

Использование клиента VMware vRealize Orchestrator

6 октября 2020 г.
vRealize Orchestrator 8.2

Актуальная техническая документация доступна на веб-сайте VMware:

<https://docs.vmware.com/ru/>

VMware, Inc.
3401 Hillview Ave.
Palo Alto, CA 94304
www.vmware.com

VMware Россия
Россия, 125284, г. Москва
ул. Беговая, д.3, стр.1
Бизнес-центр "NORDSTAR TOWER" 30й этаж
Телефон: +7 495 212 29 00
www.vmware.com/ru

© 2008–2020 VMware, Inc. Все права защищены. [Информация об авторских правах и товарных знаках.](#)

Содержание

1	Использование клиента VMware vRealize Orchestrator	5
2	Клиент VMware vRealize Orchestrator	6
	Панель управления использованием клиента vRealize Orchestrator	7
	Упорядочение содержимого в vRealize Orchestrator Client	8
	Создание папки или вложенной папки в клиенте vRealize Orchestrator	9
3	Настройка vRealize Orchestrator Client	12
	Управление ролями и группами в клиенте vRealize Orchestrator	12
	Назначение ролей в vRealize Orchestrator Client	14
	Создание групп в vRealize Orchestrator Client	14
	Настройка ролей клиента vRealize Orchestrator в vRealize Automation	15
	Журнал версий объектов vRealize Orchestrator	16
	Откат к более ранней версии рабочего процесса	16
	Визуальное сравнение версий рабочих процессов	17
	Возврат иерархии содержимого vRealize Orchestrator в предыдущее состояние с помощью Git	18
4	Примеры использования vRealize Orchestrator	19
	Интеграция Amazon Web Services и vRealize Orchestrator с помощью Python	19
	Создание базового сценария Python	20
	Создание действия Amazon Web Services	21
	Отладка действия Amazon Web Services	22
	Обновление действия службы Amazon Web Services	25
	Использование ветвей Git для управления иерархией объектов vRealize Orchestrator	26
	Подготовка среды GitLab	27
	Настройка подключения к репозиторию Git	28
	Отправка изменений в репозиторий Git	29
	Использование модулей сторонних разработчиков для вызова API-интерфейса проекта vRealize Automation	31
	Создание сценария Python для вызова API-интерфейса проекта vRealize Automation	31
	Создание сценария Node.js для вызова API-интерфейса проекта vRealize Automation	34
	Создание сценария PowerShell для вызова API-интерфейса проекта vRealize Automation	36
5	Управление рабочими процессами	40
	Стандартные рабочие процессы в библиотеке рабочих процессов vRealize Orchestrator	41
	Создание рабочих процессов в клиенте vRealize Orchestrator	41
	Изменение рабочих процессов и действий в окне родительского рабочего процесса	42

Конструктор форм ввода vRealize Orchestrator	42
Создание диалогового окна входных параметров рабочего процесса в клиенте vRealize Orchestrator	43
Свойства входных параметров в клиенте vRealize Orchestrator	43
Использование действий для проверки входных данных рабочих процессов vRealize Orchestrator	44
Запросы на взаимодействие с пользователем в клиенте vRealize Orchestrator	46
Планирование рабочих процессов в клиенте vRealize Orchestrator	46
Редактирование назначенных задач в клиенте vRealize Orchestrator	47
6 Управление действиями	48
Создание действий в клиенте vRealize Orchestrator	48
Запуск и отладка действий	49
Запуск действий в клиенте vRealize Orchestrator	50
Отладка действий в клиенте vRealize Orchestrator	50
Основные принципы работы со сценариями Python, Node.js и PowerShell	51
Ограничения среды выполнения для сценариев Python, Node.js и PowerShell	53
7 Управление элементами конфигурации	54
Создание элементов конфигурации в клиенте vRealize Orchestrator	54
8 Управление политиками	56
Создание и применение политик в клиенте vRealize Orchestrator	56
Элементы политик в клиенте vRealize Orchestrator	57
Управление запуском политик в клиенте vRealize Orchestrator	58
9 Управление элементами ресурсов	59
10 Управление пакетами	60
Создание пакета в клиенте vRealize Orchestrator	60
Экспорт пакетов в клиенте vRealize Orchestrator	61
Импорт пакетов в клиенте vRealize Orchestrator	62
11 Устранение неполадок в клиенте vRealize Orchestrator	64
Данные показателей в клиенте vRealize Orchestrator	64
Профилирование рабочих процессов в клиенте vRealize Orchestrator	64
Использование панели управления системой vRealize Orchestrator	65
Использование воспроизведения маркера рабочего процесса в клиенте vRealize Orchestrator	66
Проверка рабочих процессов vRealize Orchestrator	67
Проверка рабочего процесса и устранение ошибок проверки в клиенте vRealize Orchestrator	68
Отладка сценариев рабочих процессов в клиенте vRealize Orchestrator	69
Отладка рабочих процессов по элементам схемы	70

Использование клиента VMware vRealize Orchestrator

1

Документ *Использование клиента VMware vRealize Orchestrator* содержит сведения о функциях автоматизации рабочих процессов и возможностях vRealize Orchestrator Client.

Целевая аудитория

Эта информация предназначена для опытных системных администраторов, которым нужны инструментальные средства для запуска рабочих процессов vRealize Orchestrator и управления ими.

Клиент VMware vRealize Orchestrator

2

vRealize Orchestrator Client используется для управления службами и объектами vRealize Orchestrator.

Можно создавать объекты vRealize Orchestrator и управлять ими с помощью vRealize Orchestrator Client. vRealize Orchestrator Client можно найти по адресу https://FQDN_оркестратора/orchestration-ui.

Взаимодействие посредством REST API

vRealize Orchestrator Client запускается на сервере центра управления vRealize Orchestrator. Клиент обменивается данными с REST API vRealize Orchestrator через прокси-сервер REST.

Управление рабочими процессами

В vRealize Orchestrator Client можно создавать, редактировать, планировать, запускать и удалять рабочие процессы.

Управление действиями

В vRealize Orchestrator Client можно создавать, редактировать и удалять действия. Редактор действий поддерживает автоматическое завершение общих элементов сценария, включенных в обозреватель API-интерфейсов vRealize Orchestrator.

Управление политиками

В vRealize Orchestrator Client можно создавать, редактировать, запускать и удалять политики.

Управление конфигурацией

В vRealize Orchestrator Client можно создавать, запускать и удалять элементы конфигурации.

Управление ресурсами

В vRealize Orchestrator Client можно выполнять экспорт, импорт и обновление элементов ресурсов.

Интеграция Git

Можно создать интеграцию с репозиторием Git и использовать ее для управления разработкой рабочих процессов и другими компонентами vRealize Orchestrator в нескольких развертываниях. См. раздел [Использование ветвей Git для управления иерархией объектов vRealize Orchestrator](#).

Данные показателей

Используйте панель управления системой vRealize Orchestrator Client и функцию профилирования для сбора полезных показателей о среде и рабочих процессах vRealize Orchestrator.

Управление пакетами

Можно создавать, удалять, экспортировать и импортировать пакеты с объектами vRealize Orchestrator в vRealize Orchestrator Client.

Управление разрешениями

Пользователи с правами администратора могут назначать в vRealize Orchestrator Client роли другим пользователям и добавлять их в группы.

Обозреватель API-интерфейсов

Ознакомьтесь с командами API-интерфейса, доступными в vRealize Orchestrator Client.

В эту главу входят следующие разделы:

- [Панель управления использованием клиента vRealize Orchestrator](#)
- [Упорядочение содержимого в vRealize Orchestrator Client](#)

Панель управления использованием клиента vRealize Orchestrator

Панель управления vRealize Orchestrator Client предоставляет удобное средство для мониторинга, управления и устранения неполадок рабочих процессов vRealize Orchestrator Client.

Информация на панели управления vRealize Orchestrator Client распределена по пяти панелям.

Окно	Описание
Выполняется рабочий процесс	Визуальные данные о количестве выполняемых, ожидающих и завершившихся сбоем рабочих процессов.
Избранные рабочие процессы	Рабочие процессы, добавленные в избранное.
Ожидание ввода	Приостановленные рабочие процессы, требующие дальнейшего взаимодействия с пользователем. Эти рабочие процессы также отображаются в меню уведомлений в правом верхнем углу пользовательского интерфейса.
Последние выполненные рабочие процессы	Управление последними выполненными рабочими процессами. Показывает имя, состояние, дату начала и дату окончания выполнения рабочего процесса.
Требуется действие	Сбойные запуски и показатели производительности рабочих процессов.

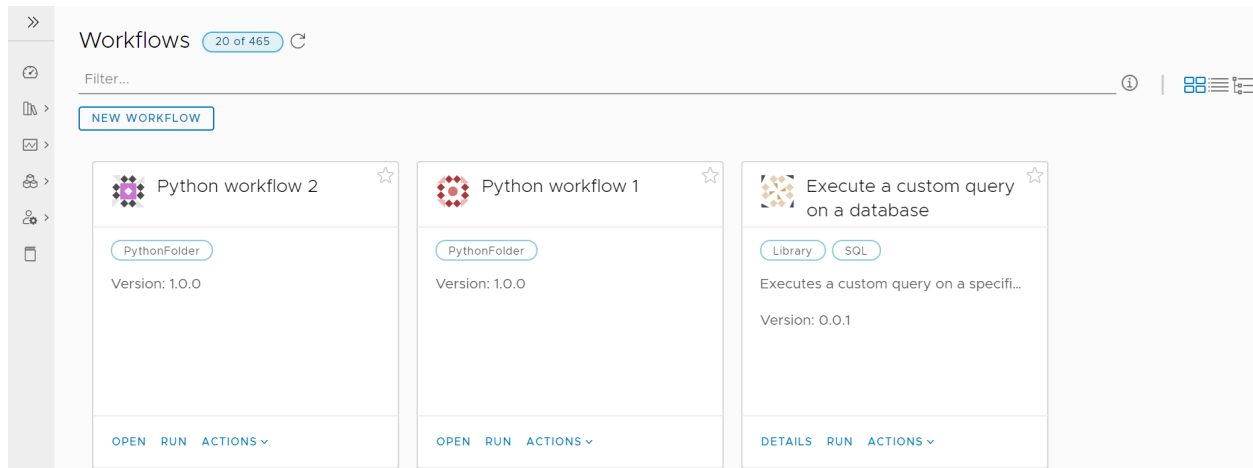
Упорядочение содержимого в vRealize Orchestrator Client

Отображением иерархии объектов vRealize Orchestrator в vRealize Orchestrator Client можно управлять.

vRealize Orchestrator Client поддерживает три режима просмотра различных объектов, таких как рабочие процессы, действия, политики, ресурсы и настройки, а именно: карточки, список и дерево. Текущий режим просмотра можно изменить в правом верхнем углу страницы.

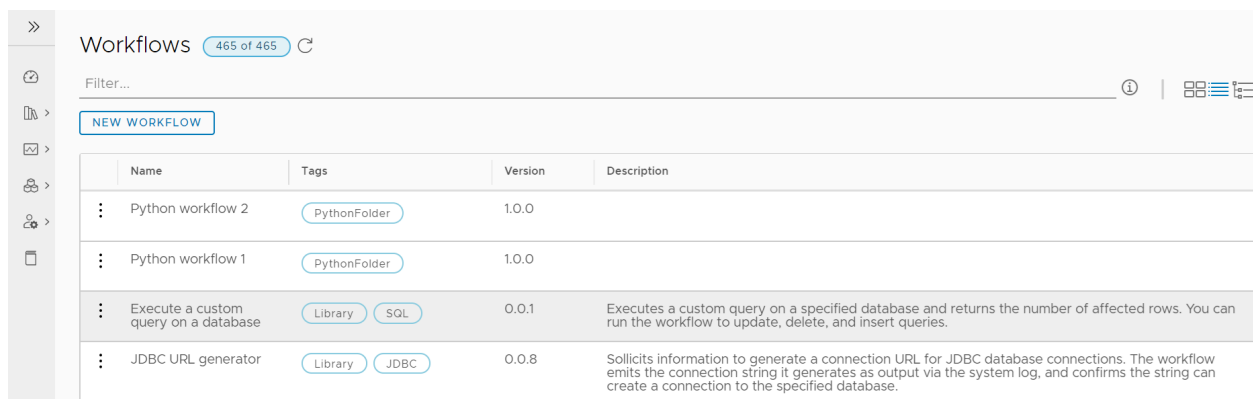
Карточки

В vRealize Orchestrator Client карточки являются режимом просмотра по умолчанию. Сведения о каждом объекте иерархии (например, рабочем процессе) отображаются в отдельном элементе-карточке.



Список

Сведения об объектах vRealize Orchestrator можно отобразить в виде списка. Для получения дополнительных сведений о действиях, которые можно выполнять с объектом, щелкните значок вертикального многоточия слева от объекта.



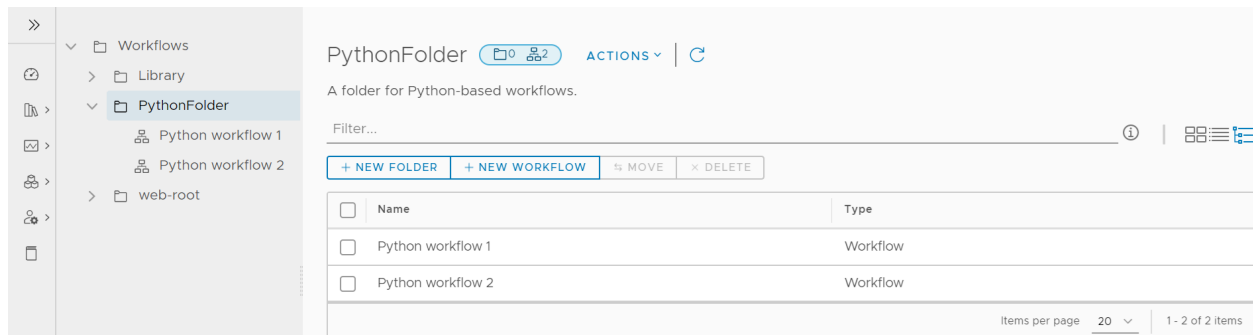
Дерево

В режиме просмотра «Дерево» иерархия объектов распределена по папкам, расположенным иерархически. Каждый тип объекта vRealize Orchestrator имеет папку корневого уровня. В корневой папке нельзя создавать новые объекты, например рабочие процессы. Необходимо создать отдельные папки, размещенные в корневой папке. Каждая папка включает в себя средства управления содержимым, например фильтры.

Примечание Каждая папка имеет отдельный фильтр содержимого. Фильтрация содержимого сразу в нескольких папках невозможна.

Дополнительные сведения о папках см. в разделе [Создание папки или вложенной папки в клиенте vRealize Orchestrator](#).

Примечание Если выбрать объект в дереве, он открывается в режиме только для чтения. Чтобы изменить содержимое объекта, например переменные или схему рабочего процесса, нажмите **Изменить** в меню параметров сверху.



Создание папки или вложенной папки в клиенте vRealize Orchestrator


Объекты vRealize Orchestrator можно упорядочить с помощью иерархической структуры папок.

Можно создавать папки и вложенные папки для упорядочения следующих типов объектов vRealize Orchestrator:

- Рабочие процессы
- Действия
- Политики
- Элементы конфигурации
- Элементы ресурсов

Процедура

1. Выполните вход в vRealize Orchestrator Client.
2. На панели навигации слева выберите страницу объекта, например **Рабочие процессы**.


3. Вверху справа щелкните значок представления «Дерево» ()
4. (дополнительно) Чтобы создать вложенную папку, выберите родительскую папку в древовидном представлении слева.
5. Щелкните **Создать папку**.
6. Введите имя и описание, затем нажмите кнопку **Сохранить**.
7. Добавьте в новую папку объекты или вложенные папки.
8. (дополнительно) Чтобы изменить имя папки, выберите команду **Действия > Изменить**.

Перемещение объектов и папок в vRealize Orchestrator Client

Чтобы упорядочить содержимое vRealize Orchestrator, его можно перенести в другую папку.

Нельзя перемещать действия между модулями действий и помещать объекты в корневую папку. Корневая папка включает в себя папки основных объектов и вложенные папки, но не может использоваться для хранения объектов.

Процедура

1. Выполните вход в vRealize Orchestrator Client.
2. На панели навигации слева выберите страницу объекта, например **Рабочие процессы**.
3. Вверху справа щелкните значок представления «Дерево» ()
4. Разверните древовидное представление и выберите объект или папку, которую необходимо переместить.
5. Перетащите объект или папку в новую родительскую папку.


Примечание Кроме того, можно перемещать объекты в новые папки непосредственно из редактора объектов. На вкладке **Сводка** щелкните кнопку **Выберите папку** и выберите новую родительскую папку для объекта. Другим вариантом перемещения является выбор объектов из таблицы на странице папки. Этот способ полезен при выполнении операций пакетного перемещения нескольких объектов vRealize Orchestrator.

Удаление папки или вложенной папки из vRealize Orchestrator Client

Устаревшие папки и вложенные папки можно удалить из vRealize Orchestrator Client.

Соответствующую папку корневого уровня для каждого типа объекта vRealize Orchestrator удалить нельзя.

Процедура

1. Выполните вход в vRealize Orchestrator Client.
2. На панели навигации слева выберите страницу объекта, например **Рабочие процессы**.
3. Вверху справа щелкните значок представления «Дерево» ()

4. Установите флажок рядом с папкой, которую необходимо удалить.

Примечание Чтобы удалить вложенную папку, выберите родительскую папку в представлении «Дерево» и установите нужный флажок.

5. Нажмите кнопку **Удалить**.
6. Если выбранная папка пуста, выполните следующие действия.
 - а) Подтвердите, что папку следует удалить.
 - б) Нажмите кнопку **Удалить**.
7. Если выбранная папка содержит объекты или вложенные папки vRealize Orchestrator Client, выполните следующие действия.
 - а) Подтвердите, что папку следует удалить.
 - б) Нажмите кнопку **Удалить**.

Появится сообщение Не удалось удалить элемент "имя_папки": папка "имя_папки" содержит объекты.
 - в) Чтобы удалить папку и все ее содержимое, нажмите **Принудительно удалить**.
 - г) Подтвердите, что папку следует удалить, и нажмите кнопку **Удалить**.

Примечание Кроме того, можно выполнить групповое удаление объектов, выбрав несколько объектов в таблице, которая находится в меню папки.

Настройка vRealize Orchestrator Client

3

Чтобы в полной мере использовать функциональные возможности vRealize Orchestrator Client, необходимо настроить разрешения пользователей и научиться применять журнал версий для управления объектами.

В эту главу входят следующие разделы:

- [Управление ролями и группами в клиенте vRealize Orchestrator](#)
- [Журнал версий объектов vRealize Orchestrator](#)

Управление ролями и группами в клиенте vRealize Orchestrator

Администратор может использовать vRealize Orchestrator Client для установки ролей пользователей и разрешений группы для компонентов и содержимого vRealize Orchestrator.

После проверки подлинности экземпляра vRealize Orchestrator администратор может установить разрешения для управления доступом к компонентам и содержимому. Разрешения в vRealize Orchestrator Client разделены на управление ролями и разрешения групп. Управление ролями позволяет управлять тем, какие компоненты vRealize Orchestrator Client пользователи могут просматривать и использовать. Разрешения для групп позволяют контролировать, какое содержимое vRealize Orchestrator Client пользователи могут просматривать и использовать. Разрешения для групп позволяют настраивать доступ к такому содержимому, как рабочие процессы, действия, политики, элементы конфигурации и элементы ресурсов. Группы можно использовать для объединения пользователей в рамках общих проектов. Например, можно создать группу, включающую в себя пользователей, которые занимаются разработкой настраиваемого подключаемого модуля vRealize Orchestrator.

Примечание Доступ к предварительно настроенному содержимому vRealize Orchestrator, например к стандартным рабочим процессам и действиям, предоставляется всем пользователям, если иное не настроено с помощью разрешений группы.

Роли

Управление ролью на стороне клиента доступно только для экземпляров vRealize Orchestrator с проверкой подлинности vSphere и лицензией vRealize Automation. Для развертываний, в которых настроена проверка подлинности vRealize Automation, необходимо использовать функцию управления учетными данными и доступом vRealize Automation. См. раздел [Настройка ролей клиента vRealize Orchestrator в vRealize Automation](#).

Роль	Описание
Администратор	<p>Может получать доступ ко всем компонентам и содержимому vRealize Orchestrator Client, включая содержимое, созданное в конкретных группах. Отвечает за настройку ролей пользователей, создание и удаление групп, а также добавление пользователей в группы.</p> <p>Примечание Администраторы арендаторов в среде vRealize Automation, используемой для проверки подлинности vRealize Orchestrator, по умолчанию имеют права администратора.</p>
Создатель рабочих процессов	<p>Может создавать, запускать, изменять и удалять содержимое vRealize Orchestrator Client. Может добавить свое содержимое в назначенную группу. Не имеет доступа к функциям администрирования и устранения неполадок в vRealize Orchestrator Client.</p>

Примечание Пользователи vRealize Automation без предварительно определенной роли могут входить в vRealize Orchestrator Client, но имеют ограниченный доступ к клиентским функциям. Если эти пользователи являются частью группы, они могут просматривать и запускать содержимое, связанное с этой группой.

Группы

Разрешения группы в vRealize Orchestrator Client можно использовать для подключения нескольких пользователей, которые работают на общем экземпляре vRealize Orchestrator, например разрабатывают настраиваемый подключаемый модуль.

Разрешения пользователей группы	Описание
Запустить и изменить	Доступно только для экземпляров vRealize Orchestrator с лицензией vRealize Automation. Может создавать, изменять, добавлять и запускать объекты vRealize Orchestrator, предназначенные для использования в группе.
Запуск	Может просматривать и запускать объекты vRealize Orchestrator, включенные в группу.

Примечание Разрешения группы связаны с системой управления ролями в vRealize Orchestrator Client. Например, пользователи, у которых нет предварительно определенной роли, могут иметь разрешение на **Запустить и изменить**, но они могут только просматривать и запускать собственное содержимое или содержимое группы и не могут создавать, изменять и добавлять содержимое.

Назначение ролей в vRealize Orchestrator Client

Администраторы могут добавлять пользователей в vRealize Orchestrator Client и разрешать им просмотр или использование определенных компонентов.

Благодаря управлению ролями можно контролировать доступ пользователей из поставщика удостоверений vRealize Orchestrator к компонентам vRealize Orchestrator Client. Функции управления ролями охватывают как пользовательский интерфейс vRealize Orchestrator Client, так и возможности API-интерфейса.

Примечание Управление ролью на стороне клиента доступно только для экземпляров vRealize Orchestrator с проверкой подлинности vSphere и лицензией vRealize Automation. Дополнительные сведения о назначении ролей экземплярам vRealize Orchestrator, прошедшим аутентификацию в vRealize Automation, см. в разделе [Настройка ролей клиента vRealize Orchestrator в vRealize Automation](#).

Процедура

1. Войдите в клиент vRealize Orchestrator как администратор.
2. Перейдите в меню **Администрирование > Управление ролями**.
3. Нажмите кнопку **Добавить**.
4. Найдите пользователя или группу, которые требуется добавить в vRealize Orchestrator Client.
5. Выберите роль пользователя. Дополнительные сведения о ролях см. в разделе [Управление ролями и группами в клиенте vRealize Orchestrator](#).
6. Нажмите кнопку **Сохранить**.

Создание групп в vRealize Orchestrator Client

Администраторы могут использовать группы, чтобы указывать, какое содержимое vRealize Orchestrator может просматриваться и быть доступным пользователям в vRealize Orchestrator Client.

С помощью vRealize Orchestrator Client можно создавать групповые разрешения для рабочих процессов, действий, политик, элементов конфигурации, элементов ресурсов и пакетов vRealize Orchestrator.

Примечание Пользователи из экземпляров vRealize Orchestrator, в которых используется проверка подлинности vSphere, могут иметь групповые разрешения только типа **Запуск**.

Процедура

1. Войдите в клиент vRealize Orchestrator как администратор.
2. Перейдите в раздел **Администрирование > Группы**.
3. Щелкните **Создать группу**.
4. На вкладке **Сводка** введите имя и описание группы.

5. На вкладке **Пользователи** нажмите **Добавить**.
 - а) Найдите пользователя, которого нужно добавить в группу.
 - б) Назначьте пользователю групповые разрешения.
 - в) Нажмите кнопку **Добавить**.
6. На вкладке **Элементы** добавьте в группу объекты vRealize Orchestrator.

Примечание Кроме того, объект можно добавить в уже существующие группы при создании этого объекта в vRealize Orchestrator Client. Чтобы добавить объект, выберите группу в раскрывающемся меню **Доступно по** на вкладке **Сводка/общие** редактора объектов.

7. Нажмите кнопку **Сохранить**.

Настройка ролей клиента vRealize Orchestrator в vRealize Automation

Для vRealize Orchestrator Client можно назначать роли служб на странице **Управление учетными данными и доступом** в vRealize Automation. Роли служб можно назначать для встроенных vRealize Orchestrator Client и автономных экземпляров vRealize Orchestrator с проверкой подлинности vRealize Automation.

Роли служб vRealize Orchestrator управляют тем, какие функции встроенного vRealize Orchestrator Client доступны пользователям. Дополнительные сведения о ролях vRealize Orchestrator см. в разделе [Управление ролями и группами в клиенте vRealize Orchestrator](#).

Примечание Автономные экземпляры vRealize Orchestrator с проверкой подлинности vSphere и лицензией vRealize Automation могут назначать роли непосредственно в vRealize Orchestrator Client. См. раздел [Назначение ролей в vRealize Orchestrator Client](#).

Необходимые условия

- Убедитесь, что соответствующие пользователи и группы импортированы из допустимого экземпляра vIDM.
- Прежде чем назначить роль службы vRealize Orchestrator пользователю, убедитесь, что у пользователя есть назначенная роль организации в vRealize Automation. См. раздел *Администрирование пользователей и групп в vRealize Automation* в руководстве *Администрирование vRealize Automation*.

Процедура

1. В раскрывающемся меню верхнего правого заголовка выберите пункт **Управление учетными данными и доступом**.
2. На вкладке **Активные пользователи** найдите адрес электронной почты пользователя, которого необходимо назначить в vRealize Orchestrator.
3. Установите флажок рядом с пользователем и нажмите кнопку **Изменить роли**.

4. Нажмите кнопку **Добавить доступ к службе**.
5. В левом раскрывающемся меню выберите пункт **Orchestrator**.
6. В правом раскрывающемся меню выберите роль, которую необходимо назначить пользователю.
7. Нажмите кнопку **Сохранить**.

Журнал версий объектов vRealize Orchestrator

vRealize Orchestrator Client поддерживает ведение журнала версий для каждого объекта vRealize Orchestrator. Используя журнал версий, можно сравнить разные версии объектов vRealize Orchestrator и вернуться к предыдущей версии.

vRealize Orchestrator создает запись журнала версий для каждого объекта vRealize Orchestrator при его сохранении. Последующие изменения в объектах vRealize Orchestrator приводят к созданию новых записей в журнале версий. Предыдущие записи журнала версий сохраняются и могут использоваться для отслеживания изменений в объекте и возврата объекта к предыдущей версии. При откате объекта к предыдущей версии создается новая запись журнала версий.

vRealize Orchestrator Client отслеживает историю версий следующих объектов vRealize Orchestrator:

- Рабочие процессы
- Действия
- Пакеты
- Политики
- Элементы конфигурации

Доступ к журналу версий объекта можно получить на вкладке **Журнал версий** на странице редактора объектов. При попытке изменить объект одновременно с другим пользователем может возникнуть конфликт слияния. Чтобы устранить конфликт слияния, нажмите кнопку **Разрешить** справа от сообщения об ошибке. В окне **Устранить конфликты** предлагается три варианта.

- **Использовать чужие.** Разрешите конфликт слияния, приняв изменения, внесенные другим пользователем.
- **Использовать свои.** Разрешите конфликт слияния, используя свои изменения.
- **Устранить.** Разрешите конфликт слияния, изменив отображаемую модель изменений. Если предоставленная модель недействительна, этот параметр недоступен.

Откат к более ранней версии рабочего процесса

Рабочий процесс можно откатить к ранее сохраненной версии.

Процедура

1. Войдите в клиент vRealize Orchestrator.
2. Перейдите в раздел **Библиотека > Рабочие процессы** и выберите рабочий процесс.

3. Перейдите на вкладку **Журнал версий**.
4. Чтобы просмотреть различия между версиями, выберите версию рабочего процесса, а затем в раскрывающемся меню **Различие с** выберите другую версию.

В окне отобразятся различия между текущей и выбранной версиями рабочего процесса.

5. Чтобы откатить рабочий процесс к другой версии, нажмите кнопку **Восстановить**.

Состояние рабочего процесса возвращается к состоянию выбранной версии.

Примечание Кроме того, версию рабочего процесса можно откатить в инструменте графического представления различий. См. раздел [Визуальное сравнение версий рабочих процессов](#).

Визуальное сравнение версий рабочих процессов

С помощью графического средства выявления различий можно сравнивать разные версии рабочих процессов.

По умолчанию различия между версиями рабочих процессов приводятся в журнале версий vRealize Orchestrator в формате YAML. Кроме того, версии рабочих процессов можно сравнить между собой визуально. Для выявления различий можно использовать:

- общую информацию о рабочем процессе, например номер версии и описание рабочего процесса;
- переменные, используемые в рабочем процессе;
- входные и выходные параметры рабочего процесса;
- схему рабочего процесса.

Необходимые условия

Создайте рабочий процесс.да

Процедура

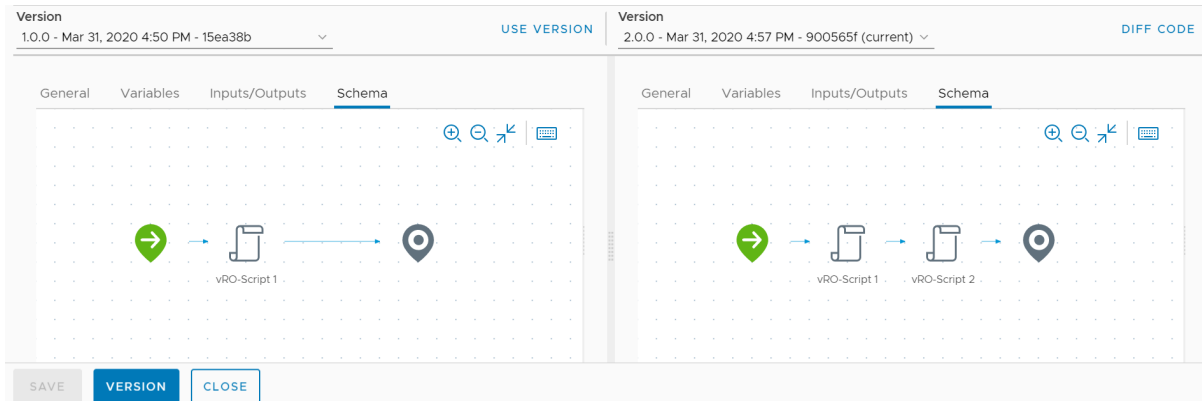
1. Выполните вход в vRealize Orchestrator Client.
2. Перейдите в раздел **Библиотека > Рабочие процессы** и выберите один из рабочих процессов.
3. Отредактируйте компоненты рабочего процесса.

Например, можно добавить дополнительный элемент **Задача с возможностью добавления сценариев** на вкладке **Схема**.

4. Нажмите кнопку **Сохранить**.
5. Перейдите на вкладку **Журнал версий**.

6. Вверху справа выберите **Визуальное представление отличий**.

Теперь можно визуально сравнить между собой две выбранные версии рабочего процесса. В раскрывающемся меню **Версия** выберите версии для сравнения.



7. (дополнительно) Чтобы откатить рабочий процесс к другой версии, выберите **Использовать версию**.

Возврат иерархии содержимого vRealize Orchestrator в предыдущее состояние с помощью Git

Используя более раннюю фиксацию в Git, можно вернуть содержимое vRealize Orchestrator к более раннему состоянию.

Можно вернуть содержимое vRealize Orchestrator к предыдущему состоянию, выбрав конкретную фиксацию.

Необходимые условия

- Настройте подключение к репозиторию GitHub или GitLab. См. раздел [Настройка подключения к репозиторию Git](#).
- Отправьте локальный набор изменений в настроенный репозиторий Git.

Процедура

1. Выполните вход в vRealize Orchestrator Client.
2. Перейдите в раздел **Администрирование > Журнал Git**.
3. Выберите набор изменений, к которому нужно выполнить возврат, и нажмите кнопку **Восстановить это**.
4. Подтвердите, что вы хотите выполнить возврат к этой фиксации, и нажмите кнопку **ОК**.

Иерархия содержимого vRealize Orchestrator возвращается к состоянию, указанному в этой фиксации. Соответствующее содержимое vRealize Orchestrator возвратится к предыдущей версии. Если содержимое не существовало во время отправки фиксации, оно удаляется из иерархии.

Следующие шаги

Чтобы восстановить иерархию vRealize Orchestrator до последнего состояния, сохраненного в репозитории Git, выполните команду Pull в окне **Журнал Git**.

Примеры использования vRealize Orchestrator

4

В этих примерах демонстрируется часть функциональных возможностей платформы vRealize Orchestrator.

Представленные сценарии содержат лишь примеры значений. Структура среды и схемы именования в вашей организации могут отличаться от представленных.

В эту главу входят следующие разделы:

- [Интеграция Amazon Web Services и vRealize Orchestrator с помощью Python](#)
- [Использование ветвей Git для управления иерархией объектов vRealize Orchestrator](#)
- [Использование модулей сторонних разработчиков для вызова API-интерфейса проекта vRealize Automation](#)

Интеграция Amazon Web Services и vRealize Orchestrator с помощью Python

В этом примере использования vRealize Orchestrator демонстрируется, как расширить возможности развертывания vRealize Orchestrator с помощью Python.

Начиная с версии 8.1, в действиях и сценариях рабочих процессов vRealize Orchestrator можно использовать три новые среды выполнения.

- Python 3.7
- Node.js 12
- PowerCLI 11/Powershell 6.2
- PowerCLI 12/Powershell 7

Примечание Среда выполнения PowerCLI включает в себя PowerShell и следующие модули: VMware.PowerCLI, PowerNSX и PowervRA.

Важно! Новые среды выполнения можно использовать только в том случае, если в развертывании vRealize Orchestrator используется лицензия vRealize Automation.

В этом примере показано, как создать сценарий Python, вызывающий экземпляры EC2 в Amazon Web Services (AWS).

Важно! Перед началом разработки настраиваемого сценария необходимо изучить основные принципы работы со сценариями Python, Node.js и PowerShell в vRealize Orchestrator. См. раздел [Основные принципы работы со сценариями Python, Node.js и PowerShell](#).

Процедура

1. Создание базового сценария Python

На локальном компьютере создайте сценарий Python, а затем упакуйте его и библиотеку boto3 в папку ZIP.

2. Создание действия Amazon Web Services

Создайте действие vRealize Orchestrator, в котором используется сценарий Python.

3. Отладка действия Amazon Web Services

В исходной версии сценария Python есть ошибка, включенная туда намеренно для обучения отладке сценария.

4. Обновление действия службы Amazon Web Services

Импортируйте обновленный сценарий Python и снова запустите действие.

Создание базового сценария Python

На локальном компьютере создайте сценарий Python, а затем упакуйте его и библиотеку boto3 в папку ZIP.

Необходимые условия

- Загрузите и установите Python 3. См. [страницу загрузки Python](#).
- Загрузите и установите Visual Studio Code. См. [страницу загрузки Visual Studio Code](#).
- Убедитесь, что установлено расширение Python для Visual Studio Code. См. [магазин Visual Studio](#).

Процедура

1. На локальном компьютере создайте папку vro-python-aws и установите SDK boto3 для Python.

```
mkdir vro-python-aws
cd vro-python-aws
mkdir lib
pip install boto3 -t lib/
```

2. Откройте редактор и создайте основной сценарий Python. В данном примере используется Visual Studio Code.

```
import boto3

def handler(context, inputs):
    ec2 = boto3.resource('ec2')
```

```
filters = [{
    'Name': 'instance-state-name',
    'Values': ['running']
}]

instances = ec2.instances.filter(Filters=filters)
for instance in instances:
    print('Instance: ' + instance.id)
```

В этом сценарии Python указаны все экземпляры EC2, выполняемые в заданной области.

3. Сохраните созданный сценарий как файл `main.py` в папке `vro-python-aws`.
4. Войдите в интерфейс командной строки.
5. Перейдите в папку `vro-python-aws`.

```
cd vro-python-aws
```

6. Создайте пакет ZIP, содержащий сценарий Python.

```
zip -r --exclude=*.zip -X vro-python-aws.zip .
```

Примечание Пакет ZIP также можно создать с помощью ZIP-утилиты (например, программы 7-Zip).

Результаты

Базовый сценарий Python создан и готов к импорту в развертывание vRealize Orchestrator.

Создание действия Amazon Web Services

Создайте действие vRealize Orchestrator, в котором используется сценарий Python.

Процедура

1. Войдите в клиент vRealize Orchestrator.
2. Перейдите в раздел **Библиотека > Действия**.
3. Щелкните **Создать действие**.
4. На вкладке **Общие** введите имя, модуль и номер версии действия.
5. На вкладке **Сценарий** выберите **Python 3.7** в качестве среды выполнения и **Zip-файл** в качестве типа сценария.
6. Щелкните **Импортировать**.
7. Перейдите в папку `vro-python-aws` и выберите пакет ZIP со сценарием Python.

8. В текстовом поле **Обработчик записей** введите **main.handler**.

Примечание В обработчике записей действия используется основной сценарий из импортированного пакета ZIP. Так как основной сценарий размещен в файле с именем `main.py` и функции с именем **handler**, обработчик записей должен называться **main.handler**. Если файл основного сценария был назван по-другому, измените значение имени обработчика записей соответствующим образом.

9. Сохраните действие и нажмите **Запустить**.

При выполнении действия возникает ошибка.

10. Перейдите на вкладку **Журналы**.

В журналах выполнения действия содержится сообщение об ошибке `botocore.exceptions.NoRegionError: You must specify a region..` Это ожидаемо, так как в базовом сценарии Python не была определена область.

Следующие шаги

Выполните отладку сценария Python. См. раздел [Отладка действия Amazon Web Services](#).

Отладка действия Amazon Web Services

В исходной версии сценария Python есть ошибка, включенная туда намеренно для обучения отладке сценария.

Необходимые условия

Войдите в учетную запись Amazon Web Services (AWS) и создайте пользователя IAM специально для этого сценария. См. раздел [Создание пользователя IAM в учетной записи AWS](#). У пользователя IAM должны быть следующие разрешения:

```
"Effect": "Allow",
"Action": "ec2:DescribeInstances",
"Resource": "*"

```

Процедура

1. Подготовьте vRealize Orchestrator Appliance.

Осторожно! Не выполняйте отладку сценариев в рабочем развертывании vRealize Orchestrator. Отладку следует выполнять в развертывании vRealize Orchestrator с одним узлом, которое предназначено для разработки и тестирования.

- а) Войдите в командную строку vRealize Orchestrator Appliance по протоколу SSH в качестве пользователя **root**.
- б) Выполните команду `vracli dev tools`.
- в) Появится запрос на подтверждение дальнейших действий. Введите **yes** (да) для продолжения или **no** (нет) для отмены.

Важно! При запуске команды `vracli dev tools` открываются порты, необходимые для отладки сценария Python. В процессе отладки текущий сеанс SSH должен оставаться открытым.

2. Запустите конфигурацию отладки.

- а) Войдите в клиент vRealize Orchestrator.
- б) Откройте действие AWS и нажмите **Отладка**.
Начнется процесс отладки, и выполнение действия будет приостановлено.
- в) Выберите вкладку **Конфигурация отладки**.
Вкладка содержит конфигурацию `.json`, которую можно удаленно вложить в IDE для отладки сценария Python.
- г) Скопируйте содержимое конфигурации вручную или нажмите **Копировать в буфер**.

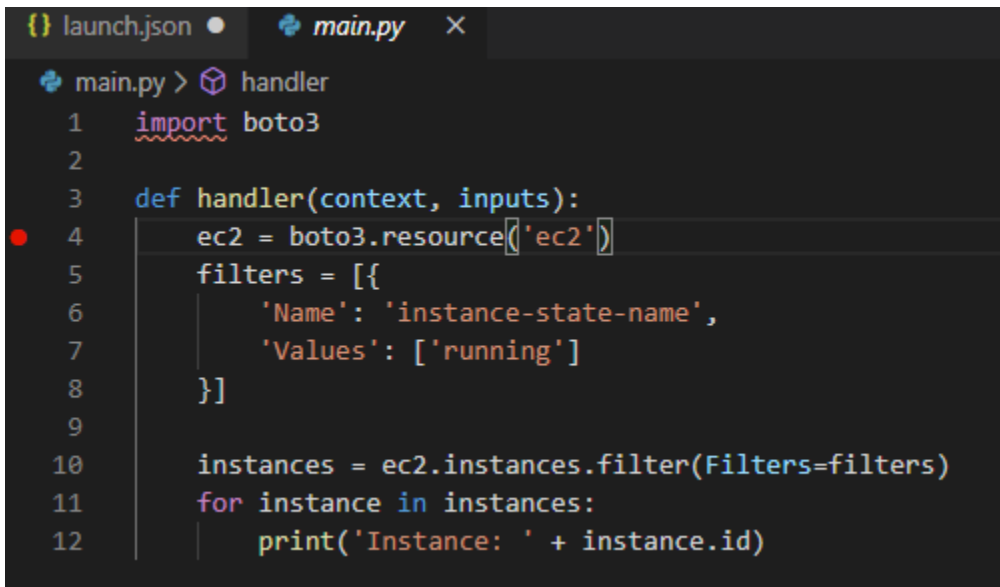
3. Выполните отладку сценария Python.

- а) Откройте Visual Studio Code.
- б) Откройте папку `vro-python-aws`.
- в) На панели навигации сверху выберите **Запустить > Открытые конфигурации**.
- г) Выберите файл **Python**.

- д) Оставьте атрибуты "version" и "configuration" в текущих позициях и вставьте содержимое .json, скопированное из клиента vRealize Orchestrator. Созданный файл launch.json должен выглядеть примерно следующим образом:

```
{
  "version": "0.2.0",
  "configurations": [
    {
      "request": "attach",
      "port": 18281,
      "name": "vRO Python debug 8302f4c7-5beb-40da-848a-5003c0296f7b",
      "host": "es-sof-vc-vm-225-190.sof-mbu.eng.vmware.com",
      "type": "python",
      "pathMappings": [
        {
          "localRoot": "${workspaceFolder}",
          "remoteRoot": "/var/run/vco-polyglot/function"
        }
      ]
    }
  ]
}
```

- е) Выберите файл сценария main.py и добавьте точку останова в строку ec2 = boto3.resource('ec2')



```
{ } launch.json  main.py x
main.py > handler
1  import boto3
2
3  def handler(context, inputs):
4  ec2 = boto3.resource('ec2')
5  filters = [{
6      'Name': 'instance-state-name',
7      'Values': ['running']
8  }]
9
10 instances = ec2.instances.filter(Filters=filters)
11 for instance in instances:
12     print('Instance: ' + instance.id)
```

- ж) На панели навигации сверху выберите Запустить > Начать отладку.

- з) Когда отладчик достигнет точки останова, выполните операцию пропуска.

При запуске отладки будет указано, что в сценарии Python не хватает указанной области и ключа доступа AWS.

- и) Вернитесь в открытый сеанс vRealize Orchestrator Appliance и нажмите клавишу **ВВОД**, чтобы закрыть порты, открытые для данного сеанса отладки.

4. Добавьте недостающие сведения в сценарий Python.

- а) В Visual Studio Code создайте файл с именем `awsconfig`, содержащий ключ доступа AWS для пользователя IAM и область AWS для эхо-тестирования с помощью сценария Python.

```
[default]
aws_access_key_id=your key ID
aws_secret_access_key=your secret access key
region=your-region
```

- б) Сохраните файл `awsconfig` как файл конфигурации (`.cfg`) в папке `vro-python-aws`.
- в) Откройте файл `main.py` и измените его так, чтобы библиотека `boto3` могла использовать файл `awsconfig.cfg`.

```
import boto3

import os
os.environ['AWS_CONFIG_FILE'] = os.getcwd() + '/awsconfig.cfg'

def handler(context, inputs):
    ec2 = boto3.resource('ec2')
    filters = [{
        'Name': 'instance-state-name',
        'Values': ['running']
    }]

    instances = ec2.instances.filter(Filters=filters)
    for instance in instances:
        print('Instance: ' + instance.id)
```

- г) Создайте новый пакет ZIP, содержащий файл `main.py`, файл `awsconfig.cfg` и библиотеку `boto3`.

```
zip -r --exclude=*.zip -X vro-python-aws.zip .
```

Примечание Пакет ZIP также можно создать с помощью ZIP-утилиты (например, программы 7-Zip).

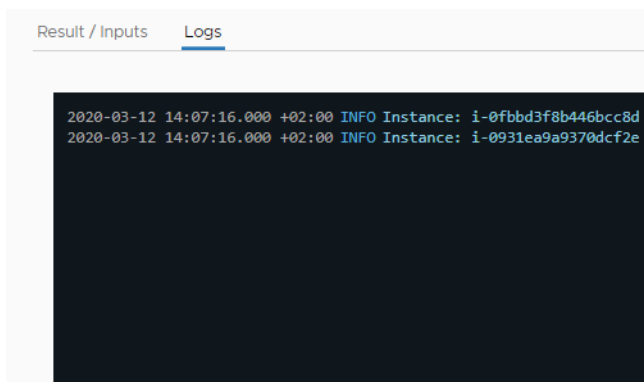
Обновление действия службы Amazon Web Services

Импортируйте обновленный сценарий Python и снова запустите действие.

Процедура

1. Войдите в клиент vRealize Orchestrator.
2. Перейдите в раздел **Библиотека > Действия** и выберите исходное действие службы Amazon Web Services (AWS).
3. (дополнительно) На вкладке **Общие** измените номер версии.
4. Удалите старый пакет ZIP и нажмите кнопку **Импорт**.
5. Выберите обновленный пакет ZIP.
6. Сохраните действие и нажмите **Запустить**.
7. По завершении выполнения действия перейдите на вкладку **Журналы**.

В журналах отображаются экземпляры EC2, запрошенные действием.



Следующие шаги

Создайте рабочий процесс vRealize Orchestrator, использующий обновленное действие AWS в качестве элемента действия.

Использование ветвей Git для управления иерархией объектов vRealize Orchestrator

Используйте структуру ветвей, чтобы упорядочить управление содержимым vRealize Orchestrator в репозитории Git.

Использование Git дает разработчикам vRealize Orchestrator большую гибкость, так как обеспечивает доступ к централизованному репозиторию. Например, с помощью Git можно управлять созданием рабочих процессов в нескольких средах vRealize Orchestrator.

Примечание Чтобы с помощью Git управлять иерархией объектов, в развертывании vRealize Orchestrator должна использоваться лицензия vRealize Automation. Дополнительные сведения см. в разделе *Активация функций vRealize Orchestrator с использованием лицензий* руководства *Установка и настройка vRealize Orchestrator*.

Начиная с версии 8.1, vRealize Orchestrator позволяет отправлять объекты в ветви и извлекать объекты из ветвей. Ветви можно использовать для управления разработкой отдельных групп объектов vRealize Orchestrator, перед тем как они снова будут объединены в главную ветвь.

В этом примере с помощью проекта GitLab осуществляется управление объектами vRealize Orchestrator, использующими среду выполнения Python. Данный сценарий представляет собой пример использования Git в vRealize Orchestrator и не является исчерпывающим описанием доступных возможностей.

Примечание Если вам привычнее работать с GitHub, можете использовать репозиторий GitHub в данном примере.

Процедура

1. Подготовка среды GitLab

Создайте ветвь Git для объектов Python в vRealize Orchestrator.

2. Настройка подключения к репозиторию Git

Администраторы могут настроить подключение между развертыванием vRealize Orchestrator и репозиторием или проектом Git.

3. Отправка изменений в репозиторий Git

Отправляйте изменения локальных объектов vRealize Orchestrator в интегрированный репозиторий Git. В этом примере выполняется отправка изменений, внесенных в действие vRealize Orchestrator на основе Python, в конкретную ветвь Git.

Подготовка среды GitLab

Создайте ветвь Git для объектов Python в vRealize Orchestrator.

Необходимые условия

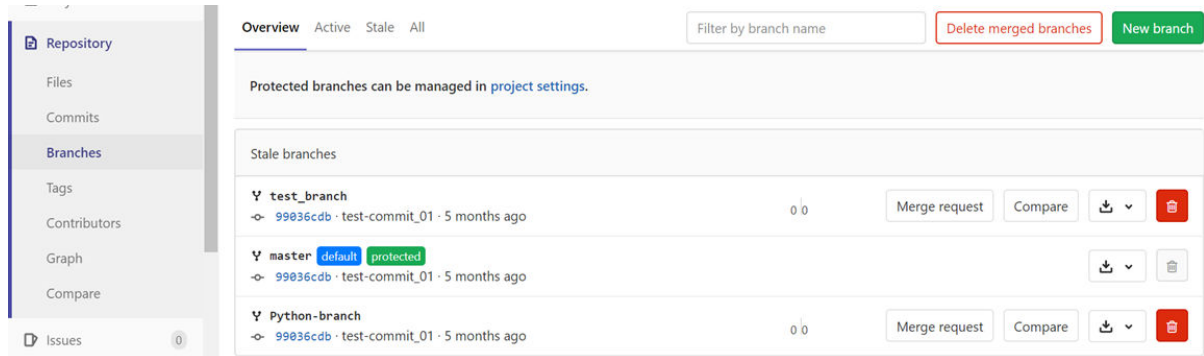
Создайте проект GitLab для среды vRealize Orchestrator. См. раздел [Создание проекта](#).

Процедура

1. Войдите в учетную запись GitLab.
2. Перейдите к проекту GitLab.
3. На панели навигации слева выберите **Repository (Репозиторий) > Branches (Ветви)**.
4. На вкладке **Overview (Обзор)** выберите **New Branch (Создать ветвь)**.
5. В поле **Branch name (Имя ветви)** введите **Python-branch (Ветвь Python)**.
6. В качестве значения параметра **Create from (Создать из)** оставьте **master (главный)**.

7. Нажмите **Create branch (Создать ветвь)**.

Ветвь для объектов vRealize Orchestrator на основе Python создана.



Настройка подключения к репозиторию Git

Администраторы могут настроить подключение между развертыванием vRealize Orchestrator и репозиторием или проектом Git.

Чтобы использовать Git для управления иерархией объектов vRealize Orchestrator, необходимо настроить подключение к репозиторию Git с помощью vRealize Orchestrator Client.

Необходимые условия

- Убедитесь, что в среде vRealize Orchestrator используется лицензия vRealize Automation.
- Создайте маркер доступа для вашего проекта GitLab и скопируйте его в буфер обмена для использования в процессе настройки. См. раздел [Создание персонального маркера доступа](#).

Примечание В данном примере используется проект GitLab. Если вам привычнее работать с GitHub, можете использовать репозиторий GitHub. Сведения о создании маркера GitHub см. в разделе [Создание персонального маркера доступа для командной строки](#).

Процедура

1. Войдите в vRealize Orchestrator Client от имени **администратора**.
2. Перейдите в раздел **Администрирование > Репозитории Git**.
3. Нажмите **Добавить репозиторий**.
4. Введите URL-адрес репозитория Git.

Например, `https://gitlab.com/имя_пользователя/репозиторий_vro`.

Примечание Кроме того, можно установить соединение с протоколом SSH.

5. Введите имя пользователя для профиля Git.
6. Введите маркер доступа для репозитория Git.
7. Чтобы проверить подключение к репозиторию Git, нажмите **Проверить**.

8. (дополнительно) Измените имя, используемое для идентификации репозитория в vRealize Orchestrator Client.
9. (дополнительно) Добавьте краткое описание подключенного репозитория Git.
10. Чтобы активировать подключенный репозиторий Git, нажмите **Создать активный репозиторий**.

Примечание Нельзя сделать несколько репозитиев Git активными одновременно. Активный репозиторий Git можно изменить на странице **Репозитории Git**.

11. Выберите ветвь, в которую будут отправлены изменения. В данном примере это ветвь **Python**. См. раздел [Подготовка среды GitLab](#).

Примечание Выбранную ветвь Git можно изменить в любое время после завершения первоначальной настройки Git.

12. Чтобы завершить процесс настройки, нажмите кнопку **Сохранить**.

Следующие шаги

Вернитесь в меню **Репозитории Git** и убедитесь, что репозиторий находится в состоянии **Активно**.

Отправка изменений в репозиторий Git

Отправляйте изменения локальных объектов vRealize Orchestrator в интегрированный репозиторий Git. В этом примере выполняется отправка изменений, внесенных в действие vRealize Orchestrator на основе Python, в конкретную ветвь Git.

Локальный набор изменений можно отправить в репозиторий Git. Каждый набор изменений может состоять из одного или нескольких измененных объектов vRealize Orchestrator.

Примечание Процесс отправки и удаления наборов изменений в репозитории Git не ограничен разрешениями группы. Следовательно, разработчики рабочих процессов из одной группы могут отправлять или отменять локальные изменения, внесенные другими разработчиками.

Необходимые условия

- Убедитесь, что ветвь Git создана. См. раздел [Подготовка среды GitLab](#).
- Убедитесь, что подключение к репозиторию Git настроено. См. раздел [Настройка подключения к репозиторию Git](#).
- Убедитесь, что интеграция Git настроена для отправки изменений в ветвь **Python-branch** (Ветвь Python) Git.
- Создайте объект vRealize Orchestrator на основе Python. Пример см. в разделе [Интеграция Amazon Web Services и vRealize Orchestrator с помощью Python](#).

Процедура

1. Выполните вход в vRealize Orchestrator Client.

2. Измените действие Python.

- Перейдите в раздел **Библиотека > Действия** и выберите действие Python.
- Внесите в действие незначительные изменения (например, измените описание).
- Сохраните действие.

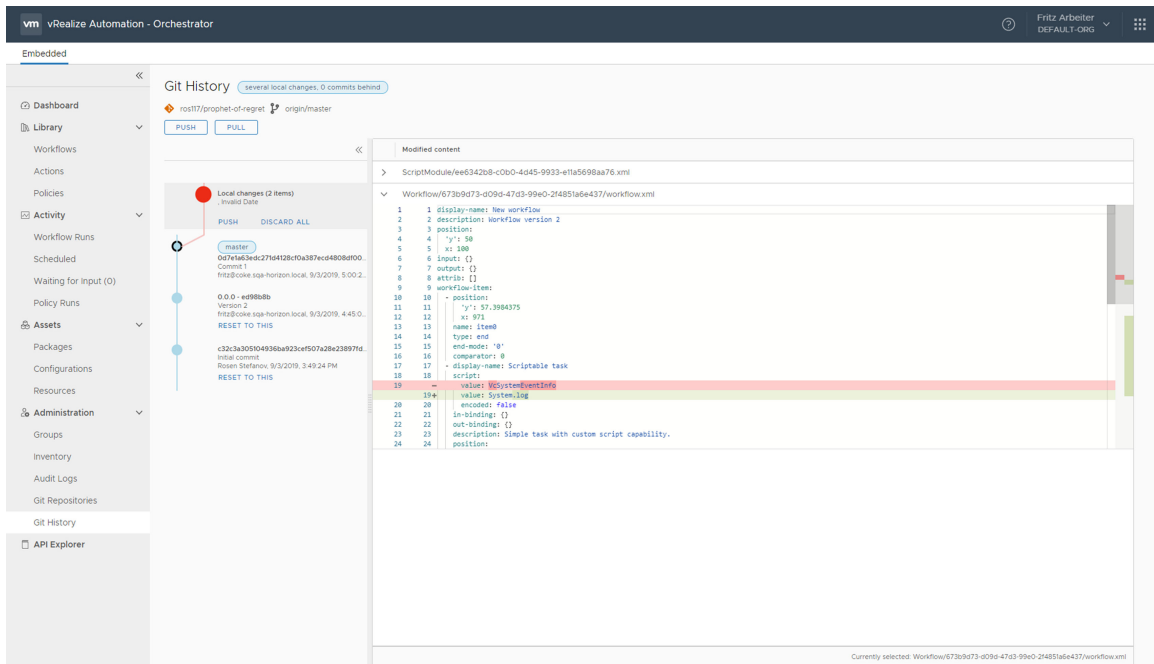
3. Отправьте изменения в репозиторий Git.

Примечание Локальные изменения можно также отправлять на уровне отдельного объекта, щелкнув параметр **Версия**, отображаемый в нижней части редактора объектов.

- Перейдите в раздел **Администрирование > Журнал Git**.

В **Журнал Git** отображаются текущие различия между ветвью локальной версии и выбранной ветвью репозитория Git. Можно развернуть запись любого измененного объекта vRealize Orchestrator, чтобы просмотреть различия между версиями.

Примечание Можно удалить локальный набор изменений, выбрав команду **удалить все**.



- Щелкните кнопку **Отправить**.
 - Введите заголовок фиксации.
 - (дополнительно) Введите краткое описание фиксации.
 - Выберите изменения действия Python, которые необходимо отправить в репозиторий Git.
- ## 4. Чтобы завершить передачу локального набора изменений в репозиторий Git, нажмите кнопку **Отправить**.

Следующие шаги

По окончании разработки в отдельной ветви Git объедините ее с главной ветвью. См. раздел [Создание запроса на слияние](#).

Использование модулей сторонних разработчиков для вызова API-интерфейса проекта vRealize Automation

В этом примере использования vRealize Orchestrator показано, как вызвать API-интерфейс проекта vRealize Automation с помощью модулей сторонних разработчиков.

В vRealize Orchestrator, начиная с версии 8.1, поддерживаются следующие среды выполнения:

- Python 3.7
- Node.js 12
- PowerCLI 11/Powershell 6.2
- PowerCLI 12/Powershell 7

Примечание Среда выполнения PowerCLI включает в себя PowerShell и следующие модули: VMware.PowerCLI, PowerNSX и PowervRA.

В этом примере демонстрируется создание действий vRealize Orchestrator, в которых для подключения к API-интерфейсу проекта vRealize Automation используются сторонние модули зависимостей.

Важно! Перед началом разработки настраиваемого сценария необходимо изучить основные принципы работы со сценариями Python, Node.js и PowerShell в vRealize Orchestrator. См. раздел [Основные принципы работы со сценариями Python, Node.js и PowerShell](#).

Создание сценария Python для вызова API-интерфейса проекта vRealize Automation

Создайте образец сценария для вызова API-интерфейса проекта vRealize Automation с помощью Python.

Необходимые условия

Убедитесь, что установлен Python 3 и установщик пакетов PIP. См. [страницу загрузок Python](#) и [указатель пакетов Python](#).

Процедура

1. На локальном компьютере откройте оболочку командной строки.
2. Создайте папку vro-python-vra.

```
mkdir vro-python-vra
```

3. Перейдите в папку vro-python-vra.

```
cd vro-python-vra
```

4. Создайте сценарий Python с именем handler.py.

```
touch handler.py
```

Сценарий handler.py должен задавать одну функцию, которая принимает два аргумента: контекст выполнения рабочего процесса vRealize Orchestrator и связанные входные данные vRealize Orchestrator.

```
def handler(context, inputs):
    print('Hello, your inputs were ' + inputs)
    return None
```

Примечание При использовании стандартных библиотек ведения журнала все, что регистрируется в действии со сценарием, также отображается в журнале рабочего процесса. Входные данные и возвращаемые значения сценария должны иметь соответствующие входные параметры и типы возвращаемых значений, настроенные в клиенте vRealize Orchestrator. Например, входные данные vRAUrl в сценарии должны иметь соответствующий входной параметр, именуемый vRAUrl, в клиенте vRealize Orchestrator. Аналогичным образом, если сценарий возвращает строковое значение, тип возвращаемого значения, настроенный в клиенте vRealize Orchestrator, также должен быть строковым. Если действие возвращает сложный объект, можно использовать тип возвращаемых значений Properties или Composite Type.

5. Установите модуль запросов Python.

Важно! Сторонние модули зависимостей должны быть установлены в папке корневого уровня в главной папке сценария vro-python-vra. В этом примере необходимо создать папку lib для модуля запросов.

а) Создайте папку lib.

```
mkdir lib
```

б) Установите модуль запросов.

```
pip3 install requests -t lib/
```

6. Добавьте модуль запросов в сценарий handler.py.

```
import requests

def handler(context, inputs):
    print('Hello, your inputs were ' + inputs)
    return None
```


7. Создайте запрос GET к API-интерфейсу проекта vRealize Automation.

```
token = ''
vRAUrl = ''
r = requests.get(vRAUrl + '/iaas/api/projects', headers={'Authorization': 'Bearer ' + token})

print('Got response ' + r.text)
```

8. Задайте значения token и vRAUrl .

- а) Получите маркер доступа с помощью API-интерфейса службы удостоверений vRealize Automation. См. раздел [Получение маркера доступа для API-интерфейса vRealize Automation](#)
- б) Для значения vRAUrl настройте сценарий таким образом, чтобы в нем использовался входной параметр vRealize Orchestrator с тем же именем.

```
vRAUrl = inputs["vRAUrl"]
```

- в) Добавьте новые значения в файл handler.py.

```
import requests

def handler(context, inputs):
    token = 'ACCESS_TOKEN'
    vRAUrl = inputs["vRAUrl"]

    r = requests.get(vRAUrl + '/iaas/api/projects', headers={'Authorization': 'Bearer ' + token})

    print('Got response ' + r.text)

    return r.json()
```

Примечание Поскольку ответ API-интерфейса проекта vRealize Automation возвращается в формате JSON, для действия vRealize Orchestrator используйте тип возвращаемого значения Properties или Composite Type.

9. Создайте пакет ZIP, содержащий файл handler.py и папку lib модуля запроса.

```
zip -r --exclude=*.zip -X vro-python-vra.zip .
```

Следующие шаги

Импортируйте сценарий PowerShell в действие vRealize Orchestrator. См. раздел [Создание действий в клиенте vRealize Orchestrator](#).

Создание сценария Node.js для вызова API-интерфейса проекта vRealize Automation

Создайте образец сценария для вызова API-интерфейса проекта vRealize Automation с помощью Node.js.

Необходимые условия

Загрузите и установите Node. Js 12. См. раздел [Загрузки Node.js](#).

Процедура

1. На локальном компьютере откройте оболочку командной строки.
2. Создайте папку vro-node-vra.

```
mkdir vro-node-vra
```

3. Перейдите в папку vro-node-vra.

```
cd vro-node-vra
```

4. Создайте сценарий Node.js с именем handler.js.

```
touch handler.js
```

Сценарий handler.js должен задавать одну функцию, которая принимает два аргумента: контекст выполнения рабочего процесса vRealize Orchestrator и связанные входные данные vRealize Orchestrator.

```
exports.handler = (context, inputs) => {
  console.log('Hello, your inputs were ' + inputs);
  return null;
}
```

Примечание При использовании стандартных библиотек ведения журнала все, что регистрируется в действии со сценарием, также отображается в журнале рабочего процесса. Входные данные и возвращаемые значения сценария должны иметь соответствующие входные параметры и типы возвращаемых значений, настроенные в клиенте vRealize Orchestrator. Например, входные данные vRAUrl в сценарии должны иметь соответствующий входной параметр, именуемый vRAUrl, в клиенте vRealize Orchestrator. Аналогичным образом, если сценарий возвращает строковое значение, тип возвращаемого значения, настроенный в клиенте vRealize Orchestrator, также должен быть строковым. Если действие возвращает сложный объект, можно использовать тип возвращаемых значений Properties или Composite Type.

5. Установите модуль запросов Node.js.

```
npm install request
```

Важно! Сторонние модули зависимостей должны быть установлены в папке корневого уровня `node_modules` в главной папке сценария `vro-node-vra`. Не перемещайте и не переименовывайте эту папку.

6. Добавьте модуль запросов в сценарий `handler.js`.

```
const request = require('request');

exports.handler = (context, inputs) => {
  console.log('Hello, your inputs were ' + inputs);
  return null;
}
```

7. Создайте запрос GET к API-интерфейсу проекта vRealize Automation.

```
const token = '';
const vRAUrl = '';
request.get(vRAUrl + '/iaas/api/projects', { 'auth': { 'bearer': token } }, function (error,
response, body) {
  console.log('Got response ' + body);
});
```

8. Задайте значения token и vRAUrl .

- а) Получите маркер доступа с помощью API-интерфейса службы удостоверений vRealize Automation. См. раздел [Получение маркера доступа для API-интерфейса vRealize Automation](#).
- б) Для значения vRAUrl настройте сценарий таким образом, чтобы в нем использовался входной параметр vRealize Orchestrator с тем же именем.

```
const vRAUrl = inputs.vRAUrl;
```

- в) Добавьте новые значения в файл handler.js.

```
const request = require('request');
exports.handler = (context, inputs, callback) => {
  const vRAUrl = inputs.vRAUrl;
  const token = 'ACCESS_TOKEN';
  request.get(vRAUrl + '/iaas/api/projects', { 'auth': { 'bearer': token } }, function
(error, response, body) {
    console.log('Got response ' + body);
    callback(null, JSON.parse(body));
  });
}
```

Примечание Поскольку ответ API-интерфейса проекта vRealize Automation возвращается в формате JSON, для действия vRealize Orchestrator используйте тип возвращаемого значения Properties или Composite Type.

9. Создайте пакет ZIP, содержащий файл handler.js и папку node_modules модуля запросов.

```
zip -r --exclude=*.zip -X vro-node-vra.zip .
```

Следующие шаги

Импортируйте сценарий Node.js в действие vRealize Orchestrator. См. раздел [Создание действий в клиенте vRealize Orchestrator](#).

Создание сценария PowerShell для вызова API-интерфейса проекта vRealize Automation

Создайте образец сценария для вызова API-интерфейса проекта vRealize Automation с помощью PowerShell.

Процедура

1. На локальном компьютере откройте оболочку командной строки.
2. Создайте папку vro-powershell-vra.

```
mkdir vro-powershell-vra
```

3. Перейдите в папку vro-powershell-vra.

```
cd vro-powershell-vra
```

4. Создайте сценарий PowerShell с именем handler.ps1.

```
touch handler.ps1
```

Сценарий handler.ps1 должен задавать одну функцию, которая принимает два аргумента: контекст выполнения рабочего процесса vRealize Orchestrator и связанные входные данные vRealize Orchestrator.

```
function Handler {
    Param($context, $inputs)

    $inputsString = $inputs | ConvertTo-Json -Compress
    Write-Host "Inputs were $inputsString"
}
```

Примечание При использовании стандартных библиотек ведения журнала все, что регистрируется в действии со сценарием, также отображается в журнале рабочего процесса. Входные данные и возвращаемые значения сценария должны иметь соответствующие входные параметры и типы возвращаемых значений, настроенные в клиенте vRealize Orchestrator. Например, входные данные vRAUrl в сценарии должны иметь соответствующий входной параметр, именуемый vRAUrl, в клиенте vRealize Orchestrator. Аналогичным образом, если сценарий возвращает строковое значение, тип возвращаемого значения, настроенный в клиенте vRealize Orchestrator, также должен быть строковым. Если действие возвращает сложный объект, можно использовать тип возвращаемых значений Properties или Composite Type.

5. Установите модуль утверждений PowerShell.

Важно! Сторонние модули зависимостей должны быть установлены в папке корневого уровня в главной папке сценария vro-powershell-vra. В этом примере необходимо создать папку Modules для модуля утверждений.

а) Создайте папку Modules.

```
mkdir Modules
```

б) Установите модуль утверждений.

```
pwsh -c "Save-Module -Name Assert -Path ./Modules/ -Repository PSGallery"
```

6. Добавьте модуль утверждений в сценарий handler.ps1.

```
Import-Module Assert

function Handler {
    Param($context, $inputs)
```

```
$inputsString = $inputs | ConvertTo-Json -Compress
Write-Host "Inputs were $inputsString"
}
```

7. Создайте запрос GET к API-интерфейсу проекта vRealize Automation, где используется командлет Invoke-RestMethod.

```
$token = ''
$vRAUrl = ''
$projectsUrl = $vRAUrl + "/project-service/api/projects"
$response = Invoke-RestMethod $projectsUrl + '/iaas/api/projects' -Headers @{'Authorization' =
"Bearer $token"} -Method 'GET'

Write-Host "Got response: $response"
```

8. Задайте значения token и vRAUrl .

- а) Получите маркер доступа с помощью API-интерфейса службы удостоверений vRealize Automation. См. раздел [Получение маркера доступа для API-интерфейса vRealize Automation](#).
- б) Добавьте атрибуты модуля утверждений Assert-NotNull и Assert-Type.

```
$token | Assert-NotNull
$token | Assert-Type String
```

- в) Для значения `vRAUrl` настройте сценарий таким образом, чтобы в нем использовался входной параметр vRealize Orchestrator с тем же именем.

```
$vRAUrl = $inputs.vRAUrl
```

- г) Добавьте новые значения в файл `handler.ps1`.

```
Import-Module Assert
$ErrorActionPreference = "Stop"
function Handler {
    Param($context, $inputs)
    $token = "ACCESS_TOKEN"
    $token | Assert-NotNull
    $token | Assert-Type String
    $vRAUrl = $inputs.vRAUrl
    $projectsUrl = $vRAUrl + "/project-service/api/projects"
    $response = Invoke-RestMethod $projectsUrl -Headers @{'Authorization' = "Bearer $token"} -
Method 'GET'

    Write-Host "Got response: $response"

    return $response
}
```

Примечание Поскольку ответ API-интерфейса проекта vRealize Automation возвращается в формате JSON, для действия vRealize Orchestrator используйте тип возвращаемого значения `Properties` или `Composite Type`.

9. Создайте пакет ZIP, содержащий файл `handler.ps1` и папку `Modules` модуля утверждения.

```
zip -r --exclude=*.zip -X vro-powershell-vra.zip .
```

Следующие шаги

Импортируйте сценарий PowerShell в действие vRealize Orchestrator. См. раздел [Создание действий в клиенте vRealize Orchestrator](#).

Управление рабочими процессами

5

Рабочий процесс — это набор действий и решений, которые выполняются последовательно. vRealize Orchestrator предоставляет библиотеку рабочих процессов, позволяющих выполнять стандартные задачи управления. vRealize Orchestrator также предоставляет библиотеки отдельных действий, которые выполняются этими рабочими процессами.

Рабочие процессы объединяют действия, решения и результаты, которые, запускаясь в определенном порядке, выполняют в виртуальной среде определенную задачу или определенный процесс. Рабочие процессы выполняют такие задачи, как подготовка виртуальных машин, резервное копирование, регулярное обслуживание, отправка электронной почты, выполнение операций по протоколу SSH, управление физической инфраструктурой, а также другие стандартные служебные операции. Рабочие процессы принимают входные данные в соответствии с их функциями. Можно создавать рабочие процессы, которые запускаются в соответствии с определенным расписанием или при возникновении определенных ожидаемых событий. Сведения могут предоставляться вами, другими пользователями, другим рабочим процессом или действием либо внешним процессом, например вызовом веб-службы из приложения. Перед запуском рабочие процессы выполняют определенную проверку и фильтрацию информации.

Рабочие процессы могут вызывать другие рабочие процессы. Например, можно создать рабочий процесс, который вызывает другой рабочий процесс для создания новой виртуальной машины.

Рабочие процессы создаются с помощью интегрированной среды разработки (Integrated Development Environment, IDE) интерфейса vRealize Orchestrator Client, которая обеспечивает доступ к библиотеке рабочих процессов и возможность запуска рабочих процессов в обработчике рабочих процессов. Обработчик рабочих процессов также может получать объекты из внешних библиотек, подключенных к vRealize Orchestrator. Эта функция позволяет настраивать процессы и реализовывать функции, предоставляемые сторонними приложениями.

В эту главу входят следующие разделы:

- [Стандартные рабочие процессы в библиотеке рабочих процессов vRealize Orchestrator](#)
- [Создание рабочих процессов в клиенте vRealize Orchestrator](#)
- [Изменение рабочих процессов и действий в окне родительского рабочего процесса](#)
- [Конструктор форм ввода vRealize Orchestrator](#)
- [Запросы на взаимодействие с пользователем в клиенте vRealize Orchestrator](#)

■ Планирование рабочих процессов в клиенте vRealize Orchestrator

Стандартные рабочие процессы в библиотеке рабочих процессов vRealize Orchestrator

vRealize Orchestrator предоставляет стандартную библиотеку рабочих процессов, которую можно использовать для автоматизации операций в виртуальной инфраструктуре. Рабочие процессы в стандартной библиотеке заблокированы и доступны только для чтения. Чтобы настроить стандартный рабочий процесс, необходимо скопировать его. Скопированные и создаваемые пользователем рабочие процессы можно редактировать.

Содержимое библиотеки рабочих процессов доступно в меню **Библиотека > Рабочие процессы** в vRealize Orchestrator Client на основе HTML5. Как стандартные, так и настраиваемые рабочие процессы упорядочиваются в клиенте с помощью тегов. Например, можно получить доступ к рабочему процессу **Создание пары ключей**, введя **SSH** в поле поиска библиотеки рабочих процессов.

Примечание К стандартному рабочему процессу нельзя добавить новые теги, их можно добавить только к его копии.

Создание рабочих процессов в клиенте vRealize Orchestrator

С помощью vRealize Orchestrator Client можно создавать и редактировать рабочие процессы.

Процедура

1. Войдите в клиент vRealize Orchestrator.
2. Перейдите в раздел **Библиотека > Рабочие процессы**.
3. Щелкните **Создать рабочий процесс**.
4. Введите имя нового рабочего процесса и нажмите **Создать**.
5. С помощью редактора рабочих процессов настройте переменные, входные и выходные, структуру схемы и представление рабочего процесса.
6. Чтобы завершить редактирование рабочего процесса, нажмите кнопку **Сохранить**.

Примечание Для отслеживания изменений в рабочих процессах используйте вкладку **Журнал версий**. Дополнительные сведения см. в разделе [Журнал версий объектов vRealize Orchestrator](#).

Следующие шаги

Для оптимизации производительности рабочих процессов можно использовать компонент воспроизведения маркеров vRealize Orchestrator. Дополнительные сведения см. в разделе [Использование воспроизведения маркера рабочего процесса в клиенте vRealize Orchestrator](#).

Изменение рабочих процессов и действий в окне родительского рабочего процесса

В vRealize Orchestrator Client можно изменять рабочие процессы и действия в окне родительского рабочего процесса.

Редактирование дочерних рабочих процессов и действий напрямую в родительских объектах упрощает их разработку.

Необходимые условия

Создайте рабочий процесс, который вызывает другой рабочий процесс, действие или и то и другое.

Процедура

1. Войдите в клиент vRealize Orchestrator.
2. Перейдите в раздел **Библиотека > Рабочие процессы** и выберите рабочий процесс.
3. Перейдите на вкладку **Схема**.
4. В зависимости от типа объекта дважды щелкните **Элемент рабочего процесса** или **Элемент действия** на холсте рабочего процесса.
5. Измените объект.
6. Чтобы завершить редактирование дочернего рабочего процесса или действия, нажмите кнопку **Сохранить**.
7. Чтобы вернуться к родительскому рабочему процессу, закройте редактор объектов.

Конструктор форм ввода vRealize Orchestrator

Если для рабочего процесса требуются входные параметры, открывается диалоговое окно, в котором пользователи вводят необходимые значения. Содержимое, макет и представление этого диалогового окна можно настроить с помощью конструктора форм ввода.

Конструктор форм ввода находится на вкладке **Форма ввода** редактора рабочих процессов. Этот конструктор состоит из меню переходов, рабочей области проекта и меню свойств. Входные и универсальные элементы можно перетаскивать из левого меню в рабочую область проекта. В рабочей области можно задать положение входных параметров, упорядочить их по вкладкам ввода и настроить их свойства.

Примечание В конструкторе входных форм нельзя использовать содержимое вкладки **Переменные** редактора рабочих процессов. Можно использовать только параметры с вкладки **Ввод-вывод**.

Универсальные элементы

В конструктор форм ввода можно добавлять универсальные элементы, такие как раскрывающиеся меню и текстовые поля для ввода пароля. Универсальные элементы не соответствуют фактическим входным параметрам, но их можно привязать к таким параметрам.

Создание диалогового окна входных параметров рабочего процесса в клиенте vRealize Orchestrator

Конструктор форм ввода можно использовать для создания и настройки диалогового окна входных параметров рабочего процесса.

Необходимые условия

Убедитесь, что рабочий процесс имеет заданный список входных параметров.

Процедура

1. Войдите в клиент vRealize Orchestrator.
2. Перейдите в раздел **Библиотека > Рабочие процессы**.
3. Выберите настраиваемый рабочий процесс.
4. Перейдите на вкладку **Форма ввода**.
5. (дополнительно) Создайте вкладки для использования в диалоговом окне ввода.

Вкладки можно использовать для упорядочения структуры диалогового окна.

6. Выберите входные параметры.
7. Измените свойства входных параметров.

Дополнительные сведения о свойствах входных параметров см. в разделе [Свойства входных параметров в клиенте vRealize Orchestrator](#).

8. (дополнительно) Добавьте в рабочую область универсальные элементы и привяжите их к входным параметрам.
9. (дополнительно) Добавьте внешнюю проверку входных параметров. Дополнительные сведения см. в разделе [Использование действий для проверки входных данных рабочих процессов vRealize Orchestrator](#).
10. Нажмите кнопку **Сохранить**.

Результаты

Вы создали макет диалогового окна рабочего процесса и задали свойства входных параметров.

Свойства входных параметров в клиенте vRealize Orchestrator

Чтобы ограничить значения входных параметров, предоставляемых пользователями при выполнении рабочих процессов vRealize Orchestrator, можно задать свойства параметров.

В vRealize Orchestrator можно определить свойства параметров, используемые для количественного измерения значений входных параметров рабочих процессов. Заданные свойства параметров накладывают ограничения на типы и значения входных параметров, которые пользователи могут вводить в рабочих процессах vRealize Orchestrator.

Свойства параметров проверяют входные параметры и изменяют представление текстовых полей, которые появляются в диалоговом окне входных параметров. Некоторые свойства параметров могут создавать зависимости между параметрами.

Свойство параметра	Описание
Метка	Метка входного параметра.
Тип отображения	Тип отображения текстового поля ввода.
Видимость	Видимость входного параметра.
Только для чтения	Текстовое поле ввода доступно только для чтения.
Настраиваемая справка	Описательное указание на входной параметр.
Значение по умолчанию	Значение входного параметра по умолчанию.
Шаг	Используется для ввода числовых типов. Степень увеличения значения входного параметра за одно нажатие.
Обязательно	Значение входного параметра является обязательным.
Регулярное выражение	Проверка входных данных с помощью регулярного выражения.
Минимальное значение	Минимальное значение или длина параметра.
Максимальное значение	Максимальное значение или длина параметра.
Соответствие текстовому полю	Соответствие значения входного параметра значению другого входного параметра.
Источник значения	Источник значений свойств параметра на вкладках Оформление , Значение и Ограничения . Примечание Можно импортировать значение внешних действий с помощью внешнего источника . Доступные действия фильтруются по типу параметра.

Использование действий для проверки входных данных рабочих процессов vRealize Orchestrator

Используйте внешние действия для проверки входных данных настраиваемых рабочих процессов.

Необходимые условия

Создайте настраиваемый рабочий процесс с входными параметрами. Дополнительные сведения см. в разделе [Создание рабочих процессов в клиенте vRealize Orchestrator](#).

С помощью конструктора форм ввода можно создавать внешние проверки для входных данных рабочего процесса. Во внешних проверках используются сценарии действий, возвращающие строковое значение, если значение входного параметра содержит ошибку. Если значение входного параметра допустимо, внешняя проверка не возвращает ничего.

Процедура

1. Войдите в клиент vRealize Orchestrator.
2. Создайте действие проверки.
 - а) Перейдите в раздел **Библиотека > Действия**.
 - б) Щелкните **Создать действие**.
 - в) Введите необходимую информацию на вкладке **Сводка**.
 - г) Введите входные параметры для действия проверки.

Примечание Имена входных параметров для действия проверки должны совпадать с именами проверяемых входных параметров для рабочего процесса.

- д) Введите сценарий действия проверки на вкладке **Сценарий**.

```
if (in_1=="invalid") {
    return "in_1 can't be invalid!";
}

if (in_2=="invalid") {
    return "in_2 can't be invalid!";
}

//inputs are valid, return nothing
```

Примечание Приведенный выше сценарий — это простой пример, который не является исчерпывающим представлением всех сценариев проверки, которые можно использовать.

- е) Нажмите кнопку **Сохранить**.
3. Примените внешнюю проверку.
 - а) Перейдите в раздел **Библиотека > Рабочие процессы**.
 - б) Выберите настраиваемый рабочий процесс.
 - в) Перейдите на вкладку **Форма ввода**.
 - г) Щелкните значок буфера обмена в левом верхнем углу экрана.
 - д) Перетащите элемент проверки vRealize Orchestrator в рабочую область.
 - е) Выберите элемент проверки, введите метку проверки и выберите действие проверки.
 - ж) (дополнительно) Создайте дополнительные элементы проверки.
 - з) Нажмите кнопку **Сохранить**.
4. Запустите рабочий процесс.

Если при проверке возникает ошибка, возвращается строка. Если проверка прошла успешно, то ничего не возвращается, и рабочий процесс продолжает выполняться.

Результаты

Вы создали внешнюю проверку для настраиваемого рабочего процесса vRealize Orchestrator.

Запросы на взаимодействие с пользователем в клиенте vRealize Orchestrator

Рабочие процессы могут запрашивать дополнительные данные, прежде чем они смогут завершить работу.

Рабочие процессы, требующие дальнейшего взаимодействия с пользователем, приостанавливают работу до тех пор, пока запрашиваемые входные параметры не будут предоставлены пользователем. Рабочие процессы определяют, какие пользователи могут предоставлять запрашиваемую информацию, и отправляют запросы на взаимодействие соответствующим образом. Рабочие процессы, ожидающие ввода данных пользователем, отображаются на панели **Последние выполненные рабочие процессы** панели управления vRealize Orchestrator Client и в меню уведомлений вверху справа.

Планирование рабочих процессов в клиенте vRealize Orchestrator

Для автоматизации выполнения рабочих процессов vRealize Orchestrator можно использовать функцию планирования.

При планировании запуска рабочего процесса необходимо установить дату, время и интервал времени, в течение которого будет запускаться назначенная задача.

Процедура

1. Войдите в клиент vRealize Orchestrator.
2. Выберите рабочий процесс в меню **Библиотека** и на панели рабочих процессов нажмите кнопку **Запланировать**.
3. Настройте параметры запланированной задачи в категориях **Общие**, **Запланировать** и **Рабочий процесс**.

Примечание Категория параметров **Рабочий процесс** доступна только для рабочих процессов, требующих входных параметров.

Параметр	Описание
Имя	Имя назначенной задачи.
Описание	Краткое описание назначенной задачи.
Пуск	Дата и время первого запланированного запуска рабочего процесса.
Запустить, если в прошлом	Выберите, запускать ли рабочий процесс, если запланированное время находится в прошлом. Да : запускает запланированный рабочий процесс немедленно. Нет : запускает рабочий процесс при следующем запланированном повторении.

Параметр	Описание
Запланировать	Установите параметры шаблона повторения и запуска события для назначенной задачи.
Дата окончания	Отображается, если выбран параметр Без повторений . Установите дату и время окончания выполнения назначенной задачи.
Рабочий процесс	Введите входные параметры рабочего процесса.

4. Щелкните **Создать**.

Результаты

Назначенная задача для рабочего процесса создана. Запланированные рабочие процессы отображаются в разделе **Действия > Запланировано**. Чтобы удалить назначенные задачи, нажмите кнопку **Удалить** на панели назначенных задач.

Редактирование назначенных задач в клиенте vRealize Orchestrator

В назначенных задачах можно изменять такие параметры, как дата, время и повторение запланированного рабочего процесса.

Необходимые условия

Создайте назначенную задачу рабочего процесса.

Процедура

1. Войдите в клиент vRealize Orchestrator.
2. Выберите эту назначенную задачу в разделе **Действие > По расписанию**.
3. На панели рабочего процесса выберите **Изменить**.
4. Отредактируйте расписание и нажмите кнопку **Сохранить**.

Примечание Входные параметры, заданные при создании назначенной задачи, доступны только для чтения и не могут быть изменены. Чтобы изменить эти параметры, создайте новую назначенную задачу для этого рабочего процесса.

Управление действиями

6

Рабочие процессы vRealize Orchestrator можно изменять, добавляя сценарии действий.

vRealize Orchestrator Client предоставляет библиотеки предварительно определенных действий и редактор действий для настраиваемых сценариев действий. Действия представляют собой отдельные функции, которые используются в качестве строительных блоков в рабочих процессах.

Действия — это функции JavaScript. Действия могут принимать несколько входных параметров, а возвращают одно значение. Действия можно вызывать для любого объекта API-интерфейса vRealize Orchestrator или объекта любого API-интерфейса, импортированного в vRealize Orchestrator с помощью подключаемого модуля.

При выполнении рабочего процесса действие получает входные параметры из переменных рабочего процесса. Эти переменные могут быть либо начальными входными параметрами рабочего процесса, либо переменными, которые задаются при запуске других элементов рабочего процесса.

Редактор действий включает в себя функцию автозаполнения для сценариев, а также обозреватель API-интерфейсов, отображающий доступные типы сценариев и их документацию.

В эту главу входят следующие разделы:

- [Создание действий в клиенте vRealize Orchestrator](#)
- [Запуск и отладка действий](#)
- [Основные принципы работы со сценариями Python, Node.js и PowerShell](#)
- [Ограничения среды выполнения для сценариев Python, Node.js и PowerShell](#)

Создание действий в клиенте vRealize Orchestrator

В vRealize Orchestrator Client можно создавать, изменять и удалять сценарии действий.

В vRealize Orchestrator, начиная с версии 8.1, поддерживаются следующие среды выполнения:

- Python 3.7
- Node.js 12
- PowerCLI 11/Powershell 6.2

■ PowerCLI 12/Powershell 7

Примечание Среда выполнения PowerCLI включает в себя PowerShell и следующие модули: VMware.PowerCLI, PowerNSX и PowervRA.

Необходимые условия

Для создания сценариев Python, Node.js или PowerShell необходимо знать основные принципы разработки совместимых сценариев для vRealize Orchestrator, рассчитанных на эти среды выполнения. См. раздел [Основные принципы работы со сценариями Python, Node.js и PowerShell](#).

Процедура

1. Войдите в клиент vRealize Orchestrator.
2. Перейдите в раздел **Библиотека > Действия**.
3. Щелкните **Создать действие**.
4. На вкладке **Общие** введите имя действия и имя модуля действия.

Примечание Каждое имя действия и имя модуля действия должно быть уникальным. Имя действия должно быть допустимой функцией JavaScript. Оно должно быть одним словом, в котором содержатся только буквы, цифры, символы доллара (\$) и подчеркивания (_). Имя модуля должно состоять из нескольких слов, разделенных точкой (.).

5. (дополнительно) Создайте описание, номер версии, теги и групповые разрешения для действия.
6. На вкладке **Сценарий** добавьте входные данные действия, выберите тип возвращаемого выходного значения и напишите сценарий.

Примечание Команда **Zip-файл** в раскрывающемся меню **Тип** позволяет импортировать внешний источник сценария, а также, при необходимости, его зависимые модули.

7. Чтобы завершить редактирование действия, нажмите кнопку **Сохранить**.

Должно появиться сообщение о том, что действие сохранено.

Следующие шаги

Пример использования действий vRealize Orchestrator см. в разделе [Интеграция Amazon Web Services и vRealize Orchestrator с помощью Python](#).

Запуск и отладка действий

Можно улучшить действия, выполняя их запуск и отладку непосредственно из редактора действий.

Начиная с версии 8.1, vRealize Orchestrator позволяет запускать и выполнять отладку действия непосредственно из редактора действий клиента vRealize Orchestrator. Эта функция позволяет гарантировать, что действия будут выполняться должным образом после их интеграции в рабочие процессы.

Запуск действий в клиенте vRealize Orchestrator

Разработчику рабочих процессов необходимо запускать действия перед их интеграцией в рабочий процесс.

Необходимые условия

Создайте действие. См. раздел [Создание действий в клиенте vRealize Orchestrator](#).

Процедура

1. Войдите в клиент vRealize Orchestrator.
2. Перейдите в раздел **Библиотека > Действия** и выберите действие, которое нужно запустить.
3. Нажмите **Запустить**.
4. Введите необходимые входные параметры и нажмите **Запустить**.

По завершении выполнения действия перейдите на вкладку **Результаты и входные данные**. Если при выполнении действия произошла ошибка, оно отображается на этой вкладке красным цветом.

Сведения о выполнении действия можно просмотреть в элементе **Результаты действия**.

Примечание Результаты выполнения действия не сохраняются.

Отладка действий в клиенте vRealize Orchestrator

Создатели рабочих процессов могут выполнять отладку действий, вставляя в сценарии точки останова.

vRealize Orchestrator содержит встроенное средство отладки, которое можно использовать для отладки сценария и входных свойств действия. Процесс отладки можно запустить в редакторе действий, вставив точки останова в строки сценария соответствующего действия.

Примечание Встроенное средство отладки работает только с действиями, использующими среду выполнения JavaScript по умолчанию. Пример отладки сценариев действий, в которых используются другие среды выполнения, см. в разделе [Отладка действия Amazon Web Services](#).

Необходимые условия

Создайте действие. См. раздел [Создание действий в клиенте vRealize Orchestrator](#).

Процедура

1. Войдите в клиент vRealize Orchestrator.
2. Перейдите в раздел **Библиотека > Действия** и выберите действие, которое требуется отладить.
3. В редакторе действий добавьте точки останова в строки сценария действия, которые необходимо отладить.
4. Нажмите **Отладка**.
5. Введите входные параметры действия и нажмите **Запустить**.

Начнется выполнение действия в режиме отладки.

6. Когда выполнение действия будет приостановлено по достижении точки останова, выберите один из следующих вариантов.

Параметр	Описание
Продолжить	Выполнение действия возобновляется, пока не будет достигнута другая точка останова или выполнение действия не завершится.
Перейти к	Переход к текущей функции действия. Если отладчик не может обработать текущую строку функции, выполняется операция Пропустить .
Пропустить	Отладчик переходит к следующей строке текущей функции.
Вернуться	Отладчик переходит к строке, которая будет выполняться при возврате текущей функции.

7. (дополнительно) На вкладке **Отладчик** добавьте выражения.
8. (дополнительно) На вкладке **Отладчик** измените значение переменных.

Основные принципы работы со сценариями Python, Node.js и PowerShell

При создании сценария для использования в vRealize Orchestrator необходимо убедиться, что он имеет правильную структуру и форматирование.

Поддерживаемые среды выполнения

В vRealize Orchestrator, начиная с версии 8.1, поддерживаются следующие среды выполнения:

- Python 3.7
- Node.js 12
- PowerCLI 11/Powershell 6.2
- PowerCLI 12/Powershell 7

Примечание Среда выполнения PowerCLI включает в себя PowerShell и следующие модули: VMware.PowerCLI, PowerNSX и PowervRA.

В новые среды выполнения можно добавить любой настраиваемый исходный код, но для принятия контекста и входных данных, а также возврата результатов в модуль vRealize Orchestrator и получения результатов из него необходимо соблюдать требуемый функциональный формат.

Рекомендации по созданию сценариев

Для простых задач создания сценариев в схему рабочего процесса можно добавить элементы **Задача с возможностью добавления сценариев**. Для более сложных задач создания сценариев можно использовать действия vRealize Orchestrator.

Благодаря применению действий можно получить два основных преимущества.

- Действия можно создавать, обновлять, импортировать и экспортировать независимо от рабочих процессов.
- Действия — это автономные объекты, которые можно выполнять и отлаживать в соответствующей среде, что повышает эффективность разработки. См. раздел [Запуск и отладка действий](#).

Требования к функции сценария

По умолчанию функции сценария присваивается имя **handler**. Функция принимает два аргумента: контекст и входные данные. Контекст — это объект сопоставления, в котором содержатся сведения о системе. Например, `vroURL` может содержать URL-адрес вызываемого экземпляра vRealize Orchestrator, а `executionId` — идентификатор маркера выполнения рабочего процесса.

Входные данные — это объект сопоставления, содержащий все входные данные, предоставленные для действий. Например, при определении входных данных в действии под именем `myInput` можно получить доступ к ним из аргумента входных данных, такого как `inputs.myInput` или `inputs["myInput"]`, в зависимости от среды выполнения. Все, что возвращается из функции, является результатом действия. Поэтому тип возвращаемых значений действия должен соответствовать типу содержимого, возвращаемого сценарием в vRealize Orchestrator. Если возвращается примитив, то тип возвращаемого значения действия должен быть числовым. Если возвращается строка, тип возвращаемого значения действия должен быть строковым. Если возвращается сложный объект, тип возвращаемого значения должен сопоставляться с `Properties` или `Composite Type`. Те же принципы применимы и к массивам.

Поддерживаемые типы входных и выходных параметров для сред выполнения Python, Node.js и PowerShell:

- String
- Number
- Boolean
- Date
- Properties
- Composite Type

Определение обработчика записей

По умолчанию обработчику записей присвоено значение `handler.handler`. Это значение указывает на то, что модуль vRealize Orchestrator выполняет поиск файла верхнего уровня в пакете ZIP с именем `handler.py`, `handler.js` или `handler.ps1`, который включает в себя функцию с именем `handler`. Любые различия в именах функции и файла обработчика должны быть отражены в значении обработчика записей. Например, если основной обработчик называется `index.js`, а функция — `callMe`, необходимо задать для обработчика записи значение **`index.callMe`**.

Отладка сценариев среды выполнения во внешней среде IDE

vRealize Orchestrator поддерживает отладку сценариев Python и Node.js во внешней среде IDE.

Отладка сценариев PowerShell во внешней среде IDE невозможна.

Ограничения среды выполнения для сценариев Python, Node.js и PowerShell

В некоторых сценариях Python, Node.js или PowerShell может потребоваться изменить значения памяти и времени ожидания в vRealize Orchestrator Client.

vRealize Orchestrator Client использует для сценариев действий Python, Node.js и PowerShell набор значений памяти и времени ожидания по умолчанию:

- память — 64 МБ;
- время ожидания — 180 секунд.

Если сценарий действия превысит одно или оба значения по умолчанию, произойдет сбой выполнения действия. Например, в сценарии действий может использоваться несколько сторонних модулей зависимостей. В таком случае ограничения памяти по умолчанию 64 МБ может оказаться недостаточно.

Чтобы избежать сбоев при выполнении действий из-за нехватки ресурсов, измените значения памяти и времени ожидания в редакторе действий.

Примечание Кроме того, можно разделить сценарий на несколько элементов задач, которые можно добавить к рабочим процессам в виде сценариев.

Процедура

1. Выполните вход в vRealize Orchestrator Client.
2. Перейдите в раздел **Библиотека > Действия** и выберите нужное действие.
3. Перейдите на вкладку **Сценарий**.
4. В разделе **Ограничения среды выполнения** измените значения памяти и времени ожидания.
5. Нажмите кнопку **Сохранить**.
6. Чтобы проверить новые ограничения среды выполнения, щелкните кнопку **Отладка**.

Управление элементами конфигурации

7

Элемент конфигурации — это список переменных, которые можно использовать для настройки постоянных в рамках всего развертывания сервера vRealize Orchestrator.

Элементы конфигурации позволяют сделать переменные доступными для всех рабочих процессов, действий и политик, выполняемых на данном сервере vRealize Orchestrator.

При создании пакета, содержащего рабочий процесс, действие или политику, которые используют переменную из элемента конфигурации, vRealize Orchestrator автоматически включает элемент конфигурации в этот пакет. При импорте пакета, содержащего элемент конфигурации, на другой сервер vRealize Orchestrator можно также импортировать значения переменных элементов конфигурации. Например, если создается рабочий процесс, использующий значения переменных, зависящие от сервера vRealize Orchestrator, на котором он выполняется, настройка этих переменных в элементе конфигурации позволит экспортировать этот рабочий процесс и использовать его на другом сервере vRealize Orchestrator. Таким образом, элементы конфигурации позволяют серверам легко обмениваться рабочими процессами, действиями и политиками.

Примечание Значения переменной элемента конфигурации, экспортированного из vRealize Orchestrator 5.1 или более ранней версии, импортировать невозможно.

В эту главу входят следующие разделы:

- [Создание элементов конфигурации в клиенте vRealize Orchestrator](#)

Создание элементов конфигурации в клиенте vRealize Orchestrator

С помощью элементов конфигурации можно задать общие переменные для сервера vRealize Orchestrator. Переменные, заданные в элементе конфигурации, могут использоваться всеми элементами, запущенными на сервере.

Процедура

1. Войдите в клиент vRealize Orchestrator.
2. Перейдите в раздел **Активы > Конфигурации**.
3. Выберите **Создать конфигурацию**.

4. Введите имя элемента конфигурации.
5. Перейдите на вкладку **Переменные**.
6. Чтобы создать локальную переменную, нажмите **Создать**.
 - а) Введите имя переменной.
 - б) Выберите тип переменной.

Примечание Чтобы создать массив переменных конфигурации, установите флажок **Массив**.

- в) (дополнительно) Введите значение переменной конфигурации.
 - г) Нажмите кнопку **Сохранить**.
7. Чтобы завершить создание элемента конфигурации, нажмите **Сохранить**.

Следующие шаги

Элемент конфигурации можно использовать в качестве источника переменных для рабочих процессов, действий и политик.

Управление политиками

8

Политики — это триггеры событий, которые отслеживают активность системы. Политики реагируют на предварительно определенные события, вызываемые изменениями в состоянии или производительности определенных объектов vRealize Orchestrator.

Политики представляют собой ряд правил, датчиков, пороговых значений и фильтров событий, которые запускают определенные рабочие процессы или сценарии при возникновении тех или иных предварительно определенных событий в vRealize Orchestrator или в технологиях, к которым vRealize Orchestrator получает доступ через подключаемые модули. Во время работы политики vRealize Orchestrator непрерывно оценивает ее правила. Например, можно внедрить датчики и пороговые значения политик, которые будут отслеживать поведение объектов типа VC:HostSystem и VC:VirtualMachine в vCenter Server.

В эту главу входят следующие разделы:

- [Создание и применение политик в клиенте vRealize Orchestrator](#)
- [Элементы политик в клиенте vRealize Orchestrator](#)
- [Управление запуском политик в клиенте vRealize Orchestrator](#)

Создание и применение политик в клиенте vRealize Orchestrator

Политики можно использовать для мониторинга активности системы vRealize Orchestrator в случае возникновения определенных событий.

Процедура

1. Войдите в клиент vRealize Orchestrator.
2. Перейдите в раздел **Библиотека > Политики**.
3. Выберите **Создать политику**.
Будет создана пустая политика.
4. Введите номер версии и название политики.
5. Перейдите на вкладку **Переменные**.

6. Чтобы создать локальную переменную, нажмите **Создать**.

- а) Введите имя переменной.
- б) Выберите тип переменной.

Примечание Чтобы создать массив переменных политики, установите флажок **Массив**.

- в) Введите значение переменной.

Примечание Чтобы импортировать значение переменной элемента конфигурации, выберите **Привязать к конфигурации**.

- г) Нажмите кнопку **Сохранить**.

7. На вкладке **Определение** добавьте элементы политики и задайте обработчики событий.

Дополнительные сведения об элементах политики см. в разделе [Элементы политик в клиенте vRealize Orchestrator](#).

8. Нажмите кнопку **Сохранить**.

Политика настроена.

Следующие шаги

Чтобы запустить политику, выберите ее и нажмите кнопку **Запустить**. Введите имя запуска политики и, если требуется, входные параметры.

Чтобы просмотреть состояние политики, перейдите в раздел **Действие > Выполняется политика**.

Элементы политик в клиенте vRealize Orchestrator

Элементы политик можно использовать для запуска предварительно определенных рабочих процессов или сценариев vRealize Orchestrator при возникновении того или иного события.

Элемент политик можно добавить для запуска рабочего процесса или сценария в качестве реакции на события, вызываемые объектами. С помощью периодического элемента событий можно запланировать выполнение рабочего процесса или сценария. С помощью корневого элемента можно задать поведение для запуска или остановки политик. В элементах политик могут содержаться обработчики событий, определяющие, когда должны выполняться элементы политик.

Примечание Обработчики событий, которые активируют элементы политик, могут быть рабочими процессами или сценариями действий. Если добавить в обработчик событий одновременно рабочий процесс и сценарий, то политика будет игнорировать срабатывание сценария и будет использовать только срабатывание рабочего процесса.

Обработчик событий	Описание
OnInit	Элемент политики запускается каждый раз при запуске политики.
OnExit	Элемент политики запускается каждый раз при остановке политики.
OnExecute	Используется периодическим элементом события. Активирует элемент политики в течение времени, указанного в периодическом элементе события.

Примечание Технологии, подключенные к базе данных vRealize Orchestrator, могут включать в себя уникальные обработчики событий. Например, подключаемый модуль SNMP позволяет использовать обработчик событий **OnTrap** при создании элементов политик на основе SNMP.

Элементы политик настраиваются на вкладке **Определение** окна изменения политики.

Управление запуском политик в клиенте vRealize Orchestrator

vRealize Orchestrator Client можно использовать для управления приоритетом политик и поведением политик при запуске сервера, когда сервер vRealize Orchestrator перезапускается.

Необходимые условия

Создайте и запустите политику. Дополнительные сведения см. в разделе [Создание и применение политик в клиенте vRealize Orchestrator](#).

Процедура

1. Войдите в клиент vRealize Orchestrator как администратор.
2. Перейдите в раздел **Действия > Выполняется политика**.
3. Выберите запуск политики, которым необходимо управлять.
4. Нажмите **Стоп**.

Состояние политики изменится на **Остановлено**.

5. На вкладке **Общие** задайте приоритет политики и поведение при запуске сервера.
6. Чтобы перезапустить политику, нажмите **Запустить**.

Состояние политики изменяется на **Запущено**.

Управление элементами ресурсов

9

В качестве атрибутов в рабочих процессах могут использоваться объекты, созданные независимо от vRealize Orchestrator. Чтобы использовать внешние объекты в качестве атрибутов в рабочих процессах, импортируйте их на сервер в качестве элементов ресурсов.

Объекты, которые рабочие процессы vRealize Orchestrator могут использовать в качестве элементов ресурсов, включают в себя файлы изображений, сценарии, шаблоны XML, HTML-файлы и т. д. Любые рабочие процессы, выполняемые на сервере vRealize Orchestrator, могут использовать любые элементы ресурсов, импортированные в vRealize Orchestrator.

После импорта объекта в vRealize Orchestrator в качестве элемента ресурсов можно изменить объект в одном месте и автоматически распространить эти изменения во все рабочие процессы, которые используют этот элемент ресурсов.

Максимальный размер элемента ресурсов составляет 16 МБ.

Элементы ресурсов можно импортировать, экспортировать, восстанавливать, обновлять и удалять.

Управление пакетами

10

Используйте vRealize Orchestrator Client для создания, экспорта и импорта пакетов. Пакеты позволяют экспортировать объекты рабочих процессов для их использования в других экземплярах vRealize Orchestrator.

Пакеты могут содержать рабочие процессы, действия, политики, элементы конфигурации и элементы ресурсов.

При добавлении элемента в пакет vRealize Orchestrator проверяет зависимости и добавляет в пакет все зависимые элементы. Например, если добавить рабочий процесс, использующий действия или другие рабочие процессы, vRealize Orchestrator добавит эти действия и рабочие процессы в пакет.

При импорте пакета сервер сравнивает версии различных элементов его содержимого с соответствующими локальными элементами. Сравнение показывает различия между версиями локальных и импортированных элементов. Пользователь может импортировать весь пакет или выбрать отдельные элементы для импорта.

Пакеты являются единственным средством экспорта и импорта большинства объектов, созданных в vRealize Orchestrator Client (за исключением элементов ресурсов).

Для контроля за тем, как принимающий сервер может использовать содержимое пакета, в пакетах используются средства управления цифровыми правами. vRealize Orchestrator подписывает и шифрует пакеты для защиты данных. Для отслеживания того, какие пользователи экспортируют и распространяют элементы, в пакетах используются сертификаты X509.

Создание пакета в клиенте vRealize Orchestrator

Используя пакеты, можно экспортировать и импортировать рабочие процессы, политики, действия, ссылки подключаемых модулей, элементы ресурсов и элементы конфигурации. Все зависимые элементы, связанные с объектами в пакете, добавляются в пакет автоматически, что обеспечивает совместимость между версиями. Чтобы удалить зависимые элементы, необходимо сначала удалить связанный с ним объект в пакете.

Пакеты являются единственным средством экспорта и импорта большинства объектов, созданных в vRealize Orchestrator Client (за исключением элементов ресурсов).

Необходимые условия

Убедитесь, что сервер vRealize Orchestrator содержит объекты, которые можно добавить в пакет, такие как рабочие процессы, действия и политики.

Процедура

1. Войдите в клиент vRealize Orchestrator.
2. Перейдите в раздел **Активы > Пакеты**.
3. Щелкните **Создать пакет**.
4. На вкладке **Общие** введите имя и описание пакета.

Примечание В именах пакетов в vRealize Orchestrator Client нельзя использовать специальные символы.

5. На вкладке **Содержимое** нажмите **Добавить**.
6. Выберите объекты, которые необходимо добавить в пакет, и нажмите кнопку **Добавить**.

Примечание Зависимые элементы добавляются в пакет автоматически, но не отображаются на вкладке **Содержимое** во время создания пакета. Для просмотра зависимых элементов перейдите на вкладку **Содержимое** после создания пакета.

7. Чтобы завершить создание пакета, нажмите **Создать**.

Экспорт пакетов в клиенте vRealize Orchestrator

С помощью vRealize Orchestrator Client можно экспортировать пакеты в другую среду vRealize Orchestrator.

Необходимые условия

Создайте пакет с объектами vRealize Orchestrator, которые требуется экспортировать. Дополнительные сведения см. в разделе [Создание пакета в клиенте vRealize Orchestrator](#).

Процедура

1. Войдите в клиент vRealize Orchestrator.
2. Перейдите в раздел **Активы > Пакеты**.
3. Выберите пакет и нажмите **Экспорт**.

4. (дополнительно) Выберите дополнительные параметры экспорта.

Параметр	Описание
Добавить значения атрибутов конфигурации в пакет	Экспорт значений атрибутов элементов конфигурации.
Добавить значения атрибутов SecureString конфигурации в пакет	Экспорт значений атрибутов конфигурации SecureString.
Добавить глобальные теги в пакет	Экспорт глобальных тегов.

5. Установите права доступа для пользователей, импортирующих пакет.

Параметр	Описание
Просмотр содержимого	Пользователь может просматривать содержимое пакета.
Добавить в пакет	Пользователь может добавлять содержимое из импортированного пакета в другие пакеты.
Изменить содержимое	Пользователь может редактировать содержимое пакета.

6. Нажмите кнопку **ОК**.

Примечание Файлы с расширением `.package` сохраняются в папке по умолчанию на локальном компьютере. Чтобы задать пользовательскую папку, измените параметры хранения в обозревателе.

Результаты

Пакет экспортирован. Теперь экспортированные объекты можно использовать в другой среде vRealize Orchestrator.

Импорт пакетов в клиенте vRealize Orchestrator

С помощью vRealize Orchestrator Client можно импортировать пакеты рабочих процессов. Импорту пакетов позволяет использовать объекты сервера vRealize Orchestrator на другом сервере.

Необходимые условия

- Создайте резервные копии всех стандартных объектов vRealize Orchestrator, которые были изменены.
- На удаленном сервере создайте и экспортируйте пакет с объектами, которые необходимо импортировать.

Процедура

1. Войдите в клиент vRealize Orchestrator.
2. Перейдите в раздел **Активы > Пакеты**.
3. Нажмите **Импорт**, найдите файл `.package`, который требуется импортировать, и щелкните **Открыть**.

4. Ознакомьтесь со сведениями об импортируемом пакете.

- а) На вкладке **Общие** содержатся сведения об импортированном пакете: его имя, описание, количество входящих в пакет элементов и сведения о сертификате.

Перед импортом файла иногда требуется указать, что вы доверяете сертификату издателя исходного экземпляра vRealize Orchestrator.
- б) На вкладке **Элементы пакета** указаны объекты, включенные в файл импорта. Если версия объекта в пакете является более поздней, чем версия на сервере, для импорта система выбирает более позднюю версию. Предыдущие версии элементов vRealize Orchestrator необходимо выбрать вручную.
- в) Если не требуется импортировать из пакета значения атрибутов элементов конфигурации, снимите флажок **Импортировать значения атрибутов конфигурации**.
- г) В раскрывающемся меню выберите, следует ли импортировать теги.

5. Щелкните **Импортировать.**

Устранение неполадок в клиенте vRealize Orchestrator

11

С помощью показателей, воспроизведения маркеров, проверки и отладки можно выполнять диагностику и мониторинг экземпляра vRealize Orchestrator.

В эту главу входят следующие разделы:

- [Данные показателей в клиенте vRealize Orchestrator](#)
- [Использование воспроизведения маркера рабочего процесса в клиенте vRealize Orchestrator](#)
- [Проверка рабочих процессов vRealize Orchestrator](#)
- [Отладка сценариев рабочих процессов в клиенте vRealize Orchestrator](#)
- [Отладка рабочих процессов по элементам схемы](#)

Данные показателей в клиенте vRealize Orchestrator

Администраторы vRealize Orchestrator могут использовать профилирование рабочего процесса и показатели панели управления системой для устранения неполадок в системе и рабочих процессах vRealize Orchestrator.

Функция профилирования собирает данные показателей о выполнении рабочих процессов.

Профилирование рабочего процесса включено по умолчанию. Автоматическое профилирование можно отключить в разделе **Центр управления > Свойства расширения > profiler-8.2.0**.

Другим источником данных показателей в vRealize Orchestrator Client является панель управления системой, которая предоставляет показатели системного уровня. Дополнительные сведения см. в разделе [Использование панели управления системой vRealize Orchestrator](#).

Профилирование рабочих процессов в клиенте vRealize Orchestrator

Выполнение рабочего процесса можно профилировать для устранения неполадок и оптимизации среды vRealize Orchestrator.

Функцию профилирования vRealize Orchestrator Client можно использовать для сбора полезных данных о выполнении рабочих процессов. Эти данные можно использовать для оптимизации производительности рабочих процессов. По умолчанию профилирование выполняемых рабочих процессов выполняется автоматически. Можно отключить автоматическое профилирование на странице **Свойства расширения** в центре управления vRealize Orchestrator и запускать профилирование вручную. Для выполнения профилирования вручную найдите рабочий процесс в библиотеке и выберите **Действия >**

Профилирование.

Необходимые условия

Запустите рабочий процесс.

Процедура

1. Войдите в клиент vRealize Orchestrator.
2. Перейдите в раздел **Действия > Выполняемые рабочие процессы**.
3. Выберите выполняемый рабочий процесс.

В схеме запуска рабочего процесса можно просмотреть данные об отдельных элементах рабочего процесса. Эти данные включают общую продолжительность выполнения, максимальную продолжительность и количество запусков элементов. Эту информацию можно отфильтровать в раскрывающемся меню в правом верхнем углу страницы.

4. Перейдите на вкладку **Быстродействие**.

На этой вкладке представлены показатели работы рабочего процесса: время ЦП, продолжительность выполнения, размер маркера и данные элемента рабочего процесса.

Примечание Если выполнение рабочего процесса приостановлено, например когда рабочий процесс ожидает дальнейшего ввода данных, показатель времени ЦП захватывает только поток среды выполнения, который выполнялся перед завершением.

Следующие шаги

Используйте данные профилирования для оптимизации рабочего процесса.

Использование панели управления системой vRealize Orchestrator

Администраторы могут использовать панель управления системой vRealize Orchestrator Client для сбора полезных данных показателей об узлах среды vRealize Orchestrator.

Чтобы открыть панель управления системой, перейдите на вкладку **Система** в верхней части страницы панели управления vRealize Orchestrator Client. Представленные данные включают:

- Состояние узла
- Свойства узла
- Настройки кластера. Настройки кластера на панели управления системой можно только просматривать. Чтобы изменить эти настройки, перейдите на страницу **Управление кластером Orchestrator** в центре управления vRealize Orchestrator.

- Сведения о потоках
- Память кучи
- Память вне кучи
- Использование файловой системы
- Данные проверки подлинности
- Пул подключений к базе данных Orchestrator
- Входные аргументы процесса

Эти данные можно использовать для мониторинга состояния отдельных узлов среды vRealize Orchestrator и устранения неполадок. Для перемещения между отдельными узлами используйте соответствующие вкладки сверху панели управления системой.

Использование воспроизведения маркера рабочего процесса в клиенте vRealize Orchestrator

Для просмотра переходов между элементами во время выполнения рабочего процесса можно использовать функцию воспроизведения маркеров.

Функция воспроизведения маркера записывает контекстные сведения для каждого перехода между элементами рабочего процесса. Для каждого элемента рабочего процесса записывается, когда рабочий процесс был запущен, завершен и какие переменные были изменены к концу выполнения элемента рабочего процесса. Воспроизведение маркера также ссылается на созданные сообщения журнала сценариев для каждого элемента рабочего процесса.

Примечание Данные о переходах элементов рабочих процессов хранятся в базе данных PostgreSQL vRealize Orchestrator. Эти данные удаляются из базы данных при удалении выполняемого рабочего процесса.

Необходимые условия

- Включите функцию воспроизведения маркеров в центре управления.
 - а) Войдите в центр управления от имени пользователя **root**.
 - б) Выберите **Свойства расширения**.
 - в) Щелкните **tokenreplay-8.2.0**.
 - г) Чтобы включить функцию воспроизведения маркеров, щелкните **Включить**.
 - д) Нажмите кнопку **Сохранить**.

Примечание Для обновления расширения на сервере vRealize Orchestrator может потребоваться до 5 минут.

- Запустите рабочий процесс.

Примечание По умолчанию воспроизведение маркера не выполняется автоматически для всех выполняемых рабочих процессов на сервере vRealize Orchestrator. Воспроизведение маркеров можно выполнять для каждого рабочего процесса по отдельности или включить расширение для воспроизведения маркеров для всех рабочих процессов на странице **Свойства расширения** в центре управления.

Процедура

1. (дополнительно) Включите воспроизведение маркеров для всех выполняемых рабочих процессов на сервере vRealize Orchestrator.

Примечание Чтобы запустить воспроизведение отдельных маркеров, не включая функцию в центре управления, щелкните **Запустить с воспроизведением** на странице редактора рабочих процессов.

- а) Войдите в центр управления от имени пользователя **root**.
- б) Выберите **Свойства расширения**.
- в) Щелкните **tokenreplay-8.2.0**.
- г) Чтобы включить функцию преобразования маркера для всех рабочих процессов, убедитесь, что параметр **Воспроизведение записи для всех выполняемых рабочих процессов** включен.
- д) Нажмите кнопку **Сохранить**.

Примечание Для обновления расширения на сервере vRealize Orchestrator может потребоваться до 5 минут.

2. Войдите в клиент vRealize Orchestrator как администратор.
3. Перейдите в раздел **Действия > Выполняемые рабочие процессы**.
4. Выберите выполняемый рабочий процесс.
5. В меню слева выберите элемент выполняемого рабочего процесса.

Теперь на вкладках **Переменная** и **Журналы** отображается информация, относящаяся к этому элементу рабочего процесса.

Проверка рабочих процессов vRealize Orchestrator

vRealize Orchestrator предоставляет средство проверки рабочих процессов. Проверка рабочего процесса помогает обнаружить ошибки в рабочем процессе и проверить, что данные правильно передаются от одного элемента к другому.

По умолчанию vRealize Orchestrator всегда выполняет проверку рабочего процесса при его запуске.

При проверке рабочего процесса средство проверки создает список всех ошибок и предупреждений. Если щелкнуть ошибку в списке, то элемент рабочего процесса, содержащий ошибку, выделяется.

Если запустить средство проверки в редакторе рабочих процессов, то оно предложит для обнаруженных ошибок быстрые исправления. Для одних быстрых исправлений требуются дополнительные сведения или входные параметры. Другие быстрые исправления позволяют сразу устранить ошибку.

При проверке рабочего процесса проверяются привязки данных и соединения между элементами. Проверка рабочего процесса не контролирует обработку данных, выполняемую каждым элементом рабочего процесса. В результате корректный рабочий процесс может выполняться неправильно и выдавать ошибочные результаты, если функция в элементе схемы является неправильной.

Проверка рабочего процесса и устранение ошибок проверки в клиенте vRealize Orchestrator

Прежде чем рабочий процесс можно будет запустить, его необходимо проверить. Ошибки, выявленные в ходе проверки, можно исправить только в том случае, если рабочий процесс открыт для редактирования.

Необходимые условия

Убедитесь, что рабочий процесс полностью готов к проверке и у него определены связанные элементы схемы и привязки.

Процедура

1. Войдите в клиент vRealize Orchestrator как администратор.
2. Перейдите в раздел **Библиотека > Рабочие процессы** и выберите рабочий процесс, который нужно проверить.
3. Щелкните элемент **Изменить**.
4. Щелкните пункт **Проверить** в меню сверху.

Если рабочий процесс прошел проверку, появится сообщение с подтверждением. Если рабочий процесс не прошел проверку, отобразится список ошибок.

5. Щелкните сообщение об ошибке для не прошедшего проверку рабочего процесса и выполните соответствующие действия, чтобы устранить проблему.

Средство проверки выделяет элемент схемы, в котором произошла ошибка, добавляя к ней красный значок. Если это возможно, средство проверки отображает действие для быстрого исправления.

- Если вы согласны с предложенным быстрым исправлением, щелкните его, чтобы выполнить действие.
- Если вы не согласны с предложенным действием быстрого исправления, закройте диалоговое окно проверки рабочего процесса и исправьте элемент схемы вручную.

Важно! Всегда проверяйте, подходит ли решение, которое предлагает vRealize Orchestrator.

Например, может быть предложено удалить неиспользуемый атрибут, хотя фактически это может быть неправильно привязанный атрибут.

6. Повторяйте предыдущие шаги, пока не будут устранены все ошибки проверки.

Результаты

Рабочий процесс проверен, а ошибки проверки устранены.

Следующие шаги

Теперь можно запустить рабочий процесс.

Отладка сценариев рабочих процессов в клиенте vRealize Orchestrator

Можно выполнять отладку выполняемых рабочих процессов, вставив точки останова в сценарий элементов рабочего процесса.

Есть несколько вариантов продолжения процесса отладки после достижения точки останова. При отладке элемента из схемы рабочего процесса можно просмотреть общие сведения о выполнении рабочего процесса, изменить переменные рабочего процесса, добавить выражения для отслеживания и просмотреть сообщения журнала.

Примечание Всегда выполняйте отладку сценариев в непродизвожденной среде.

Процедура

1. Войдите в клиент vRealize Orchestrator как администратор.
2. Выберите рабочий процесс из библиотеки.
3. Откройте схему рабочего процесса, выберите элемент рабочего процесса и перейдите на вкладку **Сценарии**.
4. Чтобы вставить точку останова, щелкните красный кружок слева от номера строки.

Примечание Точки останова можно вставлять только в элементы рабочего процесса со сценариями.

5. Чтобы запустить рабочий процесс в режиме отладки, нажмите кнопку **Отладка**.

Если для рабочего процесса требуются входные параметры, необходимо предоставить их.

6. Когда выполнение рабочего процесса будет приостановлено по достижении точки останова, выберите один из следующих вариантов.

Параметр	Описание
Продолжить	Выполнение рабочего процесса возобновляется и продолжается, пока не будет достигнута другая точка останова или выполняемый рабочий процесс не завершится.
Перейти к	Этот параметр можно использовать для перехода к элементу рабочего процесса. Во время отладки рабочего процесса в редакторе рабочих процессов нельзя выполнить переход во вложенный элемент рабочего процесса.
Пропустить	Пропускает текущий элемент в схеме и приостанавливает выполнение рабочего процесса на следующем элементе.

Примечание Можно указать отладчику, что текущую точку останова следует игнорировать, щелкнув ее. В результате символ точки останова изменится на зеленый треугольник.

7. (дополнительно) На вкладке **Отладчик** вставьте выражения для просмотра.
Можно использовать выражения для отслеживания выполнения определенных переменных.
8. (дополнительно) На вкладке **Отладчик** измените значения переменных.

Отладка рабочих процессов по элементам схемы

Создатели рабочих процессов могут выполнять отладку отдельных элементов схемы.

Процедура

1. Выполните вход в vRealize Orchestrator Client.
2. Перейдите в раздел **Библиотека > Рабочие процессы** и выберите рабочий процесс.
3. Перейдите на вкладку **Схема**.
4. Выберите элемент рабочего процесса, который необходимо отладить, и нажмите кнопку «Отладка» в левом верхнем углу элемента.

Примечание Можно добавить точку останова к элементу схемы **Элемент рабочего процесса**, после чего выполнять отладку дочерних рабочих процессов непосредственно из родительского рабочего процесса. Когда отладчик достигает элемента схемы **Элемент рабочего процесса**, открывается представление схемы дочернего рабочего процесса.

5. Повторите эти действия для всех элементов схемы, которые необходимо отладить.
6. Нажмите **Отладка**.
7. Введите требуемые значения входных параметров и нажмите **Запустить**.

Начнется выполнение рабочего процесса, которое будет приостановлено, когда отладчик достигнет элемента схемы с точкой останова.

8. При достижении точки останова выберите один из следующих параметров.

Параметр	Описание
Продолжить	Выполнение рабочего процесса возобновляется и продолжается, пока не будет достигнута другая точка останова или выполняемый рабочий процесс не завершится.
Перейти к	Переход к текущей функции рабочего процесса. Если отладчик не может обработать текущую строку функции, выполняется операция Пропустить .
Пропустить	Отладчик переходит к следующей строке текущей функции.
Вернуться	Отладчик переходит к строке, которая будет выполняться при возврате текущей функции.

9. (дополнительно) На вкладке **Переменные** измените значение переменных рабочего процесса.

The screenshot shows the debug console for 'Item1 (My Orchestrator Task)'. The 'Scripting' tab is selected, displaying the following JavaScript code:

```

1 var len=VMs.length;
2 for (var i=0; i < len; i++) {
3
4   var VM = VMs[i];
5   //var workflowLaunch = Server.getWorkflowById("workflowId");
6   var workflowLaunch = workflowId;
7   if (workflowLaunch == null) {
8     throw "Workflow not found";
9   }
10  var workflowParameters = new Properties();
11  workflowParameters.put("var", VM);
12  var wToken = workflowLaunch.execute(workflowParameters);
13  System.log ("Run workflow on " + VM.name);

```

Below the code, the 'Variables' tab is active, showing a table of variables and their values:

Variable	Value
i	Not set
VM	test
workflowToLaunch	Not set
workflowParameters	Not set
wToken	Not set
VM	test
workflowToRun	test