vCenter-Tutorial, Teil 2: Komplexe Workflows für den vCenter Orchestrator

Etüde

Guido-Arndt Söldner, Jens-Henrik Söldner

VMwares vCenter Orchestrator ist ein bei vSphere mitgeliefertes Werkzeug zum Automatisieren von IT-Prozessen in virtualisierten Umgebungen. Der zweite Teil des Tutorials beschäftigt sich mit dem Erstellen von Workflows, die komplette Prozesse abbilden und steuern können.

Im ersten Teil des Tutorials ging es nach einer Einführung in die Architektur des vCenter Orchestrator (vCO) von VMware um den konkreten Einsatz. Der zweite Teil zeigt aus welchen grundlegenden Elementen Workflows aufgebaut sein können.

Ein erster eigener Workflow

Von Haus aus bietet der vCO eine große Zahl vorgefertigter Workflows und Aktionen. In der Regel sind Erstere dafür ausgelegt, mit genau einer Elementinstanz (etwa einer virtuellen Maschine) zu interagieren und sie zu ändern. Darüber hinaus kann es sinnvoll und praktisch sein, mehrere virtuelle Maschinen (VMs) auf einmal zu ändern. Typische Szenarien sind, dass alle virtuelle Maschinen, die in einem bestimmten Ordner oder im Resource Pool abgelegt sind, modifiziert werden sollen.

Aktionen hingegen stellen elementare Bausteine dar, die in Workflows als Funktionen fungieren können. Sie dienen etwa dazu, zu überprüfen, ob ein Windows-Betriebssystem in einer VM läuft, eine IP-Adresse gültig ist oder man verwendet sie, um einen Ordner im VirtualCenter anzulegen oder Benutzerkonten im Active Directory einzurichten. VMware liefert vCO mit Hunderten fertiger Aktionen, die letzten Endes die Mächtigkeit des Systems zu großen Teilen mit ausmachen.

Das Ziel des ersten Workflows liegt auf der Hand: Er soll von jeder VM einen Snapshot erzeugen. Dazu erhält er als Eingabeparameter die Namen der virtuellen Systeme.

Damit man die eigenen Workflows im Auge behalten kann, empfiehlt es sich, im vCO in der Designansicht zunächst einen neuen Ordner anzulegen. Danach führt ein Rechtsklick auf den Ordner zur Option, einen neuen Workflow anzulegen – im Rahmen des Tutorials bietet sich als Name „Create Snapshots“ an. Daraufhin lässt sich die Workflow-Design-Maske mit ihren Registerkarten (siehe Abbildung 1) öffnen.

In der ersten Registerkarte „General“ erscheint der Name des Workflows sowie dessen Beschreibung samt Version.

Ein wichtiger Bestandteil des Workflow-Konzeptes von vCO sind Attribute. Sie dienen innerhalb eines Workflows als globale Variablen und ermöglichen es, Informationen zwischen den einzelnen Schritten auszutauschen. Wenn eine Workflow-Komponente die Attribute lesen oder schreiben soll, ist es jedoch wiederum unumgänglich, die betroffenen Attribute als Ein- oder Ausgabeparameter der aufzurufenden Komponente zu definieren. Falls ein Attribut mit der Markierung „Read-only“ versehen ist, gilt es als Konstante. Zusätzlich darf man die Attribute mit Werten vorbelegen.

Für den ersten Workflow sind zwei Attribute erforderlich:
– Das Attribut *vmNB* speichert zuerst die Gesamtzahl der zu erstellenden Snapshots bedingt durch die Zahl der VMs. Nach jedem Snapshot soll es der Workflow um eins verringern, bis 0 erreicht ist.
– Mit dem Attribut *currentVM* erhält der Workflow eine Referenz auf die aktuell zu bearbeitende VM.

Beim Anlegen von Attributen oder Ein- und Ausgabeparametern verlangt vCO eine Angabe des Typs der Variablen. Dabei gibt es folgende Typen:
– Basisdaten wie String, Number oder Boolean,
– für JavaScript wie Date oder RegExp,
– generische (wie Any, Properties),

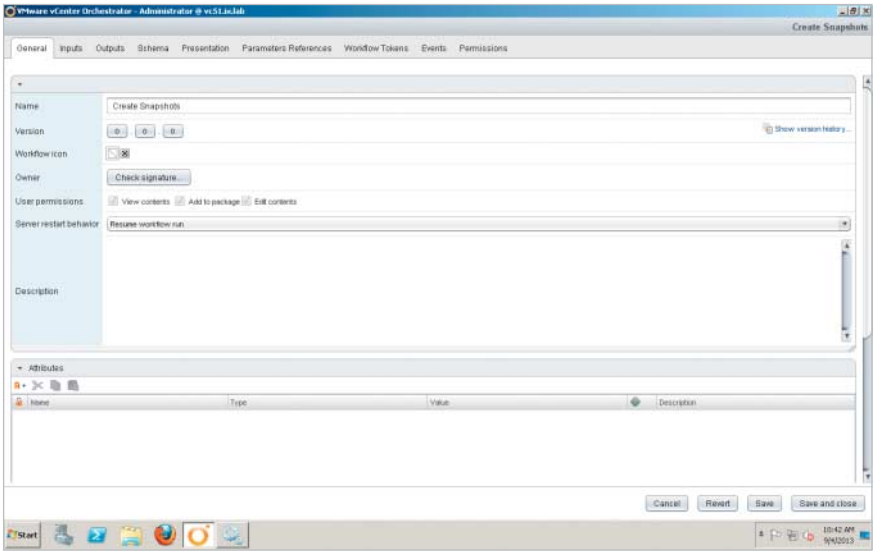
iX-TRACT

- Mit dem vCenter Orchestrator können Administratoren Workflows anlegen, ohne sich spezielle Programmierkenntnisse aneignen zu müssen.
- Das Festlegen der Parameter geschieht über die grafische Oberfläche.
- Zusätzlich lassen sich über Skripte Logs generieren, die bei der Fehleranalyse helfen.

– Composite – darunter fasst vCO Attribute beziehungsweise Eingabeparameter unterschiedlichen Typs zusammen, die aber als logische Einheit gruppiert sind,
– vCO-interne Daten wie Workflows, Actions oder Workflow-Tokens und
– Daten, wie sie Plug-ins bereitstellen.

Beim Benennen der Variablen folgt Orchestrator der „CamelCase“-Schreibweise (siehe „Alle Links“). Variablen dürfen keine Leerzeichen und sollten tunlichst keine Sonderzeichen enthalten.

Unter „Attributes“ im unteren Teilbereich des Menüs erscheint als Erstes die Variable *vmNB*. Sie soll den Datentyp *number* erhalten, die zweite *currentVM VC:VirtualMachine*. Dazu klickt man jedes Mal auf „Type“, gibt im Feld „Filter“ das gesuchte Muster ein und wählt den passenden Datentyp anschließend aus der Liste aus (siehe Abbildung 2).



Taufe: In der „General“-Ansicht kann man dem Kind einen Namen geben und die Versionshistorie einsehen (Abb. 1).

Dem Namen nach

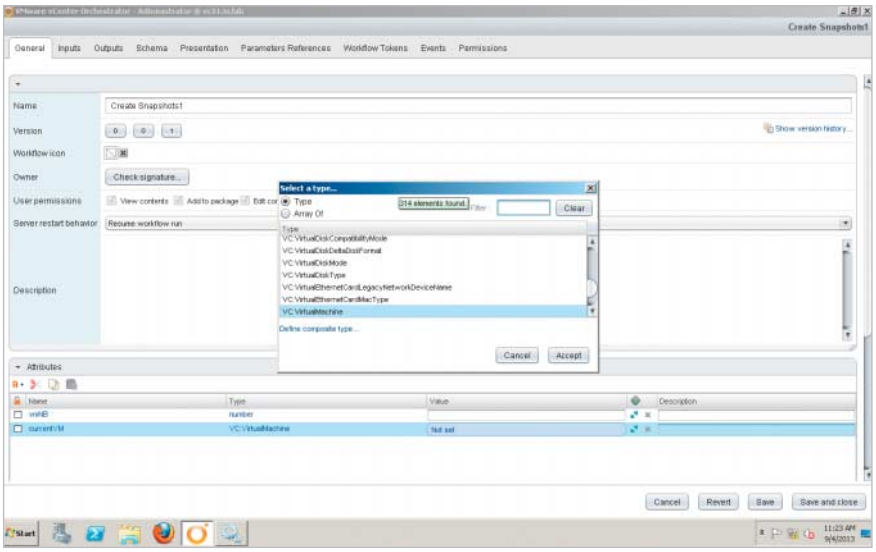
Damit der Workflow Snapshots erstellen kann, benötigt er die Namen der VMs. Diese Information akzeptiert er als Eingabeparameter. Der Benutzer kann sie entweder manuell eingeben, oder der Workflow kann sie von einem anderen, ihn aufrufenden oder einem externen System erhalten. Wie bei einem Attribut besteht jeder Eingabeparameter aus Namen und Datentyp.

Für das Beispiel soll die Registerkarte „Inputs“ den Eingabeparameter *vms* vom Typ *Array/VC:VirtualMachine* erhalten. Hier heißt es aufpassen, dass wirklich ein Feld aus virtuellen Maschinen entsteht (siehe Abbildung 3).

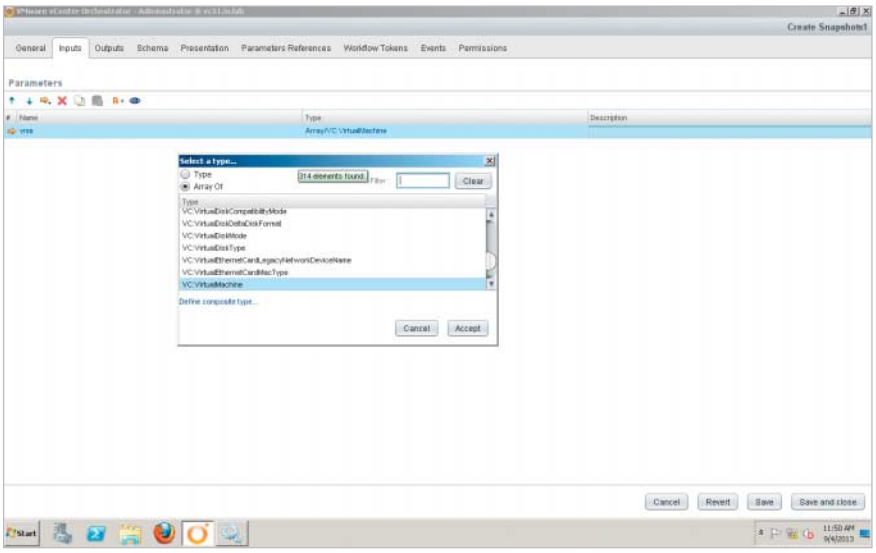
Nach diesen Vorarbeiten führt der nächste Schritt zum Implementieren der Logik, wozu ein Wechsel auf die Registrierkarte „Schema“ erforderlich ist (Abbildung 4).

Eine erste Skript-Komponente initialisiert die übergebenen Variablen und stellt die Anzahl der zu erzeugenden Snapshots fest. Eine „Custom Decision“ dient daraufhin dazu, die Aufträge abzuarbeiten. Sind für alle virtuellen Maschinen Snapshots erzeugt, endet der Workflow.

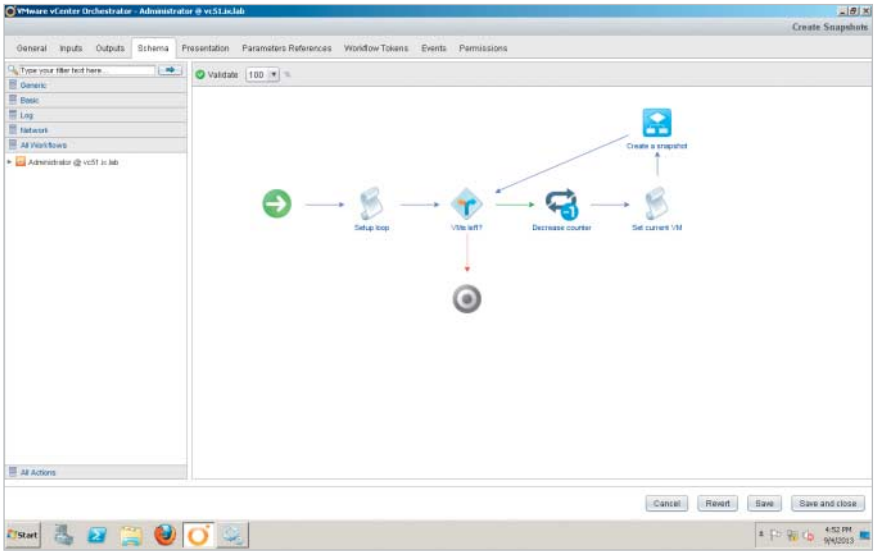
Dabei zählt die Komponente *Decrease Counter* in jedem Schleifendurchlauf die Variable *vmNB* herunter. Eine weitere legt die zu bearbeitende virtuelle Maschine fest. Und der letzte Schritt in der Schleife erzeugt schließlich den Snapshot.



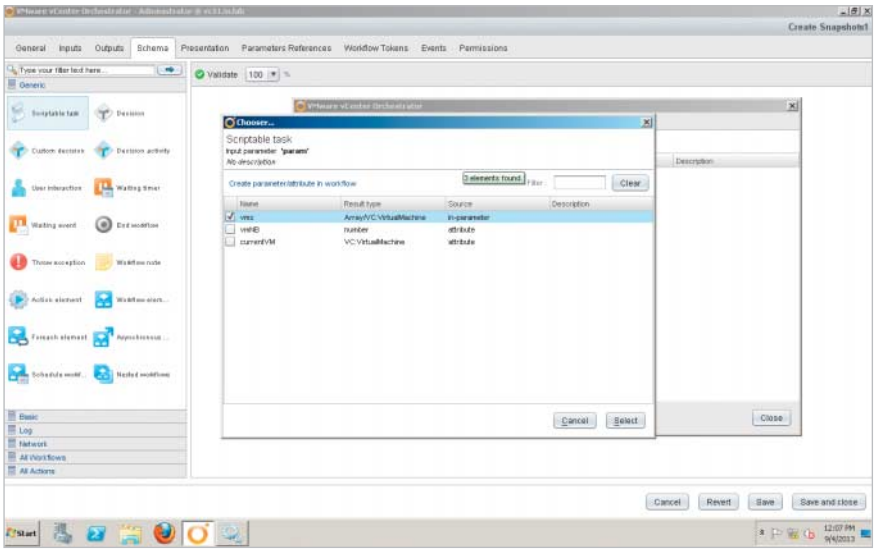
Typ gesucht: Der vCenter Orchestrator zeigt hier alle Datentypen an, die sich auf VirtualCenter beziehen (Abb. 2).



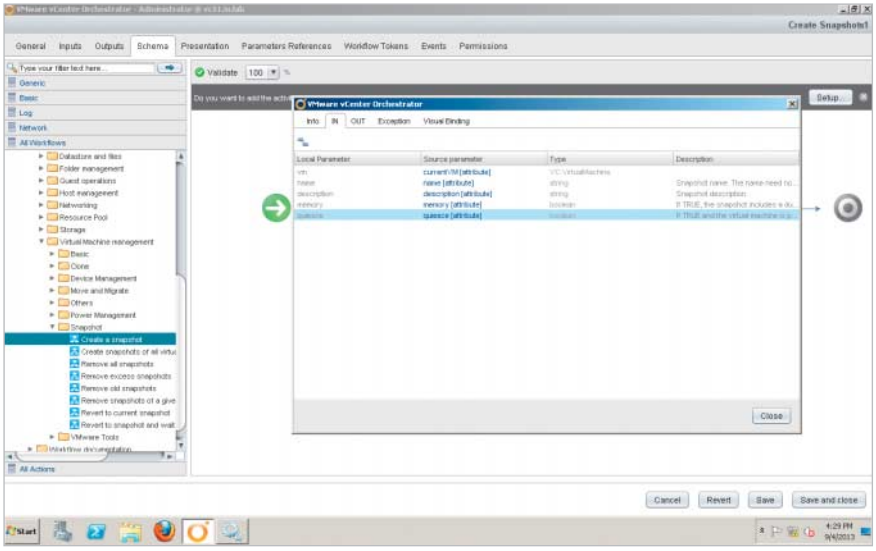
Eingabe: Das Array *VC:VirtualMachine* soll die Namen der virtuellen Maschinen aufnehmen (Abb. 3).



Ablauf: Die Logik des Workflows stellt vCO in einem Diagramm dar, im Zentrum die „Custom decision“ „VMs left?“ (Abb. 4).



Anlagen: Für den Eingabeparameter *vms* ist hier als Quelle der Array *VC:VirtualMachine* gewählt (Abb. 5).



Nach der konzeptionellen Besprechung des Workflows geht es ans schrittweise Einrichten des Workflows: Als Erstes bedarf es der Skript-Komponente (*scriptable task*) zum Initialisieren der Variablen. Diese lässt sich mit Drag & Drop vom linken Teilbereich zwischen den Anfangs- und Endknoten ziehen. Um sie einzurichten, fährt man mit dem Mauszeiger über das Element und aktiviert das Stiftsymbol, woraufhin vCO den Editiermodus öffnet. Auf der ersten Registrierkarte „Info“ hinterlegt man gleich einen passenden Namen für die Komponente wie „Setup loop“. Da es die Aufgabe der Skript-Komponente ist, die Variable *vmNB* zu setzen, braucht sie die entsprechenden Ein- und Ausgabeparameter. Auf der Registerkarte „IN“ aktiviert man dazu das „Bind to workflow parameter/attribute“-Icon und wählt den zuvor erstellten Eingabeparameter *vms* aus (siehe Abbildung 5).

Komponente konfigurieren

Ähnlich geht man mit dem Ausgabeparameter auf der Registrierkarte „OUT“ vor – die Zahl der virtuellen Maschinen soll im Attribut *vmNB* stehen. Zum Abschluss fehlt noch das auszuführende Skript, das man auf der Registrierkarte „Scripting“ eingibt. Es lautet einfach:

```
var vmNB = vms.length;
```

und belegt die Variable *vmNB* mit der Länge der *vms*-Arrays.

Die nächste zu konfigurierende Komponente lautet „Custom decision“. Deren Aufgabe ist es, die Schleifensteuerung zu übernehmen. Wie in Abbildung 4 zu erkennen, hat sie zwei Ausgänge:
– Falls vCO für alle VMs Snapshots erzeugt hat, folgt der Kontrollfluss dem roten Pfeil: Der Workflow ist beendet.
– Sind dagegen noch Snapshots zu erstellen, kehrt der Ablauf dem blauen Pfeil folgend in den Rumpf des Workflows zurück.

Nachdem man die „Custom decision“ in den Workflow gezogen hat, muss man sie noch konfigurieren. Auf der Registerkarte „IN“ kann man wie oben beschrieben die Eingabeparameter binden, in diesem Fall das Attribut *vmNB*. Als Nächstes folgt der Wechsel zur Karteikarte „Scripting“. Dort gibt man ein:

```
return vmNB > 0;
```

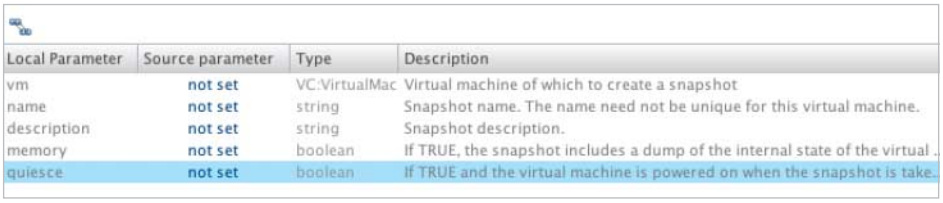
Unterpunkt: Das Erzeugen von Snapshots gehört zu den vorgefertigten Workflows (Abb. 6).

Danach kann man das Dialogfeld der „Custom decision“ schließen, nachdem man der Bedingung auf dem Karteireiter „Info“ noch einen Namen verpasst hat, etwa „VMs left?“.

Jetzt fehlt noch das Implementieren des Schleifenrumpfes. Er braucht für das Übungsbeispiel zuerst die Aktion „Decrease counter“. Man erzeugt sie, indem man das vorgefertigte Element „Decrease counter“ aus dem Inventarbereich links aus der Kategorie „Basic“ auswählt und in den Workflow zwischen dem grünen Pfeil nach der Custom Decision und dem Endelement per Drag & Drop einfügt.

Die vorgefertigte Aktion verfügt über eine lokale Variable namens *counter*, die mit *vmNB* als In- und Out-Parameter verbunden sein muss, was man unter den Karteireitern „IN“ beziehungsweise. „OUT“ direkt unter der Spalte „Source parameter“ erledigen kann. Auf der Registerkarte „Visual Binding“ lässt sich das Ergebnis noch mal optisch überprüfen.

Nachdem man den Zähler dekrementiert hat, muss als Nächstes das Attribut *currentVM* gesetzt sein. Hier behilft man



Local Parameter	Source parameter	Type	Description
vm	not set	VC:VirtualMac	Virtual machine of which to create a snapshot
name	not set	string	Snapshot name. The name need not be unique for this virtual machine.
description	not set	string	Snapshot description.
memory	not set	boolean	If TRUE, the snapshot includes a dump of the internal state of the virtual ..
quiesce	not set	boolean	If TRUE and the virtual machine is powered on when the snapshot is take..

Bindungen: Die benötigten Parameter lassen sich hier an Attribute koppeln (Abb. 7).

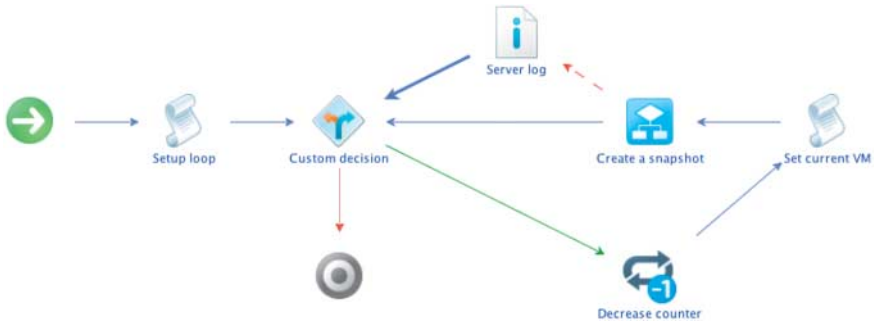
sich wieder mit der inzwischen bekannten Skript-Komponente „Scriptable task“ aus dem Inventarbereich „Generic“ und zieht diese hinter das Element „Decrease counter“ in den Workflow. Als Name für die Komponente bietet sich zum Beispiel „Set current VM“ an. Damit sie ihre Aufgabe erfüllen kann, erhält sie die Attribute *vms* und *vmNB* als Eingabeparameter auf der Registerkarte „IN“. Da *currentVM* im Anschluss die richtige virtuelle Maschine speichern soll, bindet man das Attribut in einem zweiten Schritt noch unter der Registerkarte „OUT“. Zum Schluss fehlt noch unter „Scripting“:

```
currentVM = vms[vmNB];
```

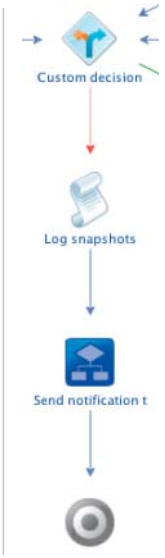
Nun kann man als Nächstes das Erzeugen der Snapshots konfigurieren. Hier kommen die bereits erwähnten vorgefertigten Workflows des Orchestrators ins Spiel, die man entweder direkt oder von einem anderen Workflow aus aufrufen kann. Für das Übungsbeispiel ist der Workflow „Create a snapshot“ relevant, der sich im grafischen Workflow-Editor in der Kategorie „All Workflows“ und dann unter „Library -> vCenter -> Virtual Machine management -> Snapshot“ befindet. Man zieht ihn einfach hinter die „Set current VM“-Komponente in die Designer-Pane.

Etwas aufwendiger als zuvor ist das Konfigurieren. Wieder geht es ans Binden der Eingabeparameter an die Workflow-Attribute, die lokale Variablen innerhalb des Workflows darstellen (siehe Abbildung 6). Der lokale Parameter *vm* kommt

Ausnahmen: Die Behandlung von Fehlern sollte mit berücksichtigt sein (Abb. 8).



Ausweg: Für die Fehlerbehandlung zieht man einfach „Serverlog“ auf „Create a snapshot“ und erhält einen hinausführenden Pfad (Abb. 9).



Berichte: Durch das Hinzufügen der Funktion „Send Notification“ erhält man Nachrichten über die Ereignisse im Workflow (Abb. 10).

zum Attribut *currentVM*. Für die anderen lokalen Parameter (*name*, *description*, *memory*, *quiesce*) existieren jedoch noch keine Attribute im Workflow.

Um das nachzuholen, klickt man einfach in der Spalte „Source parameter“ auf „not set“ und folgt im sich daraufhin öffnenden Dialogfeld dem Link „Create parameter/attribute in workflow“. Daraufhin kann man das neue Attribut im äußeren Workflow unter dem gleichen Namen wie das lokale Attribut erzeugen, indem man die Standardwerte gesetzt lässt und über OK die Auswahl verlässt (siehe Abbildung 7).

Dieses Vorgehen wiederholt sich, bis alle noch fehlenden Attribute gesetzt sind. Auf der Registerkarte „OUT“ verknüpft man mit der gleichen Vorgehensweise anschließend den Rückgabewert des „Create a snapshot“-Workflows an ein ebenfalls neu erstelltes Attribut, selbst wenn Workflow ihn nicht weiter verwendet. Ist er vom Typ *VC:VirtualMachineSnapshot*, wäre es möglich, im Nachgang noch weitere Aktionen mit dem Snapshot durchzuführen. Das Dialogfeld zum eingebundenen „Create a snapshot“-Workflow kann man nun mit „Close“ schließen.

Nun geht es zurück zur Registerkarte „General“ des Gesamt-Workflows. Sie erwartet die Standardwerte für die gerade eben erzeugten Attribute. Die Boolean-Variablen *memory* und *quiesce* erhalten den Wert „No“, für die Snapshot-Namen unter *name* und die Beschreibung bei *description* trägt man etwas Passendes ein, etwa beide Male „vCO-TestSnapshot“.

Zu guter Letzt muss der Kontrollfluss noch geändert werden, sodass er nach dem aufgerufenen „Create a snapshot“-Workflow zurück zur Schleifenbedingung „VMs left?“ springt. Das erreicht man, indem man das nach dem „Create a snapshot“ folgende Zielelement einfach auf das „VMs left?“-Entscheidungselement zieht und zur Übersichtlichkeit den „Create a snapshot“-Teil nach oben verschiebt. Der Gesamt-Workflow sollte danach wie in Abbildung 4 aussehen.

Nun ist das Erstellen des Workflows abgeschlossen. Beim Speichern validiert vCO ihn automatisch, die Prüfung lässt sich aber auch separat aufrufen. Beim Starten fragt der Orchestrator ab, welche virtuellen Maschinen er übergeben soll. Dazu stellt man menügeführt eine Verbindung zum vCenter-Server her und wählt die gewünschten aus.

Bei Fehlern Workflow fortsetzen

An dieser Stelle sei noch darauf hingewiesen, dass die ganze Ablaufsteuerung theoretisch in einem einzigen Skript implementierbar wäre. Trotzdem ist es sinnvoll, den Workflow wie beschrieben einzurichten. Denn der Orchestrator setzt während der Ausführung Checkpoints in seiner Datenbank und zwar jedes Mal, wenn er eine Aufgabe im Kontrollfluss bearbeitet hat. Der Benutzer kann somit im Fehlerfall den Workflow an der abgebrochenen Stelle weiter fortsetzen.

Tutorialinhalt

Teil 1: Einrichten des vCenter Orchestrator, erster Workflow

Teil 2: Komplexe Workflows

Teil 3: Das Anbinden von Fremdsystemen

Nichtsdestotrotz ist es darüber hinaus noch wünschenswert, im Workflow zusätzlich eine Fehlerbehandlung zu implementieren, was wichtig ist, wenn das Erzeugen des Snapshots fehlschlägt. In dem Fall sollte vCO den Fehler protokollieren und den Workflow mit der nächsten virtuellen Maschine fortsetzen.

Falls ein Fehler beim untergeordneten Workflow auftritt, der den jeweiligen Snapshot erzeugt, folgt der Kontrollfluss dem roten Pfeil in Abbildung 9. Um das in den Workflow einzubauen, zieht man einfach die Komponente „Serverlog“ aus der Kategorie „Log“ auf das Element „Create a snapshot“. Daraufhin erzeugt der vCO den Pfad für die Fehlerbehandlung. Um den Kreis der Schleife im Workflow wieder zu schließen, muss man noch den ausgehenden blauen Pfeil von der „Serverlog“-Komponente zurück zum Entscheidungselement „VMs left?“ ziehen.

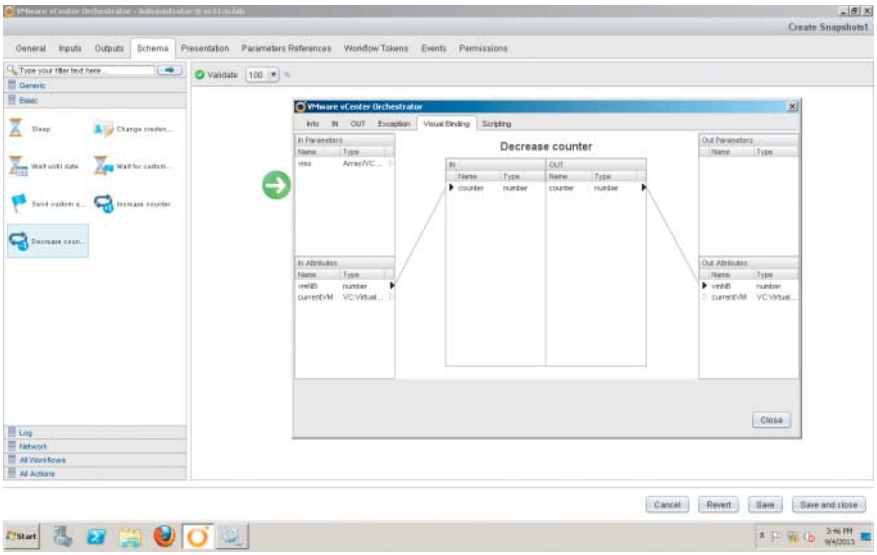
Damit die Log-Komponente die von der Snapshot-Komponente generierte Exception lesen kann, muss man die Snapshot-Komponente so konfigurieren, dass sie im Fehlerfall die Ursache in einem Attribut speichert. Hierzu wechselt man auf die Karteikarte „Exception“ (Abbildung 8) und verwendet den „Not set“-Link, um ein Attribut auszuwählen, das die Exception speichern kann. Nun möchte vCO ein neues Attribut *errorCode* erzeugen, das im Workflow bislang noch nicht vorhanden ist. Wenn man dem Link „Create parameter/attribute in workflow“ folgt und das folgende Dialogfeld mit OK bestätigt, erzeugt vCO es.

Jetzt stellt man noch die „Serverlog“-Komponente ein, damit sie die entsprechenden Werte auslesen und ins Log übertragen kann. Um das zu erreichen, legt man bei „IN“ die Parameter *text* und *object* fest. Im ersten Fall handelt es sich um ein generisches Fehlerattribut, dem man einen vordefinierten Wert, zum Beispiel „Error during snapshot creation“ ge-

Listing: Die Berichtsfunktion

```
var logText;

for(i in vms){   var vm = vms[i];   if(vm.runtime.connectionState.value =="connected" && !vm.config.template)
     {
    var actionResult = System.getModule("com.vmware.library.vc.vm.snapshot").getAllSnapshotsOfVM(vm);
     if(actionResult.length > 0){
       System.log("VM name: " + vm.name + " Number of snapshots : " + actionResult.length);
       logText = "VM name: " + vm.name + " Number of snapshots : " + actionResult.length;
       content = content + "<br>" + logText;
       }
   }
 }
 var vmsLength=vms.length; System.log("VMs have snapshots: " + vmsLength); subjects = "VMs have snapshots: " + vmsLength;
```



Zusammenhang: Die Zuordnung der Variablen an die Ein- und Ausgabe stellt vCO grafisch dar (Abb. 11).

ben kann. Dem zweiten Parameter weist man das Attribut „errorCode“ zu. Zum Schluss kann man die Log-Ausgabe in der Designansicht mit der „Custom decision“ verbinden, damit der Kontrollfluss wieder an die Schleifensteuerung zurückgeht.

Auflisten von vorhandenen Snapshots

Der Workflow ist damit fertiggestellt. In der täglichen Praxis will man jedoch wissen, wie viele Snapshots existieren, schließlich belegen sie Speicherplatz und bedürfen einer regelmäßigen Kontrolle. Deshalb könnte man den hier vorgestellten Workflow um eine Berichtsfunktion ergänzen, realisiert durch zwei weitere Komponenten: eine Protokollierungskomponente als *scriptable task*, hier „Log Snapshots“ genannt und einen im vCO mitgelieferten Workflow zum E-Mail-Versand, der im Workflow-Selektor als „Send notification“ abrufbar ist und in der Bibliothek „Library/Mail“ liegt (Abbildung 10).

Alternativ bietet sich das Erstellen eines eigenen separaten Workflows an, der einen Bericht mit allen vorhandenen Snapshots erzeugt und per E-Mail verschickt. Dank der im vCO eingebauten Scheduling-Funktion kann man diesen Bericht dann regelmäßig erzeugen und verschicken lassen.

Der Quellcode für die Skript-Komponente steht im Listing-Kasten und kann als Ausgangspunkt für eine eigene Umsetzung dienen, die hier nur skizzenhaft besprochen wird.

Um die Liste aller Snapshots zu generieren, erzeugt der Workflow zuerst ein Array, in dem er die VMs mit Snapshots speichert. In einer For-Schleife durchläuft er die Liste der virtuellen Maschinen. Falls eine VM im vCenter sichtbar und zugreifbar ist per

```
vm.runtime.connectionState.value =="connected"
```

und es sich um kein Template handelt, kann vCO die Liste aller Snapshots laden.

Im Folgenden veranlasst das Skript das Speichern der Informationen in den Variablen *content* und *subjects*. Da die E-Mail-Komponente dieselben Informationen benötigt, müssen die Variablen an entsprechende Workflow-Attribute gebunden sein. Als Eingabeparameter dient wiederum die Variable *vms*. Der letzte Schritt im Workflow besteht darin, die Komponente „Send notification to mailing list“ in den Workflow einzubauen. Dabei müssen die Betreffzeile (subject) und der Inhalt (content) an die vorher erzeugten Attribute gebunden sein. Zusätzlich muss man die anderen benötigten Attribute (etwa für den Mailserver oder -adresse) erzeugen und mit entsprechenden Werten füllen. Damit ist der Teil des Workflows zum Übermitteln von Nachrichten über den Verlauf eingebaut.

Dem Erzeugen von Workflows folgt in der nächsten Ausgabe ein weiterer Schritt: Das Einbinden von Fremdsystemen in den vCenter Orchestrator. (rh)

Dr. Guido-Arndt Söldner

ist Dozent für Wirtschaftsinformatik an der FOM Hochschule für Oekonomie & Management und beschäftigt sich mit den Themen Automatisierung und Programmierung bei der Söldner Consult GmbH in Nürnberg.

Jens-Henrik Söldner

ist Dozent für Wirtschaftsinformatik an der FOM Hochschule für Oekonomie & Management und leitet das Infrastruktur-Consulting bei der Söldner Consult GmbH in Nürnberg.

Literatur

[1] Jörg Riether; Servervirtualisierung; Doppelsteuerung; Versionsschritte in vSphere 5.1; *iX* 11/2012, S. 66

[2] Sven Ahnert, André Dannbacher, Mathias Ewald, Jörg Riether, Jens-Henrik Söldner; Virtualisierung; Alles in allem; Aufwand und Kosten der Hypervisors: Hyper-V, XenServer, vSphere und KVM; *iX* 8/2012, S. 82

Alle Links: www.ix.de/ix1310152