

VMware vRealize Orchestrator 플러그인 사용

vRealize Orchestrator 7.5

다음 VMware 웹 사이트에서 최신 기술 문서를 확인할 수 있습니다.

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

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

VMware 코리아
서울시 강남구
영동대로 517
아셈타워 13층
(우) 06164
전화: +82 2 3016 6500
팩스: +82 2 3016 6501
www.vmware.com/kr

Copyright © 2008-2018 VMware, Inc. All rights reserved. [저작권 및 상표 정보](#)

목차

VMware vRealize Orchestrator 플러그인 사용 10

1 Orchestrator 플러그인 소개 11

Orchestrator 아키텍처 11

Orchestrator 서버와 함께 설치되는 플러그인 12

Orchestrator API 탐색기 액세스 15

2 Orchestrator 플러그인 구성 16

Orchestrator 플러그인 관리 16

플러그인 제거 17

3 vCenter Server 플러그인 사용 19

vCenter Server 플러그인 구성 19

구성 워크플로 20

vCenter Server 인스턴스에 대한 연결 구성 20

vCenter Server 플러그인 스크립팅 API 22

vCenter Server 플러그인 인벤토리 사용 22

쿼리를 위한 성능 고려 사항 22

vCenter Server 플러그인으로 XPath 표현식 사용 23

vCenter Server 플러그인으로 XPath 표현식 사용 23

vCenter Server 플러그인 워크플로 라이브러리 액세스 24

vCenter Server 플러그인 워크플로 라이브러리 25

일괄 처리 워크플로 28

클러스터 및 계산 리소스 워크플로 28

구성 워크플로 29

사용자 지정 특성 워크플로 29

데이터 센터 워크플로 29

데이터스토어 및 파일 워크플로 30

데이터 센터 폴더 관리 워크플로 30

호스트 폴더 관리 워크플로 31

가상 시스템 폴더 관리 워크플로 31

게스트 작업 파일 워크플로 31

게스트 작업 프로세스 워크플로 32

전원 호스트 관리 워크플로 32

기본 호스트 관리 워크플로 33

호스트 등록 관리 워크플로 33

| | |
|------------------------|----|
| 네트워킹 워크플로 | 33 |
| 분산 가상 포트 그룹 워크플로 | 34 |
| 분산 가상 스위치 워크플로 | 34 |
| 표준 가상 스위치 워크플로 | 34 |
| 네트워킹 Virtual SAN 워크플로 | 35 |
| 리소스 풀 워크플로 | 35 |
| 스토리지 워크플로 | 36 |
| Storage DRS 워크플로 | 36 |
| 스토리지 VSAN 워크플로 | 37 |
| 기본 가상 시스템 관리 워크플로 | 37 |
| 복제 워크플로 | 38 |
| 연결된 복제 워크플로 | 39 |
| Linux 사용자 지정 복제 워크플로 | 39 |
| 도구 복제 워크플로 | 39 |
| Windows 사용자 지정 복제 워크플로 | 40 |
| 디바이스 관리 워크플로 | 41 |
| 이동 및 마이그레이션 워크플로 | 41 |
| 기타 워크플로 | 42 |
| 전원 관리 워크플로 | 43 |
| 스냅샷 워크플로 | 43 |
| VMware Tools 워크플로 | 44 |

4 vRealize Automation 플러그인 사용 45

| | |
|---|----|
| vRealize Automation용 VMware vRealize Orchestrator 플러그인 소개 | 45 |
| vRealize Automation 플러그인을 사용하는 vRealize Orchestrator의 역할 | 46 |
| vRealize Automation 플러그인 구성 | 46 |
| 구성 워크플로 | 46 |
| vRealize Automation 플러그인 워크플로 사용 | 49 |
| 제거 작업 제한 사항 | 49 |
| vRealize Automation 플러그인 인벤토리 사용 | 52 |
| vRealize Automation 플러그인 관리 워크플로 사용 | 52 |
| vRealize Automation 플러그인 인프라 관리 워크플로 사용 | 58 |
| vRealize Automation 플러그인 요청 워크플로 사용 | 61 |
| vRealize Automation 플러그인 샘플 워크플로 사용 | 62 |
| vRealize Automation 플러그인 API 액세스 | 63 |
| 예제 vRealize Automation 플러그인 스크립트 | 63 |
| CRUD 인프라 관리 작업 예제 스크립트 | 63 |
| vRealize Automation 엔터티 찾기 예제 스크립트 | 67 |
| vRealize Automation 예제 스크립트를 통해 프로비저닝된 리소스 가져오기 | 69 |
| 일반 작업 예제 스크립트 | 70 |

5 구성 플러그인 사용 74

구성 플러그인 워크플로 라이브러리 액세스 74

구성 플러그인 워크플로 라이브러리 74

6 라이브러리 플러그인 사용 76

라이브러리 플러그인 워크플로 76

7 SQL 플러그인 사용 78

SQL 플러그인 구성 78

SQL 플러그인 구성 워크플로 78

데이터베이스 추가 79

데이터베이스에 테이블 추가 80

데이터베이스 업데이트 80

SQL 샘플 워크플로 실행 81

JDBC URL 생성 81

JDBC 연결 테스트 82

JDBC를 사용하여 테이블 생성 82

JDBC 테이블에 행 삽입 83

JDBC 테이블에서 행 선택 84

JDBC 테이블에서 항목 삭제 84

JDBC 테이블에서 모든 항목 삭제 85

JDBC 테이블 삭제 86

전체 JDBC 주기 실행 86

SQL 플러그인 표준 워크플로 사용 87

SQL 플러그인 워크플로 라이브러리 87

테이블에 대한 CRUD 워크플로 생성 87

8 SSH 플러그인 사용 89

SSH 플러그인 구성 89

구성 워크플로 90

SSH 플러그인 샘플 워크플로 실행 90

키 쌍 생성 91

키 쌍 암호 변경 92

SSH 호스트에 Orchestrator 공개 키 등록 92

SSH 명령 실행 93

SSH 호스트에서 파일 복사 93

SSH 호스트에 파일 복사 94

9 XML 플러그인 사용 95

| | |
|-----------------------------|------------|
| XML 플러그인 샘플 워크플로 실행 | 95 |
| 간단한 XML 문서 생성 | 96 |
| XML 문서에서 요소 찾기 | 96 |
| XML 문서 수정 | 97 |
| XML에서 예제 주소록 생성 | 98 |
| 10 메일 플러그인 사용 | 99 |
| 기본 SMTP 연결 정의 | 99 |
| 메일 플러그인 샘플 워크플로 사용 | 100 |
| 메일 플러그인 샘플 워크플로 액세스 | 100 |
| 메일 플러그인 샘플 워크플로 | 100 |
| 11 네트워크 플러그인 사용 | 102 |
| 12 열거형 플러그인 사용 | 103 |
| 표준 시간대 코드 | 103 |
| 13 워크플로 설명서 플러그인 사용 | 107 |
| 워크플로 설명서 플러그인용 워크플로 라이브러리 | 107 |
| 워크플로 설명서 생성 | 107 |
| 14 HTTP-REST 플러그인 사용 | 109 |
| HTTP-REST 플러그인 구성 | 109 |
| 구성 워크플로 | 109 |
| Kerberos 인증 구성 | 110 |
| REST 호스트 추가 | 111 |
| REST 작업 추가 | 113 |
| REST 호스트에 스키마 추가 | 114 |
| REST 작업에서 새 워크플로 생성 | 115 |
| REST 작업 호출 | 115 |
| REST 작업 호출 | 116 |
| 15 SOAP 플러그인 사용 | 117 |
| SOAP 플러그인 구성 | 117 |
| 구성 워크플로 | 117 |
| SOAP 호스트 추가 | 118 |
| Kerberos 인증 구성 | 119 |
| SOAP 작업에서 새 워크플로 생성 | 121 |
| 사용자 지정 생성된 워크플로 테스트 | 122 |
| SOAP 작업 호출 | 122 |

16 AMQP 플러그인 사용 123

AMQP 플러그인 구성 123

구성 워크플로 123

브로커 추가 123

대기열 구독 124

브로커 업데이트 125

AMQP 플러그인 표준 워크플로 사용 125

바인딩 선언 126

대기열 선언 126

Exchange 선언 127

문자 메시지 전송 128

바인딩 삭제 129

17 SNMP 플러그인 사용 130

SNMP 디바이스 관리 130

디바이스 관리 워크플로 130

SNMP 디바이스 등록 131

SNMP 쿼리 관리 132

쿼리 관리 워크플로 132

SNMP 디바이스에 쿼리 추가 132

SNMP 트랩 호스트 관리 133

트랩 호스트 관리 워크플로 133

SNMP 트랩 포트 설정 133

SNMP 트랩 수신 134

SNMP 디바이스에서 트랩 대기 134

SNMP 트랩 정책 설정 135

SNMP 트랩 호스트 정책 구성 136

트랩 정책 편집 136

일반 SNMP 요청 워크플로 137

18 Active Directory 플러그인 사용 138

Active Directory 플러그인 구성 138

Active Directory 구성 워크플로 138

Active Directory 플러그인 워크플로 라이브러리 사용 139

Active Directory 플러그인 인벤토리 사용 139

Active Directory 플러그인 워크플로 라이브러리 액세스 139

Active Directory 플러그인 워크플로 139

19 동적 유형 플러그인 사용 142

동적 유형 구성 워크플로 142

20 PowerShell 플러그인 사용 144

VMware vRealize Orchestrator PowerShell 플러그인 소개 144

PowerShell 플러그인 구성 요소 145

WinRM 구성 147

Kerberos 인증 구성 150

PowerShell 플러그인 구성 152

구성 워크플로 152

PowerShell 호스트 추가 152

PowerShell 플러그인 인벤토리 사용 153

PowerShell 스크립트 실행 154

PowerShell 스크립트 호출 154

외부 스크립트 호출 154

작업 생성 155

PowerShell 스크립트에서 작업 생성 155

PowerShell Cmdlet용 작업 생성 156

작업 간 호출 결과 전달 157

PowerShell 플러그인과 PowerCLI 통합 157

Converter 워크플로 158

샘플 워크플로 158

PowerShell 플러그인 API 액세스 159

PowerShell 결과 작업 159

일반 PowerShell 작업의 스크립트 예제 160

문제 해결 162

Kerberos 이벤트 로깅 사용 162

Kerberos 데이터베이스에서 서버를 찾을 수 없음 163

Kerberos 티켓을 가져올 수 없음 163

다른 시간 설정으로 인한 Kerberos 인증 실패 164

Kerberos 인증 세션 모드 실패 164

영역의 키 배포 센터에 연결할 수 없음 164

기본 영역을 찾을 수 없음 165

21 Multi-Node 플러그인 사용 166

vRealize Orchestrator Multi-Node 플러그인 소개 166

Multi-Node 플러그인 구성 167

서버 구성 워크플로 167

Orchestrator 서버 추가 167

프록시 워크플로 사용 168

동기 프록시 워크플로 168

| | |
|--|------------|
| 비동기 프록시 워크플로 | 169 |
| 원격 실행 워크플로 | 169 |
| Multi-Node 플러그인 인벤토리 사용 | 170 |
| 원격 관리 워크플로 | 170 |
| Multi-Node 플러그인 API 액세스 | 171 |
| 다중 노드 플러그인 사용 사례 | 171 |
| 다중 프록시 작업 생성 | 172 |
| 원격 및 프록시 워크플로 유지 관리 | 173 |
| 로컬 서버에서 패키지 배포 | 173 |
| 22 vAPI(vCloud Suite API) 플러그인 사용 | 175 |
| vCloud Suite API 플러그인 구성 | 175 |
| vCloud Suite API 메타 모델 가져오기 | 175 |
| vCloud Suite API 끝점 추가 | 176 |
| vCloud Suite API 플러그인 API 액세스 | 177 |

VMware vRealize Orchestrator 플러그인 사용

"VMware vRealize Orchestrator 플러그인 사용"에서는 VMware® vRealize Orchestrator와 함께 설치되는 표준 플러그인 집합의 구성 및 사용에 대한 정보와 지침을 제공합니다.

대상 사용자

이 정보는 가상 시스템 기술과 데이터 센터 작업에 익숙한 고급 vSphere 관리자 및 숙련된 시스템 관리자를 대상으로 제공됩니다.

Orchestrator 플러그인 소개

1

Orchestrator 플러그인을 사용하여 외부 기술과 애플리케이션을 액세스하고 제어할 수 있습니다.

Orchestrator 플러그인에서 외부 기술을 노출하면 개체 및 기능을 워크플로에 포함하여 해당 외부 기술의 개체에서 이러한 워크플로를 실행할 수 있습니다.

플러그인을 사용하여 액세스하는 외부 기술에는 가상화 관리 도구, 이메일 시스템, 데이터베이스, 디렉토리 서비스, 원격 제어 인터페이스 등이 포함됩니다.

Orchestrator는 VMware vCenter Server API, 이메일 및 인증 기능, 기타 기술 등을 노출하는 사전 설치된 표준 플러그인 집합을 제공합니다. Orchestrator 개방형 플러그인 아키텍처를 사용하여 다른 애플리케이션에 액세스하는 플러그인을 개발할 수도 있습니다. Orchestrator는 외부 시스템과의 통합을 간소화하는 개방형 표준을 구현합니다. 사용자 지정 콘텐츠 개발에 대한 자세한 내용은 "VMware vRealize Orchestrator를 사용한 개발"을 참조하십시오.

표준 플러그인 집합은 Orchestrator 서버와 함께 자동으로 설치됩니다. 사용을 시작하기 전에 vCenter Server 플러그인과 같은 일부 플러그인을 구성해야 할 수도 있습니다.

플러그인은 새 개체 유형 및 메서드로 Orchestrator 스크립팅 엔진을 확장하며, Orchestrator와 플러그인된 기술에서 이벤트를 트리거하는 외부 시스템의 알림 이벤트를 게시합니다. 플러그인은 Orchestrator 클라이언트의 **인벤토리** 탭에서 액세스할 수 있는 JavaScript 개체의 인벤토리를 제공합니다. 각 플러그인에는 인벤토리의 개체에서 실행하여 통합된 제품의 일반적인 사용 사례를 자동화할 수 있는 워크플로 및 작업 패키지가 포함되어 있습니다.

본 장은 다음 항목을 포함합니다.

- [Orchestrator 아키텍처](#)
- [Orchestrator 서버와 함께 설치되는 플러그인](#)
- [Orchestrator API 탐색기 액세스](#)

Orchestrator 아키텍처

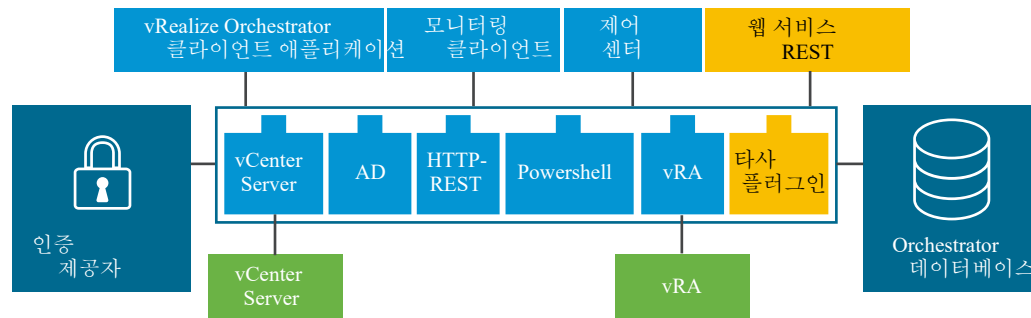
Orchestrator는 사용자가 오케스트레이션 프로세스를 자동화하는 워크플로를 생성하고 실행할 수 있는 워크플로 라이브러리와 워크플로 엔진을 포함하고 있습니다. 사용자는 Orchestrator가 일련의 플러그인을 통해 액세스하는 다양한 기술의 개체에서 워크플로를 실행합니다.

Orchestrator는 vCenter Server 및 vRealize Automation용 플러그인을 포함하는 표준 플러그인 집합을 제공해 사용자가 플러그인이 노출된 다양한 환경에서 작업을 오케스트레이션할 수 있습니다.

또한 Orchestrator는 외부 타사 애플리케이션을 오케스트레이션 플랫폼에 연결하기 위한 개방형 아키텍처를 제공합니다. 사용자는 직접 정의한 플러그인된 기술의 개체를 워크플로에서 실행할 수 있습니다.

Orchestrator는 사용자 계정을 관리하기 위해 인증 제공자에게 접속하며 실행하는 워크플로의 정보를 저장하기 위해 데이터베이스에 접속합니다. Orchestrator 클라이언트 인터페이스 또는 웹 서비스를 통해 Orchestrator, 노출되는 개체 및 Orchestrator 워크플로에 액세스할 수 있습니다. Orchestrator 워크플로 및 서비스의 모니터링과 구성은 모니터링 클라이언트 및 제어 센터를 통해 수행됩니다.

그림 1-1. VMware vRealize Orchestrator 아키텍처



Orchestrator 서버와 함께 설치되는 플러그인

Orchestrator에는 표준 플러그인 모음이 포함되어 있습니다. 각 플러그인은 Orchestrator 플랫폼에 외부 제품 API를 노출합니다. 플러그인은 인벤토리 클래스와 스크립팅 엔진에 대한 추가 개체 유형을 제공하고 외부 시스템의 알림 이벤트를 게시합니다. 또한 각 플러그인은 통합된 외부 제품의 일반적인 사용 사례를 자동화하는 워크플로 라이브러리를 제공합니다.

제어 센터의 **플러그인 관리** 페이지에서 설치된 플러그인 목록을 볼 수 있습니다. 구성이 필요한 플러그인의 경우 인터페이스에 별도의 탭이 있습니다.

표 1-1. Orchestrator와 함께 설치되는 플러그인

| 플러그인 | 용도 | 구성 |
|----------------|---|--|
| vCenter Server | vCenter Server API에 대한 액세스를 제공하여 모든 vCenter Server 개체와 기능을 Orchestrator로 자동화하는 관리 프로세스에 통합할 수 있도록 합니다. | vCenter Server 플러그인 구성 항목을 참조하십시오. |
| 구성 | Orchestrator 인증, 데이터베이스 연결, SSL 인증서 등을 구성하기 위한 워크플로를 제공합니다. | 없음 |
| 라이브러리 | 클라이언트 프로세스의 사용자 지정 및 자동화를 위한 기본적인 빌딩 블록으로 작동하는 워크플로를 제공합니다. 워크플로 라이브러리에는 수명 주기 관리, 프로비저닝, 재해 복구, 핫 백업 및 기타 표준 시스템 관리 프로세스에 대한 템플릿이 포함됩니다. 템플릿을 복사하고 편집하여 필요에 따라 수정할 수 있습니다. | 없음 |

표 1-1. Orchestrator와 함께 설치되는 플러그인 (계속)

| 플러그인 | 용도 | 구성 |
|------------------|--|--|
| SQL | Java 프로그래밍 언어와 광범위한 데이터베이스 간의 데이터베이스 독립적인 연결을 위한 산업 표준인 JDBC(Java Database Connectivity) API를 제공합니다. 데이터베이스에는 SQL 데이터베이스와 스프레드시트 또는 플랫폼 파일과 같은 기타 표 형식 데이터 소스가 포함됩니다. JDBC API는 워크플로에서 SQL 기반 데이터베이스 액세스를 위한 호출 수준 API를 제공합니다. | 없음 |
| SSH | SSH-2(보안 셸 v2) 프로토콜에 대한 구현을 제공합니다. 워크플로에서 암호 및 공용 키 기반 인증을 사용하는 원격 명령 및 파일 전송 세션을 허용합니다. 키보드를 사용한 대화형 인증을 지원합니다. 선택적으로 SSH 플러그인은 Orchestrator 클라이언트 인벤토리에서 직접 원격 파일 시스템 찾아보기를 제공할 수 있습니다. | SSH 플러그인 구성 항목을 참조하십시오. |
| XML | 워크플로에서 구현할 수 있는 완전한 DOM(문서 개체 모델) XML 파서입니다. 또는 Orchestrator JavaScript API에서 E4X(ECMAScript for XML) 구현을 사용할 수도 있습니다. | 없음 |
| 메일 | SMTP(Simple Mail Transfer Protocol)를 사용하여 워크플로에서 이메일을 전송합니다. | 사용할 <code>EmailMessage</code> 개체의 기본값을 설정합니다. 기본 SMTP 연결 정의 항목을 참조하십시오. |
| 네트워크 | Jakarta Apache Commons Net 라이브러리를 사용합니다. Telnet, FTP, POP3 및 IMAP 프로토콜 구현을 제공합니다. POP3와 IMAP 프로토콜은 이메일을 읽는 데 사용됩니다. 네트워크 플러그인은 메일 플러그인과 함께 워크플로에서 전체 이메일 전송 및 수신 기능을 제공합니다. | 없음 |
| 워크플로 설명서 | 워크플로 또는 워크플로 범주에 대해 PDF 형식으로 정보를 생성하는 데 사용할 수 있는 워크플로를 제공합니다. | 없음 |
| 열거형 | 다른 플러그인이 워크플로에서 사용할 수 있는 공통의 열거된 유형을 제공합니다. | 없음 |
| HTTP-REST | vRealize Orchestrator 및 REST 호스트 간 상호 작용을 통해 REST 웹 서비스를 관리할 수 있습니다. | HTTP-REST 플러그인 구성 항목을 참조하십시오. |
| SOAP | vRealize Orchestrator 및 SOAP 호스트 간 상호 작용을 제공하여 SOAP 웹 서비스를 관리할 수 있습니다. | SOAP 플러그인 구성 항목을 참조하십시오. |
| AMQP | 브로커라고도 하는 AMQP(Advanced Message Queuing Protocol) 서버와 상호 작용할 수 있습니다. | AMQP 플러그인 구성 항목을 참조하십시오. |
| SNMP | vRealize Orchestrator에서 SNMP 지원 시스템 및 디바이스에 연결하여 정보를 수신할 수 있습니다. | 없음 |
| Active Directory | vRealize Orchestrator 및 Microsoft Active Directory 간 상호 작용을 제공합니다. | Active Directory 플러그인 구성 항목을 참조하십시오. |
| 동적 유형 | 동적 유형을 정의하고 이러한 동적 유형을 생성 및 사용할 수 있습니다. | 장 19 동적 유형 플러그인 사용 항목을 참조하십시오. |

표 1-1. Orchestrator와 함께 설치되는 플러그인 (계속)

| 플러그인 | 용도 | 구성 |
|------------|---|---|
| 다중 노드 | 계층 관리, Orchestrator 인스턴스 관리, Orchestrator 작업의 확장을 위한 워크플로를 포함합니다. | 장 21 Multi-Node 플러그인 사용 항목을 참조하십시오. |
| PowerShell | PowerShell 호스트를 관리하고 사용자 지정 PowerShell 작업을 실행할 수 있습니다. | 장 20 PowerShell 플러그인 사용 항목을 참조하십시오. |

플러그인 구성 요소

각 플러그인은 DAR 파일 패키지입니다. DAR 파일은 Orchestrator Appliance의 `/var/lib/vco/app-server/plugins`에 저장됩니다. 워크플로 범주 및 API 모듈과 같은 각 플러그인의 구성 요소는 서로 다른 명명 규칙을 사용합니다.

표 1-2. 플러그인 구성 요소 이름

| 구성 UI의 플러그인 이름 | DAR 파일 | 워크플로 범주 | API 모듈 |
|----------------|----------------------------------|-------------------------------|--------------|
| vCenter Server | o11nplugin-vsphere.dar | vCenter | VC |
| vRO 구성 | o11nplugin-configurator.dar | 구성 | Configurator |
| 라이브러리 | o11nplugin-library.dar | 잠금 Orchestrator 문제 해결 | 해당 없음 |
| SQL | o11nplugin-database.dar | JDBC SQL | SQL |
| SSH | o11nplugin-ssh.dar | SSH | SSH |
| XML | o11nplugin-xml.dar | XML | XML |
| 메일 | o11nplugin-mail.dar | 메일 | Mail |
| 네트워크 | o11nplugin-jakartacommonsnet.dar | 없음 | Net |
| 워크플로 설명서 | o11nplugin-wfdocs.dar | 워크플로 설명서 | 워크플로 설명서 |
| 일반 열거 유형 | o11nplugin-enums.dar | 없음 | Enums |
| 동적 유형 | o11n-plugin-dynamictypes.dar | 구성 | DynamicTypes |
| HTTP-REST | o11nplugin-rest.dar | 구성 | REST |
| SOAP | o11n-plugin-soap.dar | 구성 | SOAP |
| AMQP | o11n-plugin-amqp.dar | 구성 | AMQP |
| SNMP | o11n-plugin-snmp.dar | 디바이스 관리 쿼리 관리 트랩 호스트 관리 | SNMP |

표 1-2. 플러그인 구성 요소 이름 (계속)

| 구성 UI의 플러그인 이름 | DAR 파일 | 워크플로 범주 | API 모듈 |
|------------------|---------------------------|--|------------|
| Active Directory | o11nplugin-ad.dar | 컴퓨터 구성 조직 구성 단위 사용자 사용자 그룹 | AD |
| Orchestrator | o11nplugin-multi-node.dar | 서버 구성 원격 실행 원격 관리 작업 워크플로 | VCO |
| PowerShell | o11nplugin-powershell.dar | 구성 생성 템플릿 | PowerShell |

Orchestrator API 탐색기 액세스

Orchestrator에서는 Orchestrator API를 검색하고 스크립팅된 요소에서 사용 가능한 JavaScript 개체에 대한 설명서를 보는 데 사용할 수 있는 API 탐색기를 제공합니다.

Orchestrator 설명서 홈 페이지에서 vCenter Server 플러그인에 대한 스크립팅 API의 온라인 버전을 참조할 수 있습니다.

절차

- 1 Orchestrator 클라이언트에 로그인합니다.
- 2 도구 > API 탐색기를 선택합니다.

결과

API 탐색기가 표시됩니다. 이를 사용하여 Orchestrator API의 모든 개체 및 기능을 검색할 수 있습니다.

다음에 수행할 작업

API 탐색기를 사용하여 스크립팅 가능한 요소에 대한 스크립트를 작성합니다.

Orchestrator 플러그인 구성

2

기본 Orchestrator 플러그인은 워크플로를 통해서만 구성됩니다.

기본 Orchestrator 플러그인을 구성하려면 Orchestrator 클라이언트에서 특정 워크플로를 사용해야 합니다.

본 장은 다음 항목을 포함합니다.

- Orchestrator 플러그인 관리
- 플러그인 제거

Orchestrator 플러그인 관리

제어 센터의 **플러그인 관리** 페이지에서 Orchestrator에 설치된 모든 플러그인 목록을 확인하고 기본 관리 작업을 수행할 수 있습니다.

플러그인 로깅 수준 변경

Orchestrator의 로깅 수준을 변경하는 대신 특정 플러그인에 대해서만 로깅 수준을 변경할 수 있습니다.

새 플러그인 설치

Orchestrator 플러그인을 사용하여 Orchestrator 서버를 다른 소프트웨어 제품과 통합할 수 있습니다. Orchestrator Appliance에는 사전 설치된 플러그인 집합이 포함되어 있으며, 사용자 지정 플러그인을 설치할 수도 있습니다.

모든 Orchestrator 플러그인은 제어 센터에서 설치됩니다. 사용할 수 있는 파일 확장명은 **.vmoapp** 및 **.dar**입니다. **.vmoapp** 파일은 여러 **.dar** 파일의 컬렉션을 포함할 수 있고 애플리케이션으로 설치될 수 있는 반면, **.dar** 파일은 하나의 플러그인과 연결된 모든 리소스를 포함합니다.

플러그인 사용 안 함

플러그인 이름 옆에 있는 **사용** 확인란을 선택 취소하여 플러그인을 사용하지 않도록 설정할 수 있습니다.

이 작업은 플러그인 파일을 제거하지 않습니다. Orchestrator에서 플러그인 설치 제거에 대한 자세한 내용은 **플러그인 제거**를 참조하십시오.

플러그인 제거

제어 센터를 사용하여 플러그인을 사용하지 않도록 설정할 수 있습니다. 이 작업을 수행해도 Orchestrator Appliance 파일 시스템에서 플러그인 파일이 제거되지는 않습니다. 플러그인 파일을 제거하려면 Orchestrator Appliance에 로그인하여 플러그인 파일을 수동으로 제거해야 합니다.

절차

1 Orchestrator Appliance에서 플러그인을 삭제합니다.

- a SSH를 통해 Orchestrator Appliance에 **루트**로 로그인합니다.
- b 텍스트 편집기를 사용하여 `/etc/vco/app-server/plugins/_VSOPuginInstallationVersion.xml` 파일을 엽니다.
- c 제거할 플러그인에 해당하는 코드 행을 삭제합니다.
- d `/var/lib/vco/app-server/plugins` 디렉토리로 이동합니다.
- e 제거할 플러그인이 포함된 `.dar` 압축 파일을 삭제합니다.

2 vRealize Orchestrator 서비스를 다시 시작합니다.

```
service vco-configurator restart && service vco-server restart
```

3 제어 센터에 **root**로 로그인합니다.

4 **플러그인 관리** 페이지에서 플러그인이 제거되었는지 확인합니다.

5 Orchestrator 클라이언트를 통해 플러그인과 관련된 패키지 및 폴더를 삭제합니다.

- a Orchestrator 클라이언트에 로그인합니다.
- b 왼쪽 상단의 드롭다운 메뉴에서 **디자인**을 선택합니다.
- c **패키지** 보기를 클릭합니다.
- d 삭제할 패키지를 마우스 오른쪽 버튼으로 클릭하고 **컨텐츠와 함께 요소 삭제**를 선택합니다.

참고 읽기 전용 상태로 잠겨 있는 Orchestrator 요소(예: 표준 라이브러리의 워크플로)는 삭제되지 않습니다.

- e 오른쪽 상단의 **도구** 메뉴에서 **사용자 기본 설정**을 선택합니다.

기본 설정 컨텍스트 메뉴가 열립니다.

- f **일반** 페이지에서 **비어 있지 않은 폴더 삭제 허용됨** 확인란을 선택합니다.

이제 한번의 클릭으로 하위 폴더 및 워크플로를 포함하여 전체 폴더를 삭제할 수 있습니다.

- g **워크플로** 보기를 클릭합니다.

- h 제거할 플러그인의 폴더를 삭제합니다.

i **작업** 보기를 클릭합니다.

j 제거할 플러그인의 작업 모듈을 삭제합니다.

6 vRealize Orchestrator 서비스를 다시 시작합니다.

결과

플러그인과 관련된 모든 사용자 지정 워크플로, 작업, 정책, 구성, 설정 및 리소스를 제거했습니다.

vCenter Server 플러그인 사용

3

vCenter Server 플러그인을 사용하여 여러 vCenter Server 인스턴스를 관리할 수 있습니다. vCenter Server 플러그인 API를 사용하는 워크플로를 생성하여 vCenter Server 환경에서 작업을 자동화할 수 있습니다.

vCenter Server 플러그인은 워크플로에서 사용할 수 있는 JavaScript에 vCenter Server API를 매핑합니다. 또한 워크플로에 포함할 수 있는 개별 vCenter Server 작업을 수행하는 작업을 제공합니다.

vCenter Server 플러그인은 vCenter Server 작업을 자동화하는 표준 워크플로 라이브러리를 제공합니다. 예를 들어 가상 시스템을 생성, 복제, 마이그레이션 또는 삭제하는 워크플로를 실행할 수 있습니다.

vCenter Server 플러그인은 PBM(정책 기반 관리) 및 SMS(스토리지 모니터링 서비스) API를 Orchestrator 스크립팅 API에 있는 스크립팅 개체로 포함합니다. 스토리지 정책 기반 관리 정책 및 구성 요소가 Orchestrator **인벤토리** 탭에 나타납니다.

본 장은 다음 항목을 포함합니다.

- vCenter Server 플러그인 구성
- vCenter Server 플러그인 스크립팅 API
- vCenter Server 플러그인 인벤토리 사용
- 쿼리를 위한 성능 고려 사항
- vCenter Server 플러그인으로 XPath 표현식 사용
- vCenter Server 플러그인 워크플로 라이브러리 액세스
- vCenter Server 플러그인 워크플로 라이브러리

vCenter Server 플러그인 구성

Orchestrator를 사용하여 vSphere 인벤토리에서 개체를 관리하고 개체에 대한 워크플로를 실행하려면 먼저 vCenter Server 플러그인을 구성하고 Orchestrator와 오케스트레이션할 vCenter Server 인스턴스 간의 연결 매개 변수를 정의해야 합니다.

Orchestrator 클라이언트에서 vCenter Server 구성 워크플로를 실행하여 vCenter Server 플러그인을 구성할 수 있습니다.

vSphere Web Client를 사용하여 vSphere 인벤토리에서 개체를 관리하려면 vCenter Server와 vSphere Web Client 모두가 가리키는 동일한 vCenter Single Sign-On 인스턴스에서 작동하도록 Orchestrator 서버를 구성해야 합니다. Orchestrator가 vCenter Server 확장으로 등록되어 있는지도 확인해야 합니다. 사용자 이름 및 암호를 입력하여 vCenter Server 확장을 관리할 수 있는 권한을 가진 사용자를 지정할 때 Orchestrator를 vCenter Server 확장으로 등록합니다.

구성 워크플로

vCenter Server 플러그인의 구성 워크플로 범주에는 vCenter Server 인스턴스의 연결을 관리할 수 있는 워크플로가 포함되어 있습니다.

Orchestrator 클라이언트의 **워크플로** 보기에서 **라이브러리 > vCenter > 구성**을 통해 이러한 워크플로에 액세스할 수 있습니다.

| 워크플로 이름 | 설명 |
|---|---|
| vCenter Server 인스턴스 추가 | Orchestrator를 구성하여 새 vCenter Server 인스턴스에 연결하면 vSphere 인프라에서 개체를 통해 워크플로를 실행할 수 있습니다. |
| vCenter Server의 vRealize Orchestrator 확장 목록 | vCenter Server의 모든 vRealize Orchestrator 확장을 나열합니다. |
| Orchestrator를 vCenter Server 확장으로 등록 | Orchestrator 인스턴스를 vCenter Server 확장으로 등록합니다. |
| vCenter Server 인스턴스 제거 | Orchestrator 인벤토리에서 vCenter Server 인스턴스를 제거합니다. 더 이상 이 vCenter Server 인스턴스를 오케스트레이션할 수 없습니다. |
| vCenter Server 인스턴스 업데이트 | vCenter Server 인스턴스의 연결을 업데이트합니다. 예를 들어, vCenter Server 시스템의 IP 주소가 변경되면 vCenter Server 인스턴스의 연결 매개 변수를 업데이트해야 Orchestrator를 사용하여 vSphere 인벤토리를 관리할 수 있습니다. |
| vCenter Server 확장 등록 취소 | vSphere Web Client 확장을 등록 취소합니다. |

vCenter Server 인스턴스에 대한 연결 구성

Orchestrator 클라이언트에서 vCenter Server 구성 워크플로를 실행하여 vCenter Server 인스턴스에 대한 연결을 구성할 수 있습니다.

절차

- 1 Orchestrator 클라이언트에 관리자로 로그인합니다.
- 2 Orchestrator 클라이언트에서 **워크플로** 보기를 클릭합니다.
- 3 계층형 워크플로 목록에서 **라이브러리 > vCenter > 구성**을 확장하고 **vCenter Server 인스턴스 추가** 워크플로로 이동합니다.
- 4 **vCenter 서버 인스턴스 추가** 워크플로를 마우스 오른쪽 버튼으로 클릭하고 **워크플로 시작**을 선택합니다.
- 5 추가하려는 vCenter Server 인스턴스가 설치되는 시스템의 IP 주소 또는 DNS 이름을 입력합니다.

참고 입력한 호스트 이름은 대소문자를 구별합니다.

- 6 기본 포트 값인 **443**을 유지합니다.
- 7 SDK의 기본 위치를 유지하여 vCenter Server 인스턴스에 연결합니다.
- 8 Orchestrator를 통해 vCenter Server 인스턴스 관리 여부를 선택하고 **다음**을 클릭합니다.
- 9 추가하려는 vCenter Server 인스턴스에 대한 인증서 경고 무시 여부를 선택합니다.

인증서 경고를 무시하도록 선택하면 vCenter Server 인스턴스 인증서가 자동으로 수락되고 신뢰할 수 있는 저장소에 추가됩니다.

- 10 사용하려는 메서드를 선택하여 vCenter Server 시스템에서 사용자 액세스를 관리하는 데 사용할 방법을 선택합니다.

| 옵션 | 설명 |
|----------|---|
| 고유 세션 공유 | Orchestrator가 vCenter Server에 대해 한 개 연결만 생성하도록 허용합니다. 사용자 이름 및 암호 텍스트 상자에 사용할 Orchestrator 자격 증명을 입력하고 vCenter Server 호스트에 대한 연결을 구성합니다. 선택하는 사용자는 vCenter Server 확장을 관리할 수 있는 권한과 일련의 사용자 지정된 권한을 가진 유효한 사용자여야 합니다. Orchestrator는 이러한 자격 증명을 사용하여 VirtualCenter Web 서비스를 모니터링하고, 일반적으로 Orchestrator 시스템 워크플로를 실행합니다. |
| 사용자별 세션 | vCenter Server에 대한 새 세션을 생성합니다. 이 작업 수행 시 CPU, 메모리 및 대역폭 사용량이 급격히 증가할 수 있습니다. vCenter Server가 Active Directory 도메인에 있거나 vCenter Server Single Sign-On이 사용되도록 설정된 경우에만 이 옵션을 선택합니다. 선택한 사용자는 vCenter Server 확장을 관리할 수 있는 권한을 갖는 유효한 사용자여야 합니다. |

선택한 사용자 계정은 통계 및 기타 데이터를 수집하는 정책 엔진에서도 사용됩니다. 선택한 사용자에게 유효한 권한이 없는 경우 정책 엔진은 vCenter Server 인벤토리의 필요한 부분에 액세스할 수 없고 필요한 데이터를 수집할 수 없습니다.

- 11 (선택 사항) 사용자 도메인을 입력합니다.

공유 세션을 사용하려고 선택한 경우에는 사용자 도메인 이름을 지정해야 합니다.

참고 사용자별 세션이 선택된 경우 이 텍스트 상자를 채웁니다.

- 12 (선택 사항) vSphere 스토리지 관리 끝점의 URL을 입력합니다.

PBM(정책 기반 관리) 끝점, SMS(스토리지 모니터링 서비스) 끝점 또는 두 가지 모두를 구성할 수 있습니다.

- 13 **제출**을 클릭하여 워크플로를 실행합니다.

결과

워크플로가 성공적으로 실행되면 **인벤토리** 보기에 vCenter Server 인스턴스와 해당 인스턴스에 속한 모든 vSphere 개체가 표시됩니다.

vCenter Server 플러그인 스크립팅 API

vCenter Server 스크립팅 API는 vRealize Orchestrator와 vCenter Server 간에 상호 작용을 허용하는 해당 속성, 메서드 및 생성자가 있는 클래스를 포함합니다. API를 사용하여 사용자 지정 워크플로를 개발할 수 있습니다.

사용 가능한 API 개체 목록은 <https://www.vmware.com/support/orchestrator/doc/vro-vsphere65-api/index.html>을 참조하십시오.

vCenter Server 플러그인 인벤토리 사용

vCenter Server 플러그인은 연결된 vCenter Server 인스턴스의 모든 개체를 인벤토리 보기에 노출합니다. **인벤토리** 탭을 사용하여 권한 부여 요소를 추가하거나 vCenter Server 개체에서 워크플로를 실행할 수 있습니다.

사용자 기본 설정 도구의 **인벤토리** 탭에서 **인벤토리에서 컨텍스트 메뉴 사용** 옵션을 사용하도록 설정한 경우 선택한 인벤토리 개체에서 실행할 수 있는 모든 워크플로가 팝업 메뉴에 표시됩니다.

쿼리를 위한 성능 고려 사항

vCenter Server 플러그인(vRealize Orchestrator용)을 사용하여 특정 개체에 대한 vCenter Server 인벤토리를 쿼리할 수 있습니다.

쿼리 메서드

쿼리를 수행하려면 vCSearchIndex 관리 개체 또는 플러그인 인벤토리에 포함되어 있는 개체 찾기 메서드 (예: getAllDatastores(), getAllVirtualMachines(), findAllForType() 및 기타) 중 하나를 사용할 수 있습니다.

성능

검색 쿼리에서 메서드 매개 변수에 대한 인수로 속성 집합을 지정하지 않은 경우, 기본적으로 두 가지 메서드 모두 속성을 포함하지 않고 쿼리된 개체를 반환합니다.

참고 Orchestrator 서버의 전반적인 성능에 영향을 줄 수 있으므로 Orchestrator 클라이언트가 반환된 개체의 대규모 집합을 필터링하는 것을 막기 위해 항상 `getAll...()` 및 `findAll...()` 찾기 개체가 포함된 쿼리 표현식을 사용해야 합니다.

vCenter Server 인벤토리 쿼리를 위해 두 가지 유형의 표현식을 사용할 수 있습니다.

| 표현식 유형 | 설명 |
|-----------|--|
| 이름 표현식 | 쿼리 매개 변수의 인수로 이름을 지정할 수 있습니다. 참고 vCenter Server 플러그인 인벤토리에 나타난 플러그인 개체의 이름에 따라 지정된 이름 인수를 사용하여 개체가 필터링됩니다. |
| XPath 표현식 | Xpath 쿼리 언어를 기반으로 표현식을 사용할 수 있습니다. 자세한 내용은 vCenter Server 플러그인으로 XPath 표현식 사용 항목을 참조하십시오. |

사용자 지정 속성을 지닌 vCenter Server 인벤토리 개체를 호출할 경우, 워크플로 또는 작업에서 이 개체에 대한 각 참조에서 vCenter Server에 쿼리를 전송하며 그 결과 뚜렷한 성능 오버헤드가 발생합니다. 성능을 최적화하고 워크플로 실행 시 개체를 여러 번 직렬화 및 직렬화 해제하는 것을 방지하려면, 워크플로 특성, 입력 또는 출력 매개 변수로 개체를 저장하는 대신 공유 리소스를 사용하여 개체를 저장하는 것이 가장 좋습니다. 공유 리소스는 구성 요소 또는 리소스 요소가 될 수 있습니다.

vCenter Server 플러그인으로 XPath 표현식 사용

vCenter Server 플러그인에서 찾기 메서드를 사용하여 vCenter Server 인벤토리 개체에 대해 쿼리할 수 있습니다. XPath 표현식을 사용하여 검색 매개 변수를 정의할 수 있습니다.

vCenter Server 플러그인에는 `getAllDatastores()`, `getAllResourcePools()`, `findAllForType()`과 같은 개체 찾기 메서드 집합이 포함되어 있습니다. 이러한 메서드를 사용하여 Orchestrator 서버에 연결된 vCenter Server 인스턴스의 인벤토리에 액세스하고 ID, 이름 또는 기타 속성별로 개체를 검색할 수 있습니다.

성능상의 이유로 찾기 메서드는 검색 쿼리에서 속성 집합을 지정하지 않는 한, 쿼리된 개체의 속성을 반환하지 않습니다.

Orchestrator 설명서 홈 페이지에서 vCenter Server 플러그인에 대한 스크립팅 API의 온라인 버전을 참조할 수 있습니다.

중요 XPath 표현식을 기반으로 하는 쿼리는 찾기 메서드가 vCenter Server 측에서 지정된 유형의 모든 개체를 반환하고 쿼리 필터가 vCenter Server 플러그인 측에 적용되기 때문에 Orchestrator 성능에 영향을 줄 수 있습니다.

vCenter Server 플러그인으로 XPath 표현식 사용

찾기 메서드를 호출할 때 XPath 쿼리 언어를 기반으로 하는 표현식을 사용할 수 있습니다. 검색은 XPath 표현식과 일치하는 모든 인벤토리 개체를 반환합니다. 임의의 속성을 쿼리하려면 이러한 속성을 포함하여 문자열 어레이 양식으로 스크립트를 검색할 수 있습니다.

다음 JavaScript 예제는 VcPlugin 스크립팅 개체 및 XPath 표현식을 사용하여 vCenter Server 관리 개체의 일부이고 해당 개체의 이름에 **ds** 문자열이 포함된 모든 데이터스토어 개체의 이름을 반환합니다.

```
var datastores = VcPlugin.getAllDatastores(null, "xpath:name[contains(.,'ds')]");
for each (datastore in datastores){
```

```
System.log(datastore.name);
}
```

Server 스크립팅 개체 및 `findAllForType` 찾기 메서드를 사용하여 동일한 XPath 표현식을 호출할 수 있습니다.

```
var datastores = Server.findAllForType("VC:Datastore", "xpath:name[contains(., 'ds')]");
for each (datastore in datastores){
    System.log(datastore.name);
}
```

다음 스크립트 예제는 ID가 숫자 **1**로 시작하는 모든 호스트 시스템 개체의 이름을 반환합니다.

```
var hosts = VcPlugin.getAllHostSystems(null, "xpath:id[starts-with(., '1')]");
for each (host in hosts){
    System.log(host.name);
}
```

다음 스크립트는 개체 이름에 대문자 또는 소문자로 **DC** 문자열을 포함하는 모든 데이터센터 개체의 이름 및 ID를 반환합니다. 스크립트는 **태그** 속성도 검색합니다.

```
var datacenters = VcPlugin.getAllDatacenters(['tag'], "xpath:name[contains(translate(., 'DC', 'dc'), 'dc')]");
for each (datacenter in datacenters){
    System.log(datacenter.name + " " + datacenter.id);
}
```

vCenter Server 플러그인 워크플로 라이브러리 액세스

Orchestrator 클라이언트 또는 vSphere Web Client를 사용하여 vCenter Server 플러그인 워크플로 라이브러리의 요소에 액세스해야 합니다.

사전 요구 사항

- vCenter Server 인스턴스에 대한 연결을 구성합니다.
- 로그인한 사용자 계정에 vCenter Server 워크플로를 실행하는 데 필요한 권한이 있는지 확인합니다.

절차

- 1 Orchestrator 클라이언트에서 왼쪽 상단의 드롭다운 메뉴에서 **설계** 또는 **실행**을 선택합니다.
- 2 Orchestrator 클라이언트 왼쪽 창에서 **워크플로** 보기를 클릭합니다.
- 3 계층형 목록을 **라이브러리 > vCenter**로 확장합니다.

다음에 수행할 작업

워크플로 라이브러리를 검토합니다.

vCenter Server 플러그인 워크플로 라이브러리

vCenter Server 플러그인 워크플로 라이브러리에는 vCenter Server의 관리와 관련된 자동화된 프로세스를 실행하는 데 사용할 수 있는 워크플로가 포함됩니다.

■ 일괄 처리 워크플로

일괄 처리 워크플로는 구성 요소를 채우고 선택된 vCenter Server 개체에서 워크플로를 실행합니다.

■ 클러스터 및 계산 리소스 워크플로

클러스터 및 계산 리소스 워크플로를 사용하여 클러스터를 생성하거나 이름을 변경하거나 또는 삭제할 수 있습니다. 또한, 클러스터에서 고가용성, 분산 리소스 스케줄러 및 vCloud Distributed Storage를 사용하거나 사용하지 않을 수 있습니다.

■ 구성 워크플로

vCenter Server 플러그인의 구성 워크플로 범주에는 vCenter Server 인스턴스의 연결을 관리할 수 있는 워크플로가 포함되어 있습니다.

■ 사용자 지정 특성 워크플로

사용자 지정 특성 워크플로를 사용하여 가상 시스템에 사용자 지정 특성을 추가하거나 가상 시스템의 사용자 지정 특성을 가져올 수 있습니다.

■ 데이터 센터 워크플로

데이터 센터 워크플로를 사용하여 데이터 센터를 생성, 삭제, 다시 로드, 이름 변경 또는 다시 검색할 수 있습니다.

■ 데이터스토어 및 파일 워크플로

데이터스토어 및 파일 워크플로를 사용하여 파일 목록 삭제, 데이터스토어에서 사용되지 않는 파일 찾기 등을 수행할 수 있습니다.

■ 데이터 센터 폴더 관리 워크플로

데이터 센터 폴더 관리 워크플로를 사용하여 데이터 센터 폴더를 생성 또는 삭제하거나 이름을 변경할 수 있습니다.

■ 호스트 폴더 관리 워크플로

호스트 폴더 관리 워크플로를 사용하여 호스트 폴더를 생성 또는 삭제하거나 이름을 변경할 수 있습니다.

■ 가상 시스템 폴더 관리 워크플로

가상 시스템 폴더 관리 워크플로를 사용하여 가상 시스템 폴더를 생성 또는 삭제하거나 이름을 변경할 수 있습니다.

■ 게스트 작업 파일 워크플로

게스트 작업 파일 워크플로를 사용하면 게스트 운영 체제에서 파일을 관리할 수 있습니다.

■ 게스트 작업 프로세스 워크플로

게스트 작업 프로세스 워크플로를 사용하면 게스트 운영 체제에서 실행 중인 프로세스를 제어하고 이에 대한 정보를 가져올 수 있습니다.

■ 전원 호스트 관리 워크플로

호스트 전원 관리 워크플로를 사용하여 호스트를 재부팅하거나 종료할 수 있습니다.

■ 기본 호스트 관리 워크플로

기본 호스트 관리 워크플로를 사용하여 호스트를 유지 보수 모드로 설정하고 호스트의 유지 보수 모드를 종료할 수 있습니다. 또한, 호스트를 폴더 또는 클러스터로 이동할 수 있고, 호스트에서 데이터를 다시 로드할 수 있습니다.

■ 호스트 등록 관리 워크플로

호스트 등록 관리 워크플로를 사용하여 클러스터에 호스트 추가, 클러스터에서 호스트 연결 끊기 및 다시 연결 등을 수행할 수 있습니다.

■ 네트워킹 워크플로

네트워킹 워크플로를 사용하여 분산 가상 스위치에 포트 그룹 추가, 포트 그룹으로 분산 가상 스위치 생성 등을 수행할 수 있습니다.

■ 분산 가상 포트 그룹 워크플로

분산 가상 포트 그룹 워크플로를 사용하여 포트 그룹을 업데이트하거나 삭제하고 포트 그룹을 재구성할 수 있습니다.

■ 분산 가상 스위치 워크플로

분산 가상 스위치 워크플로를 사용하여 분산 가상 스위치를 생성, 업데이트 또는 삭제하고 전용 VLAN을 생성, 삭제 또는 업데이트할 수 있습니다.

■ 표준 가상 스위치 워크플로

표준 가상 스위치 워크플로를 사용하여 표준 가상 스위치를 생성, 업데이트 또는 삭제하거나 표준 가상 스위치의 포트 그룹을 생성, 삭제 또는 업데이트할 수 있습니다.

■ 네트워킹 Virtual SAN 워크플로

Virtual SAN 워크플로를 사용하여 Virtual SAN 네트워크 트래픽을 구성할 수 있습니다.

■ 리소스 풀 워크플로

리소스 풀 워크플로를 사용하여 리소스 풀을 생성, 이름 변경, 재구성 또는 삭제하고 리소스 풀 정보를 가져올 수 있습니다.

■ 스토리지 워크플로

스토리지 워크플로를 사용하여 스토리지 관련 작업을 수행할 수 있습니다.

■ Storage DRS 워크플로

Storage DRS 워크플로를 사용하여 데이터스토어 클러스터 생성 및 구성, 클러스터에서 데이터스토어 제거, 클러스터에 스토리지 추가 등과 같은 스토리지 관련 작업을 수행합니다.

■ 스토리지 VSAN 워크플로

Virtual SAN 워크플로를 사용하여 Virtual SAN 클러스터에서 비SSD 디스크 및 디스크 그룹을 관리할 수 있습니다.

■ 기본 가상 시스템 관리 워크플로

기본 가상 시스템 관리 워크플로를 사용하여 가상 시스템 생성, 이름 변경 또는 삭제, 가상 하드웨어 업그레이드 등과 같은 가상 시스템에 대한 기본 작업을 수행할 수 있습니다.

■ 복제 워크플로

복제 워크플로를 사용하여 가상 시스템 속성 사용자 지정 여부와 관계없이 가상 시스템을 복제할 수 있습니다.

■ 연결된 복제 워크플로

연결된 복제 워크플로를 사용하여 연결된 복제에서 가상 시스템 복원, 연결된 복제 생성 등과 같은 연결된 복제 작업을 수행할 수 있습니다.

■ Linux 사용자 지정 복제 워크플로

Linux 사용자 지정 워크플로를 사용하여 Linux 가상 시스템을 복제하고 게스트 운영 체제를 사용자 지정할 수 있습니다.

■ 도구 복제 워크플로

도구 복제 워크플로를 사용하여 가상 시스템의 운영 체제에 대한 사용자 지정 정보, 가상 디바이스 업데이트에 필요한 정보 등을 가져올 수 있습니다.

■ Windows 사용자 지정 복제 워크플로

Windows 사용자 지정 복제 워크플로를 사용하여 Windows 가상 시스템을 복제하고 게스트 운영 체제를 사용자 지정할 수 있습니다.

■ 디바이스 관리 워크플로

디바이스 관리 워크플로를 사용하여 가상 시스템이나 호스트 데이터스토어에 연결된 디바이스를 관리할 수 있습니다.

■ 이동 및 마이그레이션 워크플로

이동 및 마이그레이션 워크플로를 사용하여 가상 시스템을 마이그레이션할 수 있습니다.

■ 기타 워크플로

기타 범주의 워크플로를 사용하여 FT(Fault Tolerance)를 사용 또는 사용 안 함으로 설정하고, 가상 시스템 정보를 추출하고, 연결이 끊어진 가상 시스템을 찾을 수 있습니다.

■ 전원 관리 워크플로

전원 관리 워크플로를 사용하여 가상 시스템 전원 켜기 및 끄기, 가상 시스템의 게스트 운영 체제 재부팅, 가상 시스템 일시 중단 등을 수행할 수 있습니다.

■ 스냅샷 워크플로

스냅샷 워크플로를 사용하여 스냅샷 관련 작업을 수행할 수 있습니다.

■ VMware Tools 워크플로

VMware Tools 워크플로를 사용하여 가상 시스템에서 VMware Tools 관련 작업을 수행합니다.

일괄 처리 워크플로

일괄 처리 워크플로는 구성 요소를 채우고 선택된 vCenter Server 개체에서 워크플로를 실행합니다.

Orchestrator 클라이언트의 **워크플로** 보기에서 **라이브러리 > vCenter > 일괄 처리**를 통해 일괄 처리 워크플로에 액세스할 수 있습니다.

| 워크플로 이름 | 설명 |
|------------------|--|
| 일괄 처리 구성 요소 채우기 | <p>선택한 개체 워크플로에서 워크플로 실행이 사용하는 구성 요소를 채웁니다. 다음 작업을 수행합니다.</p> <ul style="list-style-type: none"> ■ BatchObject 및 BatchAction 구성 요소를 재설정합니다. ■ 입력 매개 변수가 한 개인 모든 워크플로를 사용하여 BatchObject 구성 요소를 채웁니다. ■ 입력 매개 변수가 없거나 입력 매개 변수가 한 개 있고 어레이가 returnType인 모든 작업으로 BatchAction 구성 요소를 채웁니다. |
| 선택한 개체에서 워크플로 실행 | <p>선택한 vCenter Server 개체에서 워크플로를 실행하여 한 개 작업을 입력으로 가져옵니다. 이 작업은 워크플로를 실행하는 개체의 목록을 검색합니다. 선택한 워크플로를 실행하지 않고 개체를 반환하려면 시뮬레이션 모드에서 워크플로를 실행합니다.</p> |

클러스터 및 계산 리소스 워크플로

클러스터 및 계산 리소스 워크플로를 사용하여 클러스터를 생성하거나 이름을 변경하거나 또는 삭제할 수 있습니다. 또한, 클러스터에서 고가용성, 분산 리소스 스케줄러 및 vCloud Distributed Storage를 사용하거나 사용하지 않을 수 있습니다.

Orchestrator 클라이언트의 **워크플로** 보기에서 **라이브러리 > vCenter > 클러스터 및 계산 리소스**를 통해 클러스터 및 계산 리소스 워크플로에 액세스할 수 있습니다.

| 워크플로 이름 | 설명 |
|--|--|
| 클러스터에 DRS 가상 시스템 그룹 추가 | 클러스터에 DRS 가상 시스템 그룹을 추가합니다. |
| DRS 그룹에 가상 시스템 추가 | 기존 DRS 가상 시스템 그룹에 가상 시스템 목록을 추가합니다. |
| 클러스터 생성 | 호스트 폴더에서 클러스터를 생성합니다. |
| 클러스터 삭제 | 클러스터를 삭제합니다. |
| 클러스터에서 DRS 사용 안 함 | 클러스터에서 DRS를 사용하지 않도록 설정합니다. |
| 클러스터에서 HA 사용 안 함 | 클러스터에서 고가용성을 사용하지 않도록 설정합니다. |
| 클러스터에서 vCloud Distributed Storage 사용 안 함 | 클러스터에서 vCloud Distributed Storage를 사용하지 않도록 설정합니다. |
| 클러스터에서 DRS 사용 | 클러스터에서 DRS를 사용하도록 설정합니다. |
| 클러스터에서 HA 사용 | 클러스터에서 고가용성을 사용하도록 설정합니다. |
| 클러스터에서 vCloud Distributed Storage 사용 | 클러스터에서 vCloud Distributed Storage를 사용하도록 설정합니다. |
| 클러스터에서 가상 시스템 DRS 그룹 제거 | 클러스터에서 DRS 가상 시스템 그룹을 제거합니다. |

| 워크플로 이름 | 설명 |
|--------------------|------------------------------|
| DRS 그룹에서 가상 시스템 제거 | 클러스터 DRS 그룹에서 가상 시스템을 제거합니다. |
| 클러스터 이름 변경 | 클러스터의 이름을 변경합니다. |

구성 워크플로

vCenter Server 플러그인의 구성 워크플로 범주에는 vCenter Server 인스턴스의 연결을 관리할 수 있는 워크플로가 포함되어 있습니다.

Orchestrator 클라이언트의 **워크플로** 보기에서 **라이브러리 > vCenter > 구성**을 통해 이러한 워크플로에 액세스할 수 있습니다.

| 워크플로 이름 | 설명 |
|--------------------------------------|---|
| vCenter Server 인스턴스 추가 | Orchestrator를 구성하여 새 vCenter Server 인스턴스에 연결하면 vSphere 인프라에서 개체를 통해 워크플로를 실행할 수 있습니다. |
| vCenter Server의 Orchestrator 확장 목록 | vCenter Server의 모든 Orchestrator 확장을 나열합니다. |
| Orchestrator를 vCenter Server 확장으로 등록 | Orchestrator 인스턴스를 vCenter Server 확장으로 등록합니다. |
| vCenter Server 인스턴스 제거 | Orchestrator 인벤토리에서 vCenter Server 인스턴스를 제거합니다. 이 vCenter Server 인스턴스를 더 이상 오케스트레이션할 수 없습니다. |
| vCenter Server 인스턴스 업데이트 | vCenter Server 인스턴스의 연결을 업데이트합니다. 예를 들어, vCenter Server 시스템의 IP 주소가 변경되면 vCenter Server 인스턴스의 연결 매개 변수를 업데이트해야 Orchestrator를 사용하여 vSphere 인벤토리를 관리할 수 있습니다. |
| vCenter Server 확장 등록 취소 | vCenter Server 확장을 등록 취소합니다. |

사용자 지정 특성 워크플로

사용자 지정 특성 워크플로를 사용하여 가상 시스템에 사용자 지정 특성을 추가하거나 가상 시스템의 사용자 지정 특성을 가져올 수 있습니다.

Orchestrator 클라이언트의 **워크플로** 보기에서 **라이브러리 > vCenter > 사용자 지정 특성**을 통해 사용자 지정 특성에 액세스할 수 있습니다.

| 워크플로 이름 | 설명 |
|-------------------------|--|
| 가상 시스템에 사용자 지정 특성 추가 | 가상 시스템에 사용자 지정 특성을 추가합니다. |
| 여러 가상 시스템에 사용자 지정 특성 추가 | 선택한 가상 시스템에 사용자 지정 특성을 추가합니다. |
| 사용자 지정 특성 가져오기 | vCenter Server에서 가상 시스템의 사용자 지정 특성을 가져옵니다. |

데이터 센터 워크플로

데이터 센터 워크플로를 사용하여 데이터 센터를 생성, 삭제, 다시 로드, 이름 변경 또는 다시 검색할 수 있습니다.

Orchestrator 클라이언트의 **워크플로** 보기에서 **라이브러리 > vCenter > 데이터 센터**를 통해 데이터 센터 워크플로에 액세스할 수 있습니다.

| 워크플로 이름 | 설명 |
|------------------|--|
| 데이터 센터 생성 | 데이터 센터 폴더에서 데이터 센터를 생성합니다. |
| 데이터 센터 삭제 | 데이터 센터를 생성합니다. |
| 데이터 센터 다시 로드 | vCenter Server가 강제로 데이터 센터에서 데이터를 다시 로드합니다. |
| 데이터 센터 이름 변경 | 데이터 센터의 이름을 변경하고 작업을 완료할 때까지 대기합니다. |
| 데이터 센터 HBA 다시 검색 | 데이터 센터에서 호스트를 검색하고, 호스트 버스 어댑터를 다시 검색하여 새 스토리지를 파악합니다. |

데이터스토어 및 파일 워크플로

데이터스토어 및 파일 워크플로를 사용하여 파일 목록 삭제, 데이터스토어에서 사용되지 않는 파일 찾기 등을 수행할 수 있습니다.

Orchestrator 클라이언트의 **워크플로** 보기에서 **라이브러리 > vCenter > 데이터스토어 및 파일**을 통해 데이터스토어 및 파일 워크플로에 액세스할 수 있습니다.

| 워크플로 이름 | 설명 |
|-----------------------------------|--|
| 모든 파일 삭제 | 파일 목록을 삭제합니다. |
| 사용되지 않는 모든 데이터스토어 파일 삭제 | vCenter Server 환경에서 모든 데이터스토어를 검색하고 사용되지 않는 모든 파일을 삭제합니다. |
| 사용되지 않는 데이터스토어 파일 내보내기 | 모든 데이터스토어를 검색하고 사용되지 않는 모든 파일을 나열하는 XML 설명자 파일을 생성합니다. |
| 데이터스토어에서 사용되지 않는 파일 찾기 | Orchestrator에 등록된 모든 vCenter Server 인스턴스와 연결되지 않은, 사용하지 않은 디스크(*.vmdk), 가상 시스템(*.vmx) 및 템플릿(*.vmtx) 파일에 대해 vCenter Server 환경을 검색합니다. |
| 가상 시스템에서 모든 구성, 템플릿 및 디스크 파일 가져오기 | 모든 데이터스토어의 모든 가상 시스템 설명자 파일 목록 및 모든 가상 시스템 디스크 파일 목록을 생성합니다. |
| 모든 데이터스토어 파일 기록 | 모든 데이터스토어에서 찾은 모든 가상 시스템 구성 파일 및 모든 가상 시스템 파일에 대한 로그를 생성합니다. |
| 사용되지 않는 데이터스토어 파일 기록 | 가상 시스템에 등록된 사용되지 않는 파일에 대해 vCenter Server 환경을 검색하고 파일 로그를 텍스트 파일로 내보냅니다. |
| 데이터스토어에 파일 업로드 | 파일을 지정된 데이터스토어의 기존 폴더에 업로드합니다. 업로드된 파일은 동일한 대상 폴더에 있는 동일한 이름의 기존 파일을 덮어씁니다. |

데이터 센터 폴더 관리 워크플로

데이터 센터 폴더 관리 워크플로를 사용하여 데이터 센터 폴더를 생성 또는 삭제하거나 이름을 변경할 수 있습니다.

Orchestrator 클라이언트의 **워크플로** 보기에서 **라이브러리 > vCenter > 폴더 관리 > 데이터 센터 폴더**를 통해 데이터 센터 폴더 관리 워크플로에 액세스할 수 있습니다.

| 워크플로 이름 | 설명 |
|-----------------|--|
| 데이터 센터 폴더 생성 | 데이터 센터 폴더를 생성합니다. |
| 데이터 센터 폴더 삭제 | 데이터 센터 폴더를 삭제하고 작업을 완료할 때까지 대기합니다. |
| 데이터 센터 폴더 이름 변경 | 데이터 센터 폴더의 이름을 변경하고 작업을 완료할 때까지 대기합니다. |

호스트 폴더 관리 워크플로

호스트 폴더 관리 워크플로를 사용하여 호스트 폴더를 생성 또는 삭제하거나 이름을 변경할 수 있습니다.

Orchestrator 클라이언트의 **워크플로** 보기에서 **라이브러리 > vCenter > 폴더 관리 > 호스트 폴더**를 통해 호스트 폴더 관리 워크플로에 액세스할 수 있습니다.

| 워크플로 이름 | 설명 |
|--------------|-------------------------------------|
| 호스트 폴더 생성 | 호스트 폴더를 생성합니다. |
| 호스트 폴더 삭제 | 호스트 폴더를 삭제하고 작업을 완료할 때까지 대기합니다. |
| 호스트 폴더 이름 변경 | 호스트 폴더의 이름을 변경하고 작업을 완료할 때까지 대기합니다. |

가상 시스템 폴더 관리 워크플로

가상 시스템 폴더 관리 워크플로를 사용하여 가상 시스템 폴더를 생성 또는 삭제하거나 이름을 변경할 수 있습니다.

Orchestrator 클라이언트의 **워크플로** 보기에서 **라이브러리 > vCenter > 폴더 관리 > VM 폴더**를 통해 가상 시스템 폴더 관리 워크플로에 액세스할 수 있습니다.

| 워크플로 이름 | 설명 |
|-----------------|--|
| 가상 시스템 폴더 생성 | 가상 시스템 폴더를 생성합니다. |
| 가상 시스템 폴더 삭제 | 가상 시스템 폴더를 삭제하고 작업이 완료될 때까지 기다립니다. |
| 가상 시스템 폴더 이름 변경 | 가상 시스템 폴더의 이름을 변경하고 작업이 완료될 때까지 기다립니다. |

게스트 작업 파일 워크플로

게스트 작업 파일 워크플로를 사용하면 게스트 운영 체제에서 파일을 관리할 수 있습니다.

Orchestrator 클라이언트의 **워크플로** 보기에서 **라이브러리 > vCenter > 게스트 작업 > 파일**을 통해 게스트 작업 파일 워크플로에 액세스할 수 있습니다.

| 워크플로 이름 | 설명 |
|---------------------------|---|
| 게스트 디렉토리 확인 | 게스트 가상 시스템에 디렉토리가 있는지 확인합니다. |
| 게스트 파일 확인 | 게스트 가상 시스템에 파일이 있는지 확인합니다. |
| 게스트의 파일을 Orchestrator에 복사 | 게스트 파일 시스템의 지정된 파일을 Orchestrator 서버에 복사합니다. |
| Orchestrator의 파일을 게스트에 복사 | Orchestrator 서버의 지정된 파일을 게스트 파일 시스템에 복사합니다. |
| 게스트에서 디렉토리 생성 | 게스트 가상 시스템에서 디렉토리를 생성합니다. |

| 워크플로 이름 | 설명 |
|------------------|------------------------------|
| 게스트에서 임시 디렉토리 생성 | 게스트 가상 시스템에서 임시 디렉토리를 생성합니다. |
| 게스트에서 임시 파일 생성 | 게스트 가상 시스템에서 임시 파일을 생성합니다. |
| 게스트에서 디렉토리 삭제 | 게스트 가상 시스템에서 디렉토리를 삭제합니다. |
| 게스트에서 파일 삭제 | 게스트 가상 시스템에서 파일을 삭제합니다. |
| 게스트에서 경로 나열 | 게스트 가상 시스템에서 경로를 표시합니다. |
| 게스트에서 디렉토리 이동 | 게스트 가상 시스템에서 디렉토리를 이동합니다. |
| 게스트에서 파일 이동 | 게스트 가상 시스템에서 파일을 이동합니다. |

게스트 작업 프로세스 워크플로

게스트 작업 프로세스 워크플로를 사용하면 게스트 운영 체제에서 실행 중인 프로세스를 제어하고 이에 대한 정보를 가져올 수 있습니다.

Orchestrator 클라이언트의 **워크플로** 보기에서 **라이브러리 > vCenter > 게스트 작업 > 프로세스**를 통해 게스트 작업 파일 워크플로에 액세스할 수 있습니다.

| 워크플로 이름 | 설명 |
|------------------|---|
| 게스트에서 환경 변수 가져오기 | 게스트에서 환경 변수 목록을 반환합니다. 대화형 세션이 현재 로그인한 사용자의 변수를 반환합니다. |
| 게스트에서 프로세스 가져오기 | 게스트 운영 체제에서 실행되고 있는 프로세스와 API에 의해 시작되어 최근 완료된 프로세스 목록을 반환합니다. |
| 게스트에서 프로그램 실행 | 게스트 운영 체제에서 프로그램을 시작합니다. |
| 게스트에서 프로세스 중지 | 게스트 운영 체제에서 프로세스를 종료합니다. |

전원 호스트 관리 워크플로

호스트 전원 관리 워크플로를 사용하여 호스트를 재부팅하거나 종료할 수 있습니다.

Orchestrator 클라이언트의 **워크플로** 보기에서 **라이브러리 > vCenter > 호스트 관리 > 전원**을 통해 전원 호스트 관리 워크플로에 액세스할 수 있습니다.

호스트 재부팅

호스트를 재부팅합니다. Orchestrator 클라이언트가 호스트에 직접 연결된 경우 호스트에 대한 연결이 손실되고 반환된 작업에서 성공 표시를 받지 않습니다.

호스트 종료

호스트를 종료합니다. Orchestrator 클라이언트가 호스트에 직접 연결된 경우 호스트에 대한 연결이 손실되고 반환된 작업에서 성공 표시를 받지 않습니다.

기본 호스트 관리 워크플로

기본 호스트 관리 워크플로를 사용하여 호스트를 유지 보수 모드로 설정하고 호스트의 유지 보수 모드를 종료할 수 있습니다. 또한, 호스트를 폴더 또는 클러스터로 이동할 수 있고, 호스트에서 데이터를 다시 로드할 수 있습니다.

Orchestrator 클라이언트의 **워크플로** 보기에서 **라이브러리 > vCenter > 호스트 관리 > 기본**을 통해 기본 호스트 관리 워크플로에 액세스할 수 있습니다.

| 워크플로 이름 | 설명 |
|--------------|--|
| 유지 보수 모드 시작 | 호스트를 유지 보수 모드로 설정합니다. 작업을 취소할 수 있습니다. |
| 유지 보수 모드 종료 | 유지 보수 모드를 종료합니다. 작업을 취소할 수 있습니다. |
| 클러스터로 호스트 이동 | 기존 호스트를 클러스터로 이동합니다. 호스트가 동일한 데이터 센터에 속해야 하며, 호스트가 클러스터에 속하는 경우 호스트가 유지 보수 모드여야 합니다. |
| 폴더로 호스트 이동 | 호스트를 독립형 호스트로 폴더로 이동합니다. 호스트가 동일한 데이터 센터의 ClusterComputeResource 에 속해야 하며, 호스트가 유지 보수 모드여야 합니다. |
| 호스트 다시 로드 | vCenter Server가 강제로 호스트에서 데이터를 다시 로드합니다. |

호스트 등록 관리 워크플로

호스트 등록 관리 워크플로를 사용하여 클러스터에 호스트 추가, 클러스터에서 호스트 연결 끊기 및 다시 연결 등을 수행할 수 있습니다.

Orchestrator 클라이언트의 **워크플로** 보기에서 **라이브러리 > vCenter > 호스트 관리 > 등록**을 통해 호스트 관리 등록 워크플로에 액세스할 수 있습니다.

| 워크플로 이름 | 설명 |
|------------------|---|
| 클러스터에 호스트 추가 | 클러스터에 호스트를 추가합니다. 호스트의 SSL 인증서를 인증할 수 없는 경우 이 워크플로는 실패합니다. |
| 독립형 호스트 추가 | 호스트를 독립형 호스트로 등록합니다. |
| 호스트 연결 끊기 | vCenter Server 인스턴스에서 호스트 연결을 끊습니다. |
| 호스트 다시 연결 | 연결이 끊어진 호스트를 호스트 정보만 제공하여 다시 연결합니다. |
| 모든 정보로 호스트 다시 연결 | 연결이 끊어진 호스트를 호스트에 대한 모든 정보를 제공하여 다시 연결합니다. |
| 호스트 제거 | 호스트를 제거하고 vCenter Server 인스턴스에서 해당 호스트를 등록 취소합니다. 호스트가 클러스터에 속하는 경우 호스트를 제거하기 전에 호스트를 유지 보수 모드로 설정해야 합니다. |

네트워킹 워크플로

네트워킹 워크플로를 사용하여 분산 가상 스위치에 포트 그룹 추가, 포트 그룹으로 분산 가상 스위치 생성 등을 수행할 수 있습니다.

Orchestrator 클라이언트의 **워크플로** 보기에서 **라이브러리 > vCenter > 네트워킹**을 통해 네트워킹 워크플로에 액세스할 수 있습니다.

| 워크플로 이름 | 설명 |
|-------------------------|--------------------------------------|
| 분산 가상 스위치에 포트 그룹 추가 | 지정된 분산 가상 스위치에 새 분산 가상 포트 그룹을 추가합니다. |
| 분산 가상 스위치에 호스트 시스템 연결 | 분산 가상 스위치에 호스트를 추가합니다. |
| 포트 그룹이 포함된 분산 가상 스위치 생성 | 분산 가상 포트 그룹이 포함된 새 분산 가상 스위치를 생성합니다. |

분산 가상 포트 그룹 워크플로

분산 가상 포트 그룹 워크플로를 사용하여 포트 그룹을 업데이트하거나 삭제하고 포트 그룹을 재구성할 수 있습니다.

Orchestrator 클라이언트의 **워크플로** 보기에서 **라이브러리 > vCenter > 네트워킹 > 분산 가상 포트 그룹**을 통해 분산 가상 포트 그룹 워크플로에 액세스할 수 있습니다.

| 워크플로 이름 | 설명 |
|-------------------------------|---|
| 분산 가상 포트 그룹에 가상 시스템 NIC 번호 연결 | 지정된 가상 시스템 NIC 번호의 네트워크 연결을 다시 구성하여 지정된 분산 가상 포트 그룹에 연결합니다. NIC 번호를 지정하지 않은 경우 숫자 0이 사용됩니다. |
| 분산 가상 포트 그룹 삭제 | 지정된 분산 가상 포트 그룹을 삭제합니다. |
| 팀 구성 옵션 설정 | 분산 가상 포트 그룹에 팀 구성 옵션을 관리하기 위한 인터페이스를 제공합니다. |
| 분산 가상 포트 그룹 업데이트 | 지정된 분산 가상 포트 그룹의 구성을 업데이트합니다. |

분산 가상 스위치 워크플로

분산 가상 스위치 워크플로를 사용하여 분산 가상 스위치를 생성, 업데이트 또는 삭제하고 전용 VLAN을 생성, 삭제 또는 업데이트할 수 있습니다.

Orchestrator 클라이언트의 **워크플로** 보기에서 **라이브러리 > vCenter > 네트워킹 > 분산 가상 스위치**를 통해 분산 가상 스위치 워크플로에 액세스할 수 있습니다.

| 워크플로 이름 | 설명 |
|----------------|---|
| 분산 가상 스위치 생성 | 지정된 네트워크 폴더에 사용자가 지정하는 이름 및 업링크 포트 이름으로 분산 가상 스위치를 생성합니다. 하나 이상의 업링크 포트 이름을 지정해야 합니다. |
| 전용 VLAN 생성 | 지정된 분산 가상 스위치에서 VLAN을 생성합니다. |
| 분산 가상 스위치 삭제 | 분산 가상 스위치 및 모든 관련 요소를 삭제합니다. |
| 전용 VLAN 삭제 | 지정된 분산 가상 스위치에서 VLAN을 삭제합니다. 보조 VLAN이 있는 경우 먼저 보조 VLAN을 삭제해야 합니다. |
| 분산 가상 스위치 업데이트 | 분산 가상 스위치의 속성을 업데이트합니다. |
| 전용 VLAN 업데이트 | 지정된 분산 가상 스위치에서 VLAN을 업데이트합니다. |

표준 가상 스위치 워크플로

표준 가상 스위치 워크플로를 사용하여 표준 가상 스위치를 생성, 업데이트 또는 삭제하거나 표준 가상 스위치의 포트 그룹을 생성, 삭제 또는 업데이트할 수 있습니다.

Orchestrator 클라이언트의 **워크플로** 보기에서 **라이브러리 > vCenter > 네트워킹 > 표준 가상 스위치**를 통해 **표준 가상 스위치** 워크플로에 액세스할 수 있습니다.

| 워크플로 이름 | 설명 |
|------------------------------|---|
| 표준 가상 스위치에 포트 그룹 추가 | 표준 가상 스위치에 포트 그룹을 추가합니다. |
| 표준 가상 스위치 생성 | 표준 가상 스위치를 생성합니다. |
| 표준 가상 스위치에서 포트 그룹 삭제 | 표준 가상 스위치에 포트 그룹을 삭제합니다. |
| 표준 가상 스위치 삭제 | 호스트의 네트워크 구성에서 표준 가상 스위치를 삭제합니다. |
| 모든 표준 가상 스위치 검색 | 호스트에서 모든 표준 가상 스위치를 검색합니다. |
| 표준 가상 스위치에서 포트 그룹 업데이트 | 표준 가상 스위치에서 포트 그룹의 속성을 업데이트합니다. |
| 표준 가상 스위치 업데이트 | 표준 가상 스위치의 속성을 업데이트합니다. |
| 표준 가상 스위치에서 포트 그룹의 VNIC 업데이트 | 표준 가상 스위치에서 포트 그룹과 연관된 가상 NIC를 업데이트합니다. |

네트워킹 Virtual SAN 워크플로

Virtual SAN 워크플로를 사용하여 Virtual SAN 네트워크 트래픽을 구성할 수 있습니다.

Orchestrator 클라이언트의 **워크플로** 보기에서 **라이브러리 > vCenter > 네트워킹 > VSAN**을 통해 **네트워킹** 워크플로에 액세스할 수 있습니다.

| 워크플로 이름 | 설명 |
|-----------------------|------------------------------------|
| 클러스터 VSAN 트래픽 네트워크 설정 | 클러스터의 Virtual SAN 트래픽 네트워크를 설정합니다. |
| 호스트 VSAN 트래픽 네트워크 설정 | 호스트의 Virtual SAN 트래픽 네트워크를 설정합니다. |

리소스 풀 워크플로

리소스 풀 워크플로를 사용하여 리소스 풀을 생성, 이름 변경, 재구성 또는 삭제하고 리소스 풀 정보를 가져올 수 있습니다.

Orchestrator 클라이언트의 **워크플로** 보기에서 **라이브러리 > vCenter > 리소스 풀**을 통해 **리소스 풀** 워크플로에 액세스할 수 있습니다.

| 워크플로 이름 | 설명 |
|------------------|---|
| 리소스 풀 생성 | 기본 CPU 및 메모리 할당 값으로 리소스 풀을 생성합니다. 클러스터에서 리소스 풀을 생성하려면 클러스터에서 VMware DRS가 사용하도록 설정되어 있어야 합니다. |
| 지정된 값으로 리소스 풀 생성 | 지정한 CPU 및 메모리 할당 값으로 리소스 풀을 생성합니다. 클러스터에서 리소스 풀을 생성하려면 클러스터에서 VMware DRS가 사용하도록 설정되어 있어야 합니다. |
| 리소스 풀 삭제 | 리소스 풀을 삭제하고 작업이 완료될 때까지 기다립니다. |
| 리소스 풀 정보 가져오기 | 지정된 리소스 풀에 대한 CPU 및 메모리 정보를 반환합니다. |
| 리소스 풀 다시 구성 | 지정된 리소스 풀에 대한 CPU 및 메모리 할당 구성을 다시 구성합니다. |
| 리소스 풀 이름 변경 | 리소스 풀의 이름을 변경하고 작업이 완료될 때까지 기다립니다. |

스토리지 워크플로

스토리지 워크플로를 사용하여 스토리지 관련 작업을 수행할 수 있습니다.

Orchestrator 클라이언트의 **워크플로** 보기에서 **라이브러리 > vCenter > 스토리지**를 통해 스토리지 워크플로에 액세스할 수 있습니다.

| 워크플로 이름 | 설명 |
|-----------------------------|--|
| iSCSI/FC/로컬 SCSI에 데이터스토어 추가 | 파이버 채널, iSCSI 또는 로컬 SCSI 디스크에서 데이터스토어를 생성합니다. 기존 VMFS에서 현재 사용 중이 아닌 디스크만 새 데이터스토어 생성에 적용됩니다. 새 데이터스토어는 지정된 디스크의 사용 가능한 최대 공간을 할당합니다. |
| NFS에 데이터스토어 추가 | NFS 서버에 데이터스토어를 추가합니다. |
| iSCSI 대상 추가 | iSCSI 대상을 vCenter Server 호스트에 추가합니다. 대상은 Send 또는 Static 유형일 수 있습니다. |
| 사용 가능한 모든 디스크에 대한 VMFS 생성 | 지정된 호스트의 사용 가능한 모든 디스크에 대한 VMFS 볼륨을 생성합니다. |
| 데이터스토어 삭제 | vCenter Server 호스트에서 데이터스토어를 삭제합니다. |
| iSCSI 대상 삭제 | 이미 구성된 iSCSI 대상을 삭제합니다. 대상은 Send 또는 Static 유형일 수 있습니다. |
| iSCSI 어댑터 사용 안 함 | 지정된 호스트의 소프트웨어 iSCSI 어댑터를 사용하지 않도록 설정합니다. |
| 모든 데이터스토어 및 디스크 표시 | 지정된 호스트의 기존 데이터스토어 및 사용 가능한 디스크를 표시합니다. |
| iSCSI 어댑터 사용 | iSCSI 어댑터를 사용하도록 설정합니다. |
| 모든 스토리지 어댑터 나열 | 지정된 호스트의 모든 스토리지 어댑터를 나열합니다. |

Storage DRS 워크플로

Storage DRS 워크플로를 사용하여 데이터스토어 클러스터 생성 및 구성, 클러스터에서 데이터스토어 제거, 클러스터에 스토리지 추가 등과 같은 스토리지 관련 작업을 수행합니다.

Orchestrator 클라이언트의 **워크플로** 보기에서 **라이브러리 > vCenter > 스토리지 > Storage DRS**를 통해 Storage DRS 워크플로에 액세스할 수 있습니다.

| 워크플로 이름 | 설명 |
|------------------------------|---|
| 클러스터에 데이터스토어 추가 | 데이터스토어 클러스터에 데이터스토어를 추가합니다. 데이터스토어는 모든 호스트에 연결할 수 있어야 데이터스토어 클러스터에 포함될 수 있습니다. 데이터스토어는 동일한 연결 유형을 갖고 있어야 데이터스토어 클러스터 내에 상주할 수 있습니다. |
| 가상 시스템 구성에 따라 Storage DRS 변경 | 각 가상 시스템에 대한 Storage DRS 설정을 설정합니다. |
| 데이터스토어 클러스터 구성 | 자동화 및 런타임 규칙에 대한 데이터스토어 클러스터 설정 값을 구성합니다. |
| 단순 데이터스토어 클러스터 생성 | 기본 구성으로 단순 데이터스토어 클러스터를 생성합니다. 새 데이터스토어 클러스터에는 데이터스토어가 없습니다. |
| Storage DRS 스케줄링 작업 생성 | 데이터스토어 클러스터를 다시 구성하기 위한 스케줄링 작업을 생성합니다. 자동화 및 런타임 규칙만 설정할 수 있습니다. |

| 워크플로 이름 | 설명 |
|------------------------|--|
| 가상 시스템 반선택도 규칙 생성 | 특정 가상 시스템의 모든 가상 디스크가 서로 다른 데이터스토어에 유지되어야 함을 나타내기 위해 반선택도 규칙을 생성합니다. |
| VMDK 반선택도 규칙 생성 | 서로 다른 데이터스토어에 유지해야 할 가상 디스크를 나타내는 VMDK 반선택도 규칙을 가상 시스템에 대해 생성합니다. 이 규칙은 선택한 가상 시스템의 가상 디스크에 적용됩니다. |
| 데이터스토어 클러스터 제거 | 데이터스토어 클러스터를 제거합니다. 데이터스토어 클러스터를 제거하면 클러스터에 대한 모든 설정 및 정보도 vCenter Server 시스템에서 제거됩니다. |
| 클러스터에서 데이터스토어 제거 | 데이터스토어 클러스터에서 데이터스토어를 제거하고 데이터스토어를 데이터스토어 폴더에 배치합니다. |
| Storage DRS 스케줄링 작업 제거 | 스케줄링된 Storage DRS 작업을 제거합니다. |
| 가상 시스템 반선택도 규칙 제거 | 지정된 데이터스토어 클러스터에 대한 가상 시스템 반선택도 규칙을 제거합니다. |
| VMDK 반선택도 규칙 제거 | 지정된 데이터스토어 클러스터에 대한 VMDK 반선택도 규칙을 제거합니다. |

스토리지 VSAN 워크플로

Virtual SAN 워크플로를 사용하여 Virtual SAN 클러스터에서 비SSD 디스크 및 디스크 그룹을 관리할 수 있습니다.

Orchestrator 클라이언트의 워크플로 보기에서 **라이브러리 > vCenter > 스토리지 > VSAN**를 통해 네트워킹 워크플로에 액세스할 수 있습니다.

| 워크플로 이름 | 설명 |
|----------------------|---|
| 디스크 그룹에 디스크 추가 | Virtual SAN 디스크 그룹에 비SSD 디스크를 추가합니다. |
| 디스크 그룹에 디스크 할당 | Virtual SAN 시스템에서 사용할 디스크를 할당하고 자동으로 디스크 그룹을 생성하고 디스크를 기존 디스크 그룹으로 배포합니다. |
| 디스크 그룹 생성 | Virtual SAN 디스크 그룹을 생성합니다. |
| 호스트, 디스크 그룹 및 디스크 나열 | Virtual SAN 시스템에서 사용되거나 사용에 적합한 클러스터의 모든 호스트, 해당 디스크 그룹 및 디스크를 나열합니다. |
| 디스크 그룹 제거 | Virtual SAN 디스크 그룹을 제거합니다. |
| 디스크 그룹에서 디스크 제거 | Virtual SAN 디스크 그룹에서 비SSD 디스크를 제거합니다. |

기본 가상 시스템 관리 워크플로

기본 가상 시스템 관리 워크플로를 사용하여 가상 시스템 생성, 이름 변경 또는 삭제, 가상 하드웨어 업그레이드 등과 같은 가상 시스템에 대한 기본 작업을 수행할 수 있습니다.

Orchestrator 클라이언트의 워크플로 보기에서 **라이브러리 > vCenter > 가상 시스템 관리 > 기본**을 통해 기본 가상 시스템 관리 워크플로에 액세스할 수 있습니다.

| 워크플로 이름 | 설명 |
|--------------------------|---|
| 사용자 지정 가상 시스템 생성 | 지정된 구성 옵션 및 추가 디바이스로 가상 시스템을 생성합니다. |
| 단순 dvPortGroup 가상 시스템 생성 | 단순 가상 시스템을 생성합니다. 사용된 네트워크는 분산 가상 포트 그룹입니다. |

| 워크플로 이름 | 설명 |
|---------------------------------|--|
| 단순 가상 시스템 생성 | 가장 일반적인 디바이스 및 구성 옵션으로 가상 시스템을 생성합니다. |
| 가상 시스템 삭제 | 인벤토리 및 데이터스토어에서 가상 시스템을 제거합니다. |
| 이름으로 가상 시스템 가져오기 | 등록된 모든 vCenter Server 인스턴스에서 입력한 표현식과 일치하는 가상 시스템 목록을 반환합니다. |
| 템플릿으로 표시 | 기존 가상 시스템을 템플릿으로 변환하고 시작되도록 허용하지 않습니다. 템플릿을 사용하여 가상 시스템을 생성할 수 있습니다. |
| 가상 시스템으로 표시 | 기존 템플릿을 가상 시스템으로 변환하고 시작되도록 허용합니다. |
| 폴더로 가상 시스템 이동 | 지정된 가상 시스템 폴더로 가상 시스템을 이동합니다. |
| 리소스 풀로 가상 시스템 이동 | 리소스 풀로 가상 시스템을 이동합니다. 대상 리소스 풀이 동일한 클러스터에 없는 경우 마 이그레이션 또는 재배치 워크플로를 사용해야 합니다. |
| 폴더로 가상 시스템 이동 | 지정된 가상 시스템 폴더로 여러 가상 시스템을 이동합니다. |
| 리소스 풀로 가상 시스템 이동 | 리소스 풀로 여러 가상 시스템을 이동합니다. |
| 가상 시스템 등록 | 가상 시스템을 등록합니다. 가상 시스템 파일은 기존 데이터스토어에 배치되어야 하며 이미 등록되어 있으면 안 됩니다. |
| 가상 시스템 다시 로드 | 강제로 vCenter Server가 가상 시스템을 다시 로드하도록 합니다. |
| 가상 시스템 이름 변경 | 데이터스토어가 아니라 vCenter Server 시스템 또는 호스트에서 기존 가상 시스템의 이름을 변경합니다. |
| 가상 시스템 성능 설정 | 공유, 최소값과 최대값, 네트워크 조절 및 가상 시스템 디스크 액세스와 같은 성능 설정을 변경합니다. |
| 가상 시스템 등록 취소 | 인벤토리에서 기존 가상 시스템을 제거합니다. |
| 가상 시스템 하드웨어 업그레이드(필요한 경우 강제 적용) | 가상 시스템 하드웨어를 호스트가 지원하는 최신 개정판으로 업그레이드합니다. 이 워크플로에서는 VMware Tools가 최신 버전이 아닌 경우에도 강제로 업그레이드가 계속되도록 합니다. VMware Tools가 최신 버전이 아닌 경우 강제로 업그레이드가 계속되도록 하면 게스트 네트워크 설정이 기본 설정으로 복구됩니다. 이러한 상황을 방지하려면 워크플로를 실행하기 전에 VMware Tools를 업그레이드하십시오. |
| 가상 시스템 업그레이드 | 가상 하드웨어를 호스트가 지원하는 최신 개정판으로 업그레이드합니다. VMware Tools가 최신 버전이 아닌 경우에도 입력 매개 변수를 사용하여 업그레이드를 강제로 적용할 수 있습니다. |
| 작업 대기 및 가상 시스템 질문에 응답 | vCenter Server 작업이 완료될 때까지 또는 가상 시스템이 질문에 응답할 때까지 기다립니다. 가상 시스템에 응답이 필요한 경우 사용자 입력을 허용하고 질문에 응답합니다. |

복제 워크플로

복제 워크플로를 사용하여 가상 시스템 속성 사용자 지정 여부와 관계없이 가상 시스템을 복제할 수 있습니다.

Orchestrator 클라이언트의 워크플로 보기에서 라이브러리 > vCenter > 가상 시스템 관리 > 복제를 통해 복제 워크플로에 액세스할 수 있습니다.

| 워크플로 이름 | 설명 |
|----------------------|---|
| 속성에서 가상 시스템 복제 | 속성을 입력 매개 변수로 사용하여 가상 시스템을 복제합니다. |
| 가상 시스템 복제, 사용자 지정 없음 | 가상 시스템 UUID를 제외한 어떠한 항목도 변경하지 않고 가상 시스템을 복제합니다. |
| 속성에서 가상 시스템 사용자 지정 | 속성을 입력 매개 변수로 사용하여 가상 시스템을 사용자 지정합니다. |

연결된 복제 워크플로

연결된 복제 워크플로를 사용하여 연결된 복제에서 가상 시스템 복원, 연결된 복제 생성 등과 같은 연결된 복제 작업을 수행할 수 있습니다.

Orchestrator 클라이언트의 **워크플로** 보기에서 **라이브러리 > vCenter > 가상 시스템 관리 > 복제 > 연결된 복제** 폴더 및 하위 폴더를 통해 연결된 복제 워크플로에 액세스할 수 있습니다.

| 워크플로 이름 | 설명 |
|--|---|
| 연결된 복제에서 가상 시스템 복원 | 연결된 복제 설정에서 가상 시스템을 제거합니다. |
| 연결된 복제에 대한 가상 시스템 설정 | 연결된 복제가 되도록 가상 시스템을 준비합니다. |
| NIC가 여러 개인 Linux 시스템의 연결된 복제 생성 | Linux 가상 시스템의 연결된 복제를 생성하고, 게스트 운영 체제 사용자 지정을 수행하고, 최대 4개의 가상 네트워크 카드를 구성합니다. |
| NIC가 하나인 Linux 시스템의 연결된 복제 생성 | Linux 가상 시스템의 연결된 복제를 생성하고, 게스트 운영 체제 사용자 지정을 수행하고, 1개의 가상 네트워크 카드를 구성합니다. |
| NIC가 여러 개이고 자격 증명이 있는 Windows 시스템의 연결된 복제 생성 | Windows 가상 시스템의 연결된 복제를 생성하고 게스트 운영 체제 사용자 지정을 수행합니다. 최대 4개의 가상 네트워크 카드 및 로컬 관리자 계정을 구성합니다. |
| NIC가 하나이고 자격 증명이 있는 Windows 시스템의 연결된 복제 생성 | Windows 가상 시스템의 연결된 복제를 생성하고 게스트 운영 체제 사용자 지정을 수행합니다. 하나의 가상 네트워크 카드 및 로컬 관리자 계정을 구성합니다. |
| 사용자 지정 없이 연결된 복제 생성 | 가상 시스템에 대해 지정된 수의 연결된 복제를 생성합니다. |

Linux 사용자 지정 복제 워크플로

Linux 사용자 지정 워크플로를 사용하여 Linux 가상 시스템을 복제하고 게스트 운영 체제를 사용자 지정할 수 있습니다.

Orchestrator 클라이언트의 **워크플로** 보기에서 **라이브러리 > vCenter > 가상 시스템 관리 > 복제 > Linux 사용자 지정**을 통해 Linux 사용자 지정 복제 워크플로에 액세스할 수 있습니다.

| 워크플로 이름 | 설명 |
|-------------------------|--|
| NIC가 여러 개인 Linux 시스템 복제 | Linux 가상 시스템을 복제하고, 게스트 운영 체제 사용자 지정을 수행하고, 최대 4개의 가상 시스템 카드를 구성합니다. |
| NIC가 하나인 Linux 시스템 복제 | Linux 가상 시스템을 복제하고, 게스트 운영 체제 사용자 지정을 수행하고, 1개의 가상 시스템 카드를 구성합니다. |

도구 복제 워크플로

도구 복제 워크플로를 사용하여 가상 시스템의 운영 체제에 대한 사용자 지정 정보, 가상 디바이스 업데이트에 필요한 정보 등을 가져올 수 있습니다.

Orchestrator 클라이언트의 **워크플로** 보기에서 **라이브러리 > vCenter > 가상 시스템 관리 > 복제 > 도구**를 통해 도구 복제 워크플로에 액세스할 수 있습니다.

| 워크플로 이름 | 설명 |
|--|--|
| 네트워크를 변경하기 위해 가상 이더넷 카드 가져오기 | 가상 디바이스를 업데이트하기 위한 새 이더넷 카드를 반환합니다. 지정된 가상 디바이스의 디바이스 키 및 새 네트워크만 포함되어 있습니다. |
| Linux 사용자 지정 가져오기 | Linux 사용자 지정 준비를 반환합니다. |
| 여러 가상 이더넷 카드 디바이스 변경 사항 가져오기 | <code>VirtualEthernetCard</code> 개체에서 추가 및 제거 작업을 위한 <code>VirtualDeviceConfigSpec</code> 개체 어레이를 반환합니다. |
| NIC 설정 맵 가져오기 | <code>VimAdapterMapping</code> 을 사용하여 가상 네트워킹 카드에 대한 설정 맵을 반환합니다. |
| 자격 증명을 사용하여 Sysprep에 대한 Windows 사용자 지정 가져오기 | 자격 증명을 사용하여 Microsoft Sysprep 프로세스에 대한 사용자 지정 정보를 반환합니다. Windows 가상 시스템 복제 워크플로에서 이 워크플로를 사용합니다. |
| Unattended.txt를 사용하여 Sysprep에 대한 Windows 사용자 지정 가져오기 | Unattended.txt 파일을 사용하여 Microsoft Sysprep 프로세스에 대한 사용자 지정 정보를 반환합니다. Windows 가상 시스템 복제 워크플로에서 이 워크플로를 사용합니다. |
| Sysprep에 대한 Windows 사용자 지정 가져오기 | Microsoft Sysprep 프로세스에 대한 사용자 지정 정보를 반환합니다. Windows 가상 시스템 복제 워크플로에서 이 워크플로를 사용합니다. |

Windows 사용자 지정 복제 워크플로

Windows 사용자 지정 복제 워크플로를 사용하여 Windows 가상 시스템을 복제하고 게스트 운영 체제를 사용자 지정할 수 있습니다.

Orchestrator 클라이언트의 **워크플로** 보기에서 **라이브러리 > vCenter > 가상 시스템 관리 > 복제 > Windows 사용자 지정** 폴더 및 하위 폴더를 통해 Windows 사용자 지정 복제 워크플로에 액세스할 수 있습니다.

| 워크플로 이름 | 설명 |
|--|---|
| NIC가 하나이고 자격 증명이 있는 Windows 시스템 사용자 지정 | 게스트 운영 체제 사용자 지정을 수행하고, Windows 가상 시스템에서 하나의 가상 네트워크 카드 및 로컬 관리자 계정을 구성합니다. |
| NIC가 하나이고 자격 증명이 있는 선택 프로비저닝된 Windows 시스템 복제 | 게스트 운영 체제 사용자 지정을 수행하는 Windows 가상 시스템을 복제합니다. 하나의 가상 네트워크 카드 및 로컬 관리자 계정을 구성합니다. Sysprep 도구는 vCenter Server에서 사용 가능해야 합니다. |
| NIC가 하나이고 자격 증명이 있는 Windows 시스템 Sysprep 복제 | 게스트 운영 체제 사용자 지정을 수행하는 Windows 가상 시스템을 복제합니다. 하나의 가상 네트워크 카드 및 로컬 관리자 계정을 구성합니다. Sysprep 도구는 vCenter Server에서 사용 가능해야 합니다. |
| NIC가 여러 개이고 자격 증명이 있는 Windows 시스템 복제 | 게스트 운영 체제 사용자 지정을 수행하는 Windows 가상 시스템을 복제합니다. 로컬 관리자 계정 및 최대 4개의 가상 네트워크 카드를 구성합니다. Sysprep 도구는 vCenter Server 시스템에서 사용 가능해야 합니다. |

| 워크플로 이름 | 설명 |
|------------------------------------|--|
| NIC가 하나인 Windows 시스템 복제 | 게스트 운영 체제 사용자 지정을 수행하는 Windows 가상 시스템을 복제하고, 하나의 가상 네트워크 카드를 구성합니다. Sysprep 도구는 vCenter Server 시스템에서 사용 가능해야 합니다. |
| NIC가 하나이고 자격 증명이 있는 Windows 시스템 복제 | 게스트 운영 체제 사용자 지정을 수행하는 Windows 가상 시스템을 복제합니다. 하나의 가상 네트워크 카드 및 로컬 관리자 계정을 구성합니다. Sysprep 도구는 vCenter Server 시스템에서 사용 가능해야 합니다. |

디바이스 관리 워크플로

디바이스 관리 워크플로를 사용하여 가상 시스템이나 호스트 데이터스토어에 연결된 디바이스를 관리할 수 있습니다.

Orchestrator 클라이언트의 워크플로 보기에서 라이브러리 > vCenter > 가상 시스템 관리 > 디바이스 관리를 통해 디바이스 관리 워크플로에 액세스할 수 있습니다.

| 워크플로 이름 | 설명 |
|---------------------------------------|---|
| CD-ROM 추가 | 가상 시스템에 가상 CD-ROM을 추가합니다. 가상 시스템에 IDE 컨트롤러가 없는 경우 워크플로에서 IDE 컨트롤러를 생성합니다. |
| 디스크 추가 | 가상 시스템에 가상 디스크를 추가합니다. |
| RAM 변경 | 가상 시스템의 RAM 용량을 변경합니다. |
| 디스크를 쉐 프로비저닝으로 변환 | 가상 시스템의 쉐 프로비저닝된 디스크를 쉐 프로비저닝된 디스크로 변환합니다. |
| 독립형 디스크 변환 | 디스크에서 독립형 플래그를 제거하여 모든 독립형 가상 시스템 디스크를 보통 디스크로 변환합니다. |
| 실행 중인 가상 시스템에서 분리할 수 있는 모든 디바이스 연결 끊기 | 실행 중인 가상 시스템에서 플로피 디스크, CD-ROM 드라이브, 병렬 포트 및 직렬 포트의 연결을 끊습니다. |
| CD-ROM 마운트 | 가상 시스템의 CD-ROM을 마운트합니다. 가상 시스템에 IDE 컨트롤러 또는 CD-ROM 드라이브가 없는 경우 워크플로에서 IDE 컨트롤러 또는 CD-ROM 드라이브를 생성합니다. |
| 플로피 디스크 드라이브 마운트 | ESXi 데이터스토어에서 플로피 디스크 드라이브 FLP 파일을 마운트합니다. |

이동 및 마이그레이션 워크플로

이동 및 마이그레이션 워크플로를 사용하여 가상 시스템을 마이그레이션할 수 있습니다.

Orchestrator 클라이언트의 워크플로 보기에서 라이브러리 > vCenter > 가상 시스템 관리 > 이동 및 마이그레이션을 통해 이동 및 마이그레이션 워크플로에 액세스할 수 있습니다.

| 워크플로 이름 | 설명 |
|------------------------------------|--|
| Storage vMotion으로 가상 시스템 대량 마이그레이션 | Storage vMotion을 사용하여 단일 가상 시스템, 선택한 가상 시스템 또는 사용 가능한 모든 가상 시스템을 마이그레이션합니다. |
| vMotion으로 가상 시스템 대량 마이그레이션 | vMotion, Storage vMotion 또는 vMotion과 Storage vMotion을 둘 다 사용하여 단일 가상 시스템, 선택한 가상 시스템 또는 사용 가능한 모든 가상 시스템을 마이그레이션합니다. |

| 워크플로 이름 | 설명 |
|------------------------------------|---|
| vMotion으로 가상 시스템 마이그레이션 | vSphere API에서 MigrateVM_Task 작업을 사용하여 한 호스트에서 다른 호스트로 가상 시스템을 마이그레이션합니다. |
| 가상 시스템을 다른 vCenter Server 시스템으로 이동 | 가상 시스템 목록을 다른 vCenter Server 시스템으로 이동합니다. |
| 여러 가상 시스템 빠른 마이그레이션 | 가상 시스템의 전원이 켜진 경우 해당 시스템을 일시 중단하고 동일한 스토리지를 사용하는 다른 호스트로 마이그레이션합니다. |
| 가상 시스템 빠른 마이그레이션 | 가상 시스템의 전원이 켜진 경우 해당 시스템을 일시 중단하고 동일한 스토리지를 사용하는 다른 호스트로 마이그레이션합니다. |
| 가상 시스템 디스크 재배치 | vSphere API에서 RelocateVM_Task 작업을 사용하여 가상 시스템의 전원이 꺼진 동안 가상 시스템 디스크를 다른 호스트 또는 데이터스토어로 재배치합니다. |

기타 워크플로

기타 범주의 워크플로를 사용하여 FT(Fault Tolerance)를 사용 또는 사용 안 함으로 설정하고, 가상 시스템 정보를 추출하고, 연결이 끊어진 가상 시스템을 찾을 수 있습니다.

Orchestrator 클라이언트의 **워크플로** 보기에서 **라이브러리 > vCenter > 가상 시스템 관리 > 기타**를 통해 이러한 워크플로에 액세스할 수 있습니다.

| 워크플로 이름 | 설명 |
|-----------------------------|--|
| FT 사용 안 함 | 지정된 가상 시스템에 대해 Fault Tolerance 를 사용하지 않도록 설정합니다. |
| FT 사용 | 지정된 가상 시스템에 대해 Fault Tolerance 를 사용하도록 설정합니다. |
| 가상 시스템 정보 추출 | 지정된 가상 시스템의 가상 시스템 폴더, 호스트 시스템, 리소스 풀, 계산 리소스, 데이터스토어, 하드 드라이브 크기, CPU와 메모리, 네트워크 및 IP 주소를 반환합니다. VMware Tools가 필요할 수 있습니다. |
| 연결이 끊어진 가상 시스템 찾기 | Orchestrator 인벤토리에서 연결이 끊어진 상태인 모든 가상 시스템을 나열합니다. Orchestrator 인벤토리에서 가상 시스템과 아무런 관련이 없는, Orchestrator 인벤토리 내의 모든 데이터스토어에 대한 VMDK 및 VMTX 파일을 나열합니다. 목록을 이메일로 보냅니다(선택 사항). |
| 이름 및 BIOS UUID로 가상 시스템 가져오기 | 이름으로 가상 시스템을 검색한 다음 고유한 가상 시스템을 식별하기 위해 범용 고유 식별자(UUID)로 결과를 필터링합니다. 참고 이 워크플로는 DynamicOps가 특정 DynamicOps와 vRealize Orchestrator 가상 시스템 간의 관련성을 만들기 위해 VC:VirtualMachine 유형의 입력 매개 변수를 갖는 vRealize Orchestrator 워크플로를 호출할 때 필요합니다. |
| 이름 및 UUID로 가상 시스템 가져오기 | 이름으로 가상 시스템을 검색한 다음 고유한 가상 시스템을 식별하기 위해 범용 고유 식별자(UUID)로 결과를 필터링합니다. 참고 이 워크플로는 DynamicOps가 특정 DynamicOps와 vRealize Orchestrator 가상 시스템 간의 관련성을 만들기 위해 VC:VirtualMachine 유형의 입력 매개 변수를 갖는 vRealize Orchestrator 워크플로를 호출할 때 필요합니다. |
| 가상 시스템 UUID 가져오기 | 이름으로 가상 시스템을 검색한 다음 고유한 가상 시스템을 식별하기 위해 범용 고유 식별자(UUID)로 결과를 필터링합니다. 참고 이 워크플로는 DynamicOps가 특정 DynamicOps와 vRealize Orchestrator 가상 시스템 간의 관련성을 만들기 위해 VC:VirtualMachine 유형의 입력 매개 변수를 갖는 vRealize Orchestrator 워크플로를 호출할 때 필요합니다. |

전원 관리 워크플로

전원 관리 워크플로를 사용하여 가상 시스템 전원 켜기 및 끄기, 가상 시스템의 게스트 운영 체제 재부팅, 가상 시스템 일시 중단 등을 수행할 수 있습니다.

Orchestrator 클라이언트의 **워크플로** 보기에서 **라이브러리 > vCenter > 가상 시스템 관리 > 전원 관리**를 통해 전원 관리 워크플로에 액세스할 수 있습니다.

| 워크플로 이름 | 설명 |
|-------------------|---|
| 가상 시스템 전원 끄기 및 대기 | 가상 시스템의 전원을 끄고 프로세스가 완료될 때까지 기다립니다. |
| 게스트 OS 재부팅 | 가상 시스템의 게스트 운영 체제를 재부팅합니다. 비영구 가상 시스템을 재설정하지 않습니다. VMware Tools는 실행 중이어야 합니다. |
| 가상 시스템 재설정 및 대기 | 가상 시스템을 재설정하고 프로세스가 완료될 때까지 기다립니다. |
| 가상 시스템 재개 및 대기 | 일시 중단된 가상 시스템을 재설정하고 프로세스가 완료될 때까지 기다립니다. |
| 게스트 OS를 대기 모드로 설정 | 게스트 운영 체제를 대기 모드로 설정합니다. VMware Tools는 실행 중이어야 합니다. |
| 가상 시스템 종료 및 삭제 | 가상 시스템을 종료하고 인벤토리 및 디스크에서 삭제합니다. |
| 게스트 OS 종료 및 대기 | 게스트 운영 체제를 종료하고 프로세스가 완료될 때까지 기다립니다. |
| 가상 시스템 시작 및 대기 | 가상 시스템을 시작하고 VMware Tools가 시작될 때까지 기다립니다. |
| 가상 시스템 일시 중단 및 대기 | 가상 시스템을 일시 중단하고 프로세스가 완료될 때까지 기다립니다. |

스냅샷 워크플로

스냅샷 워크플로를 사용하여 스냅샷 관련 작업을 수행할 수 있습니다.

Orchestrator 클라이언트의 **워크플로** 보기에서 **라이브러리 > vCenter > 가상 시스템 관리 > 스냅샷**을 통해 스냅샷 워크플로에 액세스할 수 있습니다.

| 워크플로 이름 | 설명 |
|-----------------------------|---|
| 스냅샷 생성 | 스냅샷을 생성합니다. |
| 리소스 풀에 있는 모든 가상 시스템의 스냅샷 생성 | 리소스 풀에 있는 각 가상 시스템의 스냅샷을 생성합니다. |
| 모든 스냅샷 제거 | 이전 스냅샷으로 복구하지 않고 기존의 모든 스냅샷을 제거합니다. |
| 과도한 스냅샷 제거 | 지정된 수보다 많은 스냅샷이 있는 가상 시스템을 찾아 선택적으로 가장 오래된 스냅샷을 삭제합니다. 결과를 이메일로 보냅니다. |
| 오래된 스냅샷 제거 | 지정된 일 수보다 오래된 모든 스냅샷을 가져와서 사용자에게 삭제할 스냅샷을 선택하라는 메시지를 표시합니다. |
| 지정된 크기의 스냅샷 제거 | 지정된 크기보다 큰 모든 스냅샷을 가져와서 사용자에게 삭제를 확인하라는 메시지를 표시합니다. |
| 현재 스냅샷으로 복구 | 현재 스냅샷으로 복구합니다. |
| 스냅샷으로 복구 및 대기 | 특정 스냅샷으로 복구합니다. 스냅샷을 삭제하지 않습니다. |

VMware Tools 워크플로

VMware Tools 워크플로를 사용하여 가상 시스템에서 VMware Tools 관련 작업을 수행합니다.

Orchestrator 클라이언트의 **워크플로** 보기에서 **라이브러리 > vCenter > 가상 시스템 관리 > VMware Tools**를 통해 VMware Tools 워크플로에 액세스할 수 있습니다.

| 워크플로 이름 | 설명 |
|--------------------------------------|--|
| VMware Tools 설치 관리자 마운트 | VMware Tools 설치 관리자를 가상 CD-ROM에 마운트합니다. |
| 콘솔 화면 해상도 설정 | 콘솔 창의 해상도를 설정합니다. 가상 시스템의 전원을 켜야 합니다. |
| 시간 동기화 설정 | VMware Tools에서 가상 시스템과 ESXi 서버 간의 시간 동기화를 설정합니다. |
| VMware Tools 설치 관리자 마운트 해제 | VMware Tools CD-ROM을 마운트 해제합니다. |
| 재부팅 없이 Windows 가상 시스템에서 도구를 업데이트합니다. | 재부팅을 수행하지 않고 Windows 가상 시스템에서 VMware Tools를 업데이트합니다. |
| VMware Tools 업그레이드 | 가상 시스템에서 VMware Tools를 업그레이드합니다. |
| 다음 재부팅 시 VMware Tools 업그레이드 | 자동 재부팅을 수행하지 않고 가상 시스템에서 VMware Tools를 업그레이드합니다. |

vRealize Automation 플러그인 사용

4

vRealize Automation 플러그인을 사용하여 vRealize Automation에서 vRealize Orchestrator 워크플로를 실행할 수 있습니다.

플러그인과 함께 제공된 워크플로를 사용하여 vRealize Automation에서 리소스를 쉽게 배포하고 관리할 수 있습니다. 제공된 워크플로 외에 사용자 지정 워크플로를 생성할 수 있습니다.

본 장은 다음 항목을 포함합니다.

- vRealize Automation용 VMware vRealize Orchestrator 플러그인 소개
- vRealize Automation 플러그인 구성
- vRealize Automation 플러그인 워크플로 사용
- 예제 vRealize Automation 플러그인 스크립트

vRealize Automation용 VMware vRealize Orchestrator 플러그인 소개

vRealize Automation용 VMware vRealize Orchestrator 플러그인은 vRealize Orchestrator와 vRealize Automation 간의 상호 작용을 지원합니다.

vRealize Automation 플러그인을 사용하여 다음 vRealize Automation 기능에 대한 워크플로를 생성하고 실행할 수 있습니다.

- XaaS 사용자 지정 리소스 및 Blueprint 관리
- 카탈로그 항목 및 리소스 관리와 요청
- 사용 권한 구성
- 승인 정책 구성
- 작업 항목 상호 작용
- vSphere 및 vCloud Director 가상 시스템 프로비저닝 작업과 사후 프로비저닝 작업
- vRealize Automation IaaS 모델에 대한 CRUD(생성, 읽기, 업데이트 및 삭제) 작업

vRealize Automation 플러그인을 사용하는 vRealize Orchestrator의 역할

Orchestrator 클라이언트를 사용하여 워크플로를 실행 및 생성하고 플러그인 API에 액세스할 수 있습니다. vRealize Automation 설치에 포함된 vRealize Orchestrator 인스턴스 또는 외부 vRealize Orchestrator 서버를 사용할 수 있습니다.

vRealize Orchestrator는 vRealize Automation 플러그인을 지원합니다. vRealize Orchestrator는 VMware 클라우드 스택 및 타사 기술을 관리할 수 있는 확장 가능한 워크플로 라이브러리를 제공하는 개발 및 프로세스 자동화 플랫폼입니다.

vRealize Orchestrator는 개방형 플러그인 아키텍처를 통해 관리 및 운영 솔루션과의 통합을 지원합니다.

vRealize Automation 플러그인 구성

vRealize Automation 호스트와 IaaS 호스트를 추가하여 플러그인을 구성할 수 있습니다.

구성 워크플로

구성 워크플로 범주의 워크플로를 사용하여 vRealize Automation 호스트를 관리할 수 있습니다.

vRealize Automation 호스트

이러한 워크플로는 Orchestrator 클라이언트의 **워크플로** 보기에서 플러그인 라이브러리의 **구성** 하위 디렉토리를 통해 액세스할 수 있습니다.

| 워크플로 이름 | 설명 |
|------------------------------|--|
| vRA 호스트 추가 | 플러그인 인벤토리에 vRealize Automation 호스트를 추가합니다. 테넌트 관리 및 운영 작업의 경우 인벤토리 보기를 사용하여 각 테넌트에서 워크플로를 실행할 수 있습니다. 테넌트에 대해 플러그인의 전체 기능을 사용하려면 각 테넌트에 대해 전용 vRealize Automation 호스트를 생성합니다. |
| 구성 요소 레지스트리를 사용하여 vRA 호스트 추가 | 사용자별 세션 연결을 사용하여 플러그인 인벤토리에 vRealize Automation 호스트를 추가합니다. vRealize Automation 시스템 관리자의 자격 증명을 사용하여 Orchestrator 클라이언트에 로그인해야 합니다. 외부 vRealize Orchestrator 서버에서 이 기능을 사용하려면 vRealize Automation 구성 요소 레지스트리에 Orchestrator 서버를 등록해야 합니다. 참고 구성 요소 레지스트리에 외부 vRealize Orchestrator 서버를 등록하려면 vRealize Automation을 인증 제공자로 사용하도록 Orchestrator를 구성해야 합니다. 자세한 내용은 "VMware vRealize Orchestrator 설치 및 구성"을 참조하십시오. |
| vRA 호스트의 IaaS 호스트 추가 | 플러그인 인벤토리에 선택한 vRealize Automation 호스트의 IaaS 호스트를 추가합니다. |
| vRA 호스트 제거 | 플러그인 인벤토리에서 vRealize Automation 호스트를 제거합니다. |

| 워크플로 이름 | 설명 |
|----------------|---|
| vRA 호스트 업데이트 | 플러그인 인벤토리에서 vRealize Automation 호스트를 업데이트합니다. |
| vRA 호스트 유효성 검사 | vRealize Automation 호스트와 해당 연결의 유효성을 검사합니다. |

참고 vRealize Orchestrator 서버가 vRealize Automation 구성 요소 레지스트리에 등록된 경우 이름이 Default인 vRealize Automation 호스트가 자동으로 추가됩니다. Default 호스트는 기본 테넌트에 대한 사용자별 세션 연결을 사용합니다. vRealize Automation 설치에 포함된 Orchestrator 서버는 기본적으로 vRealize Automation 구성 요소 레지스트리에 등록됩니다.

vRealize Automation IaaS 호스트

이러한 워크플로는 Orchestrator 클라이언트의 **워크플로** 보기에서 플러그인 라이브러리의 **인프라 관리 > 구성** 하위 디렉토리를 통해 액세스할 수 있습니다.

vRealize Automation 설치에 포함된 vRealize Orchestrator 서버는 기본적으로 vRealize Automation 구성 요소 레지스트리에 등록됩니다.

| 워크플로 이름 | 설명 |
|-----------------|---|
| IaaS 호스트 추가 | 플러그인 인벤토리에 vRealize Automation IaaS 호스트를 추가합니다. 이 워크플로는 vRA 호스트의 IaaS 호스트 추가와 기능적으로 동일하지만 vRealize Automation 호스트가 필요 없습니다. |
| IaaS 호스트 제거 | 플러그인 인벤토리에서 vRealize Automation IaaS 호스트를 제거합니다. |
| IaaS 호스트 업데이트 | 플러그인 인벤토리에서 vRealize Automation IaaS 호스트를 업데이트합니다. |
| IaaS 호스트 유효성 검사 | vRealize Automation IaaS 호스트와 해당 연결의 유효성을 검사합니다. |

vRealize Automation 호스트 추가

워크플로를 실행하여 vRealize Automation 호스트를 추가하고 호스트 연결 매개 변수를 구성할 수 있습니다.

절차

- 1 Orchestrator 클라이언트의 드롭다운 메뉴에서 **실행** 또는 **설계**를 선택합니다.
- 2 **워크플로** 보기를 클릭합니다.
- 3 **라이브러리 > vRealize Automation > 구성**을 확장합니다.
- 4 **vRA 호스트 추가** 워크플로를 마우스 오른쪽 버튼으로 클릭하고 **워크플로 시작**을 선택합니다.
- 5 **호스트 이름** 텍스트 상자에 호스트의 고유한 이름을 입력합니다.
- 6 **호스트 URL** 텍스트 상자에 호스트의 URL 주소를 입력합니다.

예: `https://hostname`

- 7 (필수 사항) **테넌트** 텍스트 상자에 테넌트의 이름을 입력합니다.

테넌트에 대해 플러그인의 전체 기능을 사용하려면 각 테넌트에 대해 전용 vRealize Automation 호스트를 생성합니다.

- 8 사용자의 확인 없이 SSL 인증서를 자동으로 설치할지 여부를 선택합니다.
- 9 (선택 사항) vRealize Orchestrator가 vRealize Automation로부터 응답 또는 연결을 기다리는 시간을 구성하려면 **연결 시간 제한(초)** 및 **작업 시간 제한(초)** 텍스트 상자에 시간 초과 간격을 입력합니다.
- 10 **세션 모드** 드롭다운 메뉴에서 호스트에 대한 연결 유형을 선택합니다.

| 옵션 | 작업 |
|---------|---|
| 공유 세션 | 인증 사용자 이름 및 인증 암호 텍스트 상자에 vRealize Automation 사용자에게 대한 자격 증명을 입력합니다. |
| 사용자별 세션 | <p>현재 로그인되어 있는 사용자의 자격 증명을 사용하여 연결합니다. vRealize Automation 시스템 관리자의 자격 증명을 사용하여 Orchestrator 클라이언트에 로그인해야 합니다.</p> <p>외부 vRealize Orchestrator 서버에서 이 옵션을 사용하려면 vRealize Automation 구성 요소 레지스트리에 Orchestrator 서버를 등록해야 합니다.</p> <p>참고 구성 요소 레지스트리에 외부 vRealize Orchestrator 서버를 등록하려면 vRealize Automation을 인증 제공자로 사용하도록 Orchestrator를 구성해야 합니다. 자세한 내용은 "VMware vRealize Orchestrator 설치 및 구성"을 참조하십시오.</p> |

- 11 **제출**을 클릭합니다.

다음에 수행할 작업

vRealize Automation 인프라 관리 호스트를 추가합니다.

IaaS 호스트 추가

워크플로를 실행하여 vRealize Automation 호스트의 IaaS 호스트를 추가하고 연결 매개 변수를 구성할 수 있습니다.

절차

- 1 Orchestrator 클라이언트의 드롭다운 메뉴에서 **실행** 또는 **설계**를 선택합니다.
- 2 **워크플로 보기**를 클릭합니다.
- 3 **라이브러리 > vRealize Automation > 인프라 관리 > 구성**을 확장합니다.
- 4 **IaaS 호스트 추가**를 마우스 오른쪽 버튼으로 클릭하고 **워크플로 시작**을 선택합니다.
- 5 **vCAC 호스트** 드롭다운 메뉴에서 IaaS 호스트를 구성하려는 vRealize Automation 호스트를 선택합니다.
- 6 **호스트 이름** 텍스트 상자에 호스트의 고유한 이름을 입력합니다.
- 7 Model Manager가 설치되어 있는 시스템의 URL을 입력합니다.
예를 들어 `https://model_manager_machine.com`을 입력합니다.
- 8 SSL 인증서를 설치하려면 **예**를 선택합니다.

9 프록시를 사용하여 Model Manager 시스템에 액세스하려면 **예**를 선택합니다.

이 옵션을 선택하는 경우 다음 페이지에서 프록시 호스트와 프록시 포트를 제공해야 합니다.

10 **다음**을 클릭합니다.

11 명시적 프록시를 구성 중인 경우 프록시 호스트 URL과 포트를 제공합니다.

12 **다음**을 클릭합니다.

13 고유한 시간 제한 값을 구성하려면 **아니요**를 클릭합니다.

14 (선택 사항) vRealize Orchestrator가 vRealize Automation로부터 응답 또는 연결을 기다리는 시간을 구성하려면 **연결 시간 제한(초)** 및 **작업 시간 제한(초)** 텍스트 상자에 시간 초과 간격을 입력합니다.

15 **다음**을 클릭합니다.

16 호스트의 인증 유형을 선택합니다.

| 옵션 | 설명 |
|-------------|--|
| SSO | vCenter Single Sign-On을 사용하려면 이 옵션을 선택합니다. |
| NTLM | 사용 중인 Active Directory 인프라에서 NTLM 인증을 사용할 경우에만 이 옵션을 선택하여 NTLM(NT LAN Manager) 프로토콜 기반 인증을 사용하도록 설정합니다. 이 옵션을 선택할 경우 추가 NTLM 자격 증명 및 인증 옵션을 사용해야 합니다. |

17 NTLM을 선택했으면 **다음**을 클릭하고 Workstation 시스템 이름과 NetBIOS 도메인 이름을 입력합니다.

18 **제출**을 클릭합니다.

vRealize Automation 플러그인 워크플로 사용

vRealize Automation 플러그인 워크플로 라이브러리에는 카탈로그와의 상호 작용, 인프라 관리, 테넌트 및 서비스 생성 등의 일반 작업에 사용할 수 있는 워크플로가 포함됩니다.

vRealize Automation 관련 헤더 작업 및 ID와 같은 사용자 지정 HTTP 헤더를 사용하고, CRUD, 프로비저닝 및 사후 프로비저닝 워크플로에서 적용할 수 있습니다.

제거 작업 제한 사항

버전 7.0부터 일부 생성, 읽기, 업데이트 및 삭제 작업이 제한됩니다. 이전 버전에서 사용한 워크플로의 작업은 7.0 이상에서 작동하지 않습니다. 워크플로를 지원되는 작업으로 업데이트하거나 필요한 작업을 다시 사용하도록 설정할 수 있습니다.

작업을 다시 사용하도록 설정하려면 **operations.properties** 파일에서 해당 작업을 제거해야 합니다. 파일의 작업 목록은 **제한된 작업** 항목을 참조하십시오.

절차

1 vRealize Orchestrator의 드롭다운 메뉴에서 **디자인**을 선택합니다.

- 2 리소스 보기를 클릭합니다.
- 3 리소스 계층에서 **라이브러리 > VCAC > 유틸리티**를 확장합니다.
- 4 백업을 만들고 **operations.properties** 파일을 수정합니다.
 - a **operations.properties**를 마우스 오른쪽 버튼으로 클릭하고 **파일에 저장**을 선택합니다.
 - b 사본을 백업으로 저장합니다.
 - c 새 사본을 만들고 다시 사용하도록 설정할 작업을 삭제합니다.
 - d 새 파일을 저장합니다.
- 5 vRealize Orchestrator에서 기존 파일을 대체합니다.
 - a vRealize Orchestrator에서 **유틸리티** 폴더를 마우스 오른쪽 버튼으로 클릭하고 **리소스 가져오기**를 클릭합니다.
 - b **operations.properties** 파일의 새 버전을 찾아서 **열기**를 클릭합니다.
 - c **한 번 바꾸기**를 클릭하여 수정된 버전을 저장합니다.
- 6 vRealize Orchestrator 서버를 다시 시작합니다.
- 7 **operations.properties** 파일을 선택하고 **뷰어** 탭을 클릭합니다.
- 8 사용하도록 설정 중인 작업이 더 이상 파일에 없는지 확인합니다.

결과

파일에서 제거한 작업이 이제 이전 워크플로에서 작동합니다.

다음에 수행할 작업

새 워크플로를 만들 때 제한된 작업을 사용하지 않도록 합니다.

제한된 작업

operations.properties 파일의 콘텐츠에는 제한된 작업이 포함되어 있습니다. 작업을 다시 사용하도록 설정하려면 파일에서 해당 작업을 제거해야 합니다.

다음 텍스트는 **operations.properties** 파일의 기본 버전입니다. 작업을 다시 사용하도록 설정하려면 [제거 작업 제한 사항](#) 항목을 참조하십시오.

```
#Blueprints
operation.create=ManagementModelEntities.svc@VirtualMachineTemplates
operation.update=ManagementModelEntities.svc@VirtualMachineTemplates
operation.delete=ManagementModelEntities.svc@VirtualMachineTemplates
#Blueprint properties
operation.create=ManagementModelEntities.svc@VirtualMachineProperties
operation.read=ManagementModelEntities.svc@VirtualMachineProperties
operation.update=ManagementModelEntities.svc@VirtualMachineProperties
operation.delete=ManagementModelEntities.svc@VirtualMachineProperties
#Global profiles
operation.create=ManagementModelEntities.svc@GlobalProfiles
```

```

operation.read=ManagementModelEntities.svc@GlobalProfiles
operation.update=ManagementModelEntities.svc@GlobalProfiles
operation.delete=ManagementModelEntities.svc@GlobalProfiles
#Global profile properties
operation.create=ManagementModelEntities.svc@GlobalProfileProperties
operation.read=ManagementModelEntities.svc@GlobalProfileProperties
operation.update=ManagementModelEntities.svc@GlobalProfileProperties
operation.delete=ManagementModelEntities.svc@GlobalProfileProperties
#PropertySetXml
operation.create=ManagementModelEntities.svc@PropertySetXml
operation.read=ManagementModelEntities.svc@PropertySetXml
operation.update=ManagementModelEntities.svc@PropertySetXml
operation.delete=ManagementModelEntities.svc@PropertySetXml
#Property definitions
operation.create=ManagementModelEntities.svc@PropertyDefinitions
operation.read=ManagementModelEntities.svc@PropertyDefinitions
operation.update=ManagementModelEntities.svc@PropertyDefinitions
operation.delete=ManagementModelEntities.svc@PropertyDefinitions
#Property attributes
operation.create=ManagementModelEntities.svc@PropertyAttributes
operation.read=ManagementModelEntities.svc@PropertyAttributes
operation.update=ManagementModelEntities.svc@PropertyAttributes
operation.delete=ManagementModelEntities.svc@PropertyAttributes
#Property Attribute Types
operation.create=ManagementModelEntities.svc@PropertyAttributeTypes
operation.read=ManagementModelEntities.svc@PropertyAttributeTypes
operation.update=ManagementModelEntities.svc@PropertyAttributeTypes
operation.delete=ManagementModelEntities.svc@PropertyAttributeTypes
#Control layouts
operation.create=ManagementModelEntities.svc@ControlLayouts
operation.read=ManagementModelEntities.svc@ControlLayouts
operation.update=ManagementModelEntities.svc@ControlLayouts
operation.delete=ManagementModelEntities.svc@ControlLayouts
#Amazon Virtual Machine Templates
operation.create=AmazonWSModelEntities.svc@AmazonVirtualMachineTemplates
operation.read=AmazonWSModelEntities.svc@AmazonVirtualMachineTemplates
operation.update=AmazonWSModelEntities.svc@AmazonVirtualMachineTemplates
operation.delete=AmazonWSModelEntities.svc@AmazonVirtualMachineTemplates
#Openstack Virtual Machine Templates
operation.create=OpenStackModelEntities.svc@OpenstackVirtualMachineTemplates
operation.read=OpenStackModelEntities.svc@OpenstackVirtualMachineTemplates
operation.update=OpenStackModelEntities.svc@OpenstackVirtualMachineTemplates
operation.delete=OpenStackModelEntities.svc@OpenstackVirtualMachineTemplates
#Endpoint credentials
operation.create=ManagementModelEntities.svc@ConnectionCredentials
operation.update=ManagementModelEntities.svc@ConnectionCredentials
operation.delete=ManagementModelEntities.svc@ConnectionCredentials
#Management endpoints
operation.create=ManagementModelEntities.svc@ManagementEndpoints
operation.update=ManagementModelEntities.svc@ManagementEndpoints
operation.delete=ManagementModelEntities.svc@ManagementEndpoints
#Management endpoint properties
operation.create=ManagementModelEntities.svc@ManagementEndpointProperties
operation.read=ManagementModelEntities.svc@ManagementEndpointProperties

```

```
operation.update=ManagementModelEntities.svc@ManagementEndpointProperties
operation.delete=ManagementModelEntities.svc@ManagementEndpointProperties
```

vRealize Automation 플러그인 인벤토리 사용

인벤토리 보기를 사용하여 vRealize Automation 개체에서 워크플로를 실행할 수 있습니다.

인벤토리 개체에 사용할 수 있는 워크플로를 표시하려면 **도구 > 사용자 기본 설정 > 인벤토리**로 이동한 후 **인벤토리에서 컨텍스트 메뉴 사용** 확인란을 선택합니다. 이 옵션을 사용하도록 설정하면 Orchestrator 인벤토리에서 개체를 마우스 오른쪽 버튼으로 클릭했을 때 해당 개체에 사용할 수 있는 모든 워크플로가 표시됩니다.

vRealize Automation 플러그인 관리 워크플로 사용

관리 워크플로를 사용하여 vRealize Automation 서비스, 테넌트, 승인 정책, 사용 권한, 비즈니스 그룹, 카탈로그 항목 및 고급 서비스 구성 요소를 관리할 수 있습니다.

일부 워크플로에는 vRealize Automation 호스트인 vCACCAFE:VCACHost에 대한 입력 매개 변수가 포함되어 있습니다. vRealize Automation 호스트 연결을 구성한 방법에 따라 사용자가 워크플로를 실행할 때 역할이 적용되는 방식이 결정됩니다.

- 연결을 공유 세션으로 구성한 경우 공유 세션의 사용자 계정에 워크플로를 실행하는 데 필요한 역할이 있어야 합니다.
- 연결을 사용자별 세션으로 구성한 경우 vRealize Automation 사용자 인터페이스에서와 마찬가지로 워크플로를 실행하는 각 사용자에게 필요한 역할이 있어야 합니다.

이러한 워크플로는 vRealize Orchestrator 클라이언트의 **워크플로 보기**에서 **라이브러리 > vRealize Automation > 관리** 하위 디렉토리를 통해 확인할 수 있습니다.

승인 정책 하위 디렉토리의 워크플로를 사용하여 승인 정책을 생성하고 관리할 수 있습니다.

표 4-1. 승인 정책

| 워크플로 | 설명 |
|------------|---|
| 승인 정책 활성화 | 승인 정책을 활성화합니다. 승인 정책을 활성화하면 읽기 전용 상태가 됩니다. |
| 승인 수준 추가 | 항상 필요한 승인 수준을 승인에 추가합니다. 승인자에 대한 특정 사용자 및 그룹을 선택해야 합니다. |
| 승인 정책 복사 | 승인 정책을 복사합니다. |
| 승인 정책 생성 | 수준이 없거나 승인자가 없는 초안 승인 정책을 생성합니다. 승인 수준을 생성하고 정책의 승인자를 지정하려면 승인 수준 추가 워크플로를 실행합니다. |
| 승인 정책 비활성화 | 승인 정책을 비활성화합니다. 또한 승인 정책과 연결된 모든 기존 사용 권한을 삭제할 수 있습니다. |
| 승인 정책 삭제 | 초안 상태의 승인 정책을 삭제합니다. 활성 승인 정책은 읽기 전용입니다. |

비즈니스 그룹 하위 디렉토리의 워크플로를 사용하여 비즈니스 그룹 및 비즈니스 그룹 사용자 지정 속성을 생성하고 관리할 수 있습니다.

표 4-2. 비즈니스 그룹

| 워크플로 | 설명 |
|----------------|--|
| 사용자 지정 속성 추가 | 비즈니스 그룹에 사용자 지정 속성을 추가합니다. |
| 비즈니스 그룹 생성 | 비즈니스 그룹을 생성합니다. |
| 비즈니스 그룹 삭제 | 비즈니스 그룹을 삭제합니다. |
| 사용자 지정 속성 삭제 | 비즈니스 그룹에서 사용자 지정 속성을 제거합니다. |
| 비즈니스 그룹 업데이트 | 기본 시스템 접두사, 활성 디렉토리 컨테이너, 사용자 역할 등 비즈니스 그룹에 대한 세부 정보를 업데이트합니다. |
| 사용자 지정 속성 업데이트 | 비즈니스 그룹에 대한 사용자 지정 속성을 업데이트합니다. |

관리 하위 디렉토리에는 vRealize Automation 7.0 이전 버전에서 작동하는 **비즈니스 그룹(더 이상 사용되지 않음)** 하위 디렉토리가 있습니다. 기본 폴더에 있는 동일한 이름의 워크플로를 사용하십시오.

카탈로그 항목 하위 디렉토리의 워크플로를 사용하여 카탈로그 항목을 관리할 수 있습니다.

표 4-3. 카탈로그 항목

| 워크플로 | 설명 |
|-----------------|---|
| 카탈로그 항목 활성화 | 카탈로그 항목을 활성화합니다. 카탈로그 항목을 활성화하고 서비스에 할당해야 사용자가 해당 항목을 요청할 수 있습니다. |
| 서비스에 카탈로그 항목 추가 | 서비스에 카탈로그 항목을 추가합니다. 카탈로그 항목을 활성화하고 서비스에 할당해야 사용자가 해당 항목을 요청할 수 있습니다. |
| 카탈로그 항목 비활성화 | 사용자가 요청할 수 없도록 카탈로그 항목을 비활성화하고 서비스 카탈로그에서 제거합니다. |

복합 Blueprint 하위 디렉토리의 워크플로를 사용하여 디자인 캔버스에서 생성한 복합 Blueprint를 관리할 수 있습니다.

표 4-4. 복합 Blueprint

| 워크플로 | 설명 |
|--------------------|--|
| 복합 Blueprint 삭제 | 디자인 Blueprint 목록에서 게시되지 않은 Blueprint를 삭제합니다. |
| 복합 Blueprint 가져오기 | YAML 파일에서 복합 Blueprint를 가져옵니다. |
| 복합 Blueprint 게시 | 초안 상태의 복합 Blueprint를 게시합니다. |
| 복합 Blueprint 게시 취소 | 게시된 복합 Blueprint의 게시를 취소합니다. |

컨텐츠 하위 디렉토리 워크플로는 더 이상 사용되지 않습니다. Cloud Client를 사용하여 가져오기 및 내보내기 작업을 수행할 수 있습니다. Cloud Client 다운로드 및 설명서는 <https://developercenter.vmware.com/tool/cloudclient>에서 제공됩니다.

표 4-5. 콘텐츠

| 워크플로 | 설명 |
|--------------------------|---|
| 컨텐츠 내보내기(더 이상 사용되지 않음) | Cloud Client를 사용하여 가져오기 및 내보내기 작업을 수행할 수 있습니다. Cloud Client 다운로드 및 설명서는 https://developercenter.vmware.com/tool/cloudclient 에서 제공됩니다. |
| 컨텐츠 가져오기(더 이상 사용되지 않음) | Cloud Client를 사용하여 가져오기 및 내보내기 작업을 수행할 수 있습니다. Cloud Client 다운로드 및 설명서는 https://developercenter.vmware.com/tool/cloudclient 에서 제공됩니다. |
| 컨텐츠 전송(더 이상 사용되지 않음) | Cloud Client를 사용하여 가져오기 및 내보내기 작업을 수행할 수 있습니다. Cloud Client 다운로드 및 설명서는 https://developercenter.vmware.com/tool/cloudclient 에서 제공됩니다. |
| 컨텐츠 유효성 검사(더 이상 사용되지 않음) | Cloud Client를 사용하여 가져오기 및 내보내기 작업을 수행할 수 있습니다. Cloud Client 다운로드 및 설명서는 https://developercenter.vmware.com/tool/cloudclient 에서 제공됩니다. |

사용 권한 하위 디렉토리의 워크플로를 사용하여 사용 권한을 생성하고 관리할 수 있습니다.

표 4-6. 사용 권한

| 워크플로 | 설명 |
|-------------------------|--|
| 사용 권한 활성화 | 사용 권한을 활성화합니다. |
| 사용 권한에 카탈로그 항목 할당 | 하나 이상의 카탈로그 항목을 사용 권한에 할당합니다. 이 워크플로를 사용하여 승인 정책을 할당할 수도 있습니다. |
| 사용 권한에 즉시 실행 작업 할당 | 하나 이상의 즉시 실행 작업을 사용 권한에 할당합니다. 즉시 실행 작업은 요청을 생성하지 않습니다. |
| 사용 권한에 리소스 작업 할당 | 하나 이상의 리소스 작업을 사용 권한에 할당합니다. 이 워크플로를 사용하여 승인 정책을 할당할 수도 있습니다. |
| 사용 권한에 서비스 할당 | 하나 이상의 서비스를 사용 권한에 할당합니다. 이 워크플로를 사용하여 승인 정책을 할당할 수도 있습니다. |
| 사용 권한에 사용자 및 그룹 할당 | 하나 이상의 사용자 또는 그룹을 사용 권한에 할당합니다. |
| 사용 권한 생성(더 이상 사용되지 않음) | 사용 권한을 생성합니다. 하위 테넌트에 대한 사용 권한 생성을 사용하십시오. |
| 하위 테넌트에 대한 사용 권한 생성 | 사용 권한을 생성합니다. |
| 사용 권한 비활성화 | 사용 권한을 비활성화합니다. |
| 사용 권한에서 사용자 및 그룹의 할당 취소 | 사용 권한에 대한 사용자 목록에서 사용자 및 그룹을 제거합니다. |

속성 하위 디렉토리의 워크플로를 사용하여 속성 정의 및 속성 그룹을 관리할 수 있습니다. vRealize Automation 속성과의 충돌을 피하려면 모든 사용자 지정 속성 이름에 회사 또는 기능 이름과 같은 접두사 뒤에 점을 추가하여 사용하십시오.

표 4-7. 속성 정의

| 워크플로 | 설명 |
|----------|-------------------|
| 속성 정의 생성 | 사용자 지정 속성을 생성합니다. |
| 속성 정의 삭제 | 사용자 지정 속성을 삭제합니다. |

속성 그룹은 속성 정의의 컬렉션입니다.

표 4-8. 속성 그룹

| 워크플로 | 설명 |
|--------------|---------------------------------------|
| 그룹에 속성 추가 | 정의된 사용자 지정 속성을 그룹에 추가합니다. |
| 속성 그룹 생성 | 정의된 사용자 지정 속성을 추가할 수 있는 속성 그룹을 생성합니다. |
| 속성 그룹 삭제 | 속성 그룹을 삭제합니다. |
| 그룹에서 속성 제거 | 속성 그룹에서 정의된 사용자 지정 속성을 제거합니다. |
| 속성 그룹 업데이트 | 속성 그룹의 이름 또는 설명을 수정합니다. |
| 그룹에서 속성 업데이트 | 속성 그룹의 속성에 대한 이름, 값 및 동작을 수정합니다. |

서비스 하위 디렉토리의 워크플로를 사용하여 서비스를 관리할 수 있습니다.

표 4-9. 서비스

| 워크플로 | 설명 |
|-----------------|-----------------------------|
| 서비스 활성화 | 서비스를 활성화합니다. |
| 서비스에 카탈로그 항목 할당 | 하나 이상의 카탈로그 항목을 서비스에 할당합니다. |
| 서비스 복사 | 서비스를 복사합니다. |
| 서비스 생성 | 서비스를 생성합니다. |
| 서비스 비활성화 | 서비스를 비활성화합니다. |
| 서비스 삭제 | 서비스를 삭제합니다. |

테넌트 하위 디렉토리의 워크플로를 사용하여 테넌트를 생성하고 관리할 수 있습니다.

ID 저장소 워크플로는 더 이상 사용되지 않습니다. Directories Management API용으로 vRealize Automation에서 변경된 대체 워크플로가 사용됩니다.

표 4-10. 테넌트

| 워크플로 | 설명 |
|------------------------------|---|
| 관리자 추가 | 하나 이상의 테넌트 관리자 및 인프라 관리자를 테넌트에 추가합니다. |
| 테넌트에 ID 저장소 추가 | vRealize Automation 호스트의 테넌트에 ID 저장소를 추가합니다. 이 워크플로는 테넌트를 구성하는 시스템 관리자만 실행할 수 있습니다. |
| 테넌트에 ID 저장소 추가(더 이상 사용되지 않음) | 테넌트에 ID 저장소 추가 워크플로를 사용하십시오. |

표 4-10. 테넌트 (계속)

| 워크플로 | 설명 |
|--|---|
| vCAC 호스트에 ID 저장소 추가 | vRealize Automation 호스트로 구성된 테넌트에 ID 저장소를 추가합니다. 이 워크플로는 테넌트에 대한 ID 저장소를 구성하는 테넌트 관리자만 실행할 수 있습니다. |
| vCAC 호스트에 ID 저장소 추가(더 이상 사용되지 않음) | vCAC 호스트에 ID 저장소 추가 워크플로를 사용하십시오. |
| 테넌트 생성 | 테넌트를 생성합니다. 시스템 관리자 자격 증명으로 추가된 vRealize Automation 호스트를 선택해야 합니다. |
| 테넌트에서 ID 저장소 삭제 | vRealize Automation 호스트의 테넌트에서 ID 저장소를 삭제합니다. 이 워크플로는 테넌트를 구성하는 시스템 관리자만 실행할 수 있습니다. |
| vCAC 호스트에서 ID 저장소 삭제 | vRealize Automation 호스트로 구성된 테넌트에서 ID 저장소를 삭제합니다. 이 워크플로는 테넌트에 대한 ID 저장소를 구성하는 테넌트 관리자만 실행할 수 있습니다. |
| 테넌트 삭제 | 테넌트를 삭제합니다. |
| 관리자 제거 | 하나 이상의 테넌트 관리자 및 인프라 관리자를 테넌트에서 제거합니다. |
| 테넌트에 대한 ID 저장소 업데이트 | vRealize Automation 호스트의 테넌트에 대한 기존 ID 저장소를 업데이트합니다. 이 워크플로는 테넌트를 구성하는 시스템 관리자만 실행할 수 있습니다. |
| 테넌트에 대한 ID 저장소 업데이트(더 이상 사용되지 않음) | 테넌트에 대한 ID 저장소 업데이트 워크플로를 사용하십시오. |
| vCAC 호스트에 대한 ID 저장소 업데이트 | vRealize Automation 호스트로 구성된 테넌트에 대한 ID 저장소를 업데이트합니다. 이 워크플로는 테넌트에 대한 ID 저장소를 구성하는 테넌트 관리자만 실행할 수 있습니다. |
| vCAC 호스트에 대한 ID 저장소 업데이트(더 이상 사용되지 않음) | vCAC 호스트에 대한 ID 저장소 업데이트 워크플로를 사용하십시오. |
| 테넌트 업데이트 | 기존 테넌트의 이름, 설명 및 연락처 이메일 주소를 업데이트합니다. |

워크플로 구독 하위 디렉토리의 워크플로를 사용하여 이벤트 워크플로 구독을 관리할 수 있습니다.

표 4-11. 워크플로 구독

| 워크플로 | 설명 |
|------------------|---|
| 워크플로 구독 삭제 | 게시되지 않은 워크플로 구독을 삭제합니다. 이 워크플로는 시스템 및 테넌트 워크플로 구독에 적용됩니다. |
| 시스템 워크플로 구독 내보내기 | 시스템 워크플로 구독을 내보내고 JSON 형식의 vRealize Orchestrator 리소스 요소로 저장합니다. 시스템 워크플로 구독은 시스템 이벤트 및 모든 테넌트의 이벤트에 반응하는 특수한 워크플로 구독입니다. |
| 테넌트 워크플로 구독 내보내기 | 테넌트 워크플로 구독을 내보내고 JSON 형식의 리소스 요소로 저장합니다. 테넌트 관련 워크플로를 실행하는 특수한 워크플로 구독입니다. |

표 4-11. 워크플로 구독 (계속)

| 워크플로 | 설명 |
|------------------|---|
| 시스템 워크플로 구독 가져오기 | JSON 파일에서 시스템 워크플로 구독을 가져옵니다. 시스템 워크플로 구독은 시스템 이벤트에 대해 트리거되며 여러 테넌트에 걸쳐 적용될 수 있습니다. |
| 테넌트 워크플로 구독 가져오기 | JSON 파일에서 내보낸 워크플로 구독을 가져옵니다. 이러한 워크플로 구독은 테넌트와 관련이 있습니다. |
| 워크플로 구독 게시 | 초안 또는 게시되지 않은 상태의 워크플로 구독을 게시합니다. 이 워크플로는 시스템 및 테넌트 워크플로 구독에 적용됩니다. |
| 시스템 워크플로 구독 등록 | 시간 초과 및 우선 순위 값을 포함하여 시스템 워크플로 구독을 생성합니다. |
| 테넌트 워크플로 구독 등록 | 시간 초과 및 우선 순위 값을 포함하여 테넌트 관련 워크플로 구독을 생성합니다. |
| 워크플로 구독 게시 취소 | 게시된 워크플로 구독의 게시를 취소합니다. 이 워크플로는 시스템 및 테넌트 워크플로 구독에 적용됩니다. |
| 워크플로 구독 업데이트 | 이름, 설명, vRealize Orchestrator 워크플로, 구독 조건, 시간 초과 값, 상태 값 및 우선 순위 값을 변경합니다. 이벤트 항목 또는 차단 상태는 업데이트할 수 없습니다. |

XaaS 사용자 지정 리소스 하위 디렉토리의 워크플로를 사용하여 XaaS 사용자 지정 리소스를 생성하고 삭제할 수 있습니다.

표 4-12. XaaS 사용자 지정 리소스

| 워크플로 | 설명 |
|---------------|--------------------|
| 사용자 지정 리소스 생성 | 사용자 지정 리소스를 생성합니다. |
| 사용자 지정 리소스 삭제 | 사용자 지정 리소스를 제거합니다. |

XaaS 리소스 작업 하위 디렉토리의 워크플로를 사용하여 XaaS 리소스 작업을 생성하고 관리할 수 있습니다.

표 4-13. XaaS 리소스 작업

| 워크플로 | 설명 |
|--------------|-----------------------|
| 리소스 작업 복제 | 기존 리소스 작업의 사본을 생성합니다. |
| 리소스 작업 생성 | 리소스 작업을 생성합니다. |
| 리소스 작업 삭제 | 리소스 작업을 삭제합니다. |
| 리소스 작업 게시 | 리소스 작업을 게시합니다. |
| 리소스 작업 게시 취소 | 리소스 작업의 게시를 취소합니다. |

XaaS 리소스 매핑 하위 디렉토리의 워크플로를 사용하여 비 XaaS 리소스에 대한 XaaS 매핑을 생성하고 관리할 수 있습니다.

표 4-14. XaaS 리소스 매핑

| 워크플로 | 설명 |
|-----------|---|
| 리소스 매핑 생성 | 카탈로그 리소스 유형을 vRealize Orchestrator 유형에 매핑합니다. |
| 리소스 매핑 삭제 | 리소스 매핑을 삭제합니다. |
| 대상 기준 설정 | 리소스 매핑의 가용성을 결정하는 조건을 지정합니다. |

XaaS 서버 구성 하위 디렉토리의 워크플로를 사용하여 대상 Orchestrator 인스턴스를 관리할 수 있습니다.

표 4-15. XaaS 서버 구성

| 워크플로 | 설명 |
|---------------------------|---|
| Orchestrator 서버 구성 업데이트 | 포트, 호스트, 사용자 이름 및 암호 등 서버 설정을 수정합니다. |
| Orchestrator 서버 구성 유효성 검사 | vRealize Orchestrator 설정이 유효한지 확인합니다. 구성이 유효하면 TRUE 값이 반환되고, 구성이 유효하지 않으면 FALSE 값이 반환됩니다. |

XaaS 서비스 Blueprint 하위 디렉토리의 워크플로를 사용하여 XaaS Blueprint를 생성하고 관리할 수 있습니다.

표 4-16. XaaS Blueprint

| 워크플로 | 설명 |
|---------------------|---------------------------|
| 서비스 Blueprint 복제 | 서비스 Blueprint의 사본을 생성합니다. |
| 서비스 Blueprint 생성 | 서비스 Blueprint를 생성합니다. |
| 서비스 Blueprint 삭제 | 서비스 Blueprint를 삭제합니다. |
| 서비스 Blueprint 게시 | 서비스 Blueprint를 게시합니다. |
| 서비스 Blueprint 게시 취소 | 서비스 Blueprint의 게시를 취소합니다. |

vRealize Automation 플러그인 인프라 관리 워크플로 사용

인프라 관리 워크플로를 사용하여 기본 작업을 실행할 수 있습니다. 확장성 패키지를 사용하여 vRealize Orchestrator 워크플로를 프로비저닝 프로세스의 일부로 호출하거나 사용자 지정 작업 메뉴를 통해 호출하는 기능으로 vRealize Automation을 사용자 지정할 수 있습니다.

인프라 관리 워크플로는 Orchestrator 클라이언트의 **워크플로** 보기에서 플러그인 라이브러리의 **인프라 관리** 하위 디렉토리를 통해 확인할 수 있습니다.

인프라 관리 워크플로를 사용하여 가상 시스템을 프로비저닝하고 기본 생성, 읽기, 업데이트 또는 삭제 작업을 실행할 수 있습니다.

표 4-17. 인프라 관리

| 워크플로 이름 | 설명 |
|---|--|
| 가상 시스템 상태 변경 대기 | <p>가상 시스템 집합의 상태 변경을 대기합니다. 모든 가상 시스템이 성공 상태에 있는 경우 트리거가 호출되고 워크플로가 성공적으로 종료됩니다. 지정된 가상 시스템 중 하나 이상이 실패 상태이거나 존재하지 않는 경우에는 워크플로가 실패합니다. 다음 옵션에서 선택하여 성공 및 실패 상태를 입력해야 합니다.</p> <ul style="list-style-type: none"> ■ Requested ■ AwaitingApproval ■ RegisterMachine ■ BuildingMachine ■ AddingDisks ■ MachineProvisioned ■ MachineActivated ■ InstallTools(VMware에만 해당) ■ On ■ Off ■ TurningOn ■ TurningOff ■ ShuttingDown ■ Suspending ■ Resetting ■ Rebooting ■ Expired ■ DeactivateMachine ■ UnprovisionMachine ■ Disposing ■ Finalized |
| IaaS 모델 엔티티 생성 | 지정된 vRealize Automation 모델의 엔티티를 생성하고 유지합니다. |
| IaaS 모델 엔티티 삭제 | 지정된 vRealize Automation 모델 엔티티를 삭제합니다. |
| 프로비저닝 이후 작업 호출(더 이상 사용되지 않음) | 리소스 작업 요청 워크플로를 사용하십시오. |
| Blueprint에서 가상 시스템 프로비저닝(vRealize Automation 7.0에서 제거됨) | 카탈로그 항목 요청 또는 프로비저닝 요청과 함께 카탈로그 항목 요청으로 대체되었습니다. |
| 사용자 필터를 사용하여 IaaS 엔티티 읽기 | 사용자 필터를 사용하여 vRealize Automation 엔티티 목록을 읽습니다. 필터를 지정하지 않은 경우 모든 엔티티가 결과로 반환됩니다. |
| 시스템 쿼리를 사용하여 IaaS 엔티티 읽기 | OData 시스템 필터를 사용하여 vRealize Automation 엔티티 목록을 읽습니다. 시스템 필터는 OData URI 규칙에 적용됩니다. |
| IaaS 모델 엔티티 읽기 | 해당 ID를 사용하여 vRealize Automation 모델 엔티티를 읽습니다. |
| IaaS 모델 엔티티 업데이트 | 해당 ID를 사용하여 vRealize Automation 모델 엔티티를 업데이트합니다. |

확장성 하위 디렉토리의 워크플로를 사용하여 vRealize Orchestrator 워크플로를 프로비저닝 프로세스의 일부로 호출하거나 사용자 지정 작업 메뉴를 통해 호출하는 기능으로 vRealize Automation을 사용자 지정할 수 있습니다.

이 하위 디렉토리에는 IaaS 자격 증명, 끝점, 엔터프라이즈 그룹, 시스템 접두사 및 기타 엔티티를 관리하는 워크플로도 포함되어 있습니다.

표 4-18. 확장성

| 워크플로 이름 | 설명 |
|---|--|
| vCO 사용자 지정 설치 | 사용자 지정된 상태 변경 워크플로와 메뉴 작업 워크플로를 포함하여 Orchestrator 사용자 지정을 설치합니다. |
| vCO 사용자 지정 제거 | 사용자 지정된 상태 변경 워크플로와 메뉴 작업 워크플로를 포함하여 Orchestrator 사용자 지정을 제거합니다. |
| IaaS 가상 시스템의 예약 변경 | 예약, 비즈니스 그룹, 관리되는 가상 시스템과 같은 특성을 변경합니다. |
| IaaS 가상 시스템 가져오기(더 이상 사용되지 않음) | Cloud Client를 사용하십시오. Cloud Client 다운로드 및 설명서는 https://developercenter.vmware.com/tool/cloudclient 에서 제공됩니다. |
| vCenter 가상 시스템 가져오기(더 이상 사용되지 않음) | Cloud Client를 사용하십시오. Cloud Client 다운로드 및 설명서는 https://developercenter.vmware.com/tool/cloudclient 에서 제공됩니다. |
| 가상 시스템 등록 취소(vRealize Automation 7.0에서 제거됨) | 대체 워크플로가 제공되지 않습니다. |
| Blueprint 및 해당 가상 시스템에 메뉴 작업 할당(더 이상 사용되지 않음) | 가상 시스템에서 메뉴 작업을 추가하거나 업데이트합니다. 사용되는 대체 워크플로에는 사용 권한에 리소스 작업 할당 및 복합 Blueprint 가져오기가 포함됩니다. |
| 가상 시스템에 메뉴 작업 할당(더 이상 사용되지 않음) | 해당 ID를 사용하여 vRealize Automation 모델 엔티티를 업데이트합니다. 사용되는 대체 워크플로에는 사용 권한에 리소스 작업 할당 및 복합 Blueprint 가져오기가 포함됩니다. |
| Blueprint 및 해당 가상 시스템에 상태 변경 워크플로 할당(더 이상 사용되지 않음) | vRealize Automation에서 이벤트 브로커 구독으로 대체되었습니다. |
| 메뉴 작업 사용자 지정(vRealize Automation 7.0에서 제거됨) | 대체 워크플로가 제공되지 않습니다. |
| Blueprint 및 해당 가상 시스템에서 메뉴 작업 제거(vRealize Automation 7.0에서 제거됨) | 대체 워크플로가 제공되지 않습니다. |
| Blueprint 및 해당 가상 시스템에서 상태 변경 워크플로 제거 | Blueprint 및 해당 가상 시스템에서 상태 변경 워크플로를 제거합니다. |

vRealize Automation IaaS 모델 엔티티 생성

워크플로를 실행하여 사용자에게 대한 가상 시스템 참조와 같은 간단하거나 복잡한 vRealize Automation IaaS 엔티티를 만들 수 있습니다.

절차

- 1 Orchestrator 클라이언트의 드롭다운 메뉴에서 **실행** 또는 **설계**를 선택합니다.

- 2 워크플로 보기를 클릭합니다.
- 3 라이브러리 > vRealize Automation > 인프라 관리를 확장합니다.
- 4 IaaS 모델 엔티티 생성 워크플로를 마우스 오른쪽 버튼으로 클릭하고 워크플로 시작을 선택합니다.
- 5 vRealize Automation 호스트를 선택합니다.
- 6 모델 이름 텍스트 상자에 모델의 이름을 입력합니다.
- 7 엔티티 집합 이름 텍스트 상자에 엔티티 집합의 이름을 입력합니다.
스크립팅 또는 REST API를 사용하여 단순 속성, 복잡한 속성의 링크 및 HTTP 헤더 속성을 설정합니다.
- 8 제출을 클릭하여 워크플로를 실행합니다.

vRealize Automation IaaS 모델 엔티티 읽기

워크플로를 실행하여 vRealize Automation IaaS 모델 엔티티를 읽을 수 있습니다.

절차

- 1 Orchestrator 클라이언트의 드롭다운 메뉴에서 실행 또는 설계를 선택합니다.
- 2 워크플로 보기를 클릭합니다.
- 3 라이브러리 > vRealize Automation > 인프라 관리를 확장합니다.
- 4 IaaS 모델 엔티티 읽기를 마우스 오른쪽 버튼으로 클릭하고 워크플로 시작을 선택합니다.
- 5 vRealize Automation 호스트를 선택합니다.
- 6 모델 이름 텍스트 상자에 모델의 이름을 입력합니다.
- 7 엔티티 집합 이름 텍스트 상자에 엔티티 집합의 이름을 입력합니다.
스크립팅 또는 REST API를 사용하여 HTTP 헤더 속성을 설정합니다.
- 8 제출을 클릭하여 워크플로를 실행합니다.

vRealize Automation 플러그인 요청 워크플로 사용

요청 워크플로를 사용하여 카탈로그 항목 및 리소스 작업을 요청하고 작업 항목을 완료하거나 취소할 수 있습니다.

작업 항목에는 사용자 입력 또는 작업이 필요합니다. 예를 들어 워크플로 상호 작용, 승인 작업, 회수 요청에 대한 응답 등이 필요합니다.

이러한 워크플로는 vRealize Orchestrator 클라이언트의 워크플로 보기에서 플러그인 라이브러리의 요청 하위 디렉토리를 통해 액세스할 수 있습니다.

| 워크플로 | 설명 |
|----------|---|
| 작업 항목 취소 | 활성 작업 항목을 취소합니다. 이 워크플로는 시스템 관리자만 사용할 수 있습니다. |
| 작업 항목 완료 | 제공된 사용자 입력에 따라 작업 항목을 완료합니다. |

| 워크플로 | 설명 |
|------------------------|---|
| 카탈로그 항목 요청 | 워크플로를 실행하는 사용자에게 대한 카탈로그 항목을 요청합니다. 복합 Blueprint를 요청하는 워크플로가 필요한 경우 프로비저닝 요청과 함께 카탈로그 요청 워크플로를 사용합니다. |
| 사용자를 대신해 카탈로그 항목 요청 | 사용자를 대신해 카탈로그 항목에 대한 요청을 전송합니다. 대리자와 해당 사용자 모두가 권한이 있는 카탈로그 항목에 대해서만 이 워크플로를 사용할 수 있습니다. |
| 프로비저닝 요청과 함께 카탈로그 요청 | 워크플로를 실행하는 사용자에게 대한 카탈로그 항목으로 복합 Blueprint를 요청합니다. 요청에 사용자 지정 입력을 제공하는 경우 워크플로를 사용자 지정해야 합니다. 복합 Blueprint에 이 워크플로를 사용합니다. |
| 리소스 작업 요청 | 워크플로를 실행하는 사용자가 소유한 카탈로그 항목에 대한 리소스 작업을 요청합니다. |
| 사용자를 대신해 리소스 작업 요청 | 사용자를 대신해 리소스 작업에 대한 요청을 전송합니다. 대리자와 해당 사용자 모두가 권한이 있는 리소스 작업에 대해서만 이 워크플로를 사용할 수 있습니다. |
| 요청 템플릿을 사용하여 리소스 작업 요청 | <p>복잡한 매개 변수가 포함된 리소스 작업을 요청합니다. 워크플로를 복제한 다음 작업에 맞게 사용자 지정하는 것이 가장 좋습니다. 이 워크플로를 사용하여 요청 양식에 표시하지 않으려는 숨겨진 매개 변수 또는 복잡한 매개 변수를 전달할 수 있습니다. 이 워크플로의 주된 응용 분야 중 하나는 IaaS 가상 시스템 재구성 작업을 사용자 지정하는 것입니다.</p> <p>가상 시스템에서 재구성 작업을 생성하려면, 워크플로 사본을 생성한 다음 스크립트를 수정해야 합니다. vRealize Orchestrator에 나타난 매개 변수를 구성하고 <code>Cafe.Shim.VirtualMachine.Reconfigure.Requestor</code> 매개 변수를 설정합니다. 이 매개 변수는 로깅에 사용되며 비어 있지 않아야 합니다. 다음 예를 참조하십시오.</p> <pre> var requestTemplate = vCACAFERequestsHelper.getRequestForResourceAction(operation) var jsonData = vCACAFERequestsHelper.getResourceActionRequestData(requestTemplate); var json = JSON.parse(jsonData); //Change cpu example json.cpu = 2; //This is a property needed for the Reconfigure IaaS operation: json["Cafe.Shim.VirtualMachine.Reconfigure.Requestor"] = 1; vCACAFERequestsHelper.setResourceActionRequestData(requestTemplate, JSON.stringify(json)); request = System.getModule("com.vmware.library.vcacafe.request").requestResourceActionWithRequestTemplate(operation, requestTemplate); </pre> |
| 카탈로그 항목 요청 대기 | 카탈로그 항목 요청이 완료될 때까지 대기합니다. |
| 리소스 작업 요청 대기 | 리소스 작업 요청이 완료될 때까지 대기합니다. |
| 작업 항목 대기 | 작업 항목이 완료될 때까지 대기합니다. |

vRealize Automation 플러그인 샘플 워크플로 사용

샘플 워크플로를 예제로 사용하거나, 사용자 지정 워크플로를 생성하는 시작점으로 사용할 수 있습니다.

이러한 워크플로는 vRealize Orchestrator 클라이언트의 **워크플로** 보기에서 플러그인 라이브러리의 **샘플** 하위 디렉토리를 통해 확인할 수 있습니다.

| 워크플로 이름 | 설명 |
|-----------------------------|--|
| 사용 권한 생성 | 권한 부여 클라이언트 및 사용 권한 서비스와 상호 작용하여 vRealize Automation에서 사용 권한을 생성하는 샘플 스크립트를 제공합니다. |
| 테넌트 생성 | 기본 테넌트와 동일한 vRealize Automation 호스트 및 Active Directory 구성으로 테넌트를 생성합니다. 이 워크플로를 실행하려면 시스템 관리자 자격 증명으로 추가된 vRealize Automation 호스트를 선택합니다. 워크플로를 실행하기 전에 Active Directory 설정을 변경할 수 있습니다. |
| 카탈로그 항목 나열 | 선택한 테넌트의 카탈로그 항목 목록을 반환합니다. |
| 카탈로그 항목 프로비저닝 요청을 JSON으로 인쇄 | 카탈로그 항목에 대한 기본 요청 양식을 검색하여 콘솔 로그에 JSON 형식으로 추가합니다. 이 데이터를 사용하여 프로비저닝 요청을 사용자 지정할 수 있습니다. 이 정보를 사용하여 프로비저닝 요청과 함께 카탈로그 항목 요청 워크플로를 수정할 수 있습니다. |

vRealize Automation 플러그인 API 액세스

Orchestrator에서는 vRealize Automation 플러그인 API를 검색할 수 있으며 스크립팅된 요소에서 사용할 수 있는 가능한 JavaScript 개체에 대한 설명서를 볼 수 있는 API 탐색기를 제공합니다.

업데이트된 vRealize Automation API 설명서는 <https://www.vmware.com/support/pubs/vcac-pubs.html>을 참조하십시오.

절차

- 1 Orchestrator 클라이언트에 관리자로 로그인합니다.
- 2 **도구 > API 탐색기**를 선택합니다.
- 3 왼쪽 창에서 **VCAC** 및 **VCACCAFE** 모듈을 두 번 클릭하여 vRealize Automation 플러그인 API 개체의 계층형 목록을 확장합니다.

다음에 수행할 작업

API 요소에서 코드를 복사하여 스크립팅 상자에 붙여 넣을 수 있습니다. API 스크립팅에 대한 자세한 내용은 "VMware vRealize Orchestrator를 사용한 개발"을 참조하십시오.

개발 모범 사례에 대한 자세한 내용은 [vRealize Orchestrator 설명서](#)를 참조하십시오.

예제 vRealize Automation 플러그인 스크립트

제공된 JavaScript 예제를 잘라내고, 붙여 넣고, 편집하여 vRealize Automation 작업을 자동화할 수 있는 사용자 지정 스크립트를 개발할 수 있습니다.

CRUD 인프라 관리 작업 예제 스크립트

JavaScript 예제를 잘라내고, 붙여 넣고, 편집하여 CRUD vRealize Automation 작업에 대한 스크립트를 작성할 수 있습니다.

vRealize Orchestrator의 스크립팅에 대한 자세한 내용은 "VMware vRealize Orchestrator를 사용한 개발"을 참조하십시오.

예제: vRealize Automation 모델 엔티티 생성

이 예제 스크립트는 다음 작업을 수행합니다.

- 1 모델 이름 및 엔티티 집합 이름을 정의합니다.
- 2 호스트 접두사의 속성을 정의합니다.
- 3 호스트 접두사 엔티티를 저장합니다.
- 4 프로비저닝 그룹의 속성을 정의합니다.
- 5 프로비저닝 그룹을 링크로 정의합니다.
- 6 프로비저닝 그룹 엔티티를 호스트 이름 접두사와 연결하여 저장합니다.

표 4-19. 입력 변수

| 변수 | 유형 |
|------|---------------|
| host | vCAC:VcacHost |

```
var modelName = 'ManagementModelEntities.svc';
var entitySetName = 'HostNamePrefixes';
var links = null;
var headers = null;
//Create properties for prefix entity
var prefixInputProperties = {
    MachinePrefix:'test-prefix',
    NextMachineNo:1,
    MachineNumberLength:3
};
//Save the prefix
var prefixEntity = vCACEntityManager
    .createModelEntity(host.id, modelName, entitySetName, prefixInputProperties, links, headers);
entitySetName = 'ProvisioningGroups';
//Create properties for the provisioning group entity
inputProperties = {
    GroupName:'TestGroupName',
    GroupDescription:'This group was generated with a vCO workflow',
    AdministratorEmail:'test@test.com',
    AdContainer:'AD',
    IsTestGroup:false,
    Flags:2,
    GroupType:1};
//Add a reference to the newly created prefix entity
links = {
    HostNamePrefix:prefixEntity
};
//Save the provisioning group
var entity = vCACEntityManager.createModelEntity(host.id, modelName, entitySetName, inputProperties,
links, headers);
```

예제: vRealize Automation 모델 엔티티 업데이트

이 예제 스크립트는 다음 작업을 수행합니다.

- 1 제공된 엔티티에서 호스트 ID를 가져옵니다.
- 2 제공된 엔티티에서 모델 이름을 가져옵니다.
- 3 제공된 엔티티에서 엔티티 집합 이름을 가져옵니다.
- 4 제공된 엔티티에서 엔티티 ID를 가져옵니다.
- 5 업데이트할 속성 집합을 정의합니다.
- 6 엔티티를 업데이트하는 작업을 시작합니다.

표 4-20. 입력 변수

| 변수 | 유형 |
|--------------------|-------------|
| entity | vCAC:Entity |
| updatedDescription | 문자열 |

```
var hostId = entity.hostId;
var modelName = entity.modelName;
var entitySetName = entity.entitySetName;
var entityIdString = entity.keyString;
var links = null;
var headers = null;
var updateProperties = new Properties();
updateProperties.put("UserNameDescription", updatedDescription);
//Update the user description
System.getModule("com.vmware.library.vcac")
    .updateVCACEntity(hostId, modelName, entitySetName, entityIdString, updateProperties, links,
        headers);
```

예제: vRealize Automation 모델 엔티티 읽기

이 예제 스크립트는 다음 작업을 수행합니다.

- 1 모델 이름 및 엔티티 집합 이름을 정의합니다.
- 2 속성 개체와 함께 Blueprint ID를 정의합니다.
- 3 엔티티를 읽습니다.

표 4-21. 입력 변수

| 변수 | 유형 |
|-------------|---------------|
| host | vCAC:VcacHost |
| blueprintID | 문자열 |

```
var modelName = 'ManagementModelEntities.svc';
var entitySetName = 'VirtualMachineTemplates';
var links = null;
```

```
var headers = null;
//Create properties for the prefix entity
var blueprintId = {
    VirtualMachineTemplateID:blueprintId,
};
//Read the blueprint
var entity = vCACEntityManager
    .readModelEntity(host.id, modelName, entitySetName, blueprintId, headers);
```

예제: vRealize Automation 모델 엔티티 삭제

이 예제 스크립트는 다음 작업을 수행합니다.

- 1 제공된 엔티티에서 호스트 ID를 가져옵니다.
- 2 제공된 엔티티에서 모델 이름을 가져옵니다.
- 3 제공된 엔티티에서 엔티티 집합 이름을 가져옵니다.
- 4 제공된 엔티티에서 엔티티 ID를 가져옵니다.
- 5 엔티티를 삭제하는 작업을 시작합니다.

표 4-22. 입력 변수

| 변수 | 유형 |
|--------|-------------|
| entity | vCAC:Entity |

```
var hostId = entity.hostId;
var modelName = entity.modelName;
var entitySetName = entity.entitySetName;
var entityKeyString = entity.keyString;
var headers = null;
//Delete the entity
System.getModule("com.vmware.library.vcac")
    .deleteVCACEntity(hostId, modelName, entitySetName, entityKeyString, headers);
```

예제: 사용자 지정 필터를 사용하여 vRealize Automation 엔티티 읽기

이 예제 스크립트는 다음 작업을 수행합니다.

- 1 모델 이름 및 엔티티 집합 이름을 정의합니다.
- 2 엔티티를 필터링할 기준 속성을 정의합니다.
- 3 엔티티 목록을 읽습니다.

표 4-23. 입력 변수

| 변수 | 유형 |
|--------------|---------------|
| host | vCAC:VcacHost |
| templateName | 문자열 |

```

var modelName = 'ManagementModelEntities.svc';
var entitySetName = 'VirtualMachineTemplates';
var headers = null;
//Create properties for prefix entity
var properties = {
    VirtualMachineTemplateName:templateName,
};
//Read a list of entities
var entities = vCACEntityManager
    .readModelEntitiesByCustomFilter(host.id, modelName, entitySetName, properties, headers);

```

예제: 시스템 쿼리를 사용하여 vRealize Automation 엔티티 읽기

이 예제 스크립트는 다음 작업을 수행합니다.

- 1 모델 이름 및 엔티티 집합 이름을 정의합니다.
- 2 엔티티를 필터링할 기준 시스템 쿼리를 정의하고, 시스템 상태 및 구성 요소 플래그별로 필터링된 모든 가상 시스템의 상위 10개 결과를 선택합니다.
- 3 엔티티 목록을 읽습니다.

표 4-24. 입력 변수

| 변수 | 유형 |
|------|---------------|
| host | vCAC:VcacHost |

```

var modelName = 'ManagementModelEntities.svc';
var entitySetName = 'VirtualMachines';
var filter = "VirtualMachineState eq 'Off' and IsComponent eq true";
var orderBy = 'VirtualMachineName asc';
var top = 10; {
var skip = 0;;
var headers = null;
var select = null;
var entities = vCACEntityManager
    readModelEntitiesBySystemQuery(host.id, modelName, entitySetName, filter, orderBy, select, top,
    skip, headers);

```

vRealize Automation 엔티티 찾기 예제 스크립트

JavaScript 예제를 잘라내고, 붙여 넣고, 편집하여 vCACCAFEEntitiesFinder 스크립팅 유틸리티 개체를 통해 vRealize Automation 엔티티를 찾는 스크립트를 작성할 수 있습니다.

vRealize Orchestrator의 스크립팅에 대한 자세한 내용은 "VMware vRealize Orchestrator를 사용한 개발" 을 참조하십시오.

예제: 이름별로 필터링된 카탈로그 리소스 찾기

표 4-25. 입력 변수

| 변수 | 유형 |
|------|-------------------|
| host | vCACCAFE:VcacHost |

다음 예제 중 하나를 사용할 수 있습니다.

- 이 예제 스크립트는 이름 및 설명을 사용하여 *name_of_the_resource* 쿼리와 일치하는 대상 호스트의 모든 카탈로그 리소스를 가져옵니다.

```
var items = vCACCAFEEntitiesFinder.findCatalogResources(host, "name_of_the_resource");
```

- 이 예제 스크립트는 다음 작업을 수행합니다.
 - a 소비자 리소스 서비스를 가져오고 vCACCAFEPageOdataRequest 개체의 인스턴스를 Pageable 매개 변수로 전달하는 get 메서드를 호출합니다.
 - b OData 쿼리를 *name_of_the_resource* 문자열과 일치하는 name 특성의 단일 필터로 제공하여 vCACCAFEPageOdataRequest 개체를 생성합니다.

```
var service = host.createCatalogClient().getCatalogConsumerResourceService();

var filter = new Array();
filter[0] = vCACCAFEFilterParam.equal("name", vCACCAFEFilterParam.string("name_of_the_resource"));
var query = vCACCAFEQuery.query().addFilter(filter);

var items = service.getResourcesList(new vCACCAFEPageOdataRequest(query));
```

예제: 소유자별로 필터링된 카탈로그 리소스 찾기

이 예제 스크립트는 다음 작업을 수행합니다.

- 1 소비자 리소스 서비스를 가져오고 vCACCAFEPageOdataRequest 개체의 인스턴스를 Pageable 매개 변수로 전달하는 get 메서드를 호출합니다.
- 2 OData 쿼리를 *user@domain.com* 문자열과 일치하는 owner/ref 특성의 단일 필터로 제공하여 vCACCAFEPageOdataRequest 개체를 생성합니다.

owners/ref 특성은 카탈로그 리소스의 필드와 내부 구조를 기반으로 하는 컴퍼지션입니다.

vCACCAFECatalogResource 엔티티에는 vCACCAFECatalogPrincipal 엔티티의 컬렉션인 owners 특성이 있습니다. vCACCAFECatalogPrincipal 엔티티에는 사용자의 주체 ID에 대한 문자열 표현인 ref 속성이 있습니다.

```
var filter = new Array();
filter[0] = vCACCAFEFilterParam.substringOf("owners/ref",
vCACCAFEFilterParam.string("user@domain.com"));
var query = vCACCAFEQuery.query().addFilter(filter);

var items = service.getResourcesList(new vCACCAFEPageOdataRequest(query));
```

예제: 이름 및 소유자별로 필터링된 카탈로그 리소스 찾기

이 예제 스크립트는 `vCACCAFEFilterParam.and(array of conditions)` 논리 연산자를 사용하여 위 두 예제의 OData 쿼리를 단일 조건으로 결합합니다.

```
var conditions = new Array();
conditions[0] = vCACCAFEFilterParam.equal("name",
vCACCAFEFilterParam.string("name_of_the_resource_here"));
conditions[1] = vCACCAFEFilterParam.substringOf("owners/ref",
vCACCAFEFilterParam.string("user@domain.com"));

var filter = new Array();
filter[0] = vCACCAFEFilterParam.and(conditions);
var query = vCACCAFE0dataQuery.query().addFilter(filter);

var items = service.getResourcesList(new vCACCAFEPage0dataRequest(query));
```

`vCACCAFEFilterParam.group(array of parameters)`, `vCACCAFEFilterParam.not(parameter)`, `vCACCAFEFilterParam.startsWith(id, string)`, `vCACCAFEFilterParam.endsWith(id, string)`, `vCACCAFEFilterParam.greaterThan(id, number)`, `vCACCAFEFilterParam.lessThan(id, number)` 등의 여러 논리 연산자를 사용하여 다른 조건을 정의할 수 있습니다.

vRealize Automation 예제 스크립트를 통해 프로비저닝된 리소스 가져오기

JavaScript 예제를 잘라내고, 붙여 넣고, 편집하여 vRealize Automation 프로비저닝된 리소스의 실제 엔티티를 검색할 스크립트를 작성할 수 있습니다.

CatalogResource 유형은 vRealize Automation에서 프로비저닝된 리소스를 나타냅니다. 이 유형에는 다음 특성을 사용하여 카탈로그 리소스와 해당 공급자 간의 관계를 나타내는 **ProviderBinding** 유형의 특성이 있습니다.

- **bindingId** - 공급자에 고유한 엔티티의 식별자를 나타냅니다.
- **providerRef** - vRealize Automation 구성 요소 레지스트리에 등록된 서비스에 직접 해당하는 카탈로그 공급자를 식별합니다.

vRealize Orchestrator의 스크립팅에 대한 자세한 내용은 "VMware vRealize Orchestrator를 사용한 개발"을 참조하십시오.

예제: vRealize Automation 카탈로그 리소스로 프로비저닝된 가상 시스템 가져오기

이 예제에서는 vRealize Automation 호스트와 해당 IaaS 호스트를 입력 매개 변수로 사용하고, 제공된 리소스 ID에 대해 해당 IaaS 가상 시스템을 반환합니다. 스크립팅 코드는 **iaas-service** 공급자를 통해 프로비저닝된 **Virtual Machine** 유형의 카탈로그 리소스만 가져옵니다.

표 4-26. 입력 변수

| 변수 | 유형 |
|----------|-------------------|
| vcacHost | vCACCAFE:VCACHost |
| iaasHost | vCAC:VCACHost |

```
// Id of the catalog resource (or vCACCAFECatalogResource_instance.getId())
var resourceId = "c222629c-6f90-4458-8c92-8ece0ba06173";

var resource = vCACCAFEEntitiesFinder.getCatalogResource(vcacHost, resourceId);

var resourceType = resource.getResourceTypeRef().getLabel();
System.log("resource type: " + resourceType);

var providerBinding = resource.getProviderBinding();

var bindingId = providerBinding.getBindingId();
System.log("provider binding id: " + bindingId);

var provider = providerBinding.getProviderRef();
System.log("provider id: " + provider.getId());
System.log("provider name: " + provider.getLabel());

if ((resourceType == "Virtual Machine") && (provider.getLabel() == "iaas-service")) {
    System.log("It is an IaaS VM!");

    // IaaS virtual machine
    var vm = Server.findForType("vCAC:VirtualMachine", bindingId);
    System.log("IaaS VM id: " + vm.virtualMachineID);
    System.log("IaaS VM name: " + vm.displayName);

    // IaaS Entity
    var entity = System.getModule("com.vmware.library.vcac").getVirtualMachineEntityFromId(iaasHost,
bindingId);
    System.log("IaaS entity id: " + entity.keyString);
}
}
```

일반 작업 예제 스크립트

JavaScript 예제를 잘라내고, 붙여 넣고, 편집하거나, 이를 학습용 샘플로 사용하여 일반 vRealize Automation 작업을 위한 사용자 고유의 스크립트를 개발할 수 있습니다.

vRealize Orchestrator의 스크립팅에 대한 자세한 내용은 "VMware vRealize Orchestrator를 사용한 개발" 을 참조하십시오.

예제: vRealize Automation 고급 서비스 Blueprint 생성

이 예제 스크립트는 다음 작업을 수행합니다.

- 1 서비스 Blueprint를 빌드하는 데 사용되는 vRealize Orchestrator 워크플로를 설정합니다.
- 2 이 워크플로를 기반으로 서비스 Blueprint의 콘텐츠를 생성합니다.

3 서비스 Blueprint 엔티티를 생성합니다.

4 서비스 Blueprint를 게시합니다.

표 4-27. 입력 변수

| 변수 | 유형 |
|------|-------------------|
| host | vCACCAFE:VCACHost |

```
//ID of the workflow used to create the service blueprint
var workflowId = "44e42047-2fa0-4e4a-ba0c-12086540b28b";

var name = "MyBlueprint"
var description = "Blueprint description";
var workflowClient = host.createAdvancedDesignerClient().getAdvancedDesignerWorkflowService();

//Generate a service blueprint based on the workflow ID
var blueprint = workflowClient.generateServiceBlueprintByWorkflowId(workflowId);
blueprint.setTenant(host.tenant);
blueprint.setName(name);
blueprint.setDescription(description);

//Create the service blueprint
var blueprintService =
host.createAdvancedDesignerClient().getAdvancedDesignerServiceBlueprintService();
var uri = blueprintService.createServiceBlueprint(host.tenant , blueprint);

//Publish the service blueprint
var createdBlueprint = blueprintService.getServiceBlueprintByUri(uri);
blueprintService.updateServiceBlueprintStatus(host.tenant, createdBlueprint.getId(),
vCACCAFEDesignerPublishStatus.PUBLISHED);
```

예제: vRealize Automation 승인 정책 생성

이 예제 스크립트는 다음 작업을 수행합니다.

- 1 승인 정책 유형을 가져옵니다.
- 2 승인이 필요한 사용자 및 그룹을 설정합니다.
- 3 승인 수준을 설정합니다.
- 4 사전 프로비저닝 승인 단계를 정의합니다.
- 5 사후 프로비저닝 승인 단계를 정의합니다.
- 6 이름, 설명 및 유형과 같은 승인 정책 사양을 정의합니다.
- 7 승인 정책을 생성합니다.
- 8 승인 정책을 게시합니다. 승인 정책이 게시되면 읽기 전용 상태가 됩니다.

표 4-28. 입력 변수

| 변수 | 유형 |
|------|-------------------|
| host | vCACCAFE:VCACHost |

```
// Get the type of approval policy by ID
var typeService = host.createApprovalClient().getApprovalApprovalPolicyTypeService();
var type = typeService.getApprovalPolicyType("com.vmware.cafe.catalog.request");

// Set the user and group required to complete the approval
var user = new vCACCAFEApprovalPrincipal();
user.setValue("user@domain.com");
user.setType(vCACCAFEApprovalPrincipalType.USER);

var group = new vCACCAFEApprovalPrincipal();
group.setValue("group@domain.com");
group.setType(vCACCAFEApprovalPrincipalType.GROUP);

// Set the level of the approval
var level = new vCACCAFEApprovalLevel();
level.setName("IT Approval Level");
level.setDescription("IT Approval Level description");
level.setApprovalMode(vCACCAFEApprovalMode.ALL);
System.getModule("com.vmware.library.vcaccafe.util").addElementToList(level, "getApprovers", user);
System.getModule("com.vmware.library.vcaccafe.util").addElementToList(level, "getApprovers", group);
level.setLevelNumber(1);

// Set pre-provisioning phase type and the phase of the approval
var phase1Type = new vCACCAFEApprovalPhaseType();
phase1Type.setId("com.vmware.cafe.catalog.request.pre");
phase1Type.setName("Pre-Provisioning type");
phase1Type.setDescription("Pre-Provisioning type description");
phase1Type.setPhaseOrder(1);

var phase1 = new vCACCAFEPhase();
phase1.setName("Pre-Provisioning");
phase1.setDescription("Pre provisioning phase");
phase1.setPhasetype(phase1Type);
System.getModule("com.vmware.library.vcaccafe.util").addElementToList(phase1, "getLevels", level);

// Set post-provisioning phase type and the phase of the approval
var phase2Type = new vCACCAFEApprovalPhaseType();
phase2Type.setId("com.vmware.cafe.catalog.request.post");
phase2Type.setName("Post-Provisioning type");
phase2Type.setDescription("Post-Provisioning type description");
phase2Type.setPhaseOrder(1);

var phase2 = new vCACCAFEPhase();
phase2.setName("Post-Provisioning");
phase2.setDescription("Post provisioning phase");
phase2.setPhasetype(phase2Type);
System.getModule("com.vmware.library.vcaccafe.util").addElementToList(phase2, "getLevels", level);

// Create the approval policy specifications
```

```
var spec = new vCACCAFEApprovalPolicy();
spec.setName("New Policy");
spec.setDescription("New Policy description");
spec.setPolicyType(type);
System.getModule("com.vmware.library.vcaccafe.util").addElementToList(spec, "getPhases", phase1);
System.getModule("com.vmware.library.vcaccafe.util").addElementToList(spec, "getPhases", phase2);

// Create the approval policy
var approvalPolicyService = host.createApprovalClient().getApprovalApprovalPolicyService();
var approvalPolicy = approvalPolicyService.createPolicy(spec);

// Publish the approval policy
approvalPolicy.setState(vCACCAFEApprovalPolicyState.PUBLISHED);
approvalPolicy = approvalPolicyService.update(approvalPolicy);
System.log("New approval policy id: " + approvalPolicy.getId());
```

구성 플러그인 사용

5

제어 센터를 통한 Orchestrator 구성 외에도 구성 플러그인에서 워크플로를 실행하여 Orchestrator 서버 구성 설정을 수정할 수 있습니다.

구성 플러그인을 사용하여 Orchestrator 서버 키 저장소 및 신뢰할 수 있는 인증서를 구성하고 관리할 수 있습니다.

본 장은 다음 항목을 포함합니다.

- 구성 플러그인 워크플로 라이브러리 액세스
- 구성 플러그인 워크플로 라이브러리

구성 플러그인 워크플로 라이브러리 액세스

구성 플러그인 워크플로 라이브러리의 요소에 액세스하려면 Orchestrator 클라이언트를 사용해야 합니다.

절차

- 1 Orchestrator 클라이언트의 드롭다운 메뉴에서 **실행** 또는 **설계**를 선택합니다.
- 2 **워크플로** 보기를 클릭합니다.
- 3 계층형 목록을 **라이브러리 > 구성**으로 확장합니다.

다음에 수행할 작업

워크플로 라이브러리를 검토합니다.

구성 플러그인 워크플로 라이브러리

구성 플러그인 워크플로 라이브러리에는 vRealize Orchestrator의 구성과 관련된 자동화된 프로세스를 실행하는 데 사용할 수 있는 워크플로가 포함됩니다.

SSL 신뢰 관리자 워크플로

SSL 신뢰 관리자 범주에는 SSL 인증서를 삭제하고 가져오는 데 사용할 수 있는 워크플로가 포함됩니다.

이러한 워크플로는 Orchestrator 클라이언트의 **워크플로 보기**에서 **라이브러리 > 구성 > SSL 신뢰 관리자 워크플로**를 통해 액세스할 수 있습니다.

| 워크플로 이름 | 설명 |
|---------------------------------|--|
| 신뢰할 수 있는 인증서 삭제 | 서버 신뢰 저장소에서 SSL 인증서를 삭제합니다. |
| URL에서 인증서 가져오기 | URL에서 서버 신뢰 저장소로 SSL 인증서를 가져옵니다. |
| 인증된 프록시 서버를 사용하여 URL에서 인증서 가져오기 | 인증된 프록시 서버를 통해 연결 가능한 URL에서 SSL 인증서 가져오기 |
| 프록시 서버를 사용하여 URL에서 인증서 가져오기 | 프록시 서버를 통해 연결 가능한 URL에서 SSL 인증서 가져오기 |
| 인증서 별칭을 사용하여 URL에서 인증서 가져오기 | URL에서 서버 신뢰 저장소로 SSL 인증서를 가져옵니다. |
| 파일에서 신뢰할 수 있는 인증서 가져오기 | 파일에서 서버 신뢰 저장소로 SSL 인증서를 가져옵니다. |

키 저장소 워크플로

키 저장소 구성 워크플로는 Orchestrator 클라이언트의 **워크플로 보기**에서 **라이브러리 > 구성 > 키 저장소**를 통해 액세스할 수 있습니다.

| 워크플로 이름 | 설명 |
|----------|---------------------|
| 인증서 추가 | 키 저장소에 인증서를 추가합니다. |
| 키 추가 | 키를 추가합니다. |
| 키 저장소 생성 | 새 키 저장소를 생성합니다. |
| 키 저장소 삭제 | 키 저장소를 삭제합니다. |
| 인증서 삭제 | 키 저장소에서 인증서를 삭제합니다. |
| 항목 삭제 | 항목을 삭제합니다. |
| 키 삭제 | 키를 삭제합니다. |

라이브러리 플러그인 사용

6

라이브러리 플러그인 워크플로를 템플릿으로 사용하여 클라이언트 프로세스를 사용자 지정 및 자동화하고 Orchestrator 문제를 해결할 수 있습니다.

본 장은 다음 항목을 포함합니다.

■ 라이브러리 플러그인 워크플로

라이브러리 플러그인 워크플로

라이브러리 플러그인은 **잠금**, **Orchestrator** 및 **문제 해결** 워크플로 범주의 워크플로를 제공합니다.

워크플로 잠금

이러한 워크플로는 Orchestrator 클라이언트의 **워크플로** 보기에서 **라이브러리 > 잠금**을 통해 액세스할 수 있습니다.

| 워크플로 이름 | 설명 |
|------------|----------------------------|
| 모든 잠금 표시 | 모든 잠금을 표시합니다. |
| 잠금 테스트 | 잠금을 생성하는 워크플로를 테스트합니다. |
| 잠금 테스트(x5) | 5개의 잠금을 생성하는 워크플로를 테스트합니다. |
| 모든 잠금 해제 | 모든 잠금을 해제합니다. |

Orchestrator 작업 워크플로

이러한 워크플로는 Orchestrator 클라이언트의 **워크플로** 보기에서 **라이브러리 > Orchestrator > 작업**을 통해 액세스할 수 있습니다.

| 워크플로 이름 | 설명 |
|-----------|-------------------------------------|
| 되풀이 작업 생성 | 되풀이되는 작업을 생성하고 새로 생성된 작업을 반환합니다. |
| 작업 생성 | 이후 시간 및 날짜에 실행할 워크플로를 작업으로 스케줄링합니다. |

Orchestrator 워크플로

이러한 워크플로는 Orchestrator 클라이언트의 **워크플로 보기**에서 **라이브러리 > Orchestrator > 워크플로**를 통해 액세스할 수 있습니다.

| 워크플로 이름 | 설명 |
|-----------------------------|---|
| 새로 고침 상태 워크플로가 대기 중 상태에서 실행 | 지정된 원격 서버에 대해 대기 상태에 있는 모든 워크플로 실행을 처리하고 원격 워크플로 실행에 따라 워크플로 상태를 업데이트합니다. 워크플로 실행 간에 데이터 손실이 있으면(예: Orchestrator 서버 간의 연결 끊어짐) 이 워크플로를 사용할 수 있습니다. |
| 연속으로 워크플로 시작 | 인스턴스 단위로 워크플로를 여러 번 연속으로 실행합니다. 워크플로 매개 변수를 어레이로 제공합니다. 또한 시작되는 워크플로의 각 인스턴스에 대해 워크플로 입력당 하나의 속성이 있는 속성 목록을 제공합니다. 어레이의 속성 수는 워크플로 실행 수를 정의합니다. |
| 병렬로 워크플로 시작 | 서로 다른 매개 변수를 사용하여 워크플로를 여러 번 실행합니다. 워크플로 매개 변수를 어레이로 제공합니다. 또한 시작되는 워크플로의 각 인스턴스에 대해 워크플로 입력당 하나의 속성이 있는 속성 목록을 제공합니다. 어레이의 속성 수는 워크플로 실행 수를 정의합니다. |

태그 지정 워크플로

이러한 워크플로는 Orchestrator 클라이언트의 **워크플로 보기**에서 **라이브러리 > 태그 지정**을 통해 액세스할 수 있습니다.

| 워크플로 이름 | 설명 |
|------------|--|
| 태그별 개체 찾기 | 할당된 태그로 개체를 찾습니다. 태그의 이름과 값을 제공하면 워크플로에서 이러한 태그가 적용된 개체 목록을 반환합니다. |
| 워크플로 태그 나열 | 입력 매개 변수로 지정하여 워크플로에 할당한 태그를 나열합니다. |
| 태그 지정 예제 | 워크플로 태그 지정을 보여 줍니다. |
| 워크플로 태그 지정 | 워크플로에 태그를 할당합니다. 태그를 지정할 워크플로와 태그 이름 및 값을 지정해야 합니다. |
| 워크플로 태그 해제 | 워크플로에서 태그를 제거합니다. 태그를 해제할 워크플로와 지정된 워크플로에서 제거할 태그를 지정해야 합니다. |

SQL 플러그인 사용

7

SQL 플러그인에서 제공되는 API를 사용하여 SQL 데이터베이스와 기타 테이블 형식 데이터 소스(예: 스프레드시트 또는 플랫폼 파일)에 대한 연결을 구현할 수 있습니다.

JDBC를 기반으로 하는 SQL 플러그인 API는 SQL 기반 데이터베이스 액세스용 호출 수준 API를 제공합니다. 또한 SQL 플러그인은 워크플로에서 API를 사용하는 방법을 보여 주는 샘플 워크플로를 제공합니다.

본 장은 다음 항목을 포함합니다.

- [SQL 플러그인 구성](#)
- [SQL 샘플 워크플로 실행](#)
- [SQL 플러그인 표준 워크플로 사용](#)

SQL 플러그인 구성

SQL 플러그인에 포함된 워크플로를 사용할 수 있으며 Orchestrator 클라이언트에서 이를 실행하여 SQL 플러그인을 구성하고 데이터베이스를 추가, 업데이트 또는 제거할 수 있습니다.

SQL 플러그인 구성 워크플로

SQL 플러그인의 구성 워크플로 범주에는 데이터베이스 및 데이터베이스 테이블을 관리할 수 있는 워크플로가 포함됩니다.

이러한 워크플로는 Orchestrator 클라이언트의 **워크플로 보기**에서 **라이브러리 > SQL > 구성**을 통해 액세스할 수 있습니다.

| 워크플로 이름 | 설명 |
|-----------------|---|
| 데이터베이스 추가 | 데이터베이스 플러그인 인벤토리에 데이터베이스 개체를 추가합니다. |
| 데이터베이스에 테이블 추가 | 데이터베이스 플러그인 인벤토리의 데이터베이스에 데이터베이스 테이블을 추가합니다. |
| 데이터베이스 제거 | 데이터베이스 플러그인 인벤토리에서 데이터베이스 개체를 제거합니다. |
| 데이터베이스에서 테이블 제거 | 데이터베이스 플러그인 인벤토리의 데이터베이스에서 데이터베이스 테이블을 제거합니다. |
| 데이터베이스 업데이트 | 데이터베이스 플러그인 인벤토리에서 데이터베이스 개체의 구성을 업데이트합니다. |
| 데이터베이스 유효성 검사 | 데이터베이스 플러그인 인벤토리에서 데이터베이스의 유효성을 검사합니다. |

데이터베이스 추가

워크플로를 실행하여 Orchestrator 서버에 데이터베이스를 추가하고 호스트 연결 매개 변수를 구성할 수 있습니다.

보안 연결이 필요한 데이터베이스를 추가할 경우 데이터베이스 SSL 인증서를 가져와야 합니다. 제어 센터의 **신뢰할 수 있는 인증서** 탭에서 SSL 인증서를 가져올 수 있습니다.

절차

- 1 Orchestrator 클라이언트에 관리자로 로그인합니다.
- 2 Orchestrator 클라이언트에서 **워크플로** 보기를 클릭합니다.
- 3 계층형 워크플로 목록에서 **라이브러리 > SQL > 구성**을 확장하여 **데이터베이스 추가** 워크플로로 이동합니다.
- 4 **데이터베이스 추가** 워크플로를 마우스 오른쪽 버튼으로 클릭하고 **워크플로 시작**을 선택합니다.
- 5 **이름** 텍스트 상자에서 데이터베이스의 이름을 입력합니다.
- 6 데이터베이스 유형을 선택합니다.
- 7 **연결 URL** 텍스트 상자에 데이터베이스의 주소를 입력합니다.

| 데이터베이스 유형 | 구문 |
|---------------------------------|---|
| Oracle | <code>jdbc:oracle:thin:@database_url:port_number:SID</code> |
| Microsoft SQL(SQL 인증 사용) | <code>jdbc:jtds:sqlserver://database_url:port_number/database_name</code> |
| Microsoft SQL(Windows 계정 인증 사용) | <code>jdbc:jtds:sqlserver://database_url:port_number/database_name;useNTLMv2=true;domain=domain_name</code> |
| PostgreSQL | <code>jdbc:postgresql://database_url:port_number/database_name</code> |
| MySQL | <code>jdbc:mysql://database_url:port_number/database_name</code> |

- 8 플러그인에서 데이터베이스에 연결하는 데 사용할 세션 모드를 선택합니다.

| 옵션 | 설명 |
|---------|---|
| 공유 세션 | 플러그인에서 공유 자격 증명을 사용하여 데이터베이스에 연결합니다. 공유 세션에 대한 데이터베이스 자격 증명을 제공해야 합니다. |
| 사용자별 세션 | Orchestrator 클라이언트가 로그인된 사용자로부터 자격 증명을 검색합니다. 참고 사용자별 세션 모드를 사용하려면 사용자 이름만 사용하여 인증해야 합니다. <code>도메인\사용자</code> 또는 <code>사용자@도메인</code> 을 인증에 사용해서는 안 됩니다. |

- 9 **제출**을 클릭하여 워크플로를 실행합니다.

결과

워크플로가 성공적으로 실행되면 **인벤토리** 보기에 데이터베이스와 해당 데이터베이스에 속한 모든 테이블이 표시됩니다.

데이터베이스에 테이블 추가

워크플로를 실행하여 데이터베이스 플러그인 인벤토리에 있는 데이터베이스에 테이블을 추가할 수 있습니다.

사전 요구 사항

- Orchestrator 클라이언트에 관리자로 로그인되어 있는지 확인합니다.
- 데이터베이스에 연결되었는지 여부를 **인벤토리** 보기에서 확인합니다.

절차

- 1 Orchestrator 클라이언트에서 **워크플로** 보기를 클릭합니다.
- 2 계층형 워크플로 목록에서 **라이브러리 > SQL > 구성**을 확장하여 **데이터베이스에 테이블 추가** 워크플로로 이동합니다.
- 3 **데이터베이스에 테이블 추가** 워크플로를 마우스 오른쪽 버튼으로 클릭하고 **워크플로 시작**을 선택합니다.
- 4 테이블을 추가할 데이터베이스를 선택합니다.
- 5 추가할 테이블을 선택합니다.
- 6 **제출**을 클릭하여 워크플로를 실행합니다.

결과

워크플로가 성공적으로 실행되면 Orchestrator 클라이언트의 **인벤토리** 보기에 추가한 데이터베이스 테이블이 표시됩니다.

데이터베이스 업데이트

워크플로를 실행하여 플러그인 인벤토리에 있는 데이터베이스의 구성을 업데이트할 수 있습니다.

절차

- 1 Orchestrator 클라이언트에 관리자로 로그인합니다.
- 2 Orchestrator 클라이언트에서 **워크플로** 보기를 클릭합니다.
- 3 계층형 **워크플로** 목록에서 **라이브러리 > SQL > 구성**을 확장하여 **데이터베이스 업데이트** 워크플로로 이동합니다.
- 4 **데이터베이스 업데이트** 워크플로를 마우스 오른쪽 버튼으로 클릭하고 **워크플로 시작**을 선택합니다.
- 5 업데이트할 데이터베이스를 선택합니다.
- 6 **이름** 텍스트 상자에 데이터베이스의 새 이름을 입력합니다.
인벤토리 보기에 데이터베이스가 지정한 이름으로 표시됩니다.
- 7 데이터베이스 유형을 선택합니다.
- 8 **연결 URL** 텍스트 상자에 데이터베이스의 새 주소를 입력합니다.

9 플러그인에서 데이터베이스에 연결하는 데 사용할 세션 모드를 선택합니다.

| 옵션 | 설명 |
|---------|---|
| 공유 세션 | 플러그인에서 공유 자격 증명을 사용하여 데이터베이스에 연결합니다. 공유 세션에 대한 데이터베이스 자격 증명을 제공해야 합니다. |
| 사용자별 세션 | Orchestrator 클라이언트가 로그인된 사용자로부터 자격 증명을 검색합니다. 참고 사용자별 세션 모드를 사용하려면 사용자 이름만 사용하여 인증해야 합니다. <code>도메인\사용자</code> 또는 <code>사용자@도메인</code> 을 인증에 사용해서는 안 됩니다. |

10 제출을 클릭하여 워크플로를 실행합니다.

SQL 샘플 워크플로 실행

SQL 플러그인 워크플로를 실행하여 JDBC URL 생성, JDBC 연결 테스트 및 JDBC 테이블의 행 관리와 같은 JDBC 작업을 수행할 수 있습니다. 또한 SQL 플러그인 워크플로를 실행하여 데이터베이스 및 데이터베이스 테이블을 관리하고 SQL 작업을 실행할 수 있습니다.

JDBC URL 생성

Orchestrator 클라이언트에서 워크플로를 실행하여 JDBC 연결 URL을 생성할 수 있습니다.

사전 요구 사항

로그인한 사용자 계정에 JDBC 워크플로를 실행하는 데 필요한 권한이 있는지 확인합니다.

절차

- 1 Orchestrator 클라이언트에서 **워크플로** 보기를 클릭합니다.
- 2 워크플로 계층형 목록에서 **라이브러리 > JDBC**를 확장하여 JDBC URL 생성기 워크플로로 이동합니다.
- 3 JDBC URL 생성기 워크플로를 마우스 오른쪽 버튼으로 클릭하고 **워크플로 시작**을 선택합니다.
- 4 URL을 생성할 데이터베이스 유형을 선택합니다.

참고 Microsoft 데이터베이스를 사용하는 경우 **다음**을 클릭하고 데이터베이스 인스턴스 이름 및 데이터베이스 사용자 도메인 이름을 제공해야 할 수도 있습니다.

- 5 데이터베이스 URL을 생성하는 데 필요한 정보를 제공합니다.

- a 데이터베이스 서버 이름 또는 IP 주소를 입력합니다.
- b 데이터베이스 이름을 입력합니다.
- c (선택 사항) 데이터베이스 포트 번호를 입력합니다.

포트 번호를 지정하지 않으면 워크플로에서 기본 포트 번호를 사용합니다.

- d 데이터베이스 액세스를 위한 사용자 이름을 입력합니다.
- e 데이터베이스 액세스를 위한 암호를 입력합니다.

6 제출을 클릭하여 워크플로를 실행합니다.

JDBC 연결 테스트

Orchestrator 클라이언트에서 워크플로를 실행하여 데이터베이스 연결을 테스트할 수 있습니다.

사전 요구 사항

로그인한 사용자 계정에 JDBC 워크플로를 실행하는 데 필요한 권한이 있는지 확인합니다.

절차

- 1** Orchestrator 클라이언트에서 **워크플로** 보기를 클릭합니다.
- 2** 워크플로 계층형 목록에서 **라이브러리 > JDBC > JDBC 예제**를 확장하여 JDBC 연결 예제 워크플로로 이동합니다.
- 3** JDBC 연결 예제 워크플로를 마우스 오른쪽 버튼으로 클릭하고 **워크플로 시작**을 선택합니다.
- 4** 데이터베이스 연결을 테스트하는 데 필요한 정보를 제공합니다.
 - a 데이터베이스 액세스를 위한 사용자 이름을 입력합니다.
 - b 테스트할 URL을 입력합니다.
 - c 데이터베이스 액세스를 위한 암호를 입력합니다.
- 5 제출**을 클릭하여 워크플로를 실행합니다.

JDBC를 사용하여 테이블 생성

Orchestrator 클라이언트에서 워크플로를 실행하여 데이터베이스를 만들 수 있습니다.

사전 요구 사항

로그인한 사용자 계정에 JDBC 워크플로를 실행하는 데 필요한 권한이 있는지 확인합니다.

절차

- 1** Orchestrator 클라이언트에서 **워크플로** 보기를 클릭합니다.
- 2** 워크플로 계층형 목록에서 **라이브러리 > JDBC > JDBC 예제**를 확장하여 JDBC 테이블 생성 예제 워크플로로 이동합니다.
- 3** JDBC 테이블 생성 예제 워크플로를 마우스 오른쪽 버튼으로 클릭하고 **워크플로 시작**을 선택합니다.

- 4 필요한 정보를 제공하고 **다음**을 클릭합니다.
 - a 데이터베이스 액세스를 위한 암호를 입력합니다.
 - b 데이터베이스 연결 URL을 입력합니다.
 - c 데이터베이스 액세스를 위한 사용자 이름을 입력합니다.

5 SQL create 문을 입력합니다.

예제 구문은 다음과 같습니다.

```
CREATE TABLE "table_name"
("column1" "data_type_for_column1",
"column2" "data_type_for_column2")
```

6 제출을 클릭하여 워크플로를 실행합니다.

JDBC 테이블에 행 삽입

Orchestrator 클라이언트에서 워크플로를 실행하여 JDBC 테이블에 행 삽입을 테스트할 수 있습니다.

사전 요구 사항

로그인한 사용자 계정에 JDBC 워크플로를 실행하는 데 필요한 권한이 있는지 확인합니다.

절차

- 1 Orchestrator 클라이언트에서 **워크플로** 보기를 클릭합니다.
- 2 워크플로 계층형 목록에서 **라이브러리 > JDBC > JDBC 예제**를 확장하여 JDBC 테이블에 삽입 예제 워크플로로 이동합니다.
- 3 JDBC 테이블에 삽입 예제 워크플로를 마우스 오른쪽 버튼으로 클릭하고 **워크플로 시작**을 선택합니다.
- 4 필요한 정보를 제공하고 **다음**을 클릭합니다.
 - a 데이터베이스 연결 URL을 입력합니다.
 - b 데이터베이스 액세스를 위한 사용자 이름을 입력합니다.
 - c 데이터베이스 액세스를 위한 암호를 입력합니다.

5 SQL insert 문을 입력하고 **다음**을 클릭합니다.

예제 구문은 다음과 같습니다.

```
INSERT INTO "table_name" ("column1", "column2")
VALUES ("value1", "value2")
```

- 6 행에 삽입할 값을 입력합니다.
- 7 **제출**을 클릭하여 워크플로를 실행합니다.

JDBC 테이블에서 행 선택

Orchestrator 클라이언트에서 워크플로를 실행하여 JDBC 테이블에서 행을 선택할 수 있습니다.

사전 요구 사항

로그인한 사용자 계정에 JDBC 워크플로를 실행하는 데 필요한 권한이 있는지 확인합니다.

절차

- 1 Orchestrator 클라이언트에서 **워크플로** 보기를 클릭합니다.
- 2 워크플로 계층형 목록에서 **라이브러리 > JDBC > JDBC 예제**를 확장하여 JDBC 테이블에서 선택 예제 워크플로로 이동합니다.
- 3 JDBC 테이블에서 선택 예제 워크플로를 마우스 오른쪽 버튼으로 클릭하고 **워크플로 시작**을 선택합니다.
- 4 필요한 정보를 제공하고 **다음**을 클릭합니다.
 - a 데이터베이스 연결 URL을 입력합니다.
 - b 데이터베이스 액세스를 위한 사용자 이름을 입력합니다.
 - c 데이터베이스 액세스를 위한 암호를 입력합니다.
- 5 SQL select 문을 입력합니다.
예제 구문은 다음과 같습니다.

```
SELECT * FROM "table_name"
```

- 6 **제출**을 클릭하여 워크플로를 실행합니다.

JDBC 테이블에서 항목 삭제

Orchestrator 클라이언트에서 워크플로를 실행하여 JDBC 테이블에서 항목 삭제를 테스트할 수 있습니다.

사전 요구 사항

로그인한 사용자 계정에 JDBC 워크플로를 실행하는 데 필요한 권한이 있는지 확인합니다.

절차

- 1 Orchestrator 클라이언트에서 **워크플로** 보기를 클릭합니다.
- 2 워크플로 계층형 목록에서 **라이브러리 > JDBC > JDBC 예제**를 확장하여 JDBC 테이블에서 항목 삭제 예제 워크플로로 이동합니다.
- 3 JDBC 테이블에서 항목 삭제 예제 워크플로를 마우스 오른쪽 버튼으로 클릭하고 **워크플로 시작**을 선택합니다.

- 4 필요한 정보를 제공하고 **다음**을 클릭합니다.
 - a 삭제할 사용자 항목의 이름을 입력합니다.
 - b 데이터베이스 액세스를 위한 사용자 이름을 입력합니다.
 - c JDBC 연결 URL을 입력합니다.
 - d 삭제할 사용자 항목의 성을 입력합니다.
 - e 데이터베이스 액세스를 위한 암호를 입력합니다.

- 5 SQL delete 문을 입력합니다.

예제 구문은 다음과 같습니다.

```
DELETE FROM "table_name" where ("column1" = ?, "column2" = ?)
```

- 6 **제출**을 클릭하여 워크플로를 실행합니다.

JDBC 테이블에서 모든 항목 삭제

Orchestrator 클라이언트에서 워크플로를 실행하여 JDBC 테이블에서 모든 항목을 삭제할 수 있습니다.

사전 요구 사항

로그인한 사용자 계정에 JDBC 워크플로를 실행하는 데 필요한 권한이 있는지 확인합니다.

절차

- 1 Orchestrator 클라이언트에서 **워크플로** 보기를 클릭합니다.
- 2 워크플로 계층형 목록에서 **라이브러리 > JDBC > JDBC 예제**를 확장하여 JDBC 테이블에서 모든 항목 삭제 예제 워크플로로 이동합니다.
- 3 JDBC 테이블에서 모든 항목 삭제 예제 워크플로를 마우스 오른쪽 버튼으로 클릭하고 **워크플로 시작**을 선택합니다.
- 4 필요한 정보를 제공하고 **다음**을 클릭합니다.
 - a 데이터베이스 연결 URL을 입력합니다.
 - b 데이터베이스 액세스를 위한 사용자 이름을 입력합니다.
 - c 데이터베이스 액세스를 위한 암호를 입력합니다.

- 5 SQL delete 문을 입력합니다.

예제 구문은 다음과 같습니다.

```
DELETE FROM "table_name"
```

- 6 **제출**을 클릭하여 워크플로를 실행합니다.

JDBC 테이블 삭제

Orchestrator 클라이언트에서 워크플로를 실행하여 JDBC 테이블 삭제를 테스트할 수 있습니다.

사전 요구 사항

로그인한 사용자 계정에 JDBC 워크플로를 실행하는 데 필요한 권한이 있는지 확인합니다.

절차

- 1 Orchestrator 클라이언트에서 **워크플로** 보기를 클릭합니다.
- 2 워크플로 계층형 목록에서 **라이브러리 > JDBC > JDBC 예제**를 확장하여 JDBC 테이블 삭제 예제 워크플로로 이동합니다.
- 3 JDBC 테이블 삭제 예제 워크플로를 마우스 오른쪽 버튼으로 클릭하고 **워크플로 시작**을 선택합니다.
- 4 필요한 정보를 제공하고 **다음**을 클릭합니다.
 - a 데이터베이스 액세스를 위한 암호를 입력합니다.
 - b 데이터베이스 연결 URL을 입력합니다.
 - c 데이터베이스 액세스를 위한 사용자 이름을 입력합니다.
- 5 SQL drop 문을 입력합니다.
예제 구문은 다음과 같습니다.

```
DROP TABLE "table_name"
```

- 6 **제출**을 클릭하여 워크플로를 실행합니다.

전체 JDBC 주기 실행

Orchestrator 클라이언트에서 워크플로를 실행하여 하나의 전체 주기에서 모든 JDBC 예제 워크플로를 테스트할 수 있습니다.

사전 요구 사항

로그인한 사용자 계정에 JDBC 워크플로를 실행하는 데 필요한 권한이 있는지 확인합니다.

절차

- 1 Orchestrator 클라이언트에서 **워크플로** 보기를 클릭합니다.
- 2 워크플로 계층형 목록에서 **라이브러리 > JDBC > JDBC 예제**를 확장하여 전체 JDBC 주기 예제 워크플로로 이동합니다.
- 3 전체 JDBC 주기 예제 워크플로를 마우스 오른쪽 버튼으로 클릭하고 **워크플로 시작**을 선택합니다.

4 필요한 정보를 제공하고 **다음**을 클릭합니다.

- a 데이터베이스 연결 URL을 입력합니다.
- b 데이터베이스 액세스를 위한 사용자 이름을 입력합니다.
- c 데이터베이스 액세스를 위한 암호를 입력합니다.

5 데이터베이스의 항목으로 사용할 값을 입력합니다.

6 **제출**을 클릭하여 워크플로를 실행합니다.

SQL 플러그인 표준 워크플로 사용

SQL 워크플로를 사용하여 SQL 작업을 실행할 수 있습니다.

SQL 플러그인 워크플로 라이브러리

SQL 플러그인 워크플로를 실행하여 데이터베이스 및 데이터베이스 테이블을 관리하고 SQL 작업을 실행할 수 있습니다.

데이터베이스 구성 워크플로는 Orchestrator 클라이언트의 **워크플로 보기**에서 **라이브러리 > SQL > 구성**을 통해 액세스할 수 있습니다.

| 워크플로 이름 | 설명 |
|-----------------|--|
| 데이터베이스 추가 | 플러그인 인벤토리에 데이터베이스 개체를 추가합니다. |
| 데이터베이스에 테이블 추가 | 플러그인 인벤토리의 데이터베이스에 데이터베이스 테이블을 추가합니다. |
| 데이터베이스 제거 | 플러그인 인벤토리에서 데이터베이스 개체를 제거합니다. |
| 데이터베이스에서 테이블 제거 | 플러그인 인벤토리의 데이터베이스에서 데이터베이스 테이블을 제거합니다. |
| 데이터베이스 업데이트 | 플러그인 인벤토리에서 데이터베이스 개체의 구성을 업데이트합니다. |
| 데이터베이스 유효성 검사 | 플러그인 인벤토리에서 데이터베이스의 유효성을 검사합니다. |

SQL 작업 워크플로는 Orchestrator 클라이언트의 **워크플로 보기**에서 **라이브러리 > SQL**을 통해 액세스할 수 있습니다.

| 워크플로 이름 | 설명 |
|-----------------------|---|
| 데이터베이스에서 사용자 지정 쿼리 실행 | 지정된 데이터베이스에서 사용자 지정 쿼리를 실행하고 영향을 받는 행 수를 반환합니다. 워크플로를 실행하여 쿼리를 업데이트, 삭제, 삽입 및 작성할 수 있습니다. |
| 테이블에 대한 CRUD 워크플로 생성 | 특정 테이블에 대한 생성, 읽기, 업데이트 및 삭제 워크플로를 생성합니다. |
| 데이터베이스에서 사용자 지정 쿼리 읽기 | 지정된 데이터베이스에서 사용자 지정 쿼리를 실행하고 속성 어레이에 결과를 반환합니다. 워크플로를 실행하여 쿼리를 선택하고 읽을 수 있습니다. |

테이블에 대한 CRUD 워크플로 생성

워크플로를 실행하여 특정 테이블에 대한 생성, 읽기, 업데이트 및 삭제 워크플로를 생성할 수 있습니다.

사전 요구 사항

- Orchestrator 클라이언트에 관리자로 로그인되어 있는지 확인합니다.
- 데이터베이스에 연결되었는지 여부를 **인벤토리** 보기에서 확인합니다.

절차

- 1 Orchestrator 클라이언트에서 **워크플로** 보기를 클릭합니다.
- 2 계층형 워크플로 목록에서 **라이브러리 > SQL**을 확장하여 **테이블에 대한 CRUD 워크플로 생성** 워크플로로 이동합니다.
- 3 **테이블에 대한 CRUD 워크플로 생성** 워크플로를 마우스 오른쪽 버튼으로 클릭하고 **워크플로 시작**을 선택합니다.
- 4 워크플로를 생성할 테이블을 선택합니다.
- 5 워크플로를 생성할 워크플로 폴더를 선택합니다.
- 6 기존 워크플로를 덮어쓸지 여부를 선택합니다.

| 옵션 | 설명 |
|-----|---|
| 예 | 생성된 워크플로가 이름이 같은 기존 워크플로를 덮어씁니다. |
| 아니요 | 이름이 같은 워크플로가 폴더에 존재하는 경우 새 워크플로가 생성되지 않습니다. |

- 7 (선택 사항) 값을 채우지 않아야 하는 열을 선택합니다.
선택한 열은 생성된 **CRUD** 워크플로로 편집할 수 없습니다.
- 8 **제출**을 클릭하여 워크플로를 실행합니다.

결과

워크플로가 성공적으로 실행되면 선택한 워크플로 폴더에 **CRUD** 워크플로가 표시됩니다.

다음에 수행할 작업

선택한 데이터베이스 테이블에서 생성된 워크플로를 실행할 수 있습니다.

SSH 플러그인 사용

8

SSH 플러그인 워크플로를 사용하여 SSH를 지원하는 원격 호스트에서 SSH 명령을 실행하고, 보안 연결을 통해 Orchestrator 서버와 원격 호스트 간에 파일을 전송할 수 있습니다.

본 장은 다음 항목을 포함합니다.

- SSH 플러그인 구성
- SSH 플러그인 샘플 워크플로 실행

SSH 플러그인 구성

암호화된 연결을 위해 SSH 플러그인을 설정할 수 있습니다.

절차

- 1 Orchestrator 클라이언트에 관리자로 로그인합니다.
- 2 Orchestrator 클라이언트에서 **워크플로** 보기를 클릭합니다.
- 3 계층형 워크플로 목록에서 **라이브러리 > SSH**를 확장하여 SSH 호스트 추가 워크플로로 이동합니다.
- 4 메일 구성 워크플로를 마우스 오른쪽 버튼으로 클릭하고 **워크플로 시작**을 선택합니다.
- 5 **호스트 이름** 텍스트 상자에 Orchestrator를 통해 SSH로 액세스할 호스트의 이름을 입력합니다.
- 6 대상 포트를 입력합니다. 기본 SSH 포트는 22입니다.
호스트가 SSH 연결 목록에 추가됩니다.
- 7 (선택 사항) 서버에서 항목 경로를 구성합니다.
 - a **새 루트 폴더**를 클릭합니다.
 - b 새 경로를 입력하고 **값 삽입**을 클릭합니다.
- 8 SSH 명령을 실행하는 데 필요한 권한이 있는 사용자의 사용자 이름을 입력합니다.

9 인증 유형을 선택합니다.

| 옵션 | 작업 |
|-----|-----------------------------------|
| 예 | 암호 인증을 사용하려면 암호를 입력합니다. |
| 아니오 | 키 인증을 사용하려면 개인 키의 경로 및 암호를 입력합니다. |

10 제출을 클릭하여 워크플로를 실행합니다.

결과

SSH 호스트는 Orchestrator 클라이언트의 **인벤토리** 보기에서 사용할 수 있습니다.

구성 워크플로

SSH 플러그인의 구성 범주에는 Orchestrator와 SSH 호스트 간의 연결을 관리할 수 있는 워크플로가 포함되어 있습니다.

Orchestrator 클라이언트의 **워크플로** 보기에서 **라이브러리 > SSH > 구성**을 통해 이러한 워크플로에 액세스할 수 있습니다.

| 워크플로 이름 | 설명 |
|---------------------|-----------------------------------|
| 루트 폴더를 SSH 호스트에 추가 | 루트 폴더를 SSH 호스트에 대한 기존 연결에 추가합니다. |
| SSH 호스트 추가 | SSH 호스트에 대한 새 연결을 기존 구성에 추가합니다. |
| 루트 폴더를 SSH 호스트에서 제거 | 루트 폴더를 SSH 호스트에 대한 기존 연결에서 제거합니다. |
| SSH 호스트 제거 | SSH 호스트에 대한 기존 연결을 기존 구성에서 제거합니다. |
| SSH 호스트 업데이트 | SSH 호스트에 대한 기존 연결을 업데이트합니다. |

SSH 플러그인 샘플 워크플로 실행

Orchestrator 클라이언트에서 SSH 플러그인 샘플 워크플로를 실행하여 Orchestrator 서버와 SSH 호스트 간의 연결을 테스트할 수 있습니다.

■ 키 쌍 생성

Orchestrator 클라이언트에서 워크플로를 실행하여 키 쌍을 생성할 수 있습니다. 이러한 키 쌍을 사용하여 암호 없이 SSH 호스트에 연결할 수 있습니다.

■ 키 쌍 암호 변경

Orchestrator 클라이언트에서 워크플로를 실행하여 가장 최근에 생성한 키 쌍의 암호를 변경할 수 있습니다.

■ SSH 호스트에 Orchestrator 공개 키 등록

암호 대신 공개 키를 사용할 수 있습니다. SSH 호스트에 Orchestrator 공개 키를 등록하려면 Orchestrator 클라이언트에서 워크플로를 실행하면 됩니다.

■ SSH 명령 실행

Orchestrator 클라이언트에서 워크플로를 실행하여 원격 SSH 서버에서 SSH 명령을 실행할 수 있습니다.

■ SSH 호스트에서 파일 복사

Orchestrator 클라이언트에서 워크플로를 실행하여 SSH 호스트에서 Orchestrator 서버로 파일을 복사할 수 있습니다.

■ SSH 호스트에 파일 복사

Orchestrator 클라이언트에서 워크플로를 실행하여 Orchestrator 서버에서 SSH 호스트로 파일을 복사할 수 있습니다.

키 쌍 생성

Orchestrator 클라이언트에서 워크플로를 실행하여 키 쌍을 생성할 수 있습니다. 이러한 키 쌍을 사용하여 암호 없이 SSH 호스트에 연결할 수 있습니다.

키 쌍은 공개 키와 개인 키로 구성됩니다. Orchestrator에서는 개인 키를 사용하여 SSH 호스트의 공개 키에 연결할 수 있습니다. 암호를 사용하여 보안을 강화할 수 있습니다.

경고 올바른 권한 집합을 가진 모든 Orchestrator 사용자는 개인 키를 읽고, 사용하고, 덮어쓸 수 있습니다.

사전 요구 사항

로그인한 사용자 계정에 SSH 워크플로를 실행하는 데 필요한 권한이 있는지 확인합니다.

절차

- 1 Orchestrator 클라이언트에서 **워크플로** 보기를 클릭합니다.
- 2 계층형 워크플로 목록에서 **라이브러리 > SSH**를 확장하여 키 쌍 생성 워크플로로 이동합니다.
- 3 키 쌍 생성 워크플로를 마우스 오른쪽 버튼으로 클릭하고 **워크플로 시작**을 선택합니다.
- 4 필요한 정보를 제공합니다.
 - a 키 유형을 선택합니다.
 - b 키 크기를 선택합니다.
 - c (선택 사항) 암호를 입력합니다.

참고 나중에 암호를 변경할 수 있습니다.

- d (선택 사항) 설명을 입력합니다.
- 5 **제출**을 클릭하여 워크플로를 실행합니다.

키 쌍이 존재하는 경우 새 키 쌍이 기존 키 쌍을 덮어씁니다.

키 쌍 암호 변경

Orchestrator 클라이언트에서 워크플로를 실행하여 가장 최근에 생성한 키 쌍의 암호를 변경할 수 있습니다.

사전 요구 사항

로그인한 사용자 계정에 SSH 워크플로를 실행하는 데 필요한 권한이 있는지 확인합니다.

절차

- 1 Orchestrator 클라이언트에서 **워크플로** 보기를 클릭합니다.
- 2 계층형 워크플로 목록에서 **라이브러리 > SSH**를 확장하여 키 쌍 암호 변경 워크플로로 이동합니다.
- 3 키 쌍 암호 변경 워크플로를 마우스 오른쪽 버튼으로 클릭하고 **워크플로 시작**을 선택합니다.
- 4 키 쌍 암호를 재설정합니다.
 - a 현재 암호를 입력합니다.
 - b 새 암호를 입력합니다.
- 5 **제출**을 클릭하여 워크플로를 실행합니다.

SSH 호스트에 Orchestrator 공개 키 등록

암호 대신 공개 키를 사용할 수 있습니다. SSH 호스트에 Orchestrator 공개 키를 등록하려면 Orchestrator 클라이언트에서 워크플로를 실행하면 됩니다.

사전 요구 사항

로그인한 사용자 계정에 SSH 워크플로를 실행하는 데 필요한 권한이 있는지 확인합니다.

절차

- 1 Orchestrator 클라이언트에서 **워크플로** 보기를 클릭합니다.
- 2 계층형 워크플로 목록에서 **라이브러리 > SSH**를 확장하여 호스트에 vCO 공개 키 등록 워크플로로 이동합니다.
- 3 호스트에 vCO 공개 키 등록 워크플로를 마우스 오른쪽 버튼으로 클릭하고 **워크플로 시작**을 선택합니다.
- 4 SSH 호스트 이름과 이 호스트에 로그인할 사용자 이름 및 암호를 제공합니다.

참고 SSH 호스트에 등록된 자격 증명을 제공해야 합니다.

- 5 **제출**을 클릭하여 워크플로를 실행합니다.

결과

등록된 사용자로 SSH 호스트에 연결할 때 암호 인증 대신 공개 키 인증을 사용할 수 있습니다.

SSH 명령 실행

Orchestrator 클라이언트에서 워크플로를 실행하여 원격 SSH 서버에서 SSH 명령을 실행할 수 있습니다.

사전 요구 사항

로그인한 사용자 계정에 SSH 워크플로를 실행하는 데 필요한 권한이 있는지 확인합니다.

절차

- 1 Orchestrator 클라이언트에서 **워크플로** 보기를 클릭합니다.
- 2 계층형 워크플로 목록에서 **라이브러리 > SSH**를 확장하여 SSH 명령 실행 워크플로로 이동합니다.
- 3 SSH 명령 실행 워크플로를 마우스 오른쪽 버튼으로 클릭하고 **워크플로 시작**을 선택합니다.
- 4 SSH 호스트 이름 또는 IP 주소를 입력하고 **다음**을 클릭합니다.
- 5 실행할 SSH 명령을 입력하고 **다음**을 클릭합니다.

참고 기본 SSH 명령은 **uptime**입니다. 이 명령은 서버가 활성화된 기간 및 해당 기간 동안의 사용자 부하를 보여 줍니다.

- 6 **예**를 선택하여 암호 인증을 사용하고 **다음**을 클릭합니다.

참고 기본 옵션은 키 파일 인증을 사용하는 것입니다.

- 7 사용자 이름을 입력하고 **다음**을 클릭합니다.
- 8 인증 방법에 암호가 필요한 경우 암호를 입력합니다. 그렇지 않으면 개인 키 경로를 입력한 다음 개인 키의 암호를 입력합니다.
- 9 **제출**을 클릭하여 워크플로를 실행합니다.

SSH 호스트에서 파일 복사

Orchestrator 클라이언트에서 워크플로를 실행하여 SSH 호스트에서 Orchestrator 서버로 파일을 복사할 수 있습니다.

SSH 플러그인은 SFTP를 구현하는 Java JCraft 라이브러리를 사용합니다. SCP get 명령 워크플로는 SFTP를 사용하여 파일을 전송합니다.

사전 요구 사항

로그인한 사용자 계정에 SSH 워크플로를 실행하는 데 필요한 권한이 있는지 확인합니다.

참고 Orchestrator에서 폴더에 쓰기 위해서는 명시적인 쓰기 권한이 있어야 합니다.

절차

- 1 Orchestrator 클라이언트에서 **워크플로** 보기를 클릭합니다.
- 2 계층형 워크플로 목록에서 **라이브러리 > SSH**를 확장하여 SCP get 명령 워크플로로 이동합니다.

- 3 SCP get 명령 워크플로를 마우스 오른쪽 버튼으로 클릭하고 **워크플로 시작**을 선택합니다.
- 4 필요한 정보를 제공하고 **다음**을 클릭합니다.
 - a SSH 호스트 이름 또는 IP 주소를 입력합니다.
 - b SSH 인증 정보를 입력합니다.
- 5 파일 정보를 입력합니다.
 - a 파일을 복사할 Orchestrator 서버의 디렉토리 경로를 입력합니다.
 - b 원격 SSH 호스트에서 가져올 파일의 경로를 입력합니다.
- 6 **제출**을 클릭하여 워크플로를 실행합니다.

SSH 호스트에 파일 복사

Orchestrator 클라이언트에서 워크플로를 실행하여 Orchestrator 서버에서 SSH 호스트로 파일을 복사할 수 있습니다.

SSH 플러그인은 SFTP를 구현하는 Java JCraft 라이브러리를 사용합니다. SCP put 명령 워크플로는 SFTP를 사용하여 파일을 전송합니다.

사전 요구 사항

로그인한 사용자 계정에 SSH 워크플로를 실행하는 데 필요한 권한이 있는지 확인합니다.

절차

- 1 Orchestrator 클라이언트에서 **워크플로** 보기를 클릭합니다.
- 2 계층형 워크플로 목록에서 **라이브러리 > SSH**를 확장하여 SCP put 명령 워크플로로 이동합니다.
- 3 SCP put 명령 워크플로를 마우스 오른쪽 버튼으로 클릭하고 **워크플로 시작**을 선택합니다.
- 4 필요한 정보를 제공하고 **다음**을 클릭합니다.
 - a SSH 호스트 이름 또는 IP 주소를 입력합니다.
 - b SSH 인증 정보를 입력합니다.
- 5 파일 정보를 입력합니다.
 - a 로컬 Orchestrator 서버에서 원격 SSH 호스트로 복사할 파일의 경로를 입력합니다.
 - b 파일을 복사할 원격 SSH 호스트의 디렉토리 경로를 입력합니다.
- 6 **제출**을 클릭하여 워크플로를 실행합니다.

XML 플러그인 사용

9

XML 플러그인을 사용하여 XML 문서를 생성하고 수정하는 워크플로를 실행할 수 있습니다.

XML 플러그인은 DOM(문서 개체 모델) XML 구문 분석기 구현을 Orchestrator JavaScript API에 추가합니다. 또한 XML 플러그인은 워크플로에서 XML 문서를 생성하고 수정할 수 있는 방법을 보여 주는 몇 가지 샘플 워크플로를 제공합니다.

Orchestrator JavaScript API의 E4X(ECMAScript for XML) 구현을 사용하여 JavaScript에서 직접 XML 문서를 처리할 수도 있습니다. E4X 스크립팅 예제는 "VMware vRealize Orchestrator를 사용한 개발"을 참조하십시오.

E4X에 대해 자세히 알아보려면 ECMA-357 표준을 관리하는 조직의 웹 사이트로 이동하십시오.

본 장은 다음 항목을 포함합니다.

- XML 플러그인 샘플 워크플로 실행

XML 플러그인 샘플 워크플로 실행

Orchestrator 클라이언트에서 XML 플러그인 샘플 워크플로를 실행하여 테스트용으로 XML 문서를 만들고 수정할 수 있습니다.

워크플로에서 파일을 생성하거나 읽거나 수정할 수 있으므로 작업 디렉토리에 대한 충분한 액세스 권한이 있어야 합니다.

Orchestrator는 서버 시스템의 루트에 있는 **orchestrator**라는 이름의 폴더에 대한 읽기, 쓰기 및 실행 권한을 가지고 있습니다. 워크플로가 이 폴더에서 읽기, 쓰기 및 실행 사용 권한을 가지고 있지만 서버 시스템에서 폴더를 생성해야 합니다. Orchestrator Appliance를 사용하는 경우 해당 폴더는 **vco**라고 명명되고 **/var/run/vco**에 위치하게 됩니다.

워크플로 및 JavaScript의 서버 파일 시스템 액세스 설정을 변경하여 다른 폴더에 대한 액세스를 허용할 수 있습니다. "VMware vRealize Orchestrator 설치 및 구성, 워크플로 및 작업의 서버 파일 시스템 액세스 설정"을 참조하십시오.

- 간단한 XML 문서 생성

Orchestrator 클라이언트에서 워크플로를 실행하여 테스트용으로 간단한 XML 문서를 만들 수 있습니다.

■ XML 문서에서 요소 찾기

Orchestrator 클라이언트에서 워크플로를 실행하여 간단한 XML 문서 생성 워크플로를 통해 만든 XML에서 요소를 찾을 수 있습니다.

■ XML 문서 수정

Orchestrator 클라이언트에서 워크플로를 실행하여 간단한 XML 문서 생성 워크플로를 통해 만든 XML을 수정할 수 있습니다.

■ XML에서 예제 주소록 생성

Orchestrator 클라이언트에서 워크플로를 실행하여 테스트용으로 주소록을 만들 수 있습니다.

간단한 XML 문서 생성

Orchestrator 클라이언트에서 워크플로를 실행하여 테스트용으로 간단한 XML 문서를 만들 수 있습니다.

사전 요구 사항

- 로그인한 사용자 계정에 XML 워크플로를 실행하는 데 필요한 권한이 있는지 확인합니다.
- Orchestrator 서버 시스템의 루트에 `c:/orchestrator` 폴더를 생성했는지 또는 다른 폴더에 대한 액세스 권한을 설정했는지 확인합니다.

절차

- 1 Orchestrator 클라이언트에서 **워크플로** 보기를 클릭합니다.
- 2 계층형 워크플로 목록에서 **라이브러리 > XML > 샘플 XML(단순형)**을 열어 간단한 XML 문서 생성 워크플로로 이동합니다.
- 3 간단한 XML 문서 생성 워크플로를 마우스 오른쪽 버튼으로 클릭하고 **워크플로 시작**을 선택합니다.
- 4 생성할 XML 문서의 파일 경로를 입력합니다.

예를 들면 `c:/orchestrator/filename.xml`과 같습니다.

- 5 **제출**을 클릭하여 워크플로를 실행합니다.

결과

워크플로에서 사용자 목록이 포함된 XML 문서를 생성합니다. 각 항목의 특성은 `user ID` 및 `name`입니다.

XML 문서에서 요소 찾기

Orchestrator 클라이언트에서 워크플로를 실행하여 간단한 XML 문서 생성 워크플로를 통해 만든 XML에서 요소를 찾을 수 있습니다.

사전 요구 사항

- 로그인한 사용자 계정에 XML 워크플로를 실행하는 데 필요한 권한이 있는지 확인합니다.
- Orchestrator 서버 시스템의 루트에 `c:/orchestrator` 폴더를 생성했는지 또는 다른 폴더에 대한 액세스 권한을 설정했는지 확인합니다.

절차

- 1 Orchestrator 클라이언트에서 **워크플로** 보기를 클릭합니다.
- 2 계층형 워크플로 목록에서 **라이브러리 > XML > 샘플 XML(단순형)**을 열어 문서에서 요소 찾기 워크플로로 이동합니다.
- 3 문서에서 요소 찾기 워크플로를 마우스 오른쪽 버튼으로 클릭하고 **워크플로 시작**을 선택합니다.
- 4 XML 문서의 파일 경로를 입력합니다.

예를 들면 `c:/orchestrator/filename.xml`과 같습니다.

- 5 **제출**을 클릭하여 워크플로를 실행합니다.

워크플로에서 요소를 검색하여 시스템 로그에 결과를 표시합니다.

다음에 수행할 작업

결과를 보려면 Orchestrator 클라이언트에서 완료된 워크플로 실행을 선택하고 **스키마** 탭에서 **로그**를 클릭합니다.

XML 문서 수정

Orchestrator 클라이언트에서 워크플로를 실행하여 간단한 XML 문서 생성 워크플로를 통해 만든 XML을 수정할 수 있습니다.

사전 요구 사항

- 로그인한 사용자 계정에 XML 워크플로를 실행하는 데 필요한 권한이 있는지 확인합니다.
- Orchestrator 서버 시스템의 루트에 `c:/orchestrator` 폴더를 생성했는지 또는 다른 폴더에 대한 액세스 권한을 설정했는지 확인합니다.

절차

- 1 Orchestrator 클라이언트에서 **워크플로** 보기를 클릭합니다.
- 2 계층형 워크플로 목록에서 **라이브러리 > XML > 샘플 XML(단순형)**을 열어 XML 문서 수정 워크플로로 이동합니다.
- 3 XML 문서 수정 워크플로를 마우스 오른쪽 버튼으로 클릭하고 **워크플로 시작**을 선택합니다.
- 4 입력 및 출력 파일 경로를 제공합니다.

- a 수정할 XML 문서의 파일 경로를 입력합니다.

예를 들면 `c:/orchestrator/filename.xml`과 같습니다.

- b 수정한 XML 문서의 파일 경로를 입력합니다.

예를 들면 `c:/orchestrator/filename.xml`과 같습니다.

참고 두 필드 모두에 같은 파일 경로를 입력한 경우 워크플로는 원래 파일을 수정된 파일로 덮어씹습니다. 출력 파일을 존재하지 않는 파일로 입력한 경우 워크플로는 수정된 파일을 생성합니다.

5 제출을 클릭하여 워크플로를 실행합니다.

결과

워크플로에서 요소를 검색하고 해당 요소가 발견된 항목을 수정합니다.

XML에서 예제 주소록 생성

Orchestrator 클라이언트에서 워크플로를 실행하여 테스트용으로 주소록을 만들 수 있습니다.

사전 요구 사항

- 로그인한 사용자 계정에 XML 워크플로를 실행하는 데 필요한 권한이 있는지 확인합니다.
- Orchestrator 서버 시스템의 루트에 `c:/orchestrator` 폴더를 생성했는지 또는 다른 폴더에 대한 액세스 권한을 설정했는지 확인합니다.

절차

- 1 Orchestrator 클라이언트에서 **워크플로** 보기를 클릭합니다.
- 2 계층형 워크플로 목록에서 **라이브러리 > XML > 샘플 XML(주소록)**을 열어 전체 주소록 테스트 워크플로로 이동합니다.
- 3 전체 주소록 테스트 워크플로를 마우스 오른쪽 버튼으로 클릭하고 **워크플로 시작**을 선택합니다.
- 4 주소록 폴더의 경로를 입력합니다.

예를 들면 `c:/orchestrator/foldername`과 같습니다.

폴더가 없는 경우 워크플로에서 자동으로 생성합니다.

5 제출을 클릭하여 워크플로를 실행합니다.

결과

워크플로에서 DTD, XML 및 CSS 파일을 생성하고, 스타일시트를 추가하고, 지정된 폴더에 파일을 저장합니다.

SMTP(Simple Mail Transfer Protocol)를 사용하는 메일 플러그인을 사용하여 워크플로에서 이메일 메시지를 전송할 수 있습니다. 예를 들어 워크플로에 사용자 상호 작용이 필요하거나 워크플로 실행이 완료된 경우 지정된 주소로 이메일을 전송하는 워크플로를 생성할 수 있습니다.

본 장은 다음 항목을 포함합니다.

- 기본 SMTP 연결 정의
- 메일 플러그인 샘플 워크플로 사용

기본 SMTP 연결 정의

메일 플러그인은 Orchestrator 서버와 함께 설치되며, 이메일 알림을 보내고 받는 데 사용됩니다. 이메일 알림을 보내고 받기 위해 SMTP 서버에 인증할 수 있는 기본 이메일 계정을 설정할 수 있습니다.

참고 Orchestrator에서 메일을 구성할 때 로드 밸런서를 방지하십시오. SMTP_HOST_UNREACHABLE 오류가 발생할 수 있습니다.

절차

- 1 Orchestrator 클라이언트에 관리자로 로그인합니다.
- 2 Orchestrator 클라이언트에서 **워크플로** 보기를 클릭합니다.
- 3 워크플로 계층 목록에서 **라이브러리 > 메일**을 확장하여 메일 구성 워크플로로 이동합니다.
- 4 메일 구성 워크플로를 마우스 오른쪽 버튼으로 클릭하고 **워크플로 시작**을 선택합니다.
- 5 필요한 정보를 입력합니다.

| 텍스트 상자 | 설명 |
|----------|---|
| SMTP 호스트 | SMTP 서버의 IP 주소 또는 도메인 이름을 입력합니다. |
| SMTP 포트 | SMTP 구성과 일치시킬 포트 번호를 입력합니다. 기본 SMTP 포트는 25입니다. |
| 사용자 이름 | 유효한 이메일 계정을 입력합니다. Orchestrator에서 이메일을 보내는 데 사용하는 이메일 계정입니다. |

| 텍스트 상자 | 설명 |
|---------------|---|
| 암호 | 사용자 이름과 연결된 암호를 입력합니다. |
| 보낸 사람 이름 및 주소 | Orchestrator에서 보내는 모든 이메일에 표시할 보낸 사람 정보를 입력합니다. |

6 제출을 클릭하여 워크플로를 실행합니다.

메일 플러그인 샘플 워크플로 사용

사용자 지정 워크플로에서 메일 플러그인의 샘플 워크플로를 호출하여 사용자 지정 워크플로에 이메일 기능을 구현할 수 있습니다. 예제 워크플로를 실행하여 Orchestrator와 SMTP 서버 간의 상호 작용을 테스트할 수 있습니다.

■ 메일 플러그인 샘플 워크플로 액세스

Orchestrator 클라이언트를 통해 메일 플러그인 샘플 워크플로에 액세스할 수 있습니다.

■ 메일 플러그인 샘플 워크플로

샘플 메일 플러그인 워크플로를 통합하여 사용자 지정 워크플로를 개선할 수 있습니다.

메일 플러그인 샘플 워크플로 액세스

Orchestrator 클라이언트를 통해 메일 플러그인 샘플 워크플로에 액세스할 수 있습니다.

사전 요구 사항

로그인한 사용자 계정에 메일 워크플로를 실행하는 데 필요한 권한이 있는지 확인합니다.

절차

1 Orchestrator 클라이언트에서 **워크플로** 보기를 클릭합니다.

2 계층형 목록을 **라이브러리 > 메일**로 확장합니다.

다음에 수행할 작업

샘플 워크플로를 검토하고 실행합니다.

메일 플러그인 샘플 워크플로

샘플 메일 플러그인 워크플로를 통합하여 사용자 지정 워크플로를 개선할 수 있습니다.

메일 워크플로는 Orchestrator 클라이언트의 **워크플로** 보기에서 **라이브러리 > 메일**을 통해 액세스할 수 있습니다.

| 워크플로 이름 | 설명 |
|-----------------------|---|
| 메일 구성 | SMTP 서버 연결, SMTP 인증 계정, 보낸 사람의 주소 및 표시 이름 등을 정의합니다. |
| 메시지 검색 | POP3 프로토콜을 사용하여 지정된 이메일 계정의 메시지를 검색합니다. |
| 메시지 검색(MailClient 사용) | MailClient 클래스에서 제공되는 새 스크립팅 API를 사용하여 특정 이메일 계정의 메시지를 삭제하지 않고 검색합니다. |

| 워크플로 이름 | 설명 |
|----------------|--|
| 알림 보내기 | 지정된 콘텐츠가 포함된 이메일을 지정된 이메일 주소로 전송합니다. 선택적 매개 변수를 지정하지 않으면 워크플로에서 메일 구성 워크플로를 통해 설정된 기본값을 사용합니다. |
| 메일링 목록에 알림 보내기 | 지정된 콘텐츠가 포함된 이메일을 지정된 이메일 주소 목록, CC 목록 및 BCC 목록으로 전송합니다. 선택적 매개 변수를 지정하지 않으면 워크플로에서 메일 구성 워크플로를 통해 설정된 기본값을 사용합니다. |

네트워크 플러그인 사용

11

네트워크 플러그인을 사용하여 워크플로에서 텔넷, FTP, POP3 및 IMAP 프로토콜을 구현할 수 있습니다. POP3 및 IMAP 구현은 이메일 다운로드 및 읽기를 지원합니다. 네트워크 플러그인은 메일 플러그인과 함께 워크플로에서 전체 이메일 전송 및 수신 기능을 제공합니다.

열거형 플러그인 사용

12

열거형 플러그인을 사용하여 워크플로에서 일반적인 열거 유형을 구현할 수 있습니다.

본 장은 다음 항목을 포함합니다.

■ 표준 시간대 코드

표준 시간대 코드

표준 시간대 코드를 `Enums:MSTimeZone` 열거형의 가능한 값으로 사용할 수 있습니다.

| 표준 시간대 코드 | 표준 시간대 이름 | 설명 |
|-----------|-------------|-------------------------------------|
| 000 | 날짜 변경선 표준시 | (GMT-12:00) 날짜 변경선 서쪽 |
| 001 | 사모아 표준시 | (GMT-11:00) 미드웨이 군도, 사모아 |
| 002 | 하와이 표준시 | (GMT-10:00) 하와이 |
| 003 | 알래스카 표준시 | (GMT-09:00) 알래스카 |
| 004 | 태평양 표준시 | (GMT-08:00) 태평양 표준시(미국 및 캐나다), 티후아나 |
| 010 | 산지 표준시 | (GMT-07:00) 산지 표준시(미국 및 캐나다) |
| 013 | 멕시코 표준시 2 | (GMT-07:00) 치와와, 라파스, 마사틀란 |
| 015 | 미국 산지 표준시 | (GMT-07:00) 애리조나 |
| 020 | 중부 표준시 | (GMT-06:00) 중부 표준시(미국 및 캐나다) |
| 025 | 캐나다 중부 표준시 | (GMT-06:00) 서스캐처원 |
| 030 | 멕시코 표준시 | (GMT-06:00) 과달라하라, 멕시코시티, 몬테레이 |
| 033 | 중앙 아메리카 표준시 | (GMT-06:00) 중앙 아메리카 |
| 035 | 동부 표준시 | (GMT-05:00) 동부 표준시(미국 및 캐나다) |
| 040 | 미국 동부 표준시 | (GMT-05:00) 인디애나(동부) |
| 045 | SA 태평양 표준시 | (GMT-05:00) 보고타, 리마, 키토 |

| 표준 시간대 코드 | 표준 시간대 이름 | 설명 |
|-----------|---------------------|---|
| 050 | 대서양 표준시 | (GMT-04:00) 대서양 표준시(캐나다) |
| 055 | SA 서부 표준시 | (GMT-04:00) 카라카스, 라파스 |
| 056 | 태평양 SA 표준시 | (GMT-04:00) 산티아고 |
| 060 | 뉴펀들랜드 및 래브라도 반도 표준시 | (GMT-03:30) 뉴펀들랜드 및 래브라도 |
| 065 | 동남부 아메리카 표준시 | (GMT-03:00) 브라질리아 |
| 070 | SA 동부 표준시 | (GMT-03:00) 부에노스아이레스, 조지타운 |
| 073 | 그린란드 표준시 | (GMT-03:00) 그린란드 |
| 075 | 중부 대서양 표준시 | (GMT-02:00) 중부 대서양 |
| 080 | 아조레스 표준시 | (GMT-01:00) 아조레스 |
| 083 | 카보베르데 표준시 | (GMT-01:00) 카보베르데 제도 |
| 085 | GMT 표준시 | (GMT) 그리니치 표준시: 더블린, 에든버러, 리스본, 런던 |
| 090 | 그리니치 표준시 | (GMT) 카사블랑카, 몬로비아 |
| 095 | 중앙 유럽 표준시 | (GMT+01:00) 베오그라드, 브라티슬라바, 부다페스트, 류블랴나, 프라하 |
| 100 | 중앙 유럽 표준시 | (GMT+01:00) 사라예보, 스코페, 바르샤바, 자그레브 |
| 105 | 로망스 표준시 | (GMT+01:00) 브뤼셀, 코펜하겐, 마드리드, 파리 |
| 110 | 서유럽 표준시 | (GMT+01:00) 암스테르담, 베를린, 베른, 로마, 스톡홀름, 빈 |
| 113 | 서중앙 아프리카 표준시 | (GMT+01:00) 서중앙 아프리카 |
| 115 | 동유럽 표준시 | (GMT+02:00) 부카레스트 |
| 120 | 이집트 표준시 | (GMT+02:00) 카이로 |
| 125 | FLE 표준시 | (GMT+02:00) 헬싱키, 키예프, 리가, 소피아, 탈린, 빌뉴스 |
| 130 | GTB 표준시 | (GMT+02:00) 아테네, 이스탄불, 민스크 |
| 135 | 이스라엘 표준시 | (GMT+02:00) 예루살렘 |
| 140 | 남아프리카 표준시 | (GMT+02:00) 하라레, 프리토리아 |
| 145 | 러시아 표준시 | (GMT+03:00) 모스크바, 상트 페테르부르크, 볼고그라드 |
| 150 | 아랍 표준시 | (GMT+03:00) 쿠웨이트, 리야드 |
| 155 | 동아프리카 표준시 | (GMT+03:00) 나이로비 |
| 158 | 아랍 표준시 | (GMT+03:00) 바그다드 |
| 160 | 이란 표준시 | (GMT+03:30) 테헤란 |

| 표준 시간대 코드 | 표준 시간대 이름 | 설명 |
|-----------|----------------|--------------------------------------|
| 165 | 아랍 표준시 | (GMT+04:00) 아부다비, 무스카트 |
| 170 | 코코스스 표준시 | (GMT+04:00) 바쿠, 트빌리시, 예레반 |
| 175 | 아프가니스탄 표준시 | (GMT+04:30) 카불 |
| 180 | 예카테린부르크 표준시 | (GMT+05:00) 예카테린부르크 |
| 185 | 서아시아 표준시 | (GMT+05:00) 이슬라마바드, 카라치, 타슈켄트 |
| 190 | 인도 표준시 | (GMT+05:30) 첸나이, 콜카타, 뭄바이, 뉴델리 |
| 193 | 네팔 표준시 | (GMT+05:45) 카트만두 |
| 195 | 중앙 아시아 표준시 | (GMT+06:00) 아스타나, 다카 |
| 200 | 스리랑카 표준시 | (GMT+06:00) 스리자야와르데네푸라 |
| 201 | 북부 중앙 아시아 표준시 | (GMT+06:00) 알마티, 노브시비르스크 |
| 203 | 미얀마 표준시 | (GMT+06:30) 양곤(랑군) |
| 205 | 동남 아시아 표준시 | (GMT+07:00) 방콕, 하노이, 자카르타 |
| 207 | 북아시아 표준시 | (GMT+07:00) 크라스노야르스크 |
| 210 | 중국 표준시 | (GMT+08:00) 베이징, 충칭, 홍콩 특별 행정구, 우루무치 |
| 215 | 싱가포르 표준시 | (GMT+08:00) 쿼라룸푸르, 싱가포르 |
| 220 | 타이베이 표준시 | (GMT+08:00) 타이베이 |
| 225 | 서부 오스트레일리아 표준시 | (GMT+08:00) 퍼스 |
| 227 | 북아시아 동부 표준시 | (GMT+08:00) 이르쿠츠크, 울란바토르 |
| 230 | 대한민국 표준시 | (GMT+09:00) 서울 |
| 235 | 도쿄 표준시 | (GMT+09:00) 오사카, 삿포로, 도쿄 |
| 240 | 야쿠츠크 표준시 | (GMT+09:00) 야쿠츠크 |
| 245 | 오스트레일리아 중부 표준시 | (GMT+09:30) 다윈 |
| 250 | 중부 오스트레일리아 표준시 | (GMT+09:30) 애들레이드 |
| 255 | 오스트레일리아 동부 표준시 | (GMT+10:00) 캔버라, 멜버른, 시드니 |
| 260 | 동부 오스트레일리아 표준시 | (GMT+10:00) 브리즈번 |
| 265 | 태즈메이니아 표준시 | (GMT+10:00) 호바트 |
| 270 | 블라디보스토크 표준시 | (GMT+10:00) 블라디보스토크 |
| 275 | 서태평양 표준시 | (GMT+10:00) 램, 포트모르즈비 |
| 280 | 중앙 태평양 표준시 | (GMT+11:00) 마가단, 솔로몬 제도, 뉴칼레도니아 |
| 285 | 피지 표준시 | (GMT+12:00) 피지, 캄차카 반도, 마셜 제도 |

| 표준 시간대 코드 | 표준 시간대 이름 | 설명 |
|-----------|-----------|-----------------------|
| 290 | 뉴질랜드 표준시 | (GMT+12:00) 오클랜드, 웰링턴 |
| 300 | 통가 표준시 | (GMT+13:00) 누쿠알로파 |

워크플로 설명서 플러그인 사용

13

워크플로 설명서 플러그인을 사용하여 특정 워크플로 또는 워크플로 범주에 대한 PDF 설명서를 생성할 수 있습니다.

본 장은 다음 항목을 포함합니다.

- 워크플로 설명서 플러그인용 워크플로 라이브러리
- 워크플로 설명서 생성

워크플로 설명서 플러그인용 워크플로 라이브러리

워크플로 설명서 플러그인 워크플로를 사용하여 특정 워크플로 또는 워크플로 범주에 대한 PDF 설명서를 생성할 수 있습니다.

이러한 워크플로는 Orchestrator 클라이언트의 **워크플로 보기**에서 **라이브러리 > 워크플로 설명서**를 통해 액세스할 수 있습니다.

| 워크플로 이름 | 설명 |
|----------------------|----------------------------|
| 워크플로에 대한 설명서 가져오기 | 선택한 워크플로에 대한 정보를 생성합니다. |
| 워크플로 범주에 대한 설명서 가져오기 | 선택한 워크플로 범주에 대한 정보를 생성합니다. |

워크플로 설명서 생성

선택한 워크플로 또는 워크플로 폴더에 대한 설명서를 언제든지 PDF 형식으로 내보낼 수 있습니다.

내보낸 문서에는 선택한 워크플로 또는 폴더의 워크플로에 대한 자세한 정보가 포함되어 있습니다. 각 워크플로에 대한 정보에는 이름, 워크플로의 버전 기록, 특성, 매개 변수 프레젠테이션, 워크플로 스키마, 워크플로 작업 등이 있습니다. 또한 설명서에서는 사용된 작업에 대한 소스 코드를 제공합니다.

절차

- 1 Orchestrator 클라이언트의 드롭다운 목록에서 **실행** 또는 **설계**를 선택합니다.
- 2 **워크플로 보기**를 클릭합니다.
- 3 설명서를 생성할 워크플로 또는 워크플로 폴더로 이동하여 마우스 오른쪽 버튼으로 클릭합니다.

4 설명서 생성을 선택합니다.

5 PDF 파일을 저장할 폴더를 찾아서 파일 이름을 제공하고 **저장**을 클릭합니다.

결과

선택한 워크플로 또는 폴더의 워크플로에 대한 정보가 포함된 PDF 파일이 시스템에 저장됩니다.

HTTP-REST 플러그인 사용

14

HTTP-REST 플러그인을 사용하면 vRealize Orchestrator와 REST 호스트 간의 상호 작용을 통해 REST 웹 서비스를 관리할 수 있습니다. 구성 워크플로를 실행하여 REST 서비스와 해당 작업을 인벤토리 개체로 정의하고, 정의된 개체에서 REST 작업을 수행할 수 있습니다.

이 플러그인에는 REST 호스트 관리 및 REST 작업 호출과 관련된 표준 워크플로 집합이 포함되어 있습니다. 사용자 지정 워크플로를 생성하여 REST 환경에서 작업을 자동화할 수도 있습니다.

본 장은 다음 항목을 포함합니다.

- [HTTP-REST 플러그인 구성](#)
- [REST 작업에서 새 워크플로 생성](#)
- [REST 작업 호출](#)

HTTP-REST 플러그인 구성

Orchestrator 클라이언트를 사용하여 HTTP-REST 플러그인을 구성해야 합니다.

구성 워크플로

구성 워크플로 범주에는 REST 호스트를 관리하는 데 유용한 워크플로가 포함됩니다.

이러한 워크플로는 Orchestrator 클라이언트의 **워크플로 보기**에서 **라이브러리 > HTTP-REST > 구성**을 통해 액세스할 수 있습니다.

| 워크플로 이름 | 설명 |
|-------------------------------------|---|
| REST 호스트 추가 | 플러그인 인벤토리에 REST 호스트를 추가합니다. |
| 문자열로 제공된 Swagger 사양에 따라 REST 호스트 추가 | 문자열로 제공된 Swagger 사양 웹 리소스를 기반으로 REST 호스트를 추가합니다. |
| URL의 Swagger 사양에 따라 REST 호스트 추가 | 특정 URL에서 사용 가능한 Swagger 사양을 기반으로 REST 호스트를 추가합니다. |
| REST 작업 추가 | REST 호스트에 작업을 추가합니다. |
| REST 호스트에 스키마 추가 | REST 호스트에 XSD 스키마를 추가합니다. |
| REST 호스트 복제 | REST 호스트의 복제본을 생성합니다. |

| 워크플로 이름 | 설명 |
|-------------------|-----------------------------------|
| REST 작업 복제 | REST 작업의 복제본을 생성합니다. |
| 플러그인 구성 다시 로드 | 플러그인 인벤토리에서 REST 호스트 목록을 새로 고칩니다. |
| REST 호스트 제거 | 플러그인 인벤토리에서 REST 호스트를 제거합니다. |
| REST 작업 제거 | REST 호스트에서 작업을 제거합니다. |
| REST 호스트에서 스키마 제거 | REST 호스트에서 연결된 모든 XSD 스키마를 제거합니다. |
| REST 호스트 업데이트 | 플러그인 인벤토리에서 REST 호스트를 업데이트합니다. |
| REST 작업 업데이트 | REST 호스트에서 작업을 업데이트합니다. |

Kerberos 인증 구성

PowerShell 호스트를 추가하고 관리할 때 Kerberos 인증을 사용할 수 있습니다.

Kerberos 인증을 사용하면 도메인 사용자는 WinRM을 통해 원격 PowerShell 사용 가능 시스템에서 명령을 실행할 수 있습니다.

절차

- 1 WinRM 서비스에서 Kerberos 인증을 사용하도록 설정합니다.
 - a Kerberos 인증이 허용되는지 여부를 확인하려면 다음 명령을 실행합니다.


```
c:\> winrm get winrm/config/service
```
 - b Kerberos 인증을 사용하도록 설정하려면 다음 명령을 실행합니다.


```
c:\> winrm set winrm/config/service/auth @{Kerberos="true"}
```
- 2 WinRM 클라이언트에서 Kerberos 인증을 사용하도록 설정합니다.
 - a Kerberos 인증이 허용되는지 여부를 확인하려면 다음 명령을 실행합니다.


```
c:\> winrm get winrm/config/client
```
 - b Kerberos 인증을 사용하도록 설정하려면 다음 명령을 실행합니다.


```
c:\> winrm set winrm/config/client/auth @{Kerberos="true"}
```
- 3 WinRM 서비스에 대한 연결을 테스트하려면 다음 명령을 실행합니다.


```
c:\> winrm identify -r:http://winrm_server:5985 -auth:Kerberos -u:user_name -p:password -encoding:utf-8
```

4 krb5.conf 파일을 만들어 다음 위치에 저장합니다.

| 운영 체제 | 경로 |
|---------|---|
| Windows | C:\Program Files\Common Files\VMware\VMware vCenter Server - Java Components\lib\security\ |
| Linux | /usr/java/jre-vmware/lib/security/(외부 vRealize Orchestrator용). /etc/krb5.conf(vRealize Orchestrator용)가 vRealize Automation에 작성됩니다. |

krb5.conf 파일의 구조는 다음과 같습니다.

```
[libdefaults]
default_realm = YOURDOMAIN.COM
udp_preference_limit = 1
[realms]
YOURDOMAIN.COM = {
kdc = kdc.yourdomain.com
default_domain = yourdomain.com
}
[domain_realm]
.yourdomain.com=YOURDOMAIN.COM
yourdomain.com=YOURDOMAIN.COM
```

krb5.conf는 해당 값과 함께 특정한 구성 매개 변수를 포함해야 합니다.

| Kerberos 구성 태그 | 세부 정보 |
|----------------|---|
| default_realm | Active Directory 서버를 인증할 때 클라이언트에서 사용하는 기본 Kerberos 영역입니다. 참고 대문자로 입력해야 합니다. |
| kdc | KDC(키 배포 센터) 역할을 하고 Kerberos 티켓을 발행하는 도메인 컨트롤러입니다. |
| default_domain | 정규화된 도메인 이름을 생성하는 데 사용되는 기본 도메인입니다. 참고 이 태그는 Kerberos 4 호환성을 위해 사용됩니다. |

참고 기본적으로 Java Kerberos 구성은 UDP 프로토콜을 사용합니다. TCP 프로토콜만 사용하려면 udp_preference_limit 매개 변수를 값 **1**로 지정해야 합니다.

참고 Kerberos 인증을 위해서는 FQDN(정규화된 도메인 이름) 호스트 주소가 필요합니다.

중요 krb5.conf 파일을 추가하거나 수정한 경우 Orchestrator 서버 서비스를 다시 시작해야 합니다.

REST 호스트 추가

워크플로를 실행하여 REST 호스트를 추가하고 호스트 연결 매개 변수를 구성할 수 있습니다.

절차

- 1 Orchestrator 클라이언트에 관리자로 로그인합니다.
- 2 Orchestrator 클라이언트에서 **워크플로** 보기를 클릭합니다.
- 3 계층형 워크플로 목록에서 **라이브러리 > HTTP-REST > 구성**을 확장하여 REST 호스트 추가 워크플로로 이동합니다.
- 4 REST 호스트 추가 워크플로를 마우스 오른쪽 버튼으로 클릭하고 **워크플로 시작**을 선택합니다.
- 5 **이름** 텍스트 상자에 호스트의 이름을 입력합니다.
- 6 **URL** 텍스트 상자에 호스트의 주소를 입력합니다.

참고 Kerberos 인증을 위해서는 FQDN(정규화된 도메인 이름) 호스트 주소가 필요합니다.

- 7 **연결 시간 초과** 텍스트 상자에 연결이 시간 초과될 때까지 소요되는 시간(초)을 입력합니다.
- 8 **작업 시간 초과** 텍스트 상자에 작업이 시간 초과될 때까지 소요되는 시간(초)을 입력합니다.
- 9 **예**를 선택하여 REST 호스트 인증서를 수락합니다.

Orchestrator 서버 신뢰 저장소에 인증서가 추가됩니다.

- 10 인증 유형을 선택합니다.

| 옵션 | 설명 |
|-----------|--|
| 없음 | 인증이 필요하지 않습니다. |
| OAuth 1.0 | 필요한 인증 매개 변수를 제공합니다. |
| OAuth 2.0 | 인증 토큰을 제공합니다. |
| 기본 | 기본 액세스 인증을 제공합니다. 세션 모드를 선택합니다. <ul style="list-style-type: none"> ■ 공유 세션을 선택한 경우 공유 세션의 자격 증명을 제공합니다. ■ 사용자별 세션을 선택한 경우 Orchestrator 클라이언트가 로그인된 사용자로 부터 자격 증명을 검색합니다. |
| 다이제스트 | 암호화를 사용하는 다이제스트 액세스 인증을 제공합니다. 세션 모드를 선택합니다. <ul style="list-style-type: none"> ■ 공유 세션을 선택한 경우 공유 세션의 자격 증명을 제공합니다. ■ 사용자별 세션을 선택한 경우 Orchestrator 클라이언트가 로그인된 사용자로 부터 자격 증명을 검색합니다. |

| 옵션 | 설명 |
|-----------------|--|
| NTLM | <p>Windows SSP(보안 지원 공급자) 프레임워크 내에 NTLM(NT LAN Manager) 액세스 인증을 제공합니다.</p> <p>세션 모드를 선택합니다.</p> <ul style="list-style-type: none"> ■ 공유 세션을 선택한 경우 공유 세션의 자격 증명을 제공합니다. ■ 사용자별 세션을 선택한 경우 Orchestrator 클라이언트가 로그인된 사용자로부터 자격 증명을 검색합니다. <p>NTLM 설정을 제공합니다.</p> |
| Kerberos | <p>Kerberos 액세스 인증을 제공합니다.</p> <p>세션 모드를 선택합니다.</p> <ul style="list-style-type: none"> ■ 공유 세션을 선택한 경우 공유 세션의 자격 증명을 제공합니다. ■ 사용자별 세션을 선택한 경우 Orchestrator 클라이언트가 로그인된 사용자로부터 자격 증명을 검색합니다. |

11 프록시를 사용하려면 프록시 서버의 주소 및 포트를 입력합니다.

a (선택 사항) 프록시 인증 유형을 선택합니다.

| 옵션 | 설명 |
|-----------|---|
| 없음 | 인증이 필요하지 않습니다. |
| 기본 | <p>기본 액세스 인증을 제공합니다.</p> <p>세션 모드를 선택합니다.</p> <ul style="list-style-type: none"> ■ 공유 세션을 선택한 경우 공유 세션의 자격 증명을 제공합니다. ■ 사용자별 세션을 선택한 경우 Orchestrator 클라이언트가 로그인된 사용자로부터 자격 증명을 검색합니다. |

12 대상 호스트 이름을 서버 인증서에 저장된 이름과 일치시킬지 여부를 선택합니다.

13 (선택 사항) 서버에 인증하는 데 사용할 키 저장소 항목을 선택합니다. 키 저장소 항목은 PrivateKeyEntry 유형이어야 합니다.

14 **제출**을 클릭하여 워크플로를 실행합니다.

결과

워크플로가 성공적으로 실행되면 **인벤토리** 보기에 REST 호스트가 표시됩니다.

다음에 수행할 작업

작업 및 XSD 스키마를 REST 호스트에 추가하고 **인벤토리** 보기에서 워크플로를 실행할 수 있습니다.

REST 작업 추가

워크플로를 실행하여 플러그인 인벤토리의 REST 호스트에 작업을 추가할 수 있습니다.

사전 요구 사항

- Orchestrator 클라이언트에 관리자로 로그인되어 있는지 확인합니다.

- REST 호스트에 연결되었는지 여부를 **인벤토리** 보기에서 확인합니다.

절차

- 1 Orchestrator 클라이언트에서 **워크플로** 보기를 클릭합니다.
- 2 계층형 워크플로 목록에서 **라이브러리 > HTTP-REST > 구성**을 확장하여 REST 작업 추가 워크플로로 이동합니다.
- 3 REST 작업 추가 워크플로를 마우스 오른쪽 버튼으로 클릭하고 **워크플로 시작**을 선택합니다.
- 4 작업을 추가할 호스트를 선택합니다.
- 5 **이름** 텍스트 상자에 작업의 이름을 입력합니다.
- 6 **템플릿 URL** 텍스트 상자에 URL의 작업 부분만 입력합니다.

작업을 실행할 때 제공되는 매개 변수의 자리 표시자를 포함할 수 있습니다.

URL 구문의 예는 다음과 같습니다.

/customer/{id}/orders?date={date}

- 7 작업에서 사용할 HTTP 메서드를 선택합니다.
POST 또는 **PUT**을 선택한 경우 메서드의 Content-Type 요청 헤더를 제공할 수 있습니다.
- 8 **제출**을 클릭하여 워크플로를 실행합니다.

다음에 수행할 작업

인벤토리 보기에서 작업에 대한 워크플로를 실행할 수 있습니다.

REST 호스트에 스키마 추가

워크플로를 실행하여 플러그인 인벤토리의 REST 호스트에 XSD 스키마를 추가할 수 있습니다.

XSD 스키마는 웹 서비스에서 입력 및 출력 콘텐츠로 사용되는 XML 문서를 설명합니다. 이러한 스키마를 호스트에 연결하면 REST 작업에서 워크플로를 생성할 때 필요한 XML 요소를 입력으로 지정할 수 있습니다.

사전 요구 사항

- Orchestrator 클라이언트에 관리자로 로그인되어 있는지 확인합니다.
- REST 호스트에 연결되었는지 여부를 **인벤토리** 보기에서 확인합니다.

절차

- 1 Orchestrator 클라이언트에서 **워크플로** 보기를 클릭합니다.
- 2 계층형 워크플로 목록에서 **라이브러리 > HTTP-REST > 구성**을 확장하여 REST 호스트에 스키마 추가 워크플로로 이동합니다.
- 3 REST 호스트에 스키마 추가 워크플로를 마우스 오른쪽 버튼으로 클릭하고 **워크플로 시작**을 선택합니다.

- 4 XSD 스키마를 추가할 호스트를 선택합니다.
- 5 URL에서 스키마를 로드할지 여부를 선택합니다.

| 옵션 | 작업 |
|-----|------------------|
| 예 | 스키마의 URL을 입력합니다. |
| 아니요 | 스키마 콘텐츠를 제공합니다. |

- 6 **제출**을 클릭하여 워크플로를 실행합니다.

REST 작업에서 새 워크플로 생성

REST 작업에서 사용자 지정 워크플로를 생성할 수 있습니다.

사용자 지정 생성된 워크플로를 상위 수준 워크플로에 통합할 수 있습니다. 워크플로 개발에 대한 자세한 내용은 "vRealize Orchestrator 개발자 가이드"를 참조하십시오.

사전 요구 사항

- Orchestrator 클라이언트에 관리자로 로그인되어 있는지 확인합니다.
- REST 호스트에 연결되었는지 여부를 **인벤토리** 보기에서 확인합니다.

절차

- 1 Orchestrator 클라이언트에서 **워크플로** 보기를 클릭합니다.
- 2 계층형 워크플로 목록에서 **라이브러리 > HTTP-REST**를 확장하여 REST 작업에서 새 워크플로 생성 워크플로로 이동합니다.
- 3 REST 작업에서 새 워크플로 생성 워크플로를 마우스 오른쪽 버튼으로 클릭하고 **워크플로 시작**을 선택합니다.
- 4 사용 가능한 작업 목록에서 **REST** 작업을 선택합니다.

작업에 입력이 사용되고 XSD 스키마가 해당 호스트에 추가되는 경우 요청 입력 유형을 지정할 수 있습니다.

- 5 **이름** 텍스트 상자에 생성할 워크플로의 이름을 입력합니다.
- 6 새 워크플로를 생성할 워크플로 폴더를 선택합니다.

워크플로 라이브러리에서 기존 폴더를 선택할 수 있습니다.

- 7 **제출**을 클릭하여 워크플로를 실행합니다.

REST 작업 호출

REST 요청을 만들려면 구성된 REST 작업을 호출하거나, 구성된 REST 작업을 템플릿으로 사용하고 런타임에 매개 변수를 대체하여 REST 작업을 동적으로 호출할 수 있습니다.

REST 작업을 호출하는 몇 가지 방법이 있습니다.

- **REST 호스트 추가 및 REST 작업 추가** 워크플로를 실행하여 REST 호스트를 구성하고 이러한 호스트에 REST 작업을 연결합니다. 등록된 REST 호스트 및 REST 작업은 영구적이며, **인벤토리** 및 **리소스** 보기에서 확인할 수 있습니다.
- 이전처럼 REST 호스트를 구성하고 REST 작업을 추가하지 않고도 **라이브러리 > HTTP-REST 샘플**에서 **동적 REST 작업 호출** 워크플로를 실행하여 REST 작업을 호출합니다. 이 워크플로를 사용하면 REST 호스트 기준 URL 및 작업 매개 변수를 제공할 수 있습니다. 이 데이터는 영구적이지 않으므로 **인벤토리** 및 **리소스** 보기에서 사용할 수 없습니다.
- **라이브러리 > HTTP-REST 샘플**에서 **동적 매개 변수를 사용하여 REST 호스트 호출** 및 **동적 매개 변수를 사용하여 REST 작업 호출** 워크플로를 실행하여 REST 호스트를 구성하고, 이러한 호스트에 REST 작업을 연결하고, 나중에 사용하기 위해 구성된 REST 호스트 및 REST 작업을 템플릿으로 사용합니다. 워크플로를 실행할 때 이미 구성된 REST 호스트 및 REST 작업의 일부 매개 변수를 대체할 수 있습니다. 원래 REST 호스트 및 REST 작업은 영향을 받지 않습니다.

REST 작업 호출

REST 작업을 직접 호출합니다.

사전 요구 사항

- Orchestrator 클라이언트에 관리자로 로그인되어 있는지 확인합니다.
- REST 호스트에 연결되었는지 여부를 **인벤토리** 보기에서 확인합니다.

절차

- 1 Orchestrator 클라이언트에서 **워크플로** 보기를 클릭합니다.
- 2 계층형 워크플로 목록에서 **라이브러리 > HTTP-REST**를 확장하여 **REST 작업 호출** 워크플로로 이동합니다.
- 3 **REST 작업 호출** 워크플로를 마우스 오른쪽 버튼으로 클릭하고 **워크플로 시작**을 선택합니다.
- 4 사용 가능한 작업 목록에서 REST 작업을 선택합니다.
- 5 작업에 필요한 콘텐츠 및 입력 매개 변수를 제공합니다.
- 6 **제출**을 클릭하여 워크플로를 실행합니다.

SOAP 플러그인 사용

15

SOAP 플러그인을 사용하면 vRealize Orchestrator와 SOAP 호스트 간의 상호 작용을 통해 SOAP 웹 서비스를 관리할 수 있습니다. 구성 워크플로를 실행하여 SOAP 서비스를 인벤토리 개체로 정의하고, 정의된 개체에서 SOAP 작업을 수행할 수 있습니다.

이 플러그인에는 SOAP 호스트 관리 및 SOAP 작업 호출과 관련된 표준 워크플로 집합이 포함되어 있습니다. 사용자 지정 워크플로를 생성하여 SOAP 환경에서 작업을 자동화할 수도 있습니다.

본 장은 다음 항목을 포함합니다.

- SOAP 플러그인 구성
- SOAP 작업에서 새 워크플로 생성
- SOAP 작업 호출

SOAP 플러그인 구성

Orchestrator 클라이언트를 사용하여 SOAP 플러그인을 구성해야 합니다.

구성 워크플로

구성 워크플로 범주에는 SOAP 호스트를 관리할 수 있는 워크플로가 포함됩니다.

이러한 워크플로는 Orchestrator 클라이언트의 **워크플로 보기**에서 **라이브러리 > SOAP > 구성**을 통해 액세스할 수 있습니다.

| 워크플로 이름 | 설명 |
|---------------|---|
| SOAP 호스트 추가 | 플러그인 인벤토리에 SOAP 호스트를 추가합니다. |
| 플러그인 구성 다시 로드 | 플러그인 인벤토리에서 SOAP 호스트 목록을 새로 고칩니다. |
| SOAP 호스트 제거 | 플러그인 인벤토리에서 SOAP 호스트를 제거합니다. 경고 인벤토리에서 호스트를 제거하면 해당 호스트에서 생성된 모든 워크플로의 작동이 중지됩니다. |

| 워크플로 이름 | 설명 |
|--------------------------|--|
| SOAP 호스트 업데이트 | 플러그인 인벤토리에서 SOAP 호스트를 업데이트합니다. |
| 끝점 URL과 함께 SOAP 호스트 업데이트 | 기본 설정된 끝점 주소와 함께 SOAP 호스트를 업데이트합니다. WSDL 내에 정의된 끝점 주소 대신 새 끝점 주소를 사용하여 SOAP 메시지를 보내고 받습니다. |

SOAP 호스트 추가

워크플로를 실행하여 SOAP 호스트를 추가하고 호스트 연결 매개 변수를 구성할 수 있습니다.

절차

- 1 Orchestrator 클라이언트에 관리자로 로그인합니다.
- 2 Orchestrator 클라이언트에서 **워크플로** 보기를 클릭합니다.
- 3 계층형 워크플로 목록에서 **라이브러리 > SOAP > 구성**을 확장하여 **SOAP 호스트 추가** 워크플로로 이동합니다.
- 4 **SOAP 호스트 추가** 워크플로를 마우스 오른쪽 버튼으로 클릭하고 **워크플로 시작**을 선택합니다.
- 5 **이름** 텍스트 상자에 호스트의 이름을 입력합니다.
- 6 WSDL 콘텐츠를 텍스트로 제공할지 여부를 선택합니다.

| 옵션 | 작업 |
|-----|---------------------------------|
| 예 | WSDL 콘텐츠 텍스트 상자에 텍스트를 복사합니다. |
| 아니요 | WSDL URI 텍스트 상자에 올바른 경로를 입력합니다. |

- 7 **연결 시간 초과** 텍스트 상자에 Orchestrator에서 SOAP 호스트에 연결해야 하는 시간(초)을 입력합니다. 이 시간 내에 연결하지 않으면 연결이 시간 초과됩니다.
- 8 **요청 시간 초과** 텍스트 상자에 SOAP 요청이 시간 초과되기 전에 성공해야 하는 시간(초)을 지정합니다.
- 9 프록시를 사용할지 여부를 선택합니다.

| 옵션 | 작업 |
|-----|-------------------------|
| 예 | 프록시 주소 및 프록시 포트를 제공합니다. |
| 아니요 | 다음 단계를 진행합니다. |

10 인증 유형을 선택합니다.

| 옵션 | 설명 |
|----------|--|
| 없음 | 인증이 필요하지 않습니다. |
| 기본 | <p>기본 액세스 인증을 제공합니다.</p> <p>세션 모드를 선택합니다.</p> <ul style="list-style-type: none"> ■ 공유 세션을 선택한 경우 공유 세션의 자격 증명을 제공합니다. ■ 사용자별 세션을 선택한 경우 Orchestrator 클라이언트가 로그인된 사용자로부터 자격 증명을 검색합니다. |
| 다이제스트 | <p>암호화를 사용하는 다이제스트 액세스 인증을 제공합니다.</p> <p>세션 모드를 선택합니다.</p> <ul style="list-style-type: none"> ■ 공유 세션을 선택한 경우 공유 세션의 자격 증명을 제공합니다. ■ 사용자별 세션을 선택한 경우 Orchestrator 클라이언트가 로그인된 사용자로부터 자격 증명을 검색합니다. |
| NTLM | <p>Windows SSP(보안 지원 공급자) 프레임워크 내에 NTLM(NT LAN Manager) 액세스 인증을 제공합니다.</p> <p>세션 모드를 선택합니다.</p> <ul style="list-style-type: none"> ■ 공유 세션을 선택한 경우 공유 세션의 자격 증명을 제공합니다. ■ 사용자별 세션을 선택한 경우 Orchestrator 클라이언트가 로그인된 사용자로부터 자격 증명을 검색합니다. <p>NTLM 설정을 제공합니다.</p> |
| Kerberos | <p>Kerberos 액세스 인증을 제공합니다.</p> <p>세션 모드를 선택합니다.</p> <ul style="list-style-type: none"> ■ 공유 세션을 선택한 경우 공유 세션의 자격 증명을 제공합니다. ■ 사용자별 세션을 선택한 경우 Orchestrator 클라이언트가 로그인된 사용자로부터 자격 증명을 검색합니다. |

11 제출을 클릭하여 워크플로를 실행합니다.

결과

워크플로가 성공적으로 실행되면 **인벤토리** 보기에 SOAP 호스트가 표시됩니다.

다음에 수행할 작업

인벤토리 보기에서 SOAP 호스트 개체를 탐색하고 해당 개체에 대한 워크플로를 실행할 수 있습니다.

Kerberos 인증 구성

PowerShell 호스트를 추가하고 관리할 때 Kerberos 인증을 사용할 수 있습니다.

Kerberos 인증을 사용하면 도메인 사용자는 WinRM을 통해 원격 PowerShell 사용 가능 시스템에서 명령을 실행할 수 있습니다.

절차

- 1 WinRM 서비스에서 Kerberos 인증을 사용하도록 설정합니다.
 - a Kerberos 인증이 허용되는지 여부를 확인하려면 다음 명령을 실행합니다.

```
c:\> winrm get winrm/config/service
```

- b Kerberos 인증을 사용하도록 설정하려면 다음 명령을 실행합니다.

```
c:\> winrm set winrm/config/service/auth @{Kerberos="true"}
```

- 2 WinRM 클라이언트에서 Kerberos 인증을 사용하도록 설정합니다.
 - a Kerberos 인증이 허용되는지 여부를 확인하려면 다음 명령을 실행합니다.

```
c:\> winrm get winrm/config/client
```

- b Kerberos 인증을 사용하도록 설정하려면 다음 명령을 실행합니다.

```
c:\> winrm set winrm/config/client/auth @{Kerberos="true"}
```

- 3 WinRM 서비스에 대한 연결을 테스트하려면 다음 명령을 실행합니다.

```
c:\> winrm identify -r:http://winrm_server:5985 -auth:Kerberos -u:user_name -p:password -encoding:utf-8
```

- 4 krb5.conf 파일을 만들어 다음 위치에 저장합니다.

| 운영 체제 | 경로 |
|---------|---|
| Windows | C:\Program Files\Common Files\VMware\VMware vCenter Server - Java Components\lib\security\ |
| Linux | /usr/java/jre-vmware/lib/security/(외부 vRealize Orchestrator용). /etc/krb5.conf(vRealize Orchestrator용)가 vRealize Automation에 작성됩니다. |

krb5.conf 파일의 구조는 다음과 같습니다.

```
[libdefaults]
default_realm = YOURDOMAIN.COM
udp_preference_limit = 1
[realms]
YOURDOMAIN.COM = {
kdc = kdc.yourdomain.com
default_domain = yourdomain.com
}
[domain_realm]
.yourdomain.com=YOURDOMAIN.COM
yourdomain.com=YOURDOMAIN.COM
```

krb5.conf는 해당 값과 함께 특정한 구성 매개 변수를 포함해야 합니다.

| Kerberos 구성 태그 | 세부 정보 |
|----------------|---|
| default_realm | Active Directory 서버를 인증할 때 클라이언트에서 사용하는 기본 Kerberos 영역입니다. 참고 대문자로 입력해야 합니다. |
| kdc | KDC(키 배포 센터) 역할을 하고 Kerberos 티켓을 발행하는 도메인 컨트롤러입니다. |
| default_domain | 정규화된 도메인 이름을 생성하는 데 사용되는 기본 도메인입니다. 참고 이 태그는 Kerberos 4 호환성을 위해 사용됩니다. |

참고 기본적으로 Java Kerberos 구성은 UDP 프로토콜을 사용합니다. TCP 프로토콜만 사용하려면 `udp_preference_limit` 매개 변수를 값 **1**로 지정해야 합니다.

참고 Kerberos 인증을 위해서는 FQDN(정규화된 도메인 이름) 호스트 주소가 필요합니다.

중요 `krb5.conf` 파일을 추가하거나 수정한 경우 Orchestrator 서버 서비스를 다시 시작해야 합니다.

SOAP 작업에서 새 워크플로 생성

SOAP 작업에서 사용자 지정 워크플로를 생성할 수 있습니다.

사용자 지정 생성된 워크플로를 상위 수준 워크플로에 통합할 수 있습니다. 워크플로 개발에 대한 자세한 내용은 "vRealize Orchestrator 개발자 가이드"를 참조하십시오.

사전 요구 사항

- Orchestrator 클라이언트에 관리자로 로그인되어 있는지 확인합니다.
- SOAP 호스트에 연결되었는지 여부를 **인벤토리** 보기에서 확인합니다.

절차

- 1 Orchestrator 클라이언트에서 **워크플로** 보기를 클릭합니다.
- 2 계층형 워크플로 목록에서 **라이브러리 > SOAP**를 확장하여 SOAP 작업에서 새 워크플로 생성 워크플로로 이동합니다.
- 3 SOAP 작업에서 새 워크플로 생성 워크플로를 마우스 오른쪽 버튼으로 클릭하고 **워크플로 시작**을 선택합니다.
- 4 사용 가능한 작업 목록에서 **SOAP** 작업을 선택합니다.
- 5 **이름** 텍스트 상자에 생성할 워크플로의 이름을 입력합니다.
- 6 새 워크플로를 생성할 워크플로 폴더를 선택합니다.
워크플로 라이브러리에서 기존 폴더를 선택할 수 있습니다.
- 7 **제출**을 클릭하여 워크플로를 실행합니다.

다음에 수행할 작업

생성된 워크플로를 테스트할 수 있습니다.

사용자 지정 생성된 워크플로 테스트

SOAP 작업에서 생성된 사용자 지정 워크플로를 실행하여 작업의 출력 매개 변수를 가져올 수 있습니다.

사전 요구 사항

- Orchestrator 클라이언트에 관리자로 로그인되어 있는지 확인합니다.
- SOAP 호스트에 연결되었는지 여부를 **인벤토리** 보기에서 확인합니다.

절차

- 1 Orchestrator 클라이언트에서 **워크플로** 보기를 클릭합니다.
- 2 워크플로 위치로 이동합니다.
- 3 사용자 지정 워크플로를 마우스 오른쪽 버튼으로 클릭하고 **워크플로 시작**을 선택합니다.
- 4 SOAP 작업에 필요한 입력 매개 변수를 제공합니다.
- 5 **제출**을 클릭하여 워크플로를 실행합니다.
- 6 (선택 사항) **로그** 탭에서 사용 가능한 출력 매개 변수 목록을 검토합니다.

SOAP 작업 호출

새 워크플로를 생성하지 않고 SOAP 작업을 직접 호출할 수 있습니다.

사전 요구 사항

- Orchestrator 클라이언트에 관리자로 로그인되어 있는지 확인합니다.
- SOAP 호스트에 연결되었는지 여부를 **인벤토리** 보기에서 확인합니다.

절차

- 1 Orchestrator 클라이언트에서 **워크플로** 보기를 클릭합니다.
- 2 계층형 워크플로 목록에서 **라이브러리 > SOAP**를 확장하여 SOAP 작업 호출 워크플로로 이동합니다.
- 3 SOAP 작업 호출 워크플로를 마우스 오른쪽 버튼으로 클릭하고 **워크플로 시작**을 선택합니다.
- 4 사용 가능한 작업 목록에서 **SOAP** 작업을 선택합니다.
- 5 SOAP 작업에 필요한 입력 매개 변수를 제공합니다.
- 6 **제출**을 클릭하여 워크플로를 실행합니다.
- 7 (선택 사항) **로그** 탭에서 사용 가능한 출력 매개 변수 목록을 검토합니다.

AMQP 플러그인을 사용하여 브로커라고도 하는 AMQP(Advanced Message Queuing Protocol) 서버와 상호 작용할 수 있습니다. 구성 워크플로를 실행하여 AMQP 브로커와 대기열 구독을 인벤토리 개체로 정의하고, 정의된 개체에서 AMQP 작업을 수행할 수 있습니다.

이 플러그인에는 AMQP 브로커 관리 및 AMQP 작업 호출과 관련된 표준 워크플로 집합이 포함되어 있습니다.

본 장은 다음 항목을 포함합니다.

- [AMQP 플러그인 구성](#)
- [AMQP 플러그인 표준 워크플로 사용](#)

AMQP 플러그인 구성

Orchestrator 클라이언트를 사용하여 AMQP 플러그인을 구성해야 합니다.

구성 워크플로

구성 워크플로 범주에는 AMQP 브로커를 관리할 수 있는 워크플로가 포함됩니다.

이러한 워크플로는 Orchestrator 클라이언트의 **워크플로 보기**에서 **라이브러리 > AMQP > 구성**을 통해 액세스할 수 있습니다.

| 워크플로 이름 | 설명 |
|------------|---------------------------------|
| 브로커 추가 | AMQP 브로커를 추가합니다. |
| 브로커 제거 | AMQP 브로커를 제거합니다. |
| 구독 제거 | AMQP 메시지 구독을 제거합니다. |
| 대기열 구독 | 새 구독 요소를 생성합니다. |
| 브로커 업데이트 | 브로커 속성을 업데이트합니다. |
| 브로커 유효성 검사 | 연결을 시작하도록 시도하여 브로커의 유효성을 검사합니다. |

브로커 추가

워크플로를 실행하여 AMQP 브로커를 추가할 수 있습니다.

절차

- 1 Orchestrator 클라이언트에 관리자로 로그인합니다.
- 2 Orchestrator 클라이언트에서 **워크플로** 보기를 클릭합니다.
- 3 계층형 워크플로 목록에서 **라이브러리 > AMQP > 구성**을 확장하여 브로커 추가 워크플로로 이동합니다.
- 4 브로커 추가 워크플로를 마우스 오른쪽 버튼으로 클릭하고 **워크플로 시작**을 선택합니다.
- 5 브로커 추가 워크플로에 필요한 정보를 제공합니다.

| 옵션 | 작업 |
|-----------|--|
| 이름 | 브로커의 이름을 입력합니다. |
| 호스트 | 호스트의 주소를 입력합니다. |
| 포트 | AMQP 브로커 서비스의 포트를 입력합니다. 기본 포트는 5672입니다. |
| 가상 호스트 | 가상 호스트의 주소를 입력합니다. 제공되는 기본값은 슬래시(/)입니다. |
| SSL 사용 | SSL 인증서를 사용할지 여부를 선택합니다. |
| 모든 인증서 수락 | 유효성 검사 없이 모든 SSL 인증서를 수락할지 여부를 선택합니다. |
| 사용자 이름 | 브로커의 사용자 이름을 입력합니다. |
| 암호 | 브로커의 암호를 입력합니다. |

- 6 **제출**을 클릭하여 워크플로를 실행합니다.

결과

워크플로가 성공적으로 실행되면 **인벤토리** 보기에 AMQP 브로커가 표시됩니다.

다음에 수행할 작업

브로커 유효성 검사 워크플로를 실행할 수 있습니다. 오류가 발생한 경우 브로커 업데이트 워크플로를 사용하여 브로커의 속성을 변경한 후 유효성 검사를 다시 수행합니다.

대기열 구독

워크플로를 실행하여 새 구독 요소를 생성할 수 있습니다.

사전 요구 사항

- Orchestrator 클라이언트에 관리자로 로그인되어 있는지 확인합니다.
- AMQP 브로커에 연결되었는지 여부를 **인벤토리** 보기에서 확인합니다.
- 선언된 구독에 포함된 모든 쿼리가 AMQP 브로커에 있는지 확인합니다.

절차

- 1 Orchestrator 클라이언트에서 **워크플로** 보기를 클릭합니다.

- 2 계층형 워크플로 목록에서 **라이브러리 > AMQP > 구성**을 확장하여 대기열 구독 워크플로로 이동합니다.
- 3 대기열 구독 워크플로를 마우스 오른쪽 버튼으로 클릭하고 **워크플로 시작**을 선택합니다.
- 4 **이름** 텍스트 상자에 표시할 대기열의 이름을 입력합니다.
- 5 구독을 추가할 브로커를 선택합니다.
- 6 메시지 구독에 대한 모든 쿼리를 선택합니다.
- 7 **제출**을 클릭하여 워크플로를 실행합니다.

결과

워크플로가 성공적으로 실행되면 **인벤토리** 보기에 브로커 하위 항목이 표시됩니다.

다음에 수행할 작업

정책을 만들 수 있습니다.

브로커 업데이트

워크플로를 실행하여 브로커 속성을 업데이트할 수 있습니다.

사전 요구 사항

- Orchestrator 클라이언트에 관리자로 로그인되어 있는지 확인합니다.
- AMQP 브로커에 연결되었는지 여부를 **인벤토리** 보기에서 확인합니다.

절차

- 1 Orchestrator 클라이언트에서 **워크플로** 보기를 클릭합니다.
- 2 계층형 워크플로 목록에서 **라이브러리 > AMQP**를 확장하여 브로커 업데이트 워크플로로 이동합니다.
- 3 브로커 업데이트 워크플로를 마우스 오른쪽 버튼으로 클릭하고 **워크플로 시작**을 선택합니다.
- 4 업데이트할 브로커를 선택합니다.
브로커의 현재 속성이 표시됩니다.
- 5 원하는 속성을 편집합니다.
- 6 **제출**을 클릭하여 워크플로를 실행합니다.

AMQP 플러그인 표준 워크플로 사용

AMQP 워크플로 범주에는 AMQP 작업을 실행할 수 있는 워크플로가 포함됩니다.

이러한 워크플로는 Orchestrator 클라이언트의 **워크플로** 보기에서 **라이브러리 > AMQP**를 통해 액세스할 수 있습니다.

| 워크플로 이름 | 설명 |
|-------------|------------------------------|
| 바인딩 | 지정된 브로커에서 바인딩을 생성합니다. |
| 대기열 선언 | 지정된 브로커에 대기열을 추가합니다. |
| Exchange 선언 | 지정된 브로커에 Exchange를 추가합니다. |
| 대기열 삭제 | 지정된 브로커에서 대기열을 삭제합니다. |
| Exchange 삭제 | 지정된 브로커에서 Exchange를 삭제합니다. |
| 문자 메시지 수신 | 지정된 브로커에서 문자 메시지를 수신합니다. |
| 문자 메시지 전송 | 지정된 브로커를 사용하여 문자 메시지를 전송합니다. |
| 바인딩 해제 | 지정된 브로커에서 바인딩을 해제합니다. |

바인딩 선언

워크플로를 실행하여 지정된 브로커에서 바인딩을 생성할 수 있습니다.

사전 요구 사항

- Orchestrator 클라이언트에 관리자로 로그인되어 있는지 확인합니다.
- AMQP 브로커에 연결되었는지 여부를 **인벤토리** 보기에서 확인합니다.

절차

- 1 Orchestrator 클라이언트에서 **워크플로** 보기를 클릭합니다.
- 2 계층형 워크플로 목록에서 **라이브러리 > AMQP**를 확장하여 바인딩 워크플로로 이동합니다.
- 3 바인딩 워크플로를 마우스 오른쪽 버튼으로 클릭하고 **워크플로 시작**을 선택합니다.
- 4 바인딩을 생성할 브로커를 선택합니다.
- 5 바인딩에 대한 정보를 제공합니다.

| 옵션 | 작업 |
|-------------|----------------------|
| 대기열 이름 | 대기열의 이름을 입력합니다. |
| Exchange 이름 | Exchange의 이름을 입력합니다. |
| 라우팅 키 | 라우팅 키를 입력합니다. |

- 6 **제출**을 클릭하여 워크플로를 실행합니다.

대기열 선언

워크플로를 실행하여 지정된 브로커에 대기열을 추가할 수 있습니다.

사전 요구 사항

- Orchestrator 클라이언트에 관리자로 로그인되어 있는지 확인합니다.
- AMQP 브로커에 연결되었는지 여부를 **인벤토리** 보기에서 확인합니다.

절차

- 1 Orchestrator 클라이언트에서 **워크플로** 보기를 클릭합니다.
- 2 계층형 워크플로 목록에서 **라이브러리 > AMQP**를 확장하여 대기열 선언 워크플로로 이동합니다.
- 3 대기열 선언 워크플로를 마우스 오른쪽 버튼으로 클릭하고 **워크플로 시작**을 선택합니다.
- 4 대기열을 추가할 브로커를 선택합니다.
- 5 **이름** 텍스트 상자에 표시할 대기열의 이름을 입력합니다.
- 6 대기열이 지속형인지 선택합니다.

| 옵션 | 설명 |
|-----|-----------------------------|
| 예 | 브로커가 다시 시작된 후 대기열이 제거됩니다. |
| 아니요 | 브로커가 다시 시작된 후에도 대기열이 유지됩니다. |

- 7 특정 대기열을 위한 전용 클라이언트를 설정할지 여부를 선택합니다.

| 옵션 | 설명 |
|-----|--------------------------------|
| 예 | 이 특정 대기열을 위한 하나의 클라이언트를 설정합니다. |
| 아니요 | 이 특정 대기열을 위한 여러 클라이언트를 설정합니다. |

- 8 활성화된 구독이 있는 대기열을 자동으로 삭제할지 여부를 선택합니다.

| 옵션 | 설명 |
|-----|--|
| 예 | 연결된 클라이언트가 더 이상 없는 경우 대기열을 자동으로 삭제합니다. 하나 이상의 클라이언트가 구독할 때까지 대기열이 유지됩니다. |
| 아니요 | 대기열을 자동으로 삭제하지 않습니다. |

- 9 **제출**을 클릭하여 워크플로를 실행합니다.

Exchange 선언

워크플로를 실행하여 지정된 브로커에 Exchange를 추가할 수 있습니다.

사전 요구 사항

- Orchestrator 클라이언트에 관리자로 로그인되어 있는지 확인합니다.
- AMQP 브로커에 연결되었는지 여부를 **인벤토리** 보기에서 확인합니다.

절차

- 1 Orchestrator 클라이언트에서 **워크플로** 보기를 클릭합니다.
- 2 계층형 워크플로 목록에서 **라이브러리 > AMQP**를 확장하여 Exchange 선언 워크플로로 이동합니다.
- 3 Exchange 선언 워크플로를 마우스 오른쪽 버튼으로 클릭하고 **워크플로 시작**을 선택합니다.

- 4 Exchange를 추가할 브로커를 선택합니다.
- 5 이름 텍스트 상자에 Exchange의 이름을 입력합니다.
- 6 Exchange 유형을 선택합니다.

| 옵션 | 설명 |
|-----|--|
| 직접 | 메시지에 제공된 라우팅 키와 대기열이 이 Exchange에 바인딩될 때 사용되는 라우팅 조건을 직접 일치시킵니다. |
| 팬아웃 | 이 Exchange에 전송되는 모든 메시지를 해당 Exchange에 바인딩된 모든 대기열로 전달합니다. 이 Exchange에 바인딩된 대기열에는 인수가 없습니다. |
| 헤더 | 대기열이 헤더 및 값을 포함할 수 있는 인수 테이블과 함께 이 Exchange에 바인딩됩니다. x-match라는 특수한 인수가 일치 알고리즘을 결정합니다. |
| 항목 | 라우팅 키와 바인딩에 지정된 라우팅 패턴 간의 와일드카드 일치를 수행합니다. |

- 7 Exchange가 지속형인지 선택합니다.

| 옵션 | 설명 |
|-----|----------------------------------|
| 예 | 브로커가 다시 시작된 후에도 Exchange가 유지됩니다. |
| 아니요 | 브로커가 다시 시작된 후 Exchange가 제거됩니다. |

- 8 활성화된 구독이 있는 Exchange를 자동으로 삭제할지 여부를 선택합니다.

| 옵션 | 설명 |
|-----|--|
| 예 | 바인딩된 대기열이 더 이상 없는 경우 Exchange를 자동으로 삭제합니다. 하나 이상의 대기열이 바인딩될 때까지 Exchange가 유지됩니다. |
| 아니요 | Exchange를 자동으로 삭제하지 않습니다. |

- 9 제출을 클릭하여 워크플로를 실행합니다.

문자 메시지 전송

워크플로를 실행하여 지정된 브로커를 통해 문자 메시지를 전송할 수 있습니다.

사전 요구 사항

- Orchestrator 클라이언트에 관리자로 로그인되어 있는지 확인합니다.
- AMQP 브로커에 연결되었는지 여부를 **인벤토리** 보기에서 확인합니다.

절차

- 1 Orchestrator 클라이언트에서 **워크플로** 보기를 클릭합니다.
- 2 계층형 워크플로 목록에서 **라이브러리 > AMQP**를 확장하여 문자 메시지 전송 워크플로로 이동합니다.
- 3 문자 메시지 전송 워크플로를 마우스 오른쪽 버튼으로 클릭하고 **워크플로 시작**을 선택합니다.

- 4 메시지를 전송할 브로커를 선택합니다.
- 5 **Exchange 이름** 텍스트 상자에 Exchange의 이름을 지정합니다.
- 6 **라우팅 키** 텍스트 상자에 라우팅 키를 지정합니다.
- 7 **내용** 텍스트 상자에 전송할 메시지를 입력합니다.
- 8 **제출**을 클릭하여 워크플로를 실행합니다.

바인딩 삭제

워크플로를 실행하여 지정된 브로커에서 바인딩을 삭제할 수 있습니다.

사전 요구 사항

- Orchestrator 클라이언트에 관리자로 로그인되어 있는지 확인합니다.
- AMQP 브로커에 연결되었는지 여부를 **인벤토리** 보기에서 확인합니다.

절차

- 1 Orchestrator 클라이언트에서 **워크플로** 보기를 클릭합니다.
- 2 계층형 워크플로 목록에서 **라이브러리 > AMQP**를 확장하여 바인딩 해제 워크플로로 이동합니다.
- 3 바인딩 해제 워크플로를 마우스 오른쪽 버튼으로 클릭하고 **워크플로 시작**을 선택합니다.
- 4 바인딩을 제거할 브로커를 선택합니다.
- 5 바인딩에 대한 정보를 제공합니다.

| 옵션 | 작업 |
|-------------|----------------------|
| 대기열 이름 | 대기열의 이름을 지정합니다. |
| Exchange 이름 | Exchange의 이름을 지정합니다. |
| 라우팅 키 | 라우팅 키를 지정합니다. |

- 6 **제출**을 클릭하여 워크플로를 실행합니다.

SNMP 플러그인 사용

17

SNMP 플러그인을 사용하면 vRealize Orchestrator에서 SNMP 지원 시스템 및 디바이스에 연결하여 정보를 수신할 수 있습니다. 워크플로를 실행하여 SNMP 디바이스를 인벤토리 개체로 정의하고, 정의된 개체에서 SNMP 작업을 수행할 수 있습니다.

이 플러그인을 사용하여 라우터, 스위치, 네트워크 프린터 및 UPS 디바이스와 같은 SNMP 디바이스에 연결할 수 있습니다. 또한 이 플러그인은 SNMP 프로토콜을 통해 vCenter Server에서 이벤트를 수신할 수 있습니다.

SNMP 플러그인은 SNMP 디바이스와 통신할 수 있는 두 가지 방법을 제공합니다.

- 특정 SNMP 변수 값 쿼리
- 디바이스에서 생성되어 등록된 SNMP 관리자에 푸시되는 이벤트(SNMP 트랩) 수신

이 플러그인에는 SNMP 디바이스/쿼리/트랩 호스트 관리 및 SNMP 작업 수행과 관련된 표준 워크플로 집합이 포함되어 있습니다. 사용자 지정 워크플로를 생성하여 SNMP 환경에서 작업을 자동화할 수도 있습니다.

본 장은 다음 항목을 포함합니다.

- [SNMP 디바이스 관리](#)
- [SNMP 쿼리 관리](#)
- [SNMP 트랩 호스트 관리](#)
- [SNMP 트랩 수신](#)
- [일반 SNMP 요청 워크플로](#)

SNMP 디바이스 관리

워크플로를 실행하여 Orchestrator에 SNMP 디바이스를 등록하고, 기존 디바이스에 대한 설정을 편집하고, 디바이스의 등록을 취소할 수 있습니다.

디바이스 관리 워크플로

디바이스 관리 워크플로 범주에는 SNMP 디바이스를 관리할 수 있는 워크플로가 포함됩니다.

이러한 워크플로는 Orchestrator 클라이언트의 **워크플로** 보기에서 **라이브러리 > SNMP > 디바이스 관리**를 통해 액세스할 수 있습니다.

| 워크플로 이름 | 설명 |
|-----------------|-----------------------------------|
| SNMP 디바이스 편집 | 등록된 SNMP 디바이스의 구성을 편집합니다. |
| SNMP 디바이스 등록 | SNMP 지원 디바이스를 플러그인 인벤토리에 등록합니다. |
| SNMP 디바이스 등록 취소 | 플러그인 인벤토리에서 SNMP 디바이스의 등록을 취소합니다. |

SNMP 디바이스 등록

워크플로를 실행하여 SNMP 디바이스를 등록하고, 필요한 경우 고급 연결 매개 변수를 구성할 수 있습니다.

절차

- 1 Orchestrator 클라이언트에 관리자로 로그인합니다.
- 2 Orchestrator 클라이언트에서 **워크플로** 보기를 클릭합니다.
- 3 계층형 워크플로 목록에서 **라이브러리 > SNMP > 디바이스 관리**를 확장하여 SNMP 디바이스 등록 워크플로로 이동합니다.
- 4 SNMP 디바이스 등록 워크플로를 마우스 오른쪽 버튼으로 클릭하고 **워크플로 시작**을 선택합니다.
- 5 **디바이스 주소** 텍스트 상자에 SNMP 디바이스의 IP 주소 또는 DNS 이름을 입력합니다.

참고 보다 안정적인 연결을 설정하려면 IP 주소를 사용해야 합니다.

- 6 (선택 사항) **이름** 텍스트 상자에 **인벤토리** 보기에 표시할 디바이스 이름을 입력합니다.
텍스트 상자를 빈 상태로 둔 경우 디바이스 주소를 사용하여 이름이 자동으로 생성됩니다.
- 7 (선택 사항) 고급 연결 매개 변수를 구성하려면 **예**를 선택합니다.
 - a **포트** 텍스트 상자에 연결 포트를 지정합니다.

기본 포트는 161입니다.

- b **버전** 드롭다운 메뉴에서 사용할 SNMP 버전을 선택하고 자격 증명을 제공합니다.

SNMPv3 지원은 MD5 인증을 사용하는 AuthPriv 보안 수준으로 제한됩니다. DES 암호는 MD5 암호와 동일합니다.

참고 SNMPv3 지원은 권장되지 않습니다.

- 8 **제출**을 클릭하여 워크플로를 실행합니다.

결과

워크플로가 성공적으로 실행되면 **인벤토리** 보기에 SNMP 디바이스가 표시됩니다.

다음에 수행할 작업

SNMP 디바이스에 쿼리를 추가하고 **인벤토리** 보기에서 워크플로를 실행할 수 있습니다.

SNMP 쿼리 관리

등록된 SNMP 디바이스에 쿼리를 추가하고, 기존 쿼리를 실행, 복사 및 편집하고, SNMP 디바이스에서 쿼리를 제거할 수 있습니다. SNMP 쿼리를 보다 복잡한 워크플로에서 구성 요소로 사용할 수 있습니다.

쿼리 관리 워크플로

쿼리 관리 워크플로 범주에는 SNMP 쿼리를 관리할 수 있는 워크플로가 포함됩니다.

이러한 워크플로는 Orchestrator 클라이언트의 **워크플로** 보기에서 **라이브러리 > SNMP > 쿼리 관리**를 통해 액세스할 수 있습니다.

| 워크플로 이름 | 설명 |
|-------------------|--------------------------|
| SNMP 디바이스에 쿼리 추가 | SNMP 디바이스에 쿼리를 추가합니다. |
| SNMP 쿼리 복사 | 디바이스 간에 SNMP 쿼리를 복사합니다. |
| SNMP 쿼리 편집 | 기존 SNMP 쿼리를 편집합니다. |
| SNMP 디바이스에서 쿼리 제거 | 디바이스에서 SNMP 쿼리를 제거합니다. |
| SNMP 쿼리 실행 | SNMP 디바이스에 대해 쿼리를 실행합니다. |

SNMP 디바이스에 쿼리 추가

워크플로를 실행하여 플러그인 인벤토리의 SNMP 디바이스에 쿼리를 추가할 수 있습니다.

사전 요구 사항

- Orchestrator 클라이언트에 관리자로서 로그인되어 있는지 확인합니다.
- SNMP 디바이스에 연결되었는지 여부를 **인벤토리** 보기에서 확인합니다.

절차

- 1 Orchestrator 클라이언트에서 **워크플로** 보기를 클릭합니다.
- 2 계층형 워크플로 목록에서 **라이브러리 > SNMP > 쿼리 관리**를 확장하여 SNMP 디바이스에 쿼리 추가 워크플로로 이동합니다.
- 3 SNMP 디바이스에 쿼리 추가 워크플로를 마우스 오른쪽 버튼으로 클릭하고 **워크플로 시작**을 선택합니다.
- 4 쿼리를 추가할 디바이스를 선택합니다.
- 5 **유형** 드롭다운 메뉴에서 쿼리 유형을 선택합니다.

6 **OID** 텍스트 상자에 쿼리할 변수의 개체 식별자를 입력합니다.

OID 값의 예는 다음과 같습니다.

- 1.3.6.1.2.1.1.5.0
- .1.3.6.1.2.1.1.5.0
- iso.3.6.1.2.1.1.5.0

참고 플러그인은 숫자이거나 **iso**로 시작하고 숫자가 이어지는 OID 값만 지원합니다.

7 (선택 사항) **이름** 텍스트 상자에 쿼리의 이름을 입력합니다.

텍스트 상자를 빈 상태로 둔 경우 유형 및 OID 매개 변수를 사용하여 이름이 자동으로 생성됩니다.

8 **제출**을 클릭하여 워크플로를 실행합니다.

다음에 수행할 작업

인벤토리 보기에서 쿼리에 대한 워크플로를 실행할 수 있습니다.

SNMP 트랩 호스트 관리

vRealize Orchestrator는 SNMP 수신기 역할을 수행할 수 있습니다. SNMP 트랩 호스트를 시작 및 중지하고, Orchestrator에서 SNMP 트랩을 수신할 포트를 변경할 수 있습니다.

SNMP 플러그인은 SNMPv1 및 SNMPv2c 트랩을 지원합니다.

참고 SNMPv3 지원은 권장되지 않습니다.

트랩 호스트 관리 워크플로

트랩 호스트 관리 워크플로 범주에는 SNMP 트랩 호스트를 관리할 수 있는 워크플로가 포함됩니다.

이러한 워크플로는 Orchestrator 클라이언트의 **워크플로** 보기에서 **라이브러리 > SNMP > 트랩 호스트 관리**를 통해 액세스할 수 있습니다.

| 워크플로 이름 | 설명 |
|---------------|--|
| SNMP 트랩 포트 설정 | Orchestrator에서 SNMP 트랩을 수신할 포트를 설정합니다. |
| 트랩 호스트 시작 | Orchestrator에서 SNMP 트랩 수신을 시작합니다. |
| 트랩 호스트 중지 | Orchestrator에서 SNMP 트랩 수신을 중지합니다. |

SNMP 트랩 포트 설정

워크플로를 실행하여 Orchestrator에서 SNMP 트랩을 수신할 포트를 설정할 수 있습니다.

SNMP 트랩의 기본 포트는 162입니다. 그러나 Linux 시스템에서는 슈퍼 사용자 권한을 사용하여 1024 미만의 포트만 열 수 있습니다.

참고 호환성 향상을 위해 SNMP 플러그인에서 SNMP 트랩을 수신하는 기본 포트는 4000으로 설정됩니다.

사전 요구 사항

- Orchestrator 클라이언트에 관리자로 로그인되어 있는지 확인합니다.
- SNMP 디바이스에 연결되었는지 여부를 **인벤토리** 보기에서 확인합니다.

절차

- 1 Orchestrator 클라이언트에서 **워크플로** 보기를 클릭합니다.
- 2 계층형 워크플로 목록에서 **라이브러리 > SNMP > 트랩 호스트 관리**를 확장하여 SNMP 트랩 포트 설정 워크플로로 이동합니다.
- 3 SNMP 트랩 포트 설정 워크플로를 마우스 오른쪽 버튼으로 클릭하고 **워크플로 시작**을 선택합니다.
- 4 **포트** 텍스트 상자에 Orchestrator에서 SNMP 트랩을 수신해야 하는 포트 번호를 입력합니다.
- 5 **제출**을 클릭하여 워크플로를 실행합니다.

결과

워크플로에서 트랩 호스트를 중지하고 새 포트를 설정한 후 트랩 호스트를 다시 시작합니다.

SNMP 트랩 수신

SNMP 플러그인은 단일 트랩 메시지를 대기하는 워크플로를 실행하거나 지속적으로 트랩을 처리할 수 있는 정책을 사용하여 SNMP 트랩을 수신할 수 있습니다. 이 플러그인은 SNMPv1 및 SNMPv2c 트랩을 지원합니다.

SNMP 디바이스에서 트랩 대기

지정된 디바이스에서 SNMP 트랩 수신을 대기하는 워크플로를 실행할 수 있습니다.

이 워크플로에는 워크플로 실행을 중지하고 계속하기 전에 SNMP 트랩을 대기하는 트리거가 있습니다. 트랩이 수신되면 워크플로 실행이 재개됩니다. 이 워크플로를 보다 복잡한 워크플로의 일부로 사용하거나, 특정 요구 사항에 맞게 사용자 지정 또는 확장할 수 있는 샘플로 사용할 수 있습니다.

사전 요구 사항

- Orchestrator 클라이언트에 관리자로 로그인되어 있는지 확인합니다.
- SNMP 디바이스에 연결되었는지 여부를 **인벤토리** 보기에서 확인합니다.

절차

- 1 Orchestrator 클라이언트에서 **워크플로** 보기를 클릭합니다.

- 2 계층형 워크플로 목록에서 **라이브러리 > SNMP**를 확장하여 **SNMP 디바이스에서 트랩 대기** 워크플로로 이동합니다.
- 3 SNMP 디바이스에서 트랩 대기 워크플로를 마우스 오른쪽 버튼으로 클릭하고 **워크플로 시작**을 선택합니다.
- 4 SNMP 트랩을 대기할 디바이스를 선택합니다.
- 5 (선택 사항) **OID** 텍스트 상자에 특정 트랩의 개체 식별자를 입력합니다.

참고 텍스트 상자를 비워 두면 지정된 **SNMP 디바이스에서 트랩**을 수신한 후 워크플로 실행이 재개됩니다.

- 6 **제출**을 클릭하여 워크플로를 실행합니다.

SNMP 트랩 정책 설정

플러그인 인벤토리에 이미 등록되어 있는 **SNMP 디바이스**에서 지속적으로 트랩을 수신하도록 정책을 설정할 수 있습니다.

사전 요구 사항

- Orchestrator 클라이언트에 관리자로 로그인되어 있는지 확인합니다.
- SNMP 디바이스에 연결되었는지 여부를 **인벤토리** 보기에서 확인합니다.

절차

- 1 Orchestrator 클라이언트의 드롭다운 메뉴에서 **관리**를 선택합니다.
- 2 **정책 템플릿** 보기를 클릭합니다.
- 3 계층형 워크플로 목록에서 **라이브러리 > SNMP**를 확장하여 **SNMP 트랩 정책 템플릿**으로 이동합니다.
- 4 SNMP 트랩 정책 템플릿을 마우스 오른쪽 버튼으로 클릭하고 **정책 적용**을 선택합니다.
- 5 **정책 이름** 텍스트 상자에 생성할 정책의 이름을 입력합니다.
- 6 (선택 사항) **정책 대상** 텍스트 상자에 정책의 설명을 입력합니다.
- 7 정책을 설정할 **SNMP 디바이스**를 선택합니다.
- 8 **제출**을 클릭하여 정책을 만듭니다.

Orchestrator 클라이언트가 **실행** 관점으로 전환됩니다.

- 9 **정책** 보기에서 생성한 정책을 마우스 오른쪽 버튼으로 클릭하고 **정책 시작**을 선택합니다.

결과

트랩 정책에서 **SNMP 트랩**을 수신하기 시작합니다.

다음에 수행할 작업

SNMP 트랩 정책을 편집할 수 있습니다.

SNMP 트랩 호스트 정책 구성

SNMP 트랩 호스트 정책을 사용하면 등록된 SNMP 디바이스로 추가되지 않을 수 있는 호스트에서 SNMP 트랩을 수신할 수 있습니다.

사전 요구 사항

- Orchestrator 클라이언트에 관리자로 로그인되어 있는지 확인합니다.
- SNMP 디바이스에 연결되었는지 여부를 **인벤토리** 보기에서 확인합니다.

절차

- 1 Orchestrator 클라이언트의 드롭다운 메뉴에서 **관리**를 선택합니다.
- 2 **정책 템플릿** 보기를 클릭합니다.
- 3 계층형 워크플로 목록에서 **라이브러리 > SNMP**를 확장하여 **SNMP 트랩 호스트 정책** 템플릿으로 이동합니다.
- 4 SNMP 트랩 호스트 정책 템플릿을 마우스 오른쪽 버튼으로 클릭하고 **정책 적용**을 선택합니다.
- 5 **정책 이름** 텍스트 상자에 생성할 정책의 이름을 입력합니다.
- 6 (선택 사항) **정책 대상** 텍스트 상자에 정책의 설명을 입력합니다.
- 7 인벤토리 트리에서 **Trap Host (Online)**를 선택합니다.
- 8 **제출**을 클릭하여 정책을 만듭니다.
Orchestrator 클라이언트가 **실행** 관점으로 전환됩니다.
- 9 정책을 마우스 오른쪽 버튼으로 클릭하고 **편집**을 선택합니다.
- 10 **스크립팅** 탭에서 **호스트 > OnTrapAll**을 확장합니다.
 - a 정책에 연결할 워크플로 또는 스크립트를 선택합니다.
- 11 **저장 후 닫기**를 클릭하여 편집한 설정을 적용합니다.
- 12 **정책** 보기에서 편집한 정책을 마우스 오른쪽 버튼으로 클릭하고 **정책 시작**을 선택합니다.

다음에 수행할 작업

SNMP 트랩 호스트 정책을 편집할 수 있습니다.

트랩 정책 편집

트랩 정책을 편집하여 특정 사용 사례에 맞게 사용자 지정할 수 있습니다. 트랩 정책을 편집할 때 해당 우선 순위 및 시작 설정을 변경하고, 해당 정책과 연관된 스크립팅 및 사용 권한을 사용자 지정할 수 있습니다.

사전 요구 사항

- Orchestrator 클라이언트에 관리자로 로그인되어 있는지 확인합니다.

- SNMP 디바이스에 연결되었는지 여부를 **인벤토리** 보기에서 확인합니다.

절차

- 1 Orchestrator 클라이언트에서 **정책** 보기를 클릭합니다.
- 2 편집하려는 정책이 실행 중인 경우 해당 정책을 마우스 오른쪽 버튼으로 클릭하고 **정책 중지**를 선택합니다.
- 3 정책을 마우스 오른쪽 버튼으로 클릭하고 **편집**을 선택합니다.
- 4 **일반** 탭에서 시작 설정, 우선 순위 및 정책의 설명을 편집합니다.
- 5 (선택 사항) **스크립팅** 탭에서 보다 복잡한 시나리오에 통합하기 위해 특정 워크플로 또는 스크립팅 코드를 정책에 연결할 수 있습니다.
트랩이 수신되면 사용자 지정 워크플로를 트리거하도록 정책을 설정할 수 있습니다.
- 6 (선택 사항) **사용 권한** 탭에서 액세스 권한을 수정할 수 있습니다.
사용자 또는 그룹에 정책 편집 권한 없이 정책을 시작할 수 있는 권한을 제공할 수 있습니다.
- 7 **저장 후 닫기**를 클릭하여 편집한 설정을 적용합니다.
- 8 **정책** 보기에서 편집한 정책을 마우스 오른쪽 버튼으로 클릭하고 **정책 시작**을 선택합니다.

일반 SNMP 요청 워크플로

SNMP 워크플로 범주에는 쿼리를 생성하지 않고도 기본 SNMP 요청을 수행할 수 있는 워크플로가 포함됩니다.

이러한 워크플로는 Orchestrator 클라이언트의 **워크플로** 보기에서 **라이브러리 > SNMP**를 통해 액세스할 수 있습니다.

| 워크플로 이름 | 설명 |
|-------------------|---|
| 대량 SNMP 값 가져오기 | SNMP 디바이스에 대해 GET BULK 쿼리를 실행합니다. |
| 다음 SNMP 값 가져오기 | SNMP 디바이스에 대해 GET NEXT 쿼리를 실행합니다. |
| SNMP 값 가져오기 | SNMP 디바이스에 대해 GET 쿼리를 실행합니다. |
| SNMP 트랩 전송 | 지정된 주소로 SNMP 트랩을 전송합니다. |
| 모든 디바이스에서 트랩 대기 | Orchestrator로 트랩을 전송하는 모든 호스트의 SNMP 트랩 수신을 대기합니다. |
| SNMP 디바이스에서 트랩 대기 | 지정된 디바이스에서 SNMP 트랩 수신을 대기합니다. |

Active Directory 플러그인 사용

18

Active Directory 플러그인(Microsoft Active Directory용 VMware vRealize Orchestrator 플러그인)은 vRealize Orchestrator와 Microsoft Active Directory 간의 상호 작용을 지원합니다. 이 플러그인을 사용하여 Active Directory 프로세스를 자동화하는 Orchestrator 워크플로를 실행할 수 있습니다.

이 플러그인에는 표준 워크플로 집합이 포함되어 있습니다. 플러그인 API를 구현하는 사용자 지정 워크플로를 생성하여 Active Directory 환경에서 작업을 자동화할 수도 있습니다.

본 장은 다음 항목을 포함합니다.

- [Active Directory 플러그인 구성](#)
- [Active Directory 플러그인 워크플로 라이브러리 사용](#)

Active Directory 플러그인 구성

Active Directory 플러그인을 사용하여 Microsoft Active Directory 인스턴스에 연결하려면 Microsoft Active Directory 인스턴스에 대한 연결 매개 변수를 구성해야 합니다.

플러그인에 포함된 구성 워크플로를 실행하여 Active Directory를 구성할 수 있습니다.

Active Directory 구성 워크플로

Active Directory 플러그인의 구성 워크플로 범주에는 Active Directory 서버를 구성하고 SSL 인증서를 관리할 수 있는 워크플로가 포함됩니다.

이러한 워크플로는 Orchestrator 클라이언트의 **워크플로 보기**에서 **라이브러리 > Microsoft > Active Directory > 구성**을 통해 액세스할 수 있습니다.

| 워크플로 이름 | 설명 |
|--------------------------------------|--|
| Active Directory 서버 추가 | Microsoft Active Directory 서버를 구성합니다. |
| Active Directory 플러그인 옵션 구성 | Active Directory 플러그인의 검색 제한 옵션을 구성합니다. |
| Active Directory 서버 구성(더 이상 사용되지 않음) | 기본 Active Directory 서버 구성을 만들거나 업데이트합니다. Active Directory 서버 업데이트를 사용하십시오. |
| Active Directory 서버 제거 | Active Directory 서버 구성을 제거합니다. |

| 워크플로 이름 | 설명 |
|--------------------------|---|
| 구성 재설정(더 이상 사용되지 않음) | 기본 Active Directory 서버 구성을 삭제합니다. Active Directory 서버 제거를 사용하십시오. |
| Active Directory 서버 업데이트 | 기존 Active Directory 서버 구성을 수정합니다. |

Active Directory 플러그인 워크플로 라이브러리 사용

Active Directory 플러그인 워크플로 라이브러리에는 Microsoft Active Directory 개체 관리와 관련된 자동화된 프로세스를 실행하는 데 사용할 수 있는 워크플로가 포함됩니다.

워크플로는 개체 유형에 따라 범주로 그룹화됩니다. 워크플로 라이브러리의 표준 워크플로를 사용자 지정 워크플로에 통합할 수 있습니다.

Active Directory 플러그인 인벤토리 사용

Active Directory 플러그인은 연결된 Microsoft Active Directory 인스턴스의 모든 개체를 **인벤토리** 보기에 노출합니다. **인벤토리** 보기를 사용하여 권한 부여 요소를 추가하거나 Microsoft Active Directory 개체에서 워크플로를 실행할 수 있습니다.

인벤토리 개체에 사용할 수 있는 워크플로를 표시하려면 **도구 > 사용자 기본 설정 > 인벤토리**로 이동한 후 **인벤토리에서 컨텍스트 메뉴 사용** 확인란을 선택합니다. 이 옵션을 사용하도록 설정하면 Orchestrator 인벤토리에서 개체를 마우스 오른쪽 버튼으로 클릭했을 때 해당 개체에 사용할 수 있는 모든 워크플로가 표시됩니다.

Active Directory 플러그인 워크플로 라이브러리 액세스

Active Directory 플러그인 워크플로 라이브러리의 요소에 액세스하려면 Orchestrator 클라이언트를 사용해야 합니다.

절차

- 1 Orchestrator 클라이언트에 관리자로 로그인합니다.
- 2 **워크플로** 보기를 클릭합니다.
- 3 계층형 목록에서 **라이브러리 > Microsoft > Active Directory**를 확장한 다음 선택 항목을 확장합니다.

Active Directory 플러그인 워크플로

Active Directory 플러그인에는 가장 일반적인 LDAP 기능에 적용되는 표준 워크플로 집합이 포함되어 있습니다. 이러한 워크플로를 빌딩 블록으로 사용하여 복잡한 사용자 지정 솔루션을 만들 수 있습니다. 표준 워크플로를 조합하여 Active Directory 환경에서 다단계 프로세스를 자동화할 수 있습니다.

컴퓨터 워크플로

컴퓨터 워크플로 범주에는 Active Directory 컴퓨터 관리와 관련된 워크플로가 포함됩니다.

이러한 워크플로는 **라이브러리 > Microsoft > Active Directory > 컴퓨터**를 통해 액세스할 수 있습니다.

| 워크플로 이름 | 설명 |
|-------------------|---|
| 그룹에서 컴퓨터 생성 | 그룹에서 Active Directory 컴퓨터를 생성합니다. |
| 조직 구성 단위에서 컴퓨터 생성 | 조직 구성 단위에서 Active Directory 컴퓨터를 생성합니다. |
| 컴퓨터 삭제 | Active Directory 인스턴스에서 컴퓨터를 삭제합니다. |
| 컴퓨터와 해당 하위 트리 삭제 | Active Directory 인스턴스에서 컴퓨터를 삭제하고 컴퓨터 하위 트리 내의 모든 개체를 삭제합니다. |
| 컴퓨터 사용 안 함 | Active Directory 인스턴스에서 컴퓨터를 사용하지 않도록 설정합니다. |
| 컴퓨터 사용 | Active Directory 인스턴스에서 컴퓨터를 사용하도록 설정합니다. |

조직 구성 단위 워크플로

조직 구성 단위 워크플로 범주에는 **Active Directory** 조직 구성 단위 관리와 관련된 워크플로가 포함됩니다.

이러한 워크플로는 **라이브러리 > Microsoft > Active Directory > 조직 구성 단위**를 통해 액세스할 수 있습니다.

| 워크플로 이름 | 설명 |
|-----------------------|--|
| 조직 구성 단위 생성 | 기존 조직 구성 단위에 조직 구성 단위를 생성합니다. |
| 조직 구성 단위 삭제 | Active Directory 인스턴스에서 조직 구성 단위를 삭제합니다. |
| 조직 구성 단위와 해당 하위 트리 삭제 | Active Directory 인스턴스에서 조직 구성 단위를 삭제하고 해당 조직 구성 단위 하위 트리 내의 모든 개체를 삭제합니다. |

사용자 워크플로

사용자 워크플로 범주에는 **Active Directory** 사용자 관리와 관련된 워크플로가 포함됩니다.

이러한 워크플로는 **라이브러리 > Microsoft > Active Directory > 사용자**를 통해 액세스할 수 있습니다.

| 워크플로 이름 | 설명 |
|--------------------------|---|
| 사용자 그룹에 사용자 추가 | 한 명의 사용자를 사용자 그룹의 구성원으로 추가합니다. |
| 사용자 암호 변경 | 사용자의 암호를 변경합니다. SSL 연결이 필요하며, 암호가 Active Directory 제한 사항을 충족해야 합니다. |
| 그룹에서 사용자 생성 | 암호를 지정하지 않고 사용자를 생성합니다. 다음에 로그인할 때 암호를 변경해야 합니다. 도메인 정책에서 사용자가 빈 암호를 사용하도록 허용해야 합니다. |
| 조직 구성 단위에서 사용자 생성 | 조직 구성 단위에서 사용자를 생성합니다. SSL 연결이 사용하지 않도록 설정된 경우 암호를 지정할 수 없습니다. 도메인 정책에서 사용자가 빈 암호를 사용하도록 허용해야 합니다. |
| 그룹에서 암호가 있는 사용자 생성 | 사용자를 생성하고 해당 사용자의 암호를 설정합니다. 다음에 로그인할 때 암호를 변경할 수 있습니다. |
| 조직 구성 단위에서 암호가 있는 사용자 생성 | 조직 구성 단위에서 사용자를 생성하고 해당 사용자의 암호를 설정합니다. 다음에 로그인할 때 암호를 변경할 수 있습니다. SSL 연결이 사용하지 않도록 설정된 경우 암호를 지정할 수 없습니다. |
| 사용자 삭제 | Active Directory 인스턴스에서 사용자를 삭제합니다. |
| 사용자 사용 안 함 | Active Directory 인스턴스에서 사용자를 사용하지 않도록 설정합니다. |

| 워크플로 이름 | 설명 |
|-----------------|---|
| 사용자 사용 | Active Directory 인스턴스에서 사용자를 사용하도록 설정합니다. |
| 사용자 그룹에서 사용자 제거 | 사용자 그룹에서 사용자를 제거합니다. |

사용자 그룹 워크플로

사용자 그룹 워크플로 범주에는 Active Directory 사용자 그룹 관리와 관련된 워크플로가 포함됩니다.

이러한 워크플로는 **라이브러리 > Microsoft > Active Directory > 사용자 그룹**을 통해 액세스할 수 있습니다.

| 워크플로 이름 | 설명 |
|---|---|
| 그룹 구성원에 컴퓨터 추가 | 하나 이상의 컴퓨터를 사용자 그룹의 구성원으로 추가합니다. |
| 그룹 구성원에 그룹 추가 | 하나 이상의 사용자 그룹을 사용자 그룹의 구성원으로 추가합니다. |
| 그룹 구성원에 사용자 추가 | 하나 이상의 사용자를 사용자 그룹의 구성원으로 추가합니다. |
| 그룹에서 사용자 그룹 생성 | 기존 컨테이너(그룹)에서 사용자 그룹을 생성합니다. |
| 그룹에서 사용자 그룹을 생성하고 “Group name (pre-Windows 2000)” 특성 설정 | 기존 컨테이너(조직 구성 단위)에서 사용자 그룹을 생성하고 Group name (pre-Windows 2000) 특성을 설정합니다. |
| 조직 구성 단위에서 사용자 그룹 생성 | 기존 컨테이너(조직 구성 단위)에서 사용자 그룹을 생성합니다. |
| 사용자 그룹 삭제 | Active Directory 인스턴스에서 사용자 그룹을 삭제합니다. |
| 그룹 구성원에서 컴퓨터 제거 | 사용자 그룹에서 하나 이상의 컴퓨터를 제거합니다. |
| 그룹 구성원에서 그룹 제거 | 사용자 그룹에서 하나 이상의 그룹을 제거합니다. |
| 그룹 구성원에서 사용자 제거 | 사용자 그룹에서 하나 이상의 사용자를 제거합니다. |

동적 유형 플러그인 사용

19

Orchestrator Dynamic Types 플러그인을 사용하여 동적 유형을 정의하고, 이러한 유형의 개체를 생성하고, 둘 사이의 관계를 설정할 수 있습니다.

동적 유형의 정의에는 이 유형의 동적 개체를 찾는 데 사용할 수 있는 찾기 워크플로 및 작업 집합과 해당 속성에 대한 설명이 포함됩니다. 동적 유형의 런타임 인스턴스를 동적 개체라고 합니다. 생성한 동적 개체에서 워크플로를 실행하고 여러 작업을 수행할 수 있습니다.

각 동적 유형은 네임스페이스에 정의되어야 합니다. 네임스페이스는 컨테이너에서 동적 유형을 그룹화할 수 있도록 해주는 도우미 동적 개체입니다.

Dynamic Types 플러그인을 **HTTP-REST** 플러그인과 함께 사용하여 타사 **REST API** 서비스를 **Orchestrator**에 통합하고 타사 개체를 **Orchestrator** 유형으로 노출할 수 있습니다.

- 1 **Dynamic Types** 플러그인에서 네임스페이스 정의 및 유형 정의 워크플로를 실행하여 새 동적 유형과 해당 속성을 정의합니다. 새 동적 유형의 개체 및 이러한 개체와 다른 개체의 관계를 찾는 찾기 및 인벤토리 워크플로 집합이 결과로 반환됩니다.
- 2 타사 **REST API**의 입력을 수신하도록 새 찾기 및 인벤토리 워크플로를 수정합니다.
 - a **HTTP-REST** 플러그인에서 **REST** 작업 추가 워크플로를 사용하여 **REST** 작업을 생성하고 이러한 작업을 해당 **REST API** 메서드에 매핑합니다.
 - b 찾기 및 인벤토리 워크플로를 수정하여 이러한 **REST** 작업을 호출하고 해당 출력을 사용합니다.

본 장은 다음 항목을 포함합니다.

■ 동적 유형 구성 워크플로

동적 유형 구성 워크플로

Dynamic Types 플러그인 구성 패키지의 워크플로를 사용하여 동적 유형을 생성하고, **XSD** 파일에서 유형 정의를 내보내거나 가져오고, 생성한 동적 유형 간의 관계를 정의할 수 있습니다.

이러한 워크플로는 **Orchestrator** 클라이언트의 **워크플로 보기**에서 **라이브러리 > 동적 유형 > 구성**을 통해 액세스할 수 있습니다.

| 워크플로 이름 | 설명 |
|------------------|---|
| 네임스페이스 정의 | 새 네임스페이스를 정의합니다. |
| 관계 정의 | 유형 간의 새 관계를 정의합니다. |
| 유형 정의 | 지정된 네임스페이스 내에서 새 유형을 정의합니다. |
| 패키지로 구성 내보내기 | 동적 유형 정의 구성을 파일 기반 구성으로 내보냅니다. 내보낸 패키지는 다른 서버로 가져오는 데 사용될 수 있습니다. |
| 패키지에서 구성 가져오기 | 파일 기반 구성을 플러그인 구성으로 가져옵니다. |
| XSD에서 유형 정의 가져오기 | XSD 파일에서 유형 정의를 가져옵니다. |
| 네임스페이스 제거 | 네임스페이스를 제거합니다. |
| 관계 제거 | 관계를 제거합니다. |
| 유형 제거 | 유형을 제거합니다. |
| 네임스페이스 업데이트 | 네임스페이스를 업데이트합니다. |
| 유형 업데이트 | 유형을 업데이트합니다. |

PowerShell 플러그인 사용

20

PowerShell 플러그인 워크플로 라이브러리에는 PowerShell 호스트를 관리하고 사용자 지정 PowerShell 작업을 실행할 수 있는 워크플로가 포함됩니다.

Orchestrator 클라이언트에서 **인벤토리** 보기를 사용하여 사용 가능한 PowerShell 리소스를 관리할 수 있습니다. 플러그인의 스크립팅 API를 사용하여 사용자 지정 워크플로를 개발할 수 있습니다.

본 장은 다음 항목을 포함합니다.

- [VMware vRealize Orchestrator PowerShell 플러그인 소개](#)
- [PowerShell 플러그인 구성](#)
- [PowerShell 플러그인 인벤토리 사용](#)
- [PowerShell 스크립트 실행](#)
- [작업 생성](#)
- [작업 간 호출 결과 전달](#)
- [PowerShell 플러그인과 PowerCLI 통합](#)
- [샘플 워크플로](#)
- [PowerShell 플러그인 API 액세스](#)
- [PowerShell 결과 작업](#)
- [일반 PowerShell 작업의 스크립트 예제](#)
- [문제 해결](#)

VMware vRealize Orchestrator PowerShell 플러그인 소개

PowerShell 플러그인을 사용하면 vRealize Orchestrator와 Windows PowerShell 간에 상호 작용이 가능합니다.

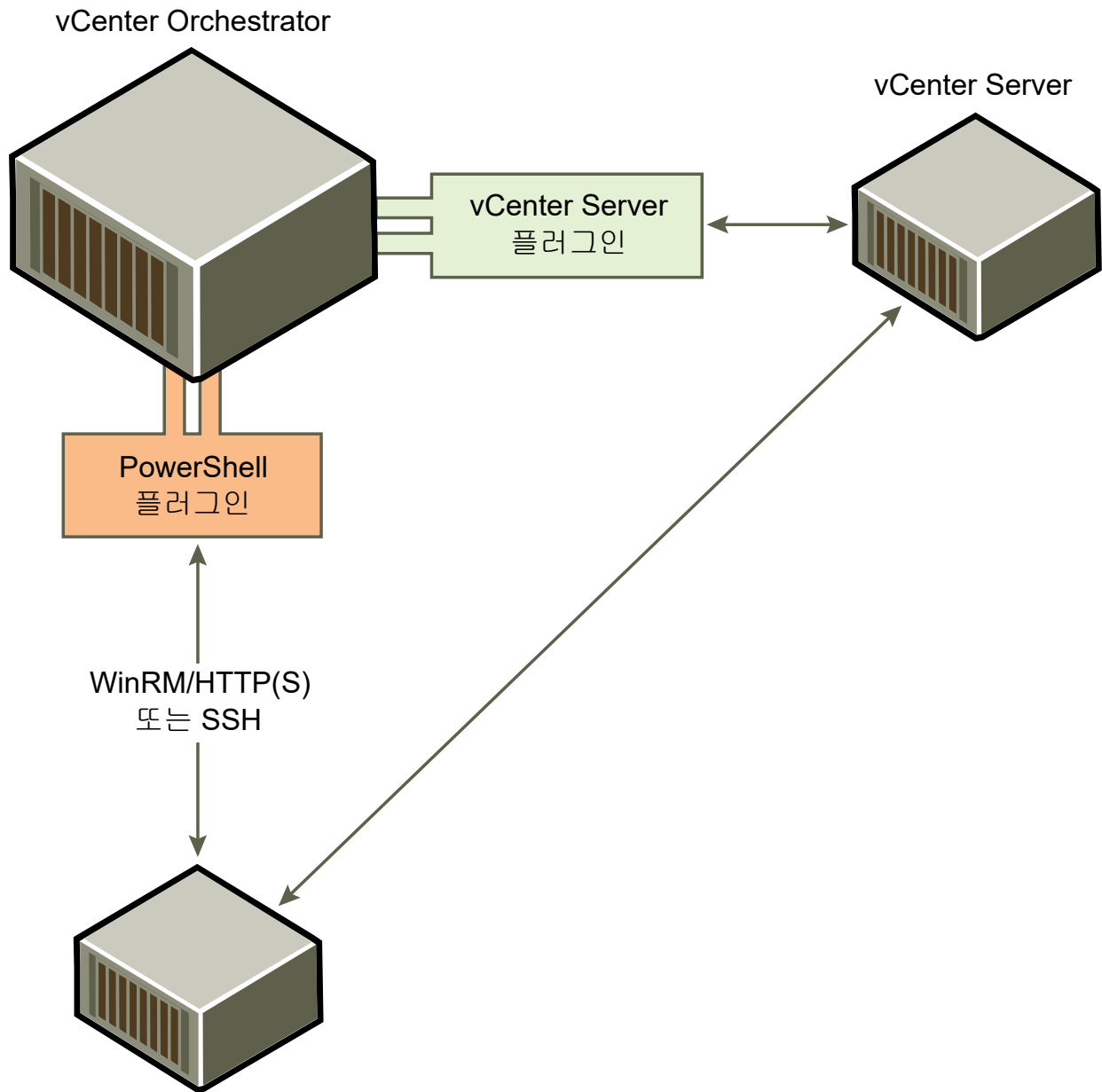
이 플러그인을 사용하면 Orchestrator 작업 및 워크플로에서 PowerShell 스크립트와 cmdlets를 호출하고 이 결과로 작업을 수행할 수 있습니다. 이 플러그인에는 표준 워크플로 집합이 포함되어 있습니다. 플러그인 API를 구현하는 사용자 지정 워크플로도 생성할 수 있습니다.

PowerShell 플러그인 구성 요소

PowerShell 플러그인은 제대로 기능하기 위해 여러 가지 구성 요소를 사용합니다.

vRealize Orchestrator 및 Windows PowerShell은 플러그인용 플랫폼을 제공하며 플러그인은 두 제품 간의 상호 작용을 제공합니다. 또한 PowerShell 플러그인은 vCenter Server 및 vSphere PowerCLI 같은 다른 구성 요소와 상호 작용할 수 있습니다.

그림 20-1. 구성 요소 관계



Windows

- WinRM 서비스 또는 OpenSSH v5.9
- PowerShell
- PowerCLI(선택 사항)

플러그인은 OpenSSH와 WinRM 통신 프로토콜을 통해 Windows PowerShell과 통신할 수 있습니다. [WinRM 구성](#) 항목을 참조하십시오.

필요한 경우 PowerShell 플러그인을 vSphere PowerCLI 및 vCenter Server와 통합할 수 있습니다. [PowerShell 플러그인과 PowerCLI 통합](#) 항목을 참조하십시오.

참고 로컬 호스트에서 모든 구성 요소를 설치할 수 있습니다. PowerShell 플러그인의 사용량, 기능 및 통신 프로토콜 요구사항은 vRealize Orchestrator 및 Windows PowerShell이 동일한 시스템에 설치된 경우 변경되지 않습니다.

PowerShell 플러그인을 사용하는 vRealize Orchestrator의 역할

Orchestrator 구성 인터페이스를 사용하여 PowerShell 플러그인을 구성해야 합니다. Orchestrator 클라이언트를 사용하여 워크플로를 실행 및 생성하고 플러그인 API에 액세스할 수 있습니다.

PowerShell 플러그인은 vRealize Orchestrator를 기반으로 합니다. Orchestrator는 VMware vCenter 인프라와 기타 기술을 관리하기 위해 확장 가능한 워크플로 라이브러리를 제공하는 개발 및 프로세스 자동화 플랫폼입니다.

Orchestrator는 개방형 플러그인 아키텍처를 통해 관리 및 운영 솔루션과의 통합을 지원합니다. PowerShell은 플러그인을 사용하여 Orchestrator에 통합할 수 있는 관리 솔루션 중 하나입니다.

Windows PowerShell과 플러그인 상호 작용

Windows PowerShell 호스트와 상호 작용하고 PowerShell 스크립트 호출과 같은 작업을 수행하는 Orchestrator 워크플로를 실행하도록 플러그인을 사용할 수 있습니다.

Windows PowerShell은 작업 기반 명령줄 셸이며 시스템 관리를 위해 설계된 스크립팅 언어입니다.

WinRM 구성

PowerShell 플러그인과 Windows PowerShell 간에 연결을 설정하려면 지원되는 통신 프로토콜 중 하나를 사용하도록 WinRM을 구성해야 합니다.

PowerShell 플러그인은 Windows 원격 관리(WinRM) 2.0을 관리 프로토콜로 지원합니다.

다음 인증 방법이 지원됩니다.

| 인증 방법 | 세부 정보 |
|----------|--|
| 기본 | 사용자 이름과 암호를 요구하는 비보안 인증 메커니즘입니다. |
| Kerberos | 클라이언트와 서버의 ID를 확인하기 위해 티켓을 사용하는 보안 인증 프로토콜입니다. |

참고 PowerShell 플러그인은 WinRM에서 사용자 자격 증명의 위임을 지원하지 않으며 CredSSP는 지원되지 않는 인증 방법입니다.

HTTP를 통한 WinRM

PowerShell 플러그인은 HTTP 프로토콜을 통해 WinRM 호스트와의 통신을 지원합니다. WinRM이 통신을 인증하는 경우에도 데이터 전송은 암호화되지 않으며 네트워크에서 일반 텍스트로 전송됩니다. 통신하는 시스템 간에 IPSec이 구성된 경우, HTTP 프로토콜을 사용해야 합니다.

기본 인증을 사용하려면, AllowUnencrypted 속성을 서비스 및 클라이언트 WinRM 구성 모두에서 **true**로 설정해야 합니다. HTTP 구성의 예는 [HTTP를 사용하도록 WinRM 구성](#)을 참조하십시오.

HTTPS를 통한 WinRM

PowerShell 플러그인은 HTTPS 프로토콜을 통해 WinRM 호스트와의 통신을 지원합니다. 보다 안전한 통신 방법으로 HTTPS 프로토콜을 사용할 수 있습니다.

HTTPS 프로토콜을 사용하려면 서버 인증을 위한 인증서를 생성하고 이 인증서를 WinRM 호스트에 설치해야 합니다. HTTPS 구성의 예는 [HTTPS를 사용하도록 WinRM 구성](#)을 참조하십시오.

HTTP를 사용하도록 WinRM 구성

HTTP 프로토콜을 통해 PowerShell 플러그인과 통신하도록 WinRM 호스트를 구성할 수 있습니다.

WinRM 호스트 시스템에서 명령을 실행하여 WinRM 구성을 수정해야 합니다. WinRM 서비스 및 WinRM 클라이언트 모두와 동일한 시스템을 사용할 수 있습니다.

중요 HTTP를 사용하도록 WinRM을 구성할 때 단계 중 하나를 건너뛰는 경우, 호스트가 추가되지 않을 수 있으며 로그 오류 메시지를 받을 수 있습니다.

```
Caused by: org.dom4j.DocumentException: Error on line -1 of document : Premature end of file.
Nested exception: Premature end of file.
at org.dom4j.io.SAXReader.read(SAXReader.java:482)
at org.dom4j.DocumentHelper.parseText(DocumentHelper.java:278)
at
com.xebialabs.overthere.cifs.winrm.connector.JdkHttpConnector.sendMessage(JdkHttpConnector.java:117)
```

절차

- 1 기본 WinRM 구성 값을 설정하려면 다음 명령을 실행합니다.

```
c:\> winrm quickconfig
```

- 2 (선택 사항) 수신기가 작동 중인지 확인하고 기본 포트를 확인하려면 다음 명령을 실행합니다.

```
c:\> winrm e winrm/config/listener
```

기본 포트는 HTTP의 경우 5985이며 HTTPS의 경우 5986입니다.

3 WinRM 서비스에서 기본 인증을 사용하도록 설정합니다.

- a 기본 인증이 허용되는지 여부를 확인하려면 다음 명령을 실행합니다.

```
c:\> winrm get winrm/config/service
```

- b 기본 인증을 사용하도록 설정하려면 다음 명령을 실행합니다.

```
c:\> winrm set winrm/config/service/auth @{Basic="true"}
```

4 WinRM 서비스에서 암호화되지 않은 데이터의 전송을 허용하려면 다음 명령을 실행합니다.

```
c:\> winrm set winrm/config/service @{AllowUnencrypted="true"}
```

5 WinRM 서비스의 채널 바인딩 토큰 강화 수준이 **강화**로 설정된 경우 해당 값을 **낮음**으로 변경합니다.

```
c:\> winrm set winrm/config/service/auth @{CbtHardeningLevel="relaxed"}
```

6 WinRM 클라이언트에서 기본 인증을 사용하도록 설정합니다.

- a 기본 인증이 허용되는지 여부를 확인하려면 다음 명령을 실행합니다.

```
c:\> winrm get winrm/config/client
```

- b 기본 인증을 사용하도록 설정하려면 다음 명령을 실행합니다.

```
c:\> winrm set winrm/config/client/auth @{Basic="true"}
```

7 WinRM 클라이언트에서 암호화되지 않은 데이터의 전송을 허용하려면 다음 명령을 실행합니다.

```
c:\> winrm set winrm/config/client @{AllowUnencrypted="true"}
```

8 WinRM 호스트 시스템이 외부 도메인에 있는 경우, 신뢰할 수 있는 호스트를 지정하려면 다음 명령을 실행합니다.

```
c:\> winrm set winrm/config/client @{TrustedHosts="host1, host2, host3"}
```

9 WinRM 서비스에 대한 연결을 테스트하려면 다음 명령을 실행합니다.

```
c:\> winrm identify -r:http://winrm_server:5985 -auth:basic -u:user_name -p:password -encoding:utf-8
```

HTTPS를 사용하도록 WinRM 구성

HTTPS 프로토콜을 통해 PowerShell 플러그인과 통신하도록 WinRM 호스트를 구성할 수 있습니다.

WinRM 호스트에는 HTTPS 프로토콜을 통해 통신하려면 인증서가 필요합니다. 인증서를 얻거나 생성할 수 있습니다. 예를 들어, .NET Framework SDK의 일부인 인증서 생성 도구(`makecert.exe`)를 사용하여 자체 서명 인증서를 생성할 수 있습니다.

사전 요구 사항

- HTTPS 프로토콜을 사용하도록 WinRM을 구성합니다. 자세한 내용은 [HTTP를 사용하도록 WinRM 구성](#) 항목을 참조하십시오.
- WinRM 호스트에서 Microsoft 관리 콘솔(`mmc.exe`)에 액세스할 수 있는지 확인합니다.

절차

1 자체 서명 인증서 생성

다음 명령줄은 `makecert.exe`를 사용하여 WinRM 호스트에서 인증서를 생성하기 위한 예시 구문을 포함합니다.

```
makecert.exe -r -pe -n "CN=host_name-3,0=organization_name" -e mm/dd/yyyy -eku
1.3.6.1.5.5.7.3.1 -ss my -sr localMachine -sky exchange -sp "Microsoft RSA
SChannel Cryptographic Provider" -sy 12 certificate_name.cer
```

2 Microsoft 관리 콘솔을 사용하여 생성한 인증서를 추가합니다.

- a `mmc.exe`를 실행합니다.
- b **파일 > 스냅인 추가/제거**를 선택합니다.
- c 사용 가능한 스냅인 목록에서 **인증서**를 선택하고 **추가**를 클릭합니다.
- d **컴퓨터 계정**을 선택하고 **다음**을 클릭합니다.
- e **완료**를 클릭합니다.
- f 이 인증서가 **콘솔 루트 > 인증서(로컬 컴퓨터) > 개인 > 인증서 및 콘솔 루트 > 인증서(로컬 컴퓨터) > 신뢰할 수 있는 루트 인증 기관 > 인증서**에 설치되어 있는지 확인합니다.

인증서가 신뢰할 수 있는 루트 인증 기관 및 개인 폴더에 설치되어 있지 않은 경우, 수동으로 설치해야 합니다.

3 올바른 지문 및 호스트 이름을 사용하여 HTTPS 수신기를 생성합니다.

다음 명령줄은 HTTPS 수신기 생성을 위한 예시 구문을 포함합니다.

```
winrm create winrm/config/Listener?Address=*&Transport=HTTPS
@{Hostname="host_name";CertificateThumbprint="certificate_thumbprint"}
```

참고 인증서 지문에서 공백을 생략합니다.

4 연결을 테스트합니다.

다음 명령줄은 연결 테스트를 위한 예시 구문을 포함합니다.

```
winrs -r:https://host_name:port_number -u:user_name -p:password hostname
```

Kerberos 인증 구성

PowerShell 호스트를 추가하고 관리할 때 Kerberos 인증을 사용할 수 있습니다.

Kerberos 인증을 사용하면 도메인 사용자는 WinRM을 통해 원격 PowerShell 사용 가능 시스템에서 명령을 실행할 수 있습니다.

절차

- 1 WinRM 서비스에서 Kerberos 인증을 사용하도록 설정합니다.
 - a Kerberos 인증이 허용되는지 여부를 확인하려면 다음 명령을 실행합니다.

```
c:\> winrm get winrm/config/service
```

- b Kerberos 인증을 사용하도록 설정하려면 다음 명령을 실행합니다.

```
c:\> winrm set winrm/config/service/auth @{Kerberos="true"}
```

- 2 WinRM 클라이언트에서 Kerberos 인증을 사용하도록 설정합니다.
 - a Kerberos 인증이 허용되는지 여부를 확인하려면 다음 명령을 실행합니다.

```
c:\> winrm get winrm/config/client
```

- b Kerberos 인증을 사용하도록 설정하려면 다음 명령을 실행합니다.

```
c:\> winrm set winrm/config/client/auth @{Kerberos="true"}
```

- 3 WinRM 서비스에 대한 연결을 테스트하려면 다음 명령을 실행합니다.

```
c:\> winrm identify -r:http://winrm_server:5985 -auth:Kerberos -u:user_name -p:password -encoding:utf-8
```

- 4 krb5.conf 파일을 만들어 다음 위치에 저장합니다.

| 운영 체제 | 경로 |
|---------|---|
| Windows | C:\Program Files\Common Files\VMware\VMware vCenter Server - Java Components\lib\security\ |
| Linux | /usr/java/jre-vmware/lib/security/(외부 vRealize Orchestrator용). /etc/krb5.conf(vRealize Orchestrator용)가 vRealize Automation에 작성됩니다. |

krb5.conf 파일의 구조는 다음과 같습니다.

```
[libdefaults]
default_realm = YOURDOMAIN.COM
udp_preference_limit = 1
[realms]
YOURDOMAIN.COM = {
kdc = kdc.yourdomain.com
default_domain = yourdomain.com
}
[domain_realm]
.yourdomain.com=YOURDOMAIN.COM
yourdomain.com=YOURDOMAIN.COM
```

krb5.conf는 해당 값과 함께 특정한 구성 매개 변수를 포함해야 합니다.

| Kerberos 구성 태그 | 세부 정보 |
|----------------|---|
| default_realm | Active Directory 서버를 인증할 때 클라이언트에서 사용하는 기본 Kerberos 영역입니다. 참고 대문자로 입력해야 합니다. |
| kdc | KDC(키 배포 센터) 역할을 하고 Kerberos 티켓을 발행하는 도메인 컨트롤러입니다. |
| default_domain | 정규화된 도메인 이름을 생성하는 데 사용되는 기본 도메인입니다. 참고 이 태그는 Kerberos 4 호환성을 위해 사용됩니다. |

참고 기본적으로 Java Kerberos 구성은 UDP 프로토콜을 사용합니다. TCP 프로토콜만 사용하려면 `udp_preference_limit` 매개 변수를 값 **1**로 지정해야 합니다.

참고 Kerberos 인증을 위해서는 FQDN(정규화된 도메인 이름) 호스트 주소가 필요합니다.

중요 `krb5.conf` 파일을 추가하거나 수정한 경우 Orchestrator 서버 서비스를 다시 시작해야 합니다.

PowerShell 플러그인 구성

Orchestrator 클라이언트를 사용하여 PowerShell 플러그인을 구성해야 합니다.

구성 워크플로

구성 워크플로 범주에는 PowerShell 호스트를 관리할 수 있는 워크플로가 포함됩니다.

이러한 워크플로는 Orchestrator 클라이언트의 **워크플로 보기**에서 **라이브러리 > PowerShell > 구성**을 통해 액세스할 수 있습니다.

| 워크플로 이름 | 설명 |
|---------------------|--|
| PowerShell 호스트 추가 | 플러그인 인벤토리에 PowerShell 호스트를 추가합니다. |
| PowerShell 호스트 제거 | 플러그인 인벤토리에서 PowerShell 호스트를 제거합니다. |
| PowerShell 호스트 업데이트 | 플러그인 인벤토리에서 지정된 PowerShell 호스트를 업데이트합니다. |
| PowerShell 호스트 검증 | 지정된 PowerShell 호스트의 구성을 검증합니다. |

PowerShell 호스트 추가

워크플로를 실행하여 PowerShell 호스트를 추가하고 호스트 연결 매개 변수를 구성합니다. 원격 또는 로컬 PowerShell 호스트에 대한 연결을 설정할 수 있습니다.

절차

- 1 Orchestrator 클라이언트에 관리자로 로그인합니다.
- 2 Orchestrator 클라이언트에서 **워크플로 보기**를 클릭합니다.

- 3 계층형 워크플로 목록에서 **라이브러리 > PowerShell > 구성**을 확장하여 PowerShell 호스트 추가 워크플로로 이동합니다.
- 4 PowerShell 호스트 추가 워크플로를 마우스 오른쪽 버튼으로 클릭하고 **워크플로 시작**을 선택합니다.
- 5 **이름** 텍스트 상자에 호스트의 이름을 입력합니다.
- 6 **호스트/IP** 텍스트 상자에 호스트의 주소를 입력합니다.

참고 Kerberos 인증을 위해서는 FQDN(정규화된 도메인 이름) 호스트 주소가 필요합니다.

- 7 (선택 사항) **포트** 텍스트 상자에 호스트의 포트를 입력합니다.
HTTP에 포트 5985를 사용하거나 HTTPS 프로토콜에 포트 5986을 사용합니다.
- 8 플러그인을 연결할 PowerShell 호스트 유형을 선택합니다.
 - a 전송 프로토콜을 선택합니다.

참고 HTTPS 전송 프로토콜을 사용하는 경우 원격 PowerShell 호스트의 인증서를 Orchestrator 키 저장소로 가져옵니다.

- b 인증 유형을 선택합니다.

중요 Kerberos 인증을 사용하려면 WinRM 서비스에서 해당 인증을 사용하도록 설정해야 합니다.

- 9 플러그인에서 PowerShell 호스트에 연결하는 데 사용할 세션 모드의 유형을 선택합니다.

| 옵션 | 설명 |
|---------|---|
| 공유 세션 | 플러그인에서 공유 자격 증명을 사용하여 원격 호스트에 연결합니다. 공유 세션에 대한 PowerShell 호스트 자격 증명을 제공해야 합니다. |
| 사용자별 세션 | Orchestrator 클라이언트가 로그인된 사용자로부터 자격 증명을 검색합니다. 사용자별 세션 모드를 사용하려면 사용자@도메인 형식으로 Orchestrator에 로그인해야 합니다. |

- 10 **Shell 코드 페이지** 드롭다운 메뉴에서 PowerShell이 사용하는 인코딩 유형을 선택합니다.
- 11 **제출**을 클릭하여 워크플로를 실행합니다.

결과

워크플로가 성공적으로 실행되면 PowerShell 호스트가 **인벤토리** 보기에 표시됩니다.

PowerShell 플러그인 인벤토리 사용

PowerShell 플러그인은 연결된 PowerShell 호스트의 모든 개체를 **인벤토리** 보기에 노출합니다. **인벤토리** 보기를 사용하여 권한 부여 요소를 추가하거나 PowerShell 개체에서 워크플로를 실행할 수 있습니다.

플러그인의 인벤토리 내에서 PowerShell 호스트와 해당 스냅인 및 cmdlet을 관리할 수 있습니다. 각 원격 호스트는 스냅인을 포함할 수 있으며, 각 스냅인은 cmdlet을 포함할 수 있습니다.

인벤토리 개체에 사용할 수 있는 워크플로를 표시하려면 **도구 > 사용자 기본 설정 > 인벤토리**로 이동한 후 **인벤토리에서 컨텍스트 메뉴 사용** 확인란을 선택합니다. 이 옵션을 사용하도록 설정하면 Orchestrator 인벤토리에서 개체를 마우스 오른쪽 버튼으로 클릭했을 때 해당 개체에 사용할 수 있는 모든 워크플로가 표시됩니다.

PowerShell 스크립트 실행

워크플로를 실행하여 PowerShell 호스트에서 외부 또는 사용자 지정 스크립트를 호출할 수 있습니다.

PowerShell 스크립트 호출

플러그인 인벤토리의 호스트에서 기존 또는 사용자 지정 PowerShell 스크립트를 실행할 수 있습니다.

사전 요구 사항

- Orchestrator 클라이언트에 관리자로 로그인되어 있는지 확인합니다.
- PowerShell 호스트에 연결되었는지 여부를 **인벤토리** 보기에서 확인합니다.

절차

- 1 Orchestrator 클라이언트에서 **워크플로** 보기를 클릭합니다.
- 2 계층형 워크플로 목록에서 **라이브러리 > PowerShell**을 확장하여 PowerShell 스크립트 호출 워크플로로 이동합니다.
- 3 PowerShell 스크립트 호출 워크플로를 마우스 오른쪽 버튼으로 클릭하고 **워크플로 시작**을 선택합니다.
- 4 스크립트를 실행할 PowerShell 호스트를 선택합니다.
- 5 **스크립트** 텍스트 상자에서 실행할 PowerShell 스크립트를 입력하거나 붙여 넣습니다.
- 6 **제출**을 클릭하여 워크플로를 실행합니다.

외부 스크립트 호출

플러그인 인벤토리의 호스트에서 외부 PowerShell 스크립트를 실행할 수 있습니다.

외부 PowerShell 스크립트는 **.ps1** 파일에 포함되어 있습니다. 실행할 **.ps1** 파일이 PowerShell 호스트에 있어야 합니다.

사전 요구 사항

- Orchestrator 클라이언트에 관리자로 로그인되어 있는지 확인합니다.
- PowerShell 호스트에 연결되었는지 여부를 **인벤토리** 보기에서 확인합니다.
- 스크립트에서 참조할 수 있는 다른 **.ps1** 파일에 대한 액세스 권한이 있는지 확인합니다.

절차

- 1 Orchestrator 클라이언트에서 **워크플로** 보기를 클릭합니다.

- 2 계층형 워크플로 목록에서 **라이브러리 > PowerShell**을 확장하여 외부 스크립트 호출 워크플로로 이동합니다.
- 3 외부 스크립트 호출 워크플로를 마우스 오른쪽 버튼으로 클릭하고 **워크플로 시작**을 선택합니다.
- 4 스크립트를 실행할 PowerShell 호스트를 선택합니다.
- 5 **이름** 텍스트 상자에 실행할 외부 .ps1 스크립트의 파일 이름을 입력합니다.

참고 .ps1 파일이 기본 폴더에 없는 경우 절대 파일 경로를 입력해야 합니다. 시스템 환경 변수를 사용하여 스크립트 경로를 지정할 수 있습니다. 예를 들면 `$env:HOME\PATH\test1.ps1`과 같습니다.

- 6 **인수** 텍스트 상자에 스크립트 인수를 입력합니다.
유효한 구문은 PowerShell 콘솔에서 사용되는 구문과 같습니다.
- 7 **제출**을 클릭하여 워크플로를 실행합니다.

작업 생성

워크플로를 실행하여 PowerShell 스크립트 또는 PowerShell cmdlet을 기반으로 작업을 생성할 수 있습니다. 생성된 작업을 사용자 지정 워크플로의 빌딩 블록으로 사용할 수 있습니다.

PowerShell 스크립트에서 작업 생성

워크플로를 실행하여 제공한 PowerShell 스크립트에서 작업을 생성할 수 있습니다. 필요한 경우 생성된 작업을 실행할 수 있는 샘플 워크플로를 생성할 수 있습니다.

자리 표시자를 사용하여 생성한 작업의 스크립트를 사용자 지정할 수 있습니다. 각 자리 표시자에 대해 워크플로는 생성된 작업에서 **string** 유형의 해당 작업 매개 변수를 생성합니다. 작업을 실행하면 실제 값을 자리 표시자를 교체하기 위한 작업 매개 변수로 제공할 수 있습니다.

사전 요구 사항

- Orchestrator 클라이언트에 관리자로 로그인되어 있는지 확인합니다.
- PowerShell 호스트에 연결되었는지 여부를 **인벤토리** 보기에서 확인합니다.

절차

- 1 Orchestrator 클라이언트에서 **워크플로** 보기를 클릭합니다.
- 2 계층형 워크플로 목록에서 **라이브러리 > PowerShell > 생성**을 확장하고 PowerShell 스크립트 워크플로에서 작업 생성으로 이동합니다.
- 3 PowerShell 스크립트 워크플로에서 작업 생성을 마우스 오른쪽 버튼으로 클릭하고 **워크플로 시작**을 선택합니다.

4 스크립트 텍스트 상자에 작업을 생성할 PowerShell 스크립트를 입력하거나 붙여 넣습니다.

참고 `{#ParamName#}`을 사용자 입력을 위한 자리 표시자로 사용할 수 있습니다. 자리 표시자가 string 유형이면 큰따옴표를 사용하여 자리 표시자의 값을 작업에 전달해야 합니다.

다음 스크립트는 생성된 작업 매개 변수를 스크립트 매개 변수에 연결하는 방법에 대한 예제입니다.

```
param($name={#ParamName#})
echo $name;
```

5 이름 텍스트 상자에 생성하려는 작업의 이름을 입력합니다.

6 작업을 생성할 기존 모듈을 선택합니다.

7 워크플로 생성 여부를 선택합니다.

| 옵션 | 설명 |
|-----|--|
| 예 | 생성된 작업을 실행할 수 있는 샘플 워크플로를 생성합니다. 워크플로를 생성할 폴더를 선택해야 합니다. 참고 생성된 워크플로의 이름은 미리 정의된 문자열 호출 스크립트 및 생성된 작업의 이름으로 구성됩니다. |
| 아니오 | 샘플 워크플로가 생성되지 않습니다. |

8 제출을 클릭하여 워크플로를 실행합니다.

다음에 수행할 작업

사용자 지정 워크플로에서 생성된 작업을 통합할 수 있습니다.

PowerShell Cmdlet용 작업 생성

워크플로를 실행하여 제공한 PowerShell Cmdlet 및 매개 변수 집합에 대한 작업을 생성할 수 있습니다. 이 작업으로 Orchestrator에서 PowerShell 기능을 사용할 수 있습니다. 필요한 경우 생성된 작업을 실행하는 샘플 워크플로를 생성할 수 있습니다.

PowerShell 스크립트 엔진으로 많은 데이터 유형을 사용할 수 있습니다. 사용할 수 있는 데이터 유형에는 Integer, Boolean, Char, .NET 어셈블리에서 사용 가능한 모든 유형 또는 사용자 정의 유형 등 기본 유형이 포함됩니다. PowerShell Cmdlet 정의에 따라 작업을 생성할 때 입력 및 출력 Cmdlet 매개 변수는 Orchestrator 플랫폼이 지원하는 유형으로 표시됩니다. PowerShell 플러그인은 유형 매핑을 정의합니다. 일반적으로 기본 유형은 해당 Orchestrator 유형에 매핑되고, 복잡한 유형은 PowerShellRemotePSObject 개체로 표시됩니다.

사전 요구 사항

- Orchestrator 클라이언트에 관리자로 로그인되어 있는지 확인합니다.
- PowerShell 호스트에 연결되었는지 여부를 **인벤토리** 보기에서 확인합니다.

절차

- 1 Orchestrator 클라이언트에서 **워크플로** 보기를 클릭합니다.
- 2 계층형 워크플로 목록에서 **라이브러리 > PowerShell > 생성**을 확장하고 PowerShell Cmdlet 워크플로에서 작업 생성으로 이동합니다.
- 3 PowerShell Cmdlet 워크플로에서 작업 생성을 마우스 오른쪽 버튼으로 클릭하고 **워크플로 시작**을 선택합니다.
- 4 생성하는 작업을 사용할 때 실행할 PowerShell Cmdlet을 선택합니다.
- 5 Cmdlet에 대한 매개 변수 집합을 선택합니다.

매개 변수 집합 정의 값이 **매개 변수 집합 정의** 텍스트 상자에 표시됩니다.

참고 매개 변수 집합 정의 텍스트 상자의 문자열을 편집하여 매개 변수 집합 정의 값을 수정할 수 없습니다. 문자열을 검토하여 매개 변수 집합이 포함하는 매개 변수의 정보에 대해 확인할 수 있습니다.

- 6 **이름** 텍스트 상자에 생성하려는 작업의 이름을 입력합니다.
- 7 작업을 생성할 기존 모듈을 선택합니다.
- 8 워크플로 생성 여부를 선택합니다.

| 옵션 | 설명 |
|-----|--|
| 예 | 생성된 작업을 실행할 수 있는 샘플 워크플로를 생성합니다. 워크플로를 생성할 폴더를 선택해야 합니다. 참고 생성된 워크플로의 이름은 미리 정의된 문자열 Execute Cmdlet 및 생성된 작업의 이름으로 구성됩니다. |
| 아니요 | 샘플 워크플로가 생성되지 않습니다. |

- 9 **제출**을 클릭하여 워크플로를 실행합니다.

다음에 수행할 작업

사용자 지정 워크플로에서 생성된 작업을 통합할 수 있습니다.

작업 간 호출 결과 전달

PowerShell 플러그인은 PowerShell 스크립트 호출 간에 결과를 매개 변수로 전달하는 기능을 지원합니다. 결과를 올바르게 전달하려면 두 호출 모두 동일한 세션에서 발생해야 합니다.

PowerShell 플러그인과 PowerCLI 통합

VMware vSphere PowerCLI와 같은 타사 스냅인에서 제공되는 기능을 PowerShell 플러그인에서 사용할 수 있습니다.

타사 스냅인 기능을 사용하려면 PowerShell 호스트에서 해당 스냅인을 사용할 수 있어야 합니다. 현재 세션에서 스냅인을 로드하려면 AddPsSnapin 작업도 호출해야 합니다. PowerCLI를 사용하는 경우 스냅인의 이름을 VMware.VimAutomation.Core로 설정해야 합니다.

PowerShell 플러그인은 미리 생성된 타사 스냅인용 작업을 제공하지 않습니다. PowerShell cmdlet용 작업 생성 워크플로를 실행하여 타사 스냅인용 작업을 생성할 수 있습니다. [PowerShell Cmdlet용 작업 생성](#) 항목을 참조하십시오.

com.vmware.library.powershell.converter 패키지에는 VC:<SomeObjectType> 개체에서 PowerCLI의 해당 개체로 변환할 수 있는 기본 빌딩 블록이 포함되어 있습니다. 이 기능을 사용하면 vCenter Server 플러그인의 워크플로가 PowerShell 플러그인의 워크플로와 상호 작용할 수 있으며 두 플러그인 간에 매개 변수를 전달할 수 있습니다.

Converter 워크플로

Converter 워크플로 범주에서 샘플 워크플로를 사용하여 PowerShell 플러그인과 PowerCLI 간의 통합을 테스트할 수 있습니다. 통합을 테스트하려면 PowerCLI가 PowerShell 호스트에 설치되어 있어야 합니다.

Converter 샘플 워크플로는 플러그인에서 사용할 수 있는 변환 기능을 보여 줍니다.

참고 PowerShell 플러그인은 PowerCLI와 vCenter Server 플러그인에서 사용할 수 있는 일부 유형을 지원하지 않습니다. 지원되지 않는 유형은 예외를 반환합니다.

이러한 워크플로는 Orchestrator 클라이언트의 **워크플로 보기**에서 **라이브러리 > PowerShell > 샘플 > Converter**를 통해 액세스할 수 있습니다.

| 워크플로 이름 | 설명 |
|--|--|
| PSObject를 vCO 개체로 변환 | PowerShellRemotePSObject를 VC:<SomeObjectType>으로 변환합니다. |
| PSObject를 vCO 개체로 변환하고 다시 PSObject로 변환 | PowerShellRemotePSObject를 VC:<SomeObjectType>으로 변환하고 다시 되돌립니다. |
| vCO 개체를 PSObject로 변환 | VC:<SomeObjectType>를 PowerShellRemotePSObject으로 변환합니다. |

샘플 워크플로

샘플 워크플로 범주에는 기본 사용 사례를 테스트할 수 있는 워크플로가 포함됩니다.

이러한 워크플로는 Orchestrator 클라이언트의 **워크플로 보기**에서 **라이브러리 > PowerShell > 샘플**을 통해 액세스할 수 있습니다.

| 워크플로 이름 | 설명 |
|-----------------|---|
| API를 통해 스크립트 호출 | 사용 가능한 스크립팅 API를 통해 PowerShell 스크립트를 호출하는 방법을 보여 줍니다. |
| 디렉토리 콘텐츠 나열 | PowerShell 호스트 파일 시스템의 디렉토리 콘텐츠를 나열합니다. |
| 파이프라인 실행 예제 | 파이프에 정렬된 여러 cmdlet을 실행할 수 있는 방법을 보여 줍니다. |
| 가상 시스템 상태 설정/해제 | 가상 시스템의 전원 상태를 설정/해제합니다. |

PowerShell 플러그인 API 액세스

Orchestrator API 탐색기에서는 PowerShell 플러그인 API를 검색할 수 있으며 스크립팅된 요소에서 사용 가능한 JavaScript 개체에 대한 설명서를 볼 수 있습니다.

절차

- 1 Orchestrator 클라이언트에 관리자로 로그인합니다.
- 2 Orchestrator 클라이언트 또는 워크플로, 정책 및 작업 편집기의 **스크립팅** 탭에서 API 탐색기에 액세스합니다.
 - Orchestrator 클라이언트에서 API 탐색기에 액세스하려면 Orchestrator 클라이언트 도구 모음에서 **도구 > API 탐색기**를 클릭합니다.
 - 워크플로, 정책 및 작업 편집기의 **스크립팅** 탭에서 API 탐색기에 액세스하려면 왼쪽에 있는 **API 검색**을 클릭합니다.
- 3 PowerShell 플러그인 API 개체의 계층형 목록을 확장하려면 왼쪽 창에서 **PowerShell** 모듈을 두 번 클릭합니다.

다음에 수행할 작업

API 요소에서 코드를 복사하여 스크립팅 상자에 붙여 넣을 수 있습니다. API 스크립팅에 대한 자세한 내용은 "VMware vRealize Orchestrator를 사용한 개발"을 참조하십시오.

PowerShell 결과 작업

PowerShell 플러그인 API의 개체를 사용하여 Windows PowerShell이 반환하는 결과로 작업할 수 있습니다.

PowerShellInvocationResult 클래스의 메서드를 사용하여 실행한 스크립트에 대한 정보를 검색할 수 있습니다.

| 방법 | 설명 |
|----------------------|---|
| getErrors() | 스크립트를 호출하는 동안 PowerShell 엔진에서 보고한 오류 목록을 반환합니다. |
| getInvocationState() | 스크립트의 상태입니다. 가능한 값은 Completed 또는 Failed 입니다. |
| getHostOutput() | PowerShell 콘솔에 표시되는 스크립트의 출력입니다. |
| getResults() | PowerShell 엔진에서 반환되는 개체입니다. 반환되는 개체의 유형은 PowershellRemotePSObject입니다. |

PowershellRemotePSObject는 PowerShell 엔진에서 반환되는 개체의 원격 표현입니다.

PowershellRemotePSObject는 getXml() 메서드를 호출하여 액세스할 수 있는 결과의 XML 직렬화를 포함합니다.

또한 PowerShell 플러그인은 XML 결과를 래핑하고 특정 개체 속성에 보다 쉽게 액세스할 수 있도록 하는 개체 모델을 제공합니다. `getRootObject()` 메서드는 개체 모델에 대한 액세스를 제공합니다. 일반적으로 `getRootObject()` 메서드는 다음 규칙을 사용하여 PowerShell 유형을 Orchestrator에서 사용 가능한 유형에 매핑합니다.

- 반환되는 개체가 기본 PowerShell 유형인 경우 해당하는 Orchestrator 기본 유형에 개체가 매핑됩니다.
- 반환되는 개체가 `collection` 유형인 경우 이 개체는 `ArrayList`로 표현됩니다.
- 반환되는 개체가 `dictionary` 유형인 경우 이 개체는 `Hashtable`로 표현됩니다.
- 반환되는 개체가 `complex` 유형인 경우 이 개체는 `PSObject`로 표현됩니다.

일반 PowerShell 작업의 스크립트 예제

JavaScript 예제를 잘라내고, 붙여 넣고, 편집하여 일반 PowerShell 작업에 대한 스크립트를 작성할 수 있습니다.

스크립팅에 대한 자세한 내용은 "vRealize Orchestrator 개발자 가이드"를 참조하십시오.

예제: API를 통해 PowerShell 스크립트 실행

JavaScript를 사용하여 플러그인 API를 통해 PowerShell 스크립트를 실행할 수 있습니다.

이 예제 스크립트는 다음 작업을 수행합니다.

- PowerShell 호스트로 세션을 엽니다.
- 실행할 스크립트를 제공합니다.
- 호출 결과를 확인합니다.
- 세션을 닫습니다.

```
var sess;
try {
    //Open session to PowerShell host
    var sess = host.openSession()
    //Set executed script
    var result = sess.invokeScript('dir')

    //Check for errors
    if (result.invocationState == 'Failed'){
        throw "PowerShellInvocationError: Errors found while executing script \n" +
result.getErrors();
    }
    //Show result
    System.log( result.getHostOutput() );
} catch (ex){
    System.error (ex)
} finally {
    if (sess) {
        //Close session
    }
}
```

```

    host.closeSession( sess.getSessionId() );
  }
}

```

예제: 결과 작업

JavaScript를 사용하여 PowerShell 스크립트 실행 결과로 작업을 할 수 있습니다.

이 예제 스크립트는 다음 작업을 수행합니다.

- 호출 상태를 확인합니다.
- 결과에서 값을 추출합니다.
- RemotePSObject 유형을 확인합니다.

```

var sess = host.openSession()
sess.addCommandFromString("dir " + directory)
var invResult = sess.invokePipeline();
//Show result
System.log( invResult.getHostOutput() );

//Check for errors
if (invResult.invocationState == 'Failed'){
System.error(invResult.getErrors());
} else {
//Get PowerShellRemotePSObject
var psObject = invResult.getResults();
var directories = psObject.getRootObject();

var isList = directories instanceof Array
if ( isList ){
    for (idx in directories){
        var item = directories[idx];
        if ( item instanceof 'System.IO.FileInfo' ){//Check type of object
            System.log( item.getProperty('FullName') );//Extract value from result
        }
    }
} else {
    System.log( directories.getProperty('FullName') );//Extract value from result
}
}

host.closeSession( sess.getSessionId());

```

예제: 사용자 지정 자격 증명 연결

JavaScript를 사용하여 사용자 지정 자격 증명으로 PowerShell 호스트에 연결할 수 있습니다.

```

var sess;
try {
    sess = host.openSessionAs(userName, password);

    var invResult = sess.invokeScript('$env:username');
}

```

```

//Check for errors
if (invResult.invocationState == 'Failed'){
    System.error(invResult.getErrors());
} else {
    //Show result
    System.log( invResult.getHostOutput() );
}
} catch (ex){
    System.error (ex)
} finally {
    if (sess) {
        host.closeSession( sess.getSessionId());
    }
}
}

```

문제 해결

PowerShell 플러그인 사용 도중 문제가 발생하는 경우, 문제를 이해하거나 해결하기 위해 문제 해결 주제를 참조하여 해결책을 찾아볼 수 있습니다.

Kerberos 이벤트 로깅 사용

문제 해결을 위해 KDC(키 배포 센터) 시스템에서 Kerberos 이벤트 로깅을 사용할 수 있습니다.

사전 요구 사항

Windows 레지스트리를 백업합니다.

절차

- 1 KDC(키 배포 센터) 역할을 하는 도메인 컨트롤러에 로그인합니다.
- 2 **관리자** 권한으로 레지스트리 편집기를 실행합니다.
- 3 레지스트리 창에서 **HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Lsa\Kerberos\Parameters**를 확장합니다.
- 4 **LogLevel** 레지스트리 키 값이 없는 경우, 마우스 오른쪽 버튼으로 클릭하여 생성합니다.
 - a **매개 변수**를 마우스 오른쪽 버튼으로 클릭하고 **새 > DWORD(32비트) 값**을 선택하고 **LogLevel**을 입력합니다.
 - b **매개 변수**를 선택하고 오른쪽 창에서 **LogLevel**을 두 번 클릭하고 **1**을 **값 데이터:** 텍스트 상자에 입력합니다.

Windows Server 2003 이상에서 새로운 설정이 재부팅 없이 적용됩니다.

결과

Kerberos 오류 이벤트 항목은 시스템 Windows 이벤트 로그에 기록됩니다.

다음에 수행할 작업

Kerberos 이벤트 로깅을 사용하지 않으려면 **LogLevel** 레지스트리 키 값을 삭제하고 해당 값 데이터를 **0**으로 변경합니다.

Kerberos 데이터베이스에서 서버를 찾을 수 없음

Kerberos 인증을 사용하여 서버를 추가한 후 해당 서버가 제대로 추가되지 않아 서버를 찾을 수 없는 경우가 있을 수 있습니다.

문제

서버에 연결하려는 경우 Kerberos 데이터베이스에서 서버를 찾을 수 없습니다.

```
No valid credentials provided (Mechanism level: No valid credentials provided (Mechanism level:
Server not found in Kerberos database (7)))
```

원인

이 오류는 여러 가지 잘못된 구성 때문에 발생할 수 있습니다.

- PowerShell 호스트는 도메인의 일부가 아닙니다.
- 영역 매핑에 대한 이 호스트는 올바르지 않습니다.
- PowerShell 호스트의 서비스 주체 이름이 잘못 작성되었습니다.

참고 Kerberos 인증은 대상이 IP 주소인 경우 작동하지 않습니다.

해결책

Kerberos 인증을 사용하여 PowerShell 호스트를 추가하는 경우 DNS 또는 NetBIOS 대상을 입력합니다.

Kerberos 티켓을 가져올 수 없음

잘못된 자격 증명을 제공한 경우 플러그인에서 Kerberos 티켓을 가져오지 못합니다.

문제

플러그인 인벤토리에 호스트를 추가할 수 없으며 다음과 같은 오류 메시지가 결과로 반환됩니다.

```
Pre-authentication information was invalid (24)
```

원인

잘못된 자격 증명을 제공했습니다.

해결책

올바른 자격 증명을 제공합니다.

다른 시간 설정으로 인한 Kerberos 인증 실패

Kerberos 구성을 사용하는 환경에서 시간 설정 차이로 인해 인증에 실패할 수 있습니다.

문제

호스트의 초기 인증 또는 리소스 액세스에 Kerberos를 사용하려는 시도가 실패하고 다음 오류 메시지가 나타납니다.

```
Clock Skew
```

원인

환경 내 컴퓨터의 시스템 시간이 도메인 컨트롤러 또는 다른 컴퓨터와 5분 넘게 차이가 나는 경우 Kerberos 인증에 실패합니다.

해결책

환경 내 시스템 시간을 동기화합니다.

Kerberos 인증 세션 모드 실패

공유 세션 또는 사용자별 세션에서 Kerberos 인증을 사용하는 경우 PowerShell 호스트를 추가하지 못할 수 있습니다.

문제

공유 세션 또는 사용자별 세션을 사용하여 플러그인 인벤토리에 PowerShell 호스트를 추가하려는 경우 워크플로에 실패하고 다음 오류가 나타납니다.

```
Null realm name (601) - default realm not specified (Dynamic Script Module name : addPowerShellHost#16)
```

원인

기본 영역이 Kerberos 구성 파일 `krb5.conf`에 지정되어 있지 않고 사용자 이름의 일부로 제공되지도 않았습니다.

해결책

Kerberos 구성 파일에서 기본 영역을 제공하거나, Kerberos를 인증할 때 사용자 이름에서 영역을 포함합니다.

영역의 키 배포 센터에 연결할 수 없음

`krb5.conf` 파일에 맞춤법 오류가 있는 경우 호스트를 추가할 때 오류가 발생할 수 있습니다.

문제

호스트를 추가할 때 Kerberos 인증에서 *사용자 영역*의 KDC(키 배포 센터)에 연결할 수 없습니다.

```
Cannot get kdc for realm YOURREALM.COM
```

원인

krb5.conf 파일의 libdefaults 및 realms 섹션에 맞춤법 오류가 있을 수 있습니다.

해결책

krb5.conf 파일의 libdefaults 및 realms 섹션에서 맞춤법이 올바른지 확인합니다.

기본 영역을 찾을 수 없음

Kerberos 인증이 필요한 Orchestrator 워크플로는 Kerberos 구성 파일에 올바른 형식 또는 인코딩이 없는 경우, 오류가 발생할 수 있습니다.

문제

Kerberos 인증은 기본 영역을 식별할 수 없습니다.

```
기본 영역을 찾을 수 없음
```

원인

Kerberos 구성 파일인 krb5.conf(vRealize Orchestrator Appliance에 업로드함)는 UNIX가 아닌 운영 체제에서 편집되었습니다. 그 결과 형식과 인코딩이 잘못되었을 수 있습니다.

해결책

Orchestrator Appliance에서 krb5.conf 파일을 읽으려면 파일의 형식이 UNIX여야 하며 문자 인코딩이 UTF-8 형식의 ANSI여야 합니다.

Multi-Node 플러그인 사용

21

Multi-Node 플러그인 워크플로 라이브러리에는 계층 오케스트레이션, Orchestrator 인스턴스 관리 및 Orchestrator 작업 확장을 위한 워크플로가 포함됩니다.

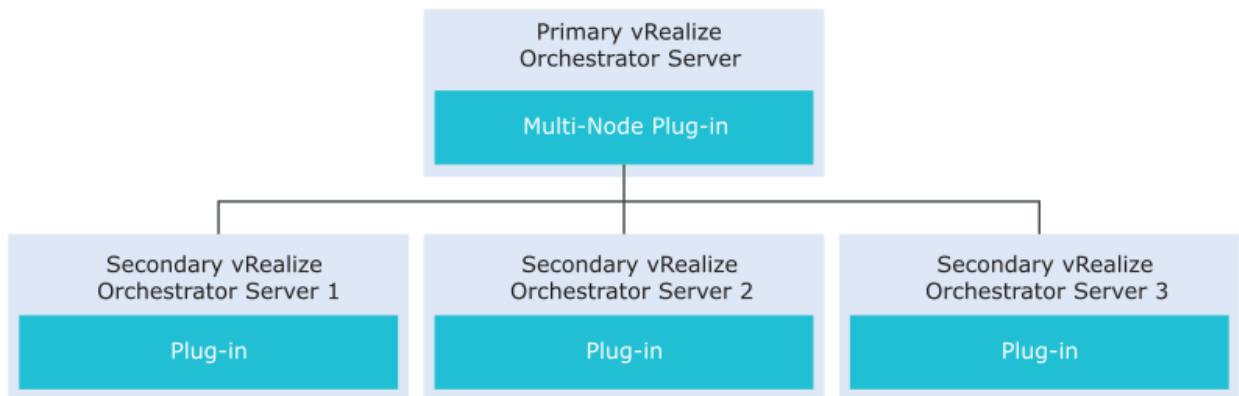
본 장은 다음 항목을 포함합니다.

- [vRealize Orchestrator Multi-Node 플러그인 소개](#)
- [Multi-Node 플러그인 구성](#)
- [프록시 워크플로 사용](#)
- [Multi-Node 플러그인 인벤토리 사용](#)
- [Multi-Node 플러그인 API 액세스](#)
- [다중 노드 플러그인 사용 사례](#)

vRealize Orchestrator Multi-Node 플러그인 소개

Multi-Node 플러그인은 vRealize Orchestrator 서버 간의 기본-보조 관계를 생성합니다. 이 관계는 패키지 관리 및 워크플로 실행 영역으로 확장됩니다.

그림 21-1. 다중 노드 플러그인 스키마



이 플러그인은 계층 오케스트레이션, vRealize Orchestrator 인스턴스 관리 및 vRealize Orchestrator 작업 확장을 위한 표준 워크플로 집합을 포함합니다.

Multi-Node 플러그인 구성

Orchestrator 클라이언트를 사용하여 Multi-Node 플러그인을 구성해야 합니다.

서버 구성 워크플로

서버 구성 워크플로 범주에는 연결된 Orchestrator 서버를 구성할 수 있는 워크플로가 포함됩니다.

이러한 워크플로는 Orchestrator 클라이언트의 **워크플로 보기**에서 **라이브러리 > Orchestrator > 서버 구성**을 통해 액세스할 수 있습니다.

| 워크플로 이름 | 설명 |
|----------------------|--|
| Orchestrator 서버 추가 | 플러그인 인벤토리에 Orchestrator 서버를 추가합니다. |
| Orchestrator 서버 삭제 | 플러그인 인벤토리에서 Orchestrator 서버를 제거하고 이 서버에 대해 생성된 모든 프로세스를 삭제합니다. |
| Orchestrator 서버 업데이트 | 해당 세부 정보를 변경하여 플러그인 인벤토리에서 Orchestrator 서버를 업데이트합니다. |

Orchestrator 서버 추가

워크플로를 실행하여 새 vRealize Orchestrator 서버에 대한 연결을 설정할 수 있습니다.

사전 요구 사항

기본 및 보조 Orchestrator 서버의 버전이 같은지 확인합니다.

절차

- 1 Orchestrator 클라이언트에 관리자로 로그인합니다.
- 2 Orchestrator 클라이언트에서 **워크플로 보기**를 클릭합니다.
- 3 계층형 워크플로 목록에서 **라이브러리 > Orchestrator > 서버 구성**을 확장하여 Orchestrator 서버 추가 워크플로로 이동합니다.
- 4 Orchestrator 서버 추가 워크플로를 마우스 오른쪽 버튼으로 클릭하고 **워크플로 시작**을 선택합니다.
- 5 새 서버의 세부 정보를 제공합니다.
- 6 연결을 공유할지 여부를 선택합니다.

| 옵션 | 설명 |
|-----|--|
| 아니요 | 원격 Orchestrator 서버에 연결하는 데 사용되는 로그인된 사용자의 자격 증명입니다. |
| 예 | 모든 사용자가 동일한 자격 증명을 사용하여 원격 Orchestrator 서버에 액세스할 수 있습니다. 공유 연결에 대한 자격 증명을 제공합니다. |

- 7 **제출**을 클릭하여 워크플로를 실행합니다.

프록시 워크플로 사용

프록시 워크플로를 사용하여 로컬 Orchestrator 서버와 원격 Orchestrator 서버 워크플로 간의 상호 작용을 관리할 수 있습니다.

Multi-Node 플러그인을 사용하여 원격 워크플로와 상호 작용하는 로컬 워크플로를 생성할 수 있습니다. 이러한 로컬 워크플로를 프록시 워크플로라고 합니다. 프록시 워크플로는 Multi-Node 플러그인의 인벤토리에서 입력 매개 변수를 가져옵니다. 프록시 워크플로를 실행하면 매개 변수가 원격 워크플로에 필요한 유형으로 변환됩니다. 원격 워크플로의 실행이 완료되면 출력 매개 변수가 기본 Orchestrator 서버의 로컬 표현으로 다시 변환됩니다.

동기 프록시 워크플로

동기 유형의 프록시 워크플로는 원격 워크플로의 작업 계약 및 API를 유지합니다.

모든 동기 프록시 워크플로의 스키마는 동일하지만 서로 다른 스크립팅을 포함합니다.



동기 프록시 워크플로는 원격 워크플로가 완료된 후 실행을 완료하고 출력 매개 변수를 제공합니다.

로컬 워크플로는 원격 워크플로의 결과를 기다리는 동안 서버 리소스를 소비하지 않습니다.

성공적인 실행이 완료되면 프록시 워크플로의 출력 매개 변수에 원격 워크플로 토큰의 로컬 표현이 포함됩니다. 단순한 유형(예: 부울, 숫자, 문자열 등)인 경우 로컬 Orchestrator 서버의 다른 워크플로에서 출력 매개 변수를 직접 사용할 수 있습니다.

비동기 프록시 워크플로

비동기 프록시 워크플로를 사용하여 원격 워크플로 실행을 최적화할 수 있습니다.

모든 비동기 프록시 워크플로의 스키마는 동일하지만 서로 다른 스크립팅을 포함합니다.



비동기 프록시 워크플로는 원격 워크플로 토큰 개체의 로컬 래퍼인 결과를 즉시 반환합니다. 프록시 워크플로는 이 토큰을 사용하여 실행 상태를 확인하며, 원격 워크플로의 실행이 완료되면 출력 매개 변수를 검색합니다. 단순한 유형(예: 부울, 숫자, 문자열 등)인 경우 로컬 Orchestrator 서버의 다른 워크플로에서 출력 매개 변수를 직접 사용할 수 있습니다.

원격 실행 워크플로

원격 실행 워크플로 범주에는 프록시 워크플로를 관리할 수 있는 워크플로가 포함됩니다.

원격 실행 표준 워크플로

프록시 워크플로를 생성하는 워크플로는 Orchestrator 클라이언트의 **워크플로 보기**에서 **라이브러리 > Orchestrator > 원격 실행**을 통해 액세스할 수 있습니다.

| 워크플로 이름 | 설명 |
|------------------|---|
| 다중 프록시 작업 생성 | 여러 서버에서 워크플로를 실행하는 다중 프록시 작업을 생성합니다. |
| 프록시 워크플로 | 원격 Orchestrator 서버에서 워크플로를 시작하는 데 사용할 수 있는 프록시 워크플로를 생성합니다. |
| 폴더에서 프록시 워크플로 생성 | 원격 Orchestrator 서버의 폴더에 있는 모든 워크플로에 대한 프록시 워크플로를 생성합니다. |

서버 프록시

서버 프록시 관리용 워크플로는 Orchestrator 클라이언트의 **워크플로 보기**에서 **라이브러리 > Orchestrator > 원격 실행 > 서버 프록시**를 통해 액세스할 수 있습니다.

| 워크플로 이름 | 설명 |
|------------------------------------|---|
| Orchestrator 서버에 대한 프록시 워크플로 생성 | 원격 서버의 구조를 미러링하여 로컬 Orchestrator 서버에 대한 프록시 워크플로를 생성합니다. |
| Orchestrator 서버에 대한 프록시 워크플로 삭제 | 로컬 Orchestrator 서버에 대한 프록시 워크플로를 제거하고 생성된 모든 워크플로를 삭제합니다. |
| Orchestrator 서버에 대한 프록시 워크플로 새로 고침 | 원격 서버에서 로컬 Orchestrator 서버에 대한 모든 프록시 워크플로를 다시 생성합니다. |

Multi-Node 플러그인 인벤토리 사용

Multi-Node 플러그인은 **인벤토리** 보기에서 연결된 vRealize Orchestrator 서버의 모든 인벤토리를 미러링합니다. **인벤토리** 보기를 사용하여 권한 부여 요소를 추가하거나 원격 Orchestrator 서버에서 워크플로를 실행할 수 있습니다.

인벤토리 개체에 사용할 수 있는 워크플로를 표시하려면 **도구 > 사용자 기본 설정 > 인벤토리**로 이동한 후 **인벤토리에서 컨텍스트 메뉴 사용** 확인란을 선택합니다. 이 옵션을 사용하도록 설정하면 Orchestrator 인벤토리에서 개체를 마우스 오른쪽 버튼으로 클릭했을 때 해당 개체에 사용할 수 있는 모든 워크플로가 표시됩니다.

단일 원격 서버의 인벤토리는 두 가지 주요 요소인 시스템 개체와 플러그인 개체로 구성됩니다. 두 개체 모두 원격 개체를 로컬로 사용할 수 있는 유형으로 래핑하는 래퍼입니다.

시스템 개체

시스템 개체는 **시스템**이라는 상위 수준 그룹에 속하며, 구성, 패키지, 워크플로, 작업 및 관련 폴더를 포함합니다. 원격 시스템 개체에는 개별 래퍼 유형이 있습니다.

플러그인 개체

플러그인 개체는 원격 Orchestrator 서버에 연결된 모든 플러그인의 인벤토리를 미러링합니다. 원격 플러그인 개체는 모두 단일 로컬 유형 **VCO:RemotePluginObject**에 래핑됩니다.

원격 관리 워크플로

원격 관리 워크플로 범주에는 원격 Orchestrator 인스턴스에서 패키지 및 워크플로를 관리할 수 있는 워크플로가 포함됩니다.

패키지

원격 패키지 관리용 워크플로는 Orchestrator 클라이언트의 **워크플로** 보기에서 **라이브러리 > Orchestrator > 원격 관리 > 패키지**를 통해 액세스할 수 있습니다.

| 워크플로 이름 | 설명 |
|----------------|--|
| 패키지 삭제 | 원격 Orchestrator 서버에서 패키지와 해당 콘텐츠를 삭제합니다. |
| 이름별 패키지 삭제 | 원격 Orchestrator 서버에서 이름별로 패키지와 해당 콘텐츠를 삭제합니다. |
| 로컬 서버에서 패키지 배포 | 로컬 Orchestrator 서버에서 원격 Orchestrator 서버로 패키지를 배포합니다. |

| 워크플로 이름 | 설명 |
|-------------------|---|
| 원격 서버에서 패키지 배포 | 하나의 원격 Orchestrator 서버에서 원격 Orchestrator 서버 목록으로 워크플로를 배포합니다. |
| 로컬 서버에서 여러 패키지 배포 | 로컬 Orchestrator 서버에서 원격 Orchestrator 서버로 여러 패키지를 배포합니다. |

워크플로

원격 워크플로 관리용 워크플로는 Orchestrator 클라이언트의 **워크플로 보기**에서 **라이브러리 > Orchestrator > 원격 관리 > 워크플로**를 통해 액세스할 수 있습니다.

| 워크플로 이름 | |
|-------------------|--|
| 원격 워크플로 삭제 | 원격 Orchestrator 서버에서 워크플로를 삭제합니다. |
| 종료된 워크플로 실행 모두 삭제 | 종료된 워크플로 실행을 원격 워크플로에서 모두 삭제합니다. |
| 로컬 서버에서 워크플로 배포 | 로컬 Orchestrator 서버에서 원격 Orchestrator 서버 목록으로 워크플로를 배포합니다. |
| 원격 서버에서 워크플로 배포 | 원격 Orchestrator 서버에서 다른 원격 Orchestrator 서버 목록으로 워크플로를 배포합니다. |

Multi-Node 플러그인 API 액세스

Orchestrator에서는 Multi-Node 플러그인 API를 검색할 수 있으며 스크립팅된 요소에서 사용 가능한 JavaScript 개체에 대한 설명서를 볼 수 있는 API 탐색기를 제공합니다.

절차

- 1 Orchestrator 클라이언트에 관리자로 로그인합니다.
- 2 Orchestrator 클라이언트 또는 워크플로, 정책 및 작업 편집기의 **스크립팅** 탭에서 API 탐색기에 액세스합니다.
 - Orchestrator 클라이언트에서 API 탐색기에 액세스하려면 Orchestrator 클라이언트 도구 모음에서 **도구 > API 탐색기**를 클릭합니다.
 - 워크플로, 정책 및 작업 편집기의 **스크립팅** 탭에서 API 탐색기에 액세스하려면 왼쪽에 있는 **API 검색**을 클릭합니다.
- 3 Multi-Node 플러그인 API 개체의 계층형 목록을 확장하려면 왼쪽 창에서 **VCO** 모듈을 두 번 클릭합니다.

다음에 수행할 작업

API 요소에서 코드를 복사하여 스크립팅 상자에 붙여 넣을 수 있습니다. API 스크립팅에 대한 자세한 내용은 "VMware vRealize Orchestrator를 사용한 개발"을 참조하십시오.

다중 노드 플러그인 사용 사례

다중 노드 플러그인 사용 사례에는 원격 및 프록시 워크플로 유지 관리 정보뿐 아니라 로컬 Orchestrator 서버에서 원격 서버로 패키지 가져오기, 다중 프록시 작업 사용 등의 사용자 시나리오가 포함됩니다.

다중 프록시 작업 생성

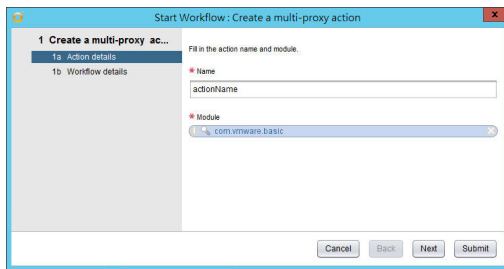
다중 프록시 작업 생성 워크플로를 실행하여 여러 서버에서 워크플로를 실행할 수 있습니다.

이후 단계에 Orchestrator 서버에서 워크플로를 실행할 수 있도록 작업을 생성할 수 있습니다.

절차

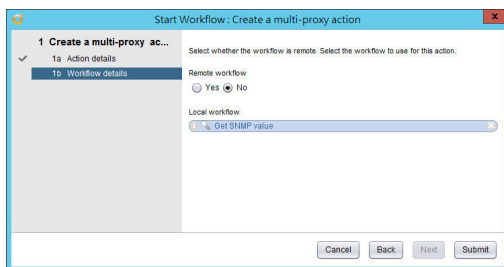
- 1 Orchestrator 클라이언트에 관리자로 로그인합니다.
- 2 Orchestrator 클라이언트에서 **워크플로** 보기를 클릭합니다.
- 3 계층형 워크플로 목록에서 **라이브러리 > Orchestrator > 원격 실행**을 확장하여 다중 프록시 작업 생성 워크플로로 이동합니다.
- 4 다중 프록시 작업 생성 워크플로를 마우스 오른쪽 버튼으로 클릭하고 **워크플로 시작**을 선택합니다.
- 5 **작업 이름** 텍스트 상자에 작업의 이름을 입력합니다.

작업 이름은 공백 없이 영숫자만 포함해야 합니다.



이름이 같은 다른 작업이 있는 경우에도 새 작업이 생성됩니다.

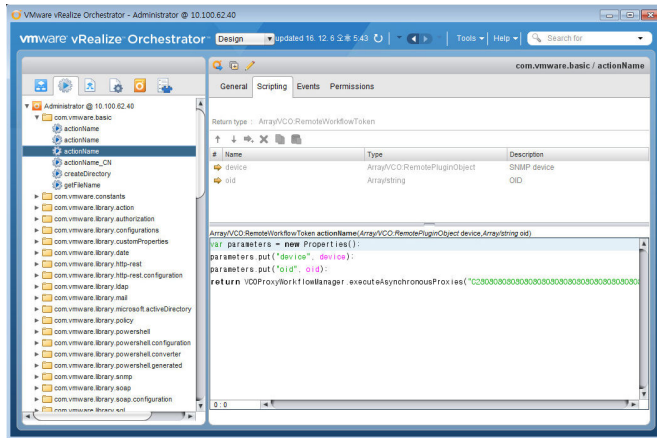
- 6 작업을 추가할 모듈을 선택합니다.
- 7 워크플로가 로컬인지 또는 원격인지 선택합니다.



- 8 이 작업에 사용할 워크플로를 선택합니다.
- 9 **제출**을 클릭하여 워크플로를 실행합니다.

결과

생성된 작업은 소스 워크플로와 동일한 매개 변수를 허용하지만 여러 개체를 선택한 경우 매개 변수를 어레이로 승격합니다. 어레이의 값은 인덱싱됩니다.



원격 및 프록시 워크플로 유지 관리

원격 및 프록시 워크플로가 변경된 경우 프록시를 업데이트하거나 삭제(더 이상 필요 없는 경우)할 수 있습니다. 유지 관리를 위해 **Multi-Node** 플러그인에서는 프록시 및 원격 워크플로 정보를 업데이트하거나 삭제할 수 있는 워크플로를 제공합니다.

프록시 워크플로 관리용 워크플로는 Orchestrator 클라이언트의 **워크플로 보기**에서 **라이브러리 > Orchestrator > 원격 실행 > 서버 프록시**를 통해 액세스할 수 있습니다.

| 워크플로 이름 | 설명 |
|------------------------------------|---|
| Orchestrator 서버에 대한 프록시 워크플로 새로 고침 | 원격 서버에서 로컬 Orchestrator 서버에 대한 모든 프록시 워크플로를 다시 생성합니다. |
| Orchestrator 서버에 대한 프록시 워크플로 삭제 | 로컬 Orchestrator 서버에 대한 프록시 워크플로를 제거하고 생성된 모든 워크플로를 삭제합니다. |

프록시 워크플로의 향후 유지 관리용 워크플로는 Orchestrator 클라이언트의 **워크플로 보기**에서 **라이브러리 > Orchestrator > 원격 관리 > 워크플로**를 통해 액세스할 수 있습니다.

| 워크플로 이름 | 설명 |
|-------------------|---|
| 종료된 워크플로 실행 모두 삭제 | 종료된 워크플로 실행을 원격 워크플로에서 모두 삭제합니다. |
| 원격 워크플로 삭제 | 원격 Orchestrator 서버에서 워크플로를 삭제합니다. |
| 로컬 서버에서 워크플로 배포 | 로컬 Orchestrator 서버에서 원격 Orchestrator 서버 목록으로 워크플로를 배포합니다. |

로컬 서버에서 패키지 배포

워크플로를 실행하여 로컬 Orchestrator 서버에서 원격 Orchestrator 서버로 패키지를 배포할 수 있습니다.

이 예제에서는 로컬 서버에서 원격 서버 어레이로 패키지를 배포할 수 있습니다.

절차

- 1 Orchestrator 클라이언트에 관리자로 로그인합니다.
- 2 Orchestrator 클라이언트에서 **워크플로 보기**를 클릭합니다.

- 3 계층형 워크플로 목록에서 **라이브러리 > Orchestrator > 원격 관리**를 확장하여 로컬 서버에서 패키지 배포 워크플로로 이동합니다.
- 4 로컬 서버에서 패키지 배포를 마우스 오른쪽 버튼으로 클릭하고 **워크플로 시작**을 선택합니다.
- 5 로컬 저장소에서 배포할 패키지를 선택합니다.
- 6 패키지를 배포할 대상 원격 서버를 선택합니다.
- 7 원격 서버 패키지를 덮어쓸지 여부를 선택합니다.

| 옵션 | 설명 |
|-----|--|
| 예 | 패키지된 요소의 버전을 무시하고 원격 서버의 패키지가 대체됩니다. |
| 아니요 | 서버의 버전 확인 및 패키지 배포가 수행됩니다. 확인에 성공한 후 패키지가 배포됩니다. |

- 8 **제출**을 클릭하여 워크플로를 실행합니다.

결과

워크플로를 실행한 후 로그 보기 및 플러그인의 인벤토리에 상태 정보가 표시됩니다.

vAPI(vCloud Suite API) 플러그인 사용

22

vCloud Suite API 플러그인은 vCloud Suite API 공급자가 노출하는 API를 사용할 수 있는 기능을 제공합니다. vCloud Suite API는 vCloud Suite 끝점을 통해 vCenter Server로 요청을 전송하여 가상 환경의 리소스에 액세스할 수 있는 서비스 지향 아키텍처를 제공합니다.

이 플러그인에는 표준 워크플로 및 예제 워크플로 집합이 포함되어 있습니다. 이 플러그인을 구현하는 사용자 지정 워크플로를 생성하여 가상 환경에서 작업을 자동화할 수도 있습니다. vCloud Suite API에 대한 자세한 내용은 "VMware vCloud Suite SDK 프로그래밍 가이드"를 참조하십시오.

본 장은 다음 항목을 포함합니다.

- [vCloud Suite API 플러그인 구성](#)
- [vCloud Suite API 플러그인 API 액세스](#)

vCloud Suite API 플러그인 구성

플러그인에 포함된 구성 워크플로를 실행하여 vCloud Suite API를 구성할 수 있습니다.

vCloud Suite API 메타 모델 가져오기

vCloud Suite API 플러그인은 vCloud Suite API 공급자 메타데이터 서비스를 쿼리하여 vCloud Suite API 서비스를 동적으로 검색합니다. 메타데이터 서비스를 노출하지 않는 vCloud Suite API 공급자는 지원되지 않습니다.

vCloud Suite API 메타 모델을 가져와 그 뒤에 끝점을 추가합니다.

절차

- 1 Orchestrator 클라이언트에 관리자로 로그인합니다.
- 2 Orchestrator 클라이언트에서 **워크플로** 보기를 클릭합니다.
- 3 계층형 워크플로 목록에서 **라이브러리 > VAPI**를 확장하여 **vAPI 메타 모델 가져오기** 워크플로로 이동합니다.
- 4 **vAPI 메타 모델 가져오기** 워크플로를 마우스 오른쪽 버튼으로 클릭하고 **워크플로 시작**을 선택합니다.
- 5 **vAPI 끝점 URL** 텍스트 상자에 vCloud Suite API 끝점의 URL을 입력합니다.

6 보안 프로토콜 연결을 사용할지 여부를 선택합니다.

| 옵션 | 설명 |
|-----|---|
| 아니요 | 보안 프로토콜 연결을 사용하지 않고 vCloud Suite API 메타 모델을 가져옵니다. |
| 예 | 보안 프로토콜 연결을 사용하여 vCloud Suite API 메타 모델을 가져오려면 <ol style="list-style-type: none"> 인증서 경고를 무시하고 vCloud Suite 끝점을 자동으로 수락할지 여부를 선택합니다. vCloud Suite 끝점을 인증할 사용자 자격 증명을 제공합니다. |

7 제출을 클릭하여 워크플로를 실행합니다.

다음에 수행할 작업

vCloud Suite API 끝점 추가

vCloud Suite API 끝점 추가

vCloud Suite API 끝점을 추가합니다.

사전 요구 사항

vCloud Suite API 메타 모델을 가져옵니다.

절차

- 1 Orchestrator 클라이언트에 관리자로 로그인합니다.
- 2 Orchestrator 클라이언트에서 **워크플로** 보기를 클릭합니다.
- 3 계층형 워크플로 목록에서 **라이브러리 > VAPI**를 확장하여 **vAPI 끝점 추가** 워크플로로 이동합니다.
- 4 **vAPI 끝점 추가** 워크플로를 마우스 오른쪽 버튼으로 클릭하고 **워크플로 시작**을 선택합니다.
- 5 **vAPI 끝점 URL** 텍스트 상자에 vCloud Suite API 끝점의 URL을 입력합니다.
- 6 보안 프로토콜 연결을 사용할지 여부를 선택합니다.

| 옵션 | 설명 |
|-----|---|
| 아니요 | 보안 프로토콜 연결을 사용하지 않고 vCloud Suite API 메타 모델을 가져옵니다. |
| 예 | 보안 프로토콜 연결을 사용하여 vCloud Suite API 메타 모델을 가져오려면 <ol style="list-style-type: none"> 인증서 경고를 무시하고 vCloud Suite 끝점을 자동으로 수락할지 여부를 선택합니다. vCloud Suite 끝점을 인증할 사용자 자격 증명을 제공합니다. |

7 제출을 클릭하여 워크플로를 실행합니다.

vCloud Suite API 플러그인 API 액세스

Orchestrator에서는 vCloud Suite API 플러그인 API를 검색할 수 있으며 스크립팅된 요소에서 사용 가능한 JavaScript 개체에 대한 설명서를 볼 수 있는 API 탐색기를 제공합니다.

절차

- 1 Orchestrator 클라이언트에 관리자로 로그인합니다.
- 2 Orchestrator 클라이언트 또는 워크플로, 정책 및 작업 편집기의 **스크립팅** 탭에서 API 탐색기에 액세스합니다.
 - Orchestrator 클라이언트에서 API 탐색기에 액세스하려면 Orchestrator 클라이언트 도구 모음에서 **도구 > API 탐색기**를 클릭합니다.
 - 워크플로, 정책 및 작업 편집기의 **스크립팅** 탭에서 API 탐색기에 액세스하려면 왼쪽에 있는 **API 검색**을 클릭합니다.
- 3 vCloud Suite API 플러그인 API 개체의 계층형 목록을 확장하려면 왼쪽 창에서 **VAPI** 모듈을 두 번 클릭합니다.

다음에 수행할 작업

API 요소에서 코드를 복사하여 스크립팅 상자에 붙여 넣을 수 있습니다. API 스크립팅에 대한 자세한 내용은 "VMware vRealize Orchestrator를 사용한 개발"을 참조하십시오.