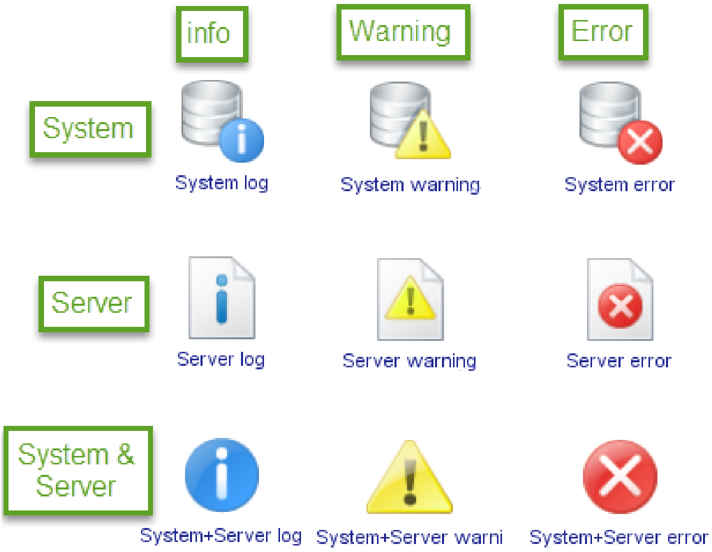


```
[2015-07-14 19:57:30.672] [I] VMX file path : '[local_pESXi_Bulk]'
[2015-07-14 19:57:37.978] [I] Task 'CreateVM_Task' error: A specified parameter was not correct: configSpec.numCPUs (Dynamic
Script Module name : vim3WaitTaskEnd#20)
[2015-07-14 19:57:38.002] [E] Banana?
```

Check the **Variables**. As you can see, the **errorCode** attribute now contains the error code that vCenter returned. If you check the logs, you will also see that the error was logged. However, you will also see the error text that we created. In the older versions, the error code may be shown instead.

It is obvious that our error text is not as clear as the one that we got from vCenter. However, this was just to showcase what each IN-parameter does. When you program, you can replace the original error message with something that either is a bit more descriptive, or contains a link to an internal document that shows the user how to solve the problem.

There are basically nine log workflow elements in the schema, and they differ in terms of where they log and the severity that they log. The info level logs are just for information and do not show up as an error. The warning logs indicate that something happened but it can be ignored. The Error log level indicates that something that has happened can or should not be ignored.



The main difference between server and system logs is that the server logs are stored as events in the Orchestrator database, and the system logs are stored in the `system.log` file. In addition to this, the system entries will appear in the **Logs** tab of a workflow token as long as the Orchestrator Client is open when the workflow runs. However, the server entries will appear in the **Events** tab of the workflow.

> Please note that the log files are rotated. This means that after some time, the older entries will be overwritten. The events are purged from the database according to the purging policy defined in the Orchestrator configuration.

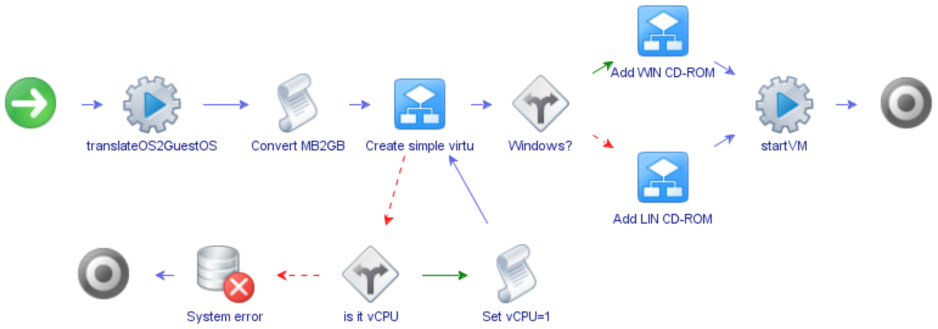
Catching and dealing with an error

This is a bit advanced, but it gives you a good idea of how powerful error handling can be. As this is going be an advanced topic, you will need to use all the skills that you have acquired so far.

1. In this example, we will lift the CPU restrictions and create an intentional error, which will then be rectified by using error handling while the program is running:
2. You can make a copy of your **InstallFreshVM** deploy workflow just to be on the safe side or increase the version.
3. Open the workflow for editing.
4. Move the **vmNbCpus** input parameter to be an attribute and reassign it to the **Create simple virtual machine** workflow element.
5. Create a new input parameter of type number called **vmCPUs**. We will use this to capture the input.
6. Add the **vmCPUs** attribute as an IN-parameter and **vmNbCpus** as an OUT-parameter to the **ConvertMB2GB** scriptable task. Now, add the `vmNbOfCpus=vmCPUs;` line to your script.

   This is done so that we can later change the attribute to another value. Remember that input parameter cannot be assigned as the OUT-parameters of the workflow elements.

7. Now, drag a (basic) **Decision** element onto the red error line and then drag a scriptable task onto the **Decision** element. After doing this, you will have to rearrange the lines. The `true` line should point to the scriptable task and the `false` line to the log element. Connect the line from the scriptable task back to the **Create simple virtual machine** element, as shown in the following figure. Please note that this task requires you to delete and create new lines, as seen in *Chapter 5*, *Combining and Modifying Workflows*:



8. In the **Decision** element, check whether **errorCode contains numCPUs**. What this does is check whether the error that we are getting is due to a problem with the CPU count. Check the error message that you received in the last example.
9. Assign the **vmNbCpus** attribute to be an OUT-parameter of the scriptable task and add the `vmNbOfCpus=1;` script.
10. Done. Now save and run the workflow and enter `0.5` vCPUs.

If you've managed to create and run the preceding example successfully, you can start calling yourself an Orchestrator developer. Well done!

As you can see, we can catch an error and then try to use the error message to rectify the situation.

Summary

In this chapter, we touched upon log files as well as ways to handle an error. We learned how to catch errors and resolve them.

The introduction to the Orchestrator debug mode should now enable you to become better at debugging the workflows that you write and improve your understanding of the function of a workflow.

In the last chapter, we will say goodbye and wrap up our work. We will learn how to export and import all the things that we have learned so far.

9
Packing It All Up

We are approaching the end of this book, and it's time to pack it all up. In this chapter, we will have a quick look at how to export and import a workflow as well as create a whole package.

Importing and exporting workflows

Orchestrator provides us with a lot of possibilities for creating scripts, and as everyone knows, sharing is caring. So, it is good to know how to export a workflow so that you can share it with others. We will focus on workflows here, but the same concept holds true for actions.

Exporting a workflow

We want to export a workflow into a file so that we can send it via e-mail, upload it, or simply make a backup.

1. Open the Orchestrator Client and drill down through the workflow library until you find the workflow that you would like to export. Choose the **InstallFreshVM** deploy workflow.
2. Right-click on the workflow and select **Export workflow**.
3. Chose a location that you want to store the workflow in and click on **Save**.

The file has been exported into a file called `InstallFreshVM.workflow` that you can now move around or e-mail.

The workflow that you exported only contains the workflow, not the action that we created for it or the other workflows that have been used. This is where packages come in handy, as they export not only the workflow, but also everything that is needed for this workflow.

Every workflow has a unique ID. We can be sure that when we import it into other Orchestrator servers, it will not overwrite anything and it will be accessible via the API by using the same ID as that of the original server.

Importing a workflow

Now that we know how to export a workflow, let's import one. As we do not normally have a new workflow, we're just going to delete the old **InstallFreshVM** workflow and then import it from the file again:

1. Delete the **InstallFreshVM** workflow, as demonstrated in *Chapter 4*, *Working with Workflows*.
2. Now, right-click on the folder you want to import the workflow into and select **Import workflow**.
3. Select the file that you would like to import and click on **Open**.

The workflow has been imported, and you can use it again.

Working with packages

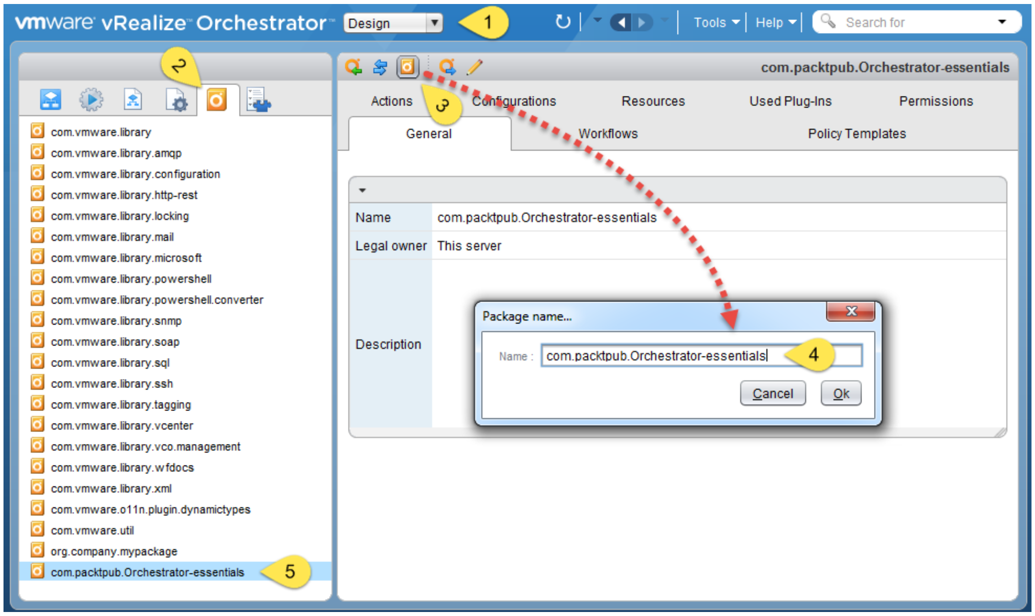
As we saw in the last section, exporting a workflow will only export it and not the objects that we used in it. Therefore, we should have a look at packages. Packages export and import all the dependent objects that are needed to run the workflow and not just a single object.

Creating a package

Before we go ahead and export a package, we have to create one, as follows:

1. In your Orchestrator Client, make sure that you are in the **Design** mode.
2. Click on **Packages**, .

3. Select **Add Package**, , from the side menu.
4. As package name, select `com.packtpub.Orchestrator-essentials`.
5. You will see that a new package has been added to the end of the package list.



com.packtpub.Orchestrator-essentials

Name: com.packtpub.Orchestrator-essentials
Legal owner: This server
Description:

Package name... Name : com.packtpub.Orchestrator-essentials

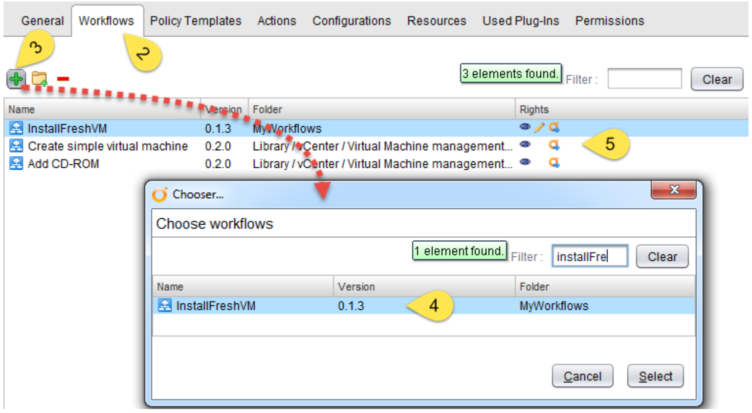
- com.vmware.library
- com.vmware.library.amqp
- com.vmware.library.configuration
- com.vmware.library.http-rest
- com.vmware.library.locking
- com.vmware.library.mail
- com.vmware.library.microsoft
- com.vmware.library.powershell
- com.vmware.library.powershell.converter
- com.vmware.library.snmp
- com.vmware.library.soap
- com.vmware.library.sql
- com.vmware.library.ssh
- com.vmware.library.tagging
- com.vmware.library.vcenter
- com.vmware.library.vco.management
- com.vmware.library.wfdocs
- com.vmware.library.xml
- com.vmware.o11n.plugin.dynamictypes
- com.vmware.util
- org.company.mypackage
- com.packtpub.Orchestrator-essentials

Filling a package

Now that we have created the package, we need to fill it with objects, as follows:

1. Starting from where we left off, right-click on the newly created package and select **Edit**.
2. Select **Workflows**.
3. Click on the green **+** button.
4. Find the **InstallFreshVM** workflow, mark it, and click on **Select**.

5. You will now see that the workflow has been added along with all the other workflows that we used in it.



Name	Version	Folder	Rights
InstallFreshVM	0.1.3	MyWorkflows	
Create simple virtual machine	0.2.0	Library / vCenter / Virtual Machine management...	
Add CD-ROM	0.2.0	Library / vCenter / Virtual Machine management...	

Name	Version	Folder
InstallFreshVM	0.1.3	MyWorkflows

6. When you click on **Actions**, you will find out that a lot of actions have also been added, along with the **translateOS2GuestOS** that we created in *Chapter 6*, *Advanced vRO Scripting with JavaScript*. These are all the actions that are used by the other workflows that have been added.
7. Click on **Save and close** to exit the workflow package and save its contents.

We have now created an Orchestrator package that contains all the spoils of our work.

> It is a good idea to check all the workflows and actions that have been added to your package.
>
> Please note that VMware suggests deleting common plug-in library actions and workflows from packages before shipping them to a client. This will make sure that a version mismatch doesn't happen.

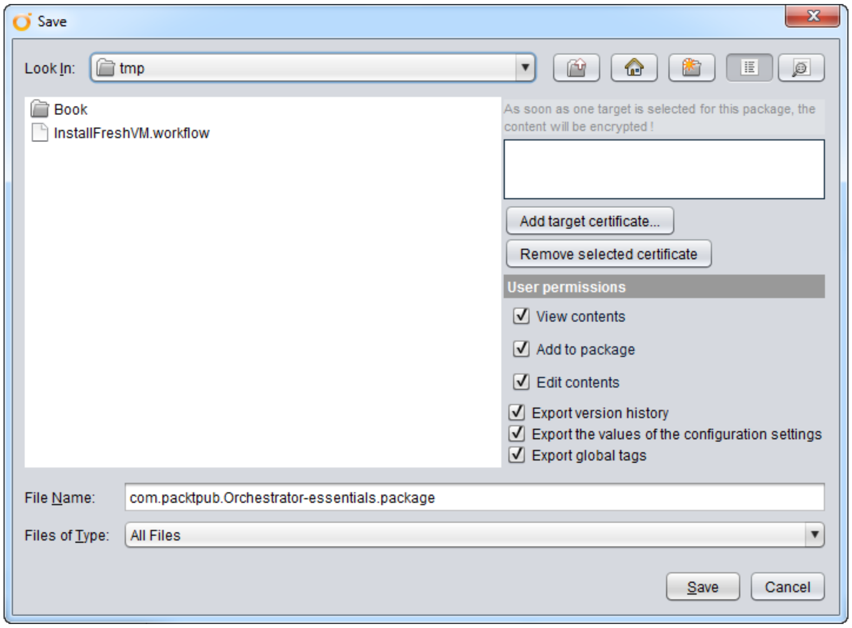
Exporting a package

Let's export this package so that we can share it with others:

1. Go to the **Package** tab and select the `com.packtpub.Orchestrator-essentials` package.
2. Right-click on the package and select **Rebuild package**. This option will refresh the package and make sure that all the essential components are in the package. Note that this will add all the common plug-in library actions and workflows to your workflow again.

3. Right-click on the workflow and select **Export package**.
4. Click on **Save**.

A file named `com.packtpub.Orchestrator-essentials.package` will be saved to your local disk.



Look In: tmp

Book
InstallFreshVM.workflow

As soon as one target is selected for this package, the content will be encrypted !

User permissions

☑ View contents
☑ Add to package
☑ Edit contents
☑ Export version history
☑ Export the values of the configuration settings
☑ Export global tags

File Name: com.packtpub.Orchestrator-essentials.package

Files of Type: All Files

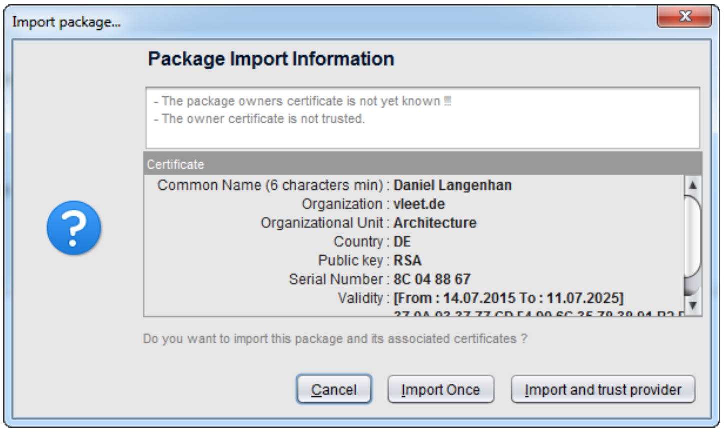
As you can see, there are quite a lot of options for exporting. The following table summarizes what each option does:

Option	Description
Add target certificate	You can specify a recipient's certificate to make sure that only the recipient can import it.
View Contents	This is not really as restrictive as one would expect. When you deselect this option, you will still see all the normal tabs in the workflow. The only thing that won't work is that you can't go to an element by double-clicking on it. For example, if the workflow contains an action, double-clicking on it won't open the action element. However, the action can be accessed normally, and you can see the scripting content.
Add to package	Deselecting this option will make it impossible for users to export a package that contains elements from this package. You can still create packages with elements that don't have the **Add to package** flag. However, you will get an error message when you try to export the package.
Edit content	Without this flag, users who import this package will not be able to edit the workflow. This flag is mostly set for all the packages that are a part of a plug-in or to make sure that changes are not possible for support reasons.
Export version history	By deselecting this option, users will not be able to export the full version history of each element. Instead, the element will be displayed in the latest version with the **imported content from package** remark.
Export values of the configuration settings	Deselecting this section will export the configuration and its attributes. However, it will not export their values.
Export global tags	This will export the global tag of the objects in the package.

Importing a package

Last but not least, we will import a package into Orchestrator. This is something that we really cannot showcase that much without a second Orchestrator, but we will still show it so that you know how to do it:

1. Go to the **Package** tab in another Orchestrator.
2. Click on the **Import package**, .
3. Select the file of the package that you would like to import and click on **Open**.
4. The **Import package** wizard will show the package certificate that we created in *Chapter 2, Deploying and Configuring the Orchestrator Appliance*, and ask whether you would like to import this certificate only once or import and trust it. Select one:



Package Import Information

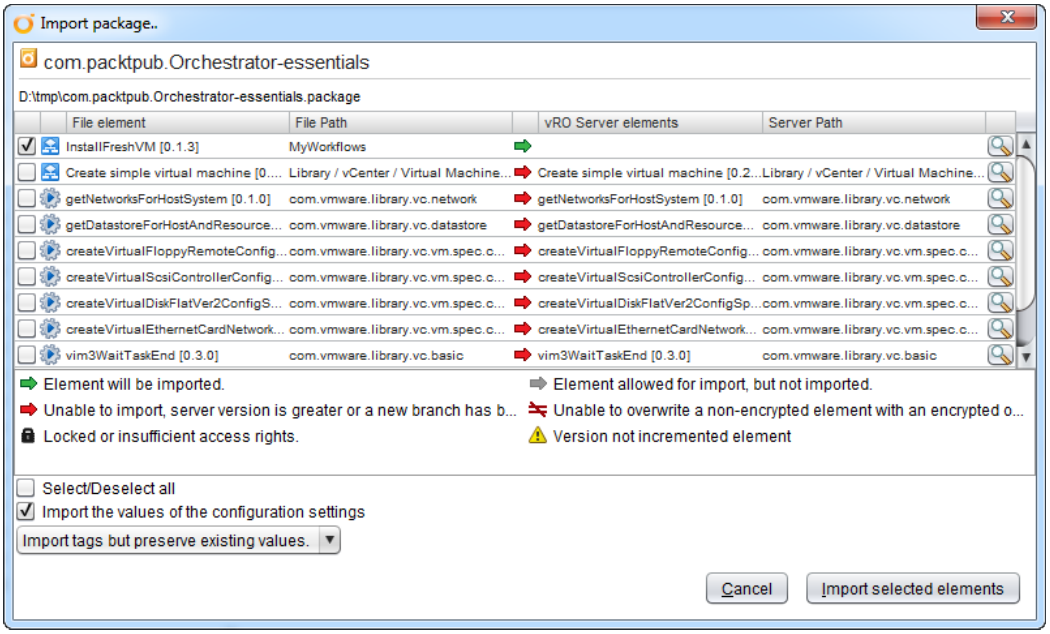
- The package owners certificate is not yet known !!!
- The owner certificate is not trusted.

Certificate

Common Name (6 characters min) :	**Daniel Langenhan**
Organization :	**vleet.de**
Organizational Unit :	**Architecture**
Country :	**DE**
Public key :	**RSA**
Serial Number :	**8C 04 88 67**
Validity :	**[From : 14.07.2015 To : 11.07.2025]**

Do you want to import this package and its associated certificates ?

5. You will now see a bigger window that shows all the workflows and actions that this package contains. Orchestrator checks the version numbers of the workflows and actions against the one that you are trying to import. A green arrow means that Orchestrator will import the workflow. A red one means that it won't, because the versions are the same or the signatures don't match. Grey arrows indicate that the versions match and hence, the import won't take place. You can override this selection by checking off the checkbox, but you should be careful with this. Click on **Import selected elements**:



com.packtpub.Orchestrator-essentials

D:\tmp\com.packtpub.Orchestrator-essentials.package

	File element	File Path		vRO Server elements	Server Path
☑	InstallFreshVM [0.1.3]	MyWorkflows	🟩		
□	Create simple virtual machine [0....	Library / vCenter / Virtual Machine...	🟥	Create simple virtual machine [0.2...	Library / vCenter / Virtual Machine...
□	getNetworksForHostSystem [0.1.0]	com.vmware.library.vc.network	🟥	getNetworksForHostSystem [0.1.0]	com.vmware.library.vc.network
□	getDatastoreForHostAndResource...	com.vmware.library.vc.datastore	🟥	getDatastoreForHostAndResource...	com.vmware.library.vc.datastore
□	createVirtualFloppyRemoteConfig...	com.vmware.library.vc.vm.spec.c...	🟥	createVirtualFloppyRemoteConfig...	com.vmware.library.vc.vm.spec.c...
□	createVirtualScsiControllerConfig...	com.vmware.library.vc.vm.spec.c...	🟥	createVirtualScsiControllerConfig...	com.vmware.library.vc.vm.spec.c...
□	createVirtualDiskFlatVer2ConfigS...	com.vmware.library.vc.vm.spec.c...	🟥	createVirtualDiskFlatVer2ConfigSp...	com.vmware.library.vc.vm.spec.c...
□	createVirtualEthernetCardNetwork...	com.vmware.library.vc.vm.spec.c...	🟥	createVirtualEthernetCardNetwork...	com.vmware.library.vc.vm.spec.c...
□	vim3WaitTaskEnd [0.3.0]	com.vmware.library.vc.basic	🟥	vim3WaitTaskEnd [0.3.0]	com.vmware.library.vc.basic

🟩 Element will be imported.
🟥 Unable to import, server version is greater or a new branch has b...
🔒 Locked or insufficient access rights.
▨ Element allowed for import, but not imported.
🟥 Unable to overwrite a non-encrypted element with an encrypted o...
⚠ Version not incremented element

□ Select/Deselect all
☑ Import the values of the configuration settings
Import tags but preserve existing values. ▼

The package is now added to Orchestrator, and you will find it in the Package section. You will also find out that the workflow folders that you created as well as the action modules are also imported. Have a look around.

Summary

Well, this has been an exciting journey. In this concluding chapter, we learned how to export and import, as well as package, our work.

You are now ready to not only develop your own workflows, but also share them.

In this book, we could not address all the things that Orchestrator is capable of doing, and we barely touched on the possibilities that it offers. If you regard Orchestrator as a tool that is worth your while, which I surely think it is, you may like to have a look at another book that I have written, *vRealize Orchestrator Cookbook*, where almost 400 pages are dedicated to all things possible. The book also comes with more than 100 ready-to-run workflows that are available for download.

So, see you around in the Orchestrator community (see *Chapter 1*, *Architectural Overview*). Have fun!

Index

Thank you for buying

VMware vRealize Orchestrator Essentials

About Packt Publishing

Packt, pronounced 'packed', published its first book, *Mastering phpMyAdmin for Effective MySQL Management*, in April 2004, and subsequently continued to specialize in publishing highly focused books on specific technologies and solutions.

Our books and publications share the experiences of your fellow IT professionals in adapting and customizing today's systems, applications, and frameworks. Our solution-based books give you the knowledge and power to customize the software and technologies you're using to get the job done. Packt books are more specific and less general than the IT books you have seen in the past. Our unique business model allows us to bring you more focused information, giving you more of what you need to know, and less of what you don't.

Packt is a modern yet unique publishing company that focuses on producing quality, cutting-edge books for communities of developers, administrators, and newbies alike. For more information, please visit our website at `www.packtpub.com`.

About Packt Enterprise

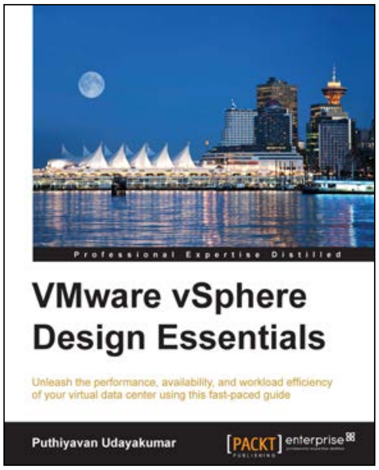
In 2010, Packt launched two new brands, Packt Enterprise and Packt Open Source, in order to continue its focus on specialization. This book is part of the Packt Enterprise brand, home to books published on enterprise software – software created by major vendors, including (but not limited to) IBM, Microsoft, and Oracle, often for use in other corporations. Its titles will offer information relevant to a range of users of this software, including administrators, developers, architects, and end users.

Writing for Packt

We welcome all inquiries from people who are interested in authoring. Book proposals should be sent to `author@packtpub.com`. If your book idea is still at an early stage and you would like to discuss it first before writing a formal book proposal, then please contact us; one of our commissioning editors will get in touch with you.

We're not just looking for published authors; if you have strong technical skills but no writing experience, our experienced editors can help you develop a writing career, or simply get some additional reward for your expertise.

[PACKT] enterprise
PUBLISHING professional expertise distilled

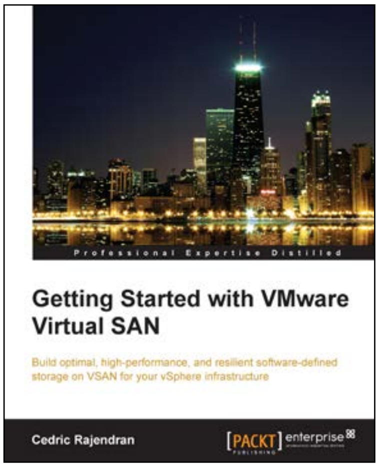

Professional Expertise Distilled
VMware vSphere Design Essentials
Unleash the performance, availability, and workload efficiency of your virtual data center using this fast-paced guide
Puthiyavan Udayakumar
[PACKT] enterprise

VMware vSphere Design Essential

ISBN: 978-1-78439-004-4 Paperback: 176 pages

Unleash the performance, availability, and workload efficiency of your virtual data center using this fast-paced guide

1. Explore methodologies that are applied during the Initial, Planning, Analysis, and Design phases of a vSphere implementation.
2. Familiarize yourself with planning, deploying, maintaining, and optimizing vSphere solutions with ease.
3. Seamlessly configure and deploy vSphere with the help of easy-to-follow examples in this fast-paced guide.


Professional Expertise Distilled
Getting Started with VMware Virtual SAN
Build optimal, high-performance, and resilient software-defined storage on VSAN for your vSphere infrastructure
Cedric Rajendran
[PACKT] enterprise

Getting Started with VMware Virtual SAN

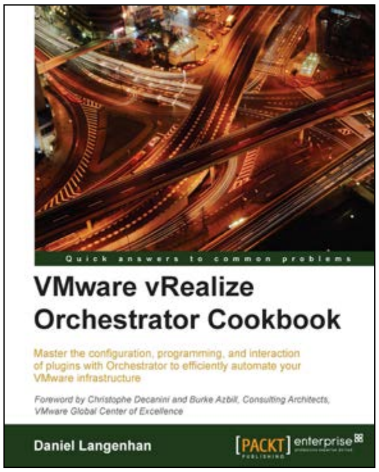
ISBN: 978-1-78439-925-2 Paperback: 162 pages

Build optimal, high-performance, and resilient software-defined storage on VSAN for your vSphere infrastructure

1. Effectively understand the fundamental concepts and features of Virtual SAN.
2. Implement and administer your VMware VSAN efficiently.
3. Ensure stability and data reliability and meet service-level agreements for each virtual machine and application.

Please check **www.PacktPub.com** for information on our titles

[PACKT] enterprise
professional expertise distilled
PUBLISHING


Quick answers to common problems
VMware vRealize Orchestrator Cookbook
Master the configuration, programming, and interaction of plugins with Orchestrator to efficiently automate your VMware infrastructure
Foreword by Christophe Decanini and Burke Azbill, Consulting Architects, VMware Global Center of Excellence
Daniel Langenhan
[PACKT] enterprise

VMware vRealize Orchestrator Cookbook

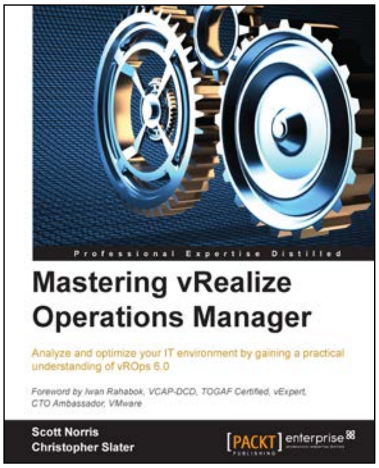
ISBN: 978-1-78439-224-6 Paperback: 382 pages

Master the configuration, programming, and interaction of plugins with Orchestrator to efficiently automate your VMware infrastructure

1. Program with Orchestrator to automate and synchronize your infrastructure.

2. Integrate the base plug-ins into your workflows.

3. Packed with over 100 example workflows, packaged for download and reuse.


Professional Expertise Distilled
Mastering vRealize Operations Manager
Analyze and optimize your IT environment by gaining a practical understanding of vROps 6.0
Foreword by Iwan Rahabok, VCAP-DCD, TOGAF Certified, vExpert, CTO Ambassador, VMware
Scott Norris
Christopher Slater
[PACKT] enterprise

Mastering vRealize Operations Manager

ISBN: 978-1-78439-254-3 Paperback: 272 pages

Analyze and optimize your IT environment by gaining a practical understanding of vROps 6.0

1. Get complete control of capacity management in your virtual environment.

2. Display the most appropriate performance metrics and assemble your own dashboard.

3. Analyze and process data from different sources into a single repository, allowing you to understand every layer of your environment.

Please check **www.PacktPub.com** for information on our titles