開發 VMware vRealize Orchestrator 的 Web Services 用戶端

vRealize Orchestrator 7.4

vmware®

您可以在 VMware 網站上找到最新的技術說明文件，網址為:

https://docs.vmware.com/tw/

如果您對此文件有何想法，請將您的回應意見提交至：

docfeedback@vmware.com

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

內容

開發用於 VMware vRealize Orchestrator 的網路服務用戶端

開發用於 VMware vRealize Orchestrator 的網路服務用戶端提供相關用於開發 VMware® vRealize Orchestrator 網路服務用戶端的資訊。

Orchestrator 提供的網路服務 API 可讓您開發透過網路存取及使用工作流程的應用程式。Orchestrator 提供的具象狀態傳輸 (REST) API 能讓您用來執行工作流程上的各種作業。

適合對象

此資訊適用於想要透過 RESTful 網路服務存取網路 Orchestrator 處理程序的網路應用程式開發人員。

開發 Web 服務用戶端

1

VMware vRealize Orchestrator 提供的 Web 服務 API 可讓您開發透過 Web 存取的應用程式。Orchestrator Web 服務 API 的主要用途是允許您將 Orchestrator 工作流程整合在自訂的 Web 型應用程式中。

Orchestrator 提供以具象狀態傳輸 (REST) API 為基礎的 Web 服務 API。Orchestrator REST API 會公開 Orchestrator 詳細目錄及已安裝外掛程式其詳細目錄中的物件，作為您可在預先定義的 URL 存取的資源。這些 URL 的 HTTP 要求會造成透過工作流程來觸發作業。Orchestrator REST API 透過一組 RESTful Web 服務公開詳細目錄作為資源，您可以使用這些資源擷取工作流程定義、執行工作流程、檢查執行中工作流程的狀態、取消執行工作流程、處理等待中使用者互動、擷取工作流程的呈現等等。

使用 vRealize Orchestrator REST API 2

Orchestrator REST API 提供的功能可讓您經由 HTTP 直接與 Orchestrator 伺服器通訊，並在工作流程上執行各種工作流程相關作業。

Orchestrator REST API 公開 Orchestrator 伺服器詳細目錄的物件，並在預先定義的 URL 將外掛程式當成資源安裝。您讓 HTTP 在這些 URL 呼叫觸發 Orchestrator 中的作業。您可透過此方式執行工作流程上的各種工作：

- 執行工作流程、排程工作流程、擷取工作流程執行、回應使用者互動及取消工作流程執行。
- 擷取有關工作流程的詳細資訊，如其輸入與輸出參數及其簡報。
- 擷取有關工作流程執行的詳細資訊，如其狀態、產生的記錄、開始日期與結束日期。
- 瀏覽 Orchestrator 詳細目錄及安裝的外掛程式。
- 匯入與匯出工作流程、動作及套件。

您可透過使用 Orchestrator REST API，在可使用任何程式語言建立的自訂應用程式中，輕鬆整合 Orchestrator 工作流程。

Orchestrator REST API 也提供 eTag 支援以及快取回應資料的機制。

本章節討論下列主題:

- 執行標記作業

針對 Orchestrator 及第三方系統進行驗證

您必須針對透過 Orchestrator REST API 提出 HTTP 要求的 Orchestrator 進行驗證。若您使用 Orchestrator REST API 存取第三方系統的資源，如 vCenter Server 或 vRealize Automation，則您也必須針對該系統進行驗證。

例如：若要存取 Orchestrator 詳細目錄中的所有工作流程，您必須針對 Orchestrator 進行驗證。然而，若要針對 vCenter Server 執行工作流程，您必須針對 Orchestrator 和 vCenter Server 進行驗證。

Orchestrator REST API 的驗證機制，會因您使用 vRealize Automation 或 vSphere 設定 Orchestrator 作為驗證提供者而有所不同。若 Orchestrator 使用 vCenter Single Sign-On，則視您組態而定，可使用 vCenter Single Sign-On 伺服器發行的金鑰持有者 Token 進行驗證。如果 Orchestrator 是以 vRealize Automation 設定，則您可以透過 OAuth bearer 存取 Token 進行驗證。

若您在 Orchestrator REST API 的最上層 URL 提出 HTTP 要求，則您不需要針對 Orchestrator 進行驗證。Orchestrator REST API 最上層的 URL 為 https://*orchestrator_host*:*port*/vco/api/。

備註 外部 Orchestrator 的預設連接埠號碼為 8281。內嵌於 vRealize Automation 的 Orchestrator 執行個體預設連接埠號碼為 443。

REST API 最上層的 URL `GET` 要求會傳回 URL 至可透過 API 存取的所有資源。若要在這些 URL 提出 HTTP 要求，您必須針對 Orchestrator 進行驗證。

透過 Orchestrator REST API 使用 vCenter Single Sign-On 驗證

若使用 vSphere 驗證模式設定 Orchestrator 啟用 vCenter Single Sign-On 伺服器，您需要主體金鑰持有者的 Token 才能透過 Orchestrator REST API 存取 Orchestrator 中的系統物件。若要透過 Orchestrator 伺服器存取 vCenter Server 或使用 vCenter Single Sign-On 伺服器的第三方系統，您需委派 Orchestrator 金鑰持有者 Token 與您的主體 Token。

若設定 Orchestrator 啟用 vCenter Single Sign-On 伺服器，您必須使用有效的憑證驗證，Orchestrator 才能管理金鑰持有者 Token。

存取 Orchestrator 中的系統物件

您可在詳細目錄 URL 存取 Orchestrator 中的系統物件，以及 REST API 的目錄服務。

- https://*orchestrator_host*:*port*/vco/api/inventory/System/
- https://*orchestrator_host*:*port*/vco/api/catalog/System/

當您存取 Orchestrator 中的系統物件後，即傳遞您在詳細目錄或目錄服務中提出 `Authorization` HTTP 要求標頭中的主體金鑰持有者 Token。

例如：若要擷取型別 `Workflow` 的所有系統物件，您需在 https://*orchestrator_host*:*port*/vco/api/catalog/System/Workflow/ 提出 `GET` 要求。若要驗證 Orchestrator，您需傳遞 `Authorization` 要求標頭中的主體金鑰持有者 Token。

存取第三方系統中的物件

若要透過 Orchestrator REST API 執行以 vCenter Single Sign-On 伺服器登錄的第三方系統作業，您必須驗證 Orchestrator 及第三方系統。您需在 HTTP 呼叫中加入兩個透過 Orchestrator REST API 提出的標頭。

- `Authorization`。您必須通過此標頭中的主體金鑰持有者 Token。
- `VCOAuthorization`。您必須傳遞此標頭中 Orchestrator 委派的主體金鑰持有者 Token。您必須取得 vCenter Single Sign-On 伺服器中由 Orchestrator 委派的 Token。Orchestrator 代表您使用委派 Token 驗證第三方系統。

例如，若要透過 Orchestrator REST API 執行使用虛擬機器的工作流程，您需在 Orchestrator 和 vCenter Server 中存取資源。若要驗證 Orchestrator 和 vCenter Server，您必須傳遞您提出 `Authorization` 標頭中的主體金鑰持有者 Token，以及 `VCOAuthorization` 標頭中的委派 Token。您可透過此方式利用主體 Token 驗證 Orchestrator，而 Orchestrator 則會代表您利用委派 Token 驗證 vCenter Server。

vCenter Single Sign-On 伺服器將 Orchestrator 當成解決方案，而每一個解決方案皆以唯一使用者名稱向 vCenter Single Sign-On 伺服器登錄。您透過傳遞 Orchestrator 解決方案使用者名稱和主體金鑰持有者 Token 至 vCenter Single Sign-On 伺服器，要求 Orchestrator 委派 Token。vCenter Single Sign-On 伺服器發出的 Token 為 Orchestrator 用於代表您驗證第三方系統的委派金鑰持有者 Token。

範例 2-1. 在 vCenter Single Sign-On 模式中取得工作階段

下列範例代碼將在 vCenter Single Sign-On 模式中取得工作階段

```
URI uri = URI.create("https://orchestrator-server:8281/vco/api");
VcoSessionFactory sessionFactory = new DefaultVcoSessionFactory(uri);

//provide the address of the vCenter Single Sign-On server
URI ssoUri = URI.create("https://sso-server:7444/ims/STSService?wsdl");

//set the tokens to be valid for an hour
long lifeTimeSeconds = 60 * 60;

//create a factory for vCenter Single Sign-On tokens
SsoAuthenticator sso = new SsoAuthenticator(URI ssoUri, URI adminUri, VcoSessionFactory
vcoSessionFactory, long lifeTimeSeconds);

//provide vCenter Single Sign-On credentials
SsoAuthentication authentication = sso.createSsoAuthentication("username", "password");

VcoSession session = sessionFactory.newSession(authentication);
//use session here
```

取得 Orchestrator 的解決方案使用者名稱

vCenter Single Sign-On 伺服器將 Orchestrator 當成解決方案，而每一個解決方案皆以唯一使用者名稱向 vCenter Single Sign-On 伺服器登錄。若要能從 vCenter Single Sign-On 伺服器要求委派用於 Orchestrator 的金鑰持有者 Token，您需要 Orchestrator 的解決方案使用者名稱。

先決條件

確認您擁有 vCenter Single Sign-On 伺服器發出的有效主體金鑰持有者 Token。

程序

1 在 Orchestrator 解決方案使用者名稱的 URL 提出 GET 要求：

```
GET https://{orchestrator_host}:{port}/vco/api/users/
```

2 在要求的 Authorization 標頭中提供主體金鑰持有者 Token。

回應的 <user solution-user="*OrchestratorSolutionUserName*"/> 元素包含 Orchestrator 解決方案使用者名稱。下列為範例 Orchestrator 解決方案使用者名稱。

```
<user xmlns="http://www.vmware.com/vco" admin-rights="true" solution-
user="vCO-15d98795afa5b0d6f47ee3aeab3">
```

下一個

使用 Orchestrator 解決方案使用者名稱及您的主體金鑰持有者 Token ，要求從 vCenter Single Sign-On 伺服器委派金鑰持有者 Token。

透過已設定的 vRealize Automation 驗證，使用 vRealize Orchestrator REST API SDK

在多承租人或單承租人環境中，您可透過已設定的 vRA 驗證使用 REST API SDK。

若要取得以下程式碼所需的驗證 Token (Oauth2.0)，請參閱使用 Oauth2.0 驗證取得 vRO REST API 授權 (2148518) 知識庫文章。

備註 在 vRealize Automation 驗證模式下取得工作階段

下列範例程式碼可在 vRealize Automation 驗證模式下，於單承租人和多承租人環境中取得工作階段。

- 若未啟用多租戶：

```
URI uri = URI.create("https://orchestrator-server:8283/vco/api");
VcoSessionFactory sessionFactory = new DefaultVcoSessionFactory(uri);
String token =
"eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJqdGkiOiJjNzg4NWNiYS1hZTFmLTRiM2UtYmYyYi04ZmRmNzY3N"
+
"GZiZWEiLCJwcm4iOiJhZG1pbmlzdHJhdG9yQFZTUEhFUkUuTE9DQUwiLCJkb21haW4iOiJ2c3BoZXJlLmxvY2FsIiwidXNlcl9
pZCI6Ij"
+
```

```
"MiLCJhdXRoX3RpbWUiOjE1MDIyMDIxMTAsImlzcyI6Imh0dHBzOi8vc2YtMjktMTAtMjkuc29mLW1idS5lbmcudm13YXJlLmN"
+
"vbS9TQUFTL3QvdnNwaGVyZS5sb2NhbC9hdXRoIiwiYXVkIjoiaHR0cHM6Ly9zZi0yOS0xMC0yOS5zb2YtbWJ1LmVuZy52bXdhc
mUuY29tL"
+
"1NBQVMvdC92c3BoZXJlLmxvY2FsL2F1dGgvb2F1dGh0b2tlbiIsImN0eCI6Ilt7XCJtdGRcIjpcInVybjpvYXNpczpuYW1lczp
"
+
"0YzpTQU1MOjIuMDphYzpjbGFzc2VzOlBhc3N3b3JkUHJvdGVjdGVkVHJhbnNwb3J0XCIsXCJpYXRcIjoxNTAyMjAyMTEwLFwia
WRcIjoxM"
+
"n1dIiwic2NwIjoidXNlciIsImlkcCI6IjAiLCJlbWwiOiJhZG1pbmlzdHJhdG9yQHNmLTI5LTEwLTI5LnNvZi1tYnUuZW5nLnZ
"
+
"td2FyZS5jb20iLCJjaWQiOiJjYWZlX2NsaS1yRlJlNEx6bURnIiwiZGlkIjoiIiwid2lkIjoiIiwiZXhwIjoxNTAyMjMwOTEwL
CJpYXQiOj"
+
"E1MDIyMDIxMTAsInN1YiI6IjQ1ZjQwNWUzLTNlNTgtNGJmZC1hNzMwLTQ1MjU4OWIxOGUxNyIsInBybl90eXBlIjoiVVNFUiJ9
."
+ "G9gEQPtmEH5jYab-
IlTK8NFYcwc3JZCEEjsmpUSH6oxLmZKEf-1JbsysBVH4ufqmGah3GMvmy6PUiTTamLRLFkCLwa50OQCQ-qgKutZl21R"
+ "m740qBKLhmBB0NQg19ysMAVJNSxapFzirmWurF_5CKpv4WM7Y8H_bY9iNmDKQTXI";

//provide OAuth2 token obtained in step 1 here
Authentication auth = new OAuthTokenAuthentication(token);

VcoSession session = sessionFactory.newSession(auth);
//Use the session here
```

- 若已啟用多租戶：
  - 適用於來自一般承租人的使用者：

```
URI uri = URI.create("https://orchestrator-server:8283/vco/api");
VcoSessionFactory sessionFactory = new DefaultVcoSessionFactory(uri);
String token =
"eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJqdGkiOiJjNzg4NWNiYS1hZTFmLTRiM2UtYmYyYi04ZmRmNzY3N"
+
"GZiZWEiLCJwcm4iOiJhZG1pbmlzdHJhdG9yQFZTUEhFUkUuTE9DQUwiLCJkb21haW4iOiJ2c3BoZXJlLmxvY2FsIiwidXN
lcl9pZCI6Ij"
+
"MiLCJhdXRoX3RpbWUiOjE1MDIyMDIxMTAsImlzcyI6Imh0dHBzOi8vc2YtMjktMTAtMjkuc29mLW1idS5lbmcudm13YXJl
LmN"
+
"vbS9TQUFTL3QvdnNwaGVyZS5sb2NhbC9hdXRoIiwiYXVkIjoiaHR0cHM6Ly9zZi0yOS0xMC0yOS5zb2YtbWJ1LmVuZy52b
XdhcmUuY29tL"
+
"1NBQVMvdC92c3BoZXJlLmxvY2FsL2F1dGgvb2F1dGh0b2tlbiIsImN0eCI6Ilt7XCJtdGRcIjpcInVybjpvYXNpczpuYW1
lczp"
+
"0YzpTQU1MOjIuMDphYzpjbGFzc2VzOlBhc3N3b3JkUHJvdGVjdGVkVHJhbnNwb3J0XCIsXCJpYXRcIjoxNTAyMjAyMTEwL
FwiaWRcIjoxM"
+
"n1dIiwic2NwIjoidXNlciIsImlkcCI6IjAiLCJlbWwiOiJhZG1pbmlzdHJhdG9yQHNmLTI5LTEwLTI5LnNvZi1tYnUuZW5
nLnZ"
```

```
+
"td2FyZS5jb20iLCJjaWQiOiJjYWZlX2NsaS1yRlJlNEx6bURnIiwiZGlkIjoiIiwid2lkIjoiIiwiZXhwIjoxNTAyMjMwO
TEwLCJpYXQiOj"
+
"E1MDIyMDIxMTAsInN1YiI6IjQ1ZjQwNWUzLTNlNTgtNGJmZC1hNzMwLTQ1MjU4OWIxOGUxNyIsInBybl90eXBlIjoiVVNF
UiJ9."
+ "G9gEQPtmEH5jYab-
IlTK8NFYcwc3JZCEEjsmpUSH6oxLmZKEf-1JbsysBVH4ufqmGah3GMvmy6PUiTTamLRLFkCLwa50OQCQ-qgKutZl21R"
+ "m740qBKLhmBB0NQg19ysMAVJNSxapFzirmWurF_5CKpv4WM7Y8H_bY9iNmDKQTXI";

//provide OAuth2 token obtained in step 1 here
Authentication auth = new OAuthTokenAuthentication(token);

VcoSession session = sessionFactory.newSession(auth);
//The operations will be executed in the scope of the tenant authenticated with the token
above.

//Use the session below
```

- 適用於解決方案使用者：

  解決方案使用者可在自己的承租人範圍及一般承租人範圍中作業。他們可以覆寫自己執行的作業範圍。

```
URI uri = URI.create("https://orchestrator-server:8283/vco/api");
VcoSessionFactory sessionFactory = new DefaultVcoSessionFactory(uri);

exapmle

String token =
"eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJqdGkiOiJjNzg4NWNiYS1hZTFmLTRiM2UtYmYyYi04ZmRmNzY3N"
+
"GZiZWEiLCJwcm4iOiJhZG1pbmlzdHJhdG9yQFZTUEhFUkUuTE9DQUwiLCJkb21haW4iOiJ2c3BoZXJlLmxvY2FsIiwidXN
lcl9pZCI6Ij"
+
"MiLCJhdXRoX3RpbWUiOjE1MDIyMDIxMTAsImlzcyI6Imh0dHBzOi8vc2YtMjktMTAtMjkuc29mLW1idS5lbmcudm13YXJl
LmN"
+
"vbS9TQUFTL3QvdnNwaGVyZS5sb2NhbC9hdXRoIiwiYXVkIjoiaHR0cHM6Ly9zZi0yOS0xMC0yOS5zb2YtbWJ1LmVuZy52b
XdhcmUuY29tL"
+
"1NBQVMvdC92c3BoZXJlLmxvY2FsL2F1dGgvb2F1dGh0b2tlbiIsImN0eCI6Ilt7XCJtdGRcIjpcInVybjpvYXNpczpuYW1
lczp"
+
"0YzpTQU1MOjIuMDphYzpjbGFzc2VzOlBhc3N3b3JkUHJvdGVjdGVkVHJhbnNwb3J0XCIsXCJpYXRcIjoxNTAyMjAyMTEwL
FwiaWRcIjoxM"
+
"n1dIiwic2NwIjoidXNlciIsImlkcCI6IjAiLCJlbWwiOiJhZG1pbmlzdHJhdG9yQHNmLTI5LTEwLTI5LnNvZi1tYnUuZW5
nLnZ"
+
"td2FyZS5jb20iLCJjaWQiOiJjYWZlX2NsaS1yRlJlNEx6bURnIiwiZGlkIjoiIiwid2lkIjoiIiwiZXhwIjoxNTAyMjMwO
TEwLCJpYXQiOj"
+
"E1MDIyMDIxMTAsInN1YiI6IjQ1ZjQwNWUzLTNlNTgtNGJmZC1hNzMwLTQ1MjU4OWIxOGUxNyIsInBybl90eXBlIjoiVVNF
```

```
UiJ9."
+ "G9gEQPtmEH5jYab-
IlTK8NFYcwc3JZCEEjsmpUSH6oxLmZKEf-1JbsysBVH4ufqmGah3GMvmy6PUiTTamLRLFkCLwa50OQCQ-qgKutZl21R"
+ "m740qBKLhmBB0NQg19ysMAVJNSxapFzirmWurF_5CKpv4WM7Y8H_bY9iNmDKQTXI";

//provide OAuth2 token obtained in step 1 here
Authentication auth = new OAuthTokenAuthentication(token);


// By default each tenant works in its tenant scope. However, solution users can overrde the
tenant in which they performa a given operation:
// Here, users of SDK should provide a value that is meanningful to their context.
String overrideWithTenant = "nonSolutionUserTenant";


VcoSession session = sessionFactory.newSession(auth, overrideWithTenant);

//Use session below
```

存取 Orchestrator REST API 的參考說明文件

Orchestrator REST API 參考說明文件包含有關 API RESTful 網路服務、適用 API 的資料模型、用於 API 的有效回應代碼、代碼範例等資訊。

Orchestrator REST API 的參考說明文件連同 Orchestrator 一起安裝。可至 https://*orchestrator_host*:*port*/vco/api/docs/ 取得參考說明文件

Swagger 的官方規格可從 https://swagger.io/specification/ 取得。

使用 Java REST SDK

您可使用 Java SDK 程式庫直接呼叫 Orchestrator REST API 上含物件的 Java 應用程式與工作作業。

Orchestrator REST SDK 的每項 RESTful 網路服務都擁有對應作業方法的換行 Java 類別，並可透過使用服務執行。

Java REST SDK 連同 Orchestrator 一起安裝。Java REST SDK 成品可在下列位置取得。

備註 如果已部署 Orchestrator Appliance，則只能存取成品。

- https://*orchestrator_host*:*port*/vco-repo/com/vmware/o11n/o11n-rest-client/
- https://*orchestrator_host*:*port*/vco-repo/com/vmware/o11n/o11n-rest-client-examples/
- https://*orchestrator_host*:*port*/vco-repo/com/vmware/o11n/o11n-rest-client-services/
- https://*orchestrator_host*:*port*/vco-repo/com/vmware/o11n/o11n-rest-client-stubs/

範例 2-2. 執行工作流程並等待完成

下列範例代碼可執行工作流程並等待完成。

```
//start a new session to Orchestrator by using specified credentials
VcoSession session = DefaultVcoSessionFactory.newLdapSession(new URI("https://orchestrator-server:
8281/vco/api/"), "username", "password");

//create the services
WorkflowService workflowService = new WorkflowService(session);
ExecutionService executionService = new ExecutionService(session);

//find a workflow by ID
Workflow workflow = workflowService.getWorkflow("1231235");

//create an ExecutionContext from the user's input
ExecutionContext context = new ExecutionContextBuilder().addParam("name", "Jerry").addParam("age",
18).build();

//run the workflow
WorkflowExecution execution = executionService.execute(workflow, context);

//wait for the workflow to reach the user interaction state, checking every 500 milliseconds
execution = executionService.awaitState(execution, 500, 10, WorkflowExecutionState.CANCELED,
WorkflowExecutionState.FAILED, WorkflowExecutionState.COMPLETED);

String nameParamValue = new ParameterExtractor().fromTheOutputOf(execution).extractString("name");
System.out.println("workflow was executed with 'name' input set to" + nameParamValue);
```

含工作流程的作業

Orchestrator REST API 提供的網路服務可讓您用來執行各種含工作流程的作業。

尋找工作流程並擷取其定義

若要能執行含工作流程的任何類型作業，您必須在 Orchestrator 詳細目錄中找到工作流程並擷取其定義。定義會列出工作流程輸入與輸出參數，並包含可用工作流程的連結、工作流程簡報與其他物件。

先決條件

請確認您已在 Orchestrator 中匯入範例工作流程套件。套件包含在 Orchestrator 範例應用程式 ZIP 檔案中，您可從 Orchestrator 說明文件頁面下載此檔案。

程序

1 尋找工作流程的詳細目錄項目。

- 若您有完整工作流程名稱或名稱的關鍵字，請套用篩選器並在工作流程服務的 URL 提出 GET 要求。

```
GET https://{orchestrator_host}:{port}/vco/api/workflows?conditions=name={workflowFullName}

GET https://{orchestrator_host}:{port}/vco/api/workflows?conditions=name~{keyWord}
```

- 在屬於工作流程詳細目錄項目輸入點的 URL 提出 GET 要求，透過類別或詳細目錄服務搜尋工作流程：

```
GET https://{orchestrator_host}:{port}/vco/api/catalog/System/Workflow/

GET https://{orchestrator_host}:{port}/vco/api/inventory/System/Workflows/
```

2 在工作流程詳細目錄項目的 URL 提出 GET 要求，擷取工作流程詳細目錄項目：

```
GET https://{orchestrator_host}:{port}/vco/api/catalog/System/Workflow/{workflowID}/
```

3 在定義的 URL 提出 GET 要求，擷取工作流程的定義：

```
GET https://{orchestrator_host}:{port}/vco/api/workflows/{workflowID}/
```

範例 2-3. 搜尋 Send Hello 工作流程

您可尋找 Send Hello 工作流程並擷取其定義：

1 若要尋找 Send Hello 工作流程，請套用篩選器並在工作流程服務的 URL 提出 GET 要求：

```
GET https://localhost:8281/vco/api/workflows?conditions=name~Hello
```

您會收到名稱中包含 Hello 的工作流程清單：

```
<xml version="1.0" encoding="UTF-8" standalone="yes">
<inventory-items xmlns="http://www.vmware.com/vco" total="2">
   <link rel="down"
      href="https://localhost:
8281/vco/api/catalog/System/Workflow/CF80808080808080808080808080808080E6808080013086668236014a0614d1
6e1/">
       <attributes>
          <attribute name="id"
value="CF80808080808080808080808080808080E6808080013086668236014a0614d16e1"/>
          <attribute name="canExecute" value="true" />
          <attribute name="description" value="" />
          <attribute name="name" value="Interactive Hello World" />
          <attribute name="type" value="Workflow"/>
          <attribute name="canEdit" value="true"/>
      </attributes>
```

```
    </link>
    <link rel="down"
      href="https://localhost:
8281/vco/api/catalog/System/Workflow/CF808080808080808080808080808080DA8080800130866682360l4a0614d1
6e1/">
      <attributes>
          <attribute name="id"
value="CF808080808080808080808080808080DA8080800130866682360l4a0614d16e1"/>
          <attribute name="canExecute" value="true" />
          <attribute name="description" value="" />
          <attribute name="name" value="Send Hello" />
          <attribute name="type" value="Workflow"/>
          <attribute name="canEdit" value="true"/>
      </attributes>
   </link>
</inventory-items>
```

2 在擁有 Send Hello 工作流程詳細目錄項目的 URL 提出 GET 要求：

```
GET https://localhost:
8281/vco/api/catalog/System/Workflow/CF808080808080808080808080808080DA8080800130866682360l4a0614d1
6e1/
```

您會在回應本體中收到 Send Hello 工作流程的詳細目錄項目：

```
<xml version="1.0" encoding="UTF-8" standalone="yes">
<inventory-item xmlns="http://www.vmware.com/vco"
   href="https://localhost:
8281/vco/api/catalog/System/Workflow/CF808080808080808080808080808080DA8080800130866682360l4a0614d1
6e1/">
   <relations>
      <link rel="down"
       href="https://localhost:
8281/vco/api/workflows/CF808080808080808080808080808080DA8080800130866682360l4a0614d16e1/" />
      </relations>
   <attributes>
      <attribute name="id"
value="CF808080808080808080808080808080DA8080800130866682360l4a0614d16e1"/>
      <attribute name="canExecute" value="true" />
      <attribute name="description" value="" />
      <attribute name="name" value="Send Hello" />
      <attribute name="type" value="Workflow"/>
      <attribute name="canEdit" value="true"/>
   </attributes>
</inventory-item>
```

3 若要擷取工作流程的定義，請在其 URL 提出 GET 要求：

```
GET https://localhost:
8281/vco/api/workflows/CF808080808080808080808080808080DA8080800130866682360l4a0614d16e1/
```

您會在回應本體中收到 Send Hello 工作流程的定義：

```
<xml version="1.0" encoding="UTF-8" standalone="yes">
<workflow xmlns="http://www.vmware.com/vco" customized-icon="false"
   href="https://localhost:
8281/vco/api/workflows/CF80808080808080808080808080808080DA8080800130866682360l4a0614d16e1/">
   <relations>
      <link rel="up"
         href="https://localhost:8281/vco/api/inventory/System/Workflows/Samples/HelloWorld/" />
        <link rel="add"
            href="https://localhost:
8281/vco/api/workflows/CF80808080808080808080808080808080DA8080800130866682360l4a0614d16e1/executions/" />
        <link rel="down"
            href="https://localhost:
8281/vco/api/workflows/CF80808080808080808080808080808080DA8080800130866682360l4a0614d16e1/executions/" />
        <link rel="down"
            href="https://localhost:
8281/vco/api/workflows/CF80808080808080808080808080808080DA8080800130866682360l4a0614d16e1/presentation/" />
        <link rel="down"
            href="https://localhost:
8281/vco/api/workflows/CF80808080808080808080808080808080DA8080800130866682360l4a0614d16e1/tasks/" />
        <link rel="down"
            href="https://localhost:
8281/vco/api/workflows/CF80808080808080808080808080808080DA8080800130866682360l4a0614d16e1/icon/" />
   </relations>
   <input-parameters>
      <parameter name="name" type="string" />
   </input-parameters>
   <output-parameters>
      <parameter name="message" type="string" />
   </output-parameters>
   <name>Send Hello</name>
      <description></description>
</workflow>
```

執行工作流程

您可為特定工作流程建立新的執行物件，透過 Orchestrator REST API 執行工作流程。

先決條件

請確認您已在 Orchestrator 中匯入範例工作流程套件。套件包含在 Orchestrator 範例應用程式 ZIP 檔案中，您可從 Orchestrator 說明文件頁面下載此檔案。

程序

1 在定義的 URL 提出 GET 要求，擷取您要執行工作流程的定義：

```
GET http://{orchestrator_host}:{port}/vco/api/workflows/{workflowID}/
```

您會在要求的回應本體中收到工作流程的定義。在工作流程定義中，您可檢視工作流程的輸入參數、工作流程描述及其他資訊。

2 在擁有工作流程執行物件的 URL 提出 POST 要求：

```
POST https://{orchestrator_host}:{port}/vco/api/workflows/{workflowID}/executions/
```

3 在要求本體的 execution-context 元素中，提供工作流程輸入參數的數值。

若您在要求本體中提供空的 execution-context，則該工作流程會執行輸入參數的預設數值。

若 POST 要求成功，您會在 Location 標頭中收到含空的回應本體狀態碼 202 及新建立執行物件的連接。

範例 2-4. 執行 Send Hello 工作流程

您可擷取 Send Hello 工作流程的定義並執行。

1 在擁有 Send Hello 工作流程定義的 URL 提出 GET 要求：

```
GET https://localhost:
8281/vco/api/workflows/CF8080808080808080808080808080808080DA8080800130866682360l4a0614d16e1/
```

您會在要求的回應本體中，收到工作流程定義：

```
<xml version="1.0" encoding="UTF-8" standalone="yes">
<workflow xmlns="http://www.vmware.com/vco" customized-icon="false"
   href="https://localhost:
8281/vco/api/workflows/CF8080808080808080808080808080808080DA8080800130866682360l4a0614d16e1/">
   <relations>
      <link rel="up"
        href="https://localhost:8281/vco/api/inventory/System/Workflows/Samples/HelloWorld/" />
      <link rel="add"
        href="https://localhost:
8281/vco/api/workflows/CF8080808080808080808080808080808080DA8080800130866682360l4a0614d16e1/execution
s/" />
      <link rel="down"
        href="https://localhost:
8281/vco/api/workflows/CF8080808080808080808080808080808080DA8080800130866682360l4a0614d16e1/execution
s/" />
      <link rel="down"
        href="https://localhost:
8281/vco/api/workflows/CF8080808080808080808080808080808080DA8080800130866682360l4a0614d16e1/presentati
on/" />
      <link rel="down"
           href="https://localhost:
8281/vco/api/workflows/CF8080808080808080808080808080808080DA8080800130866682360l4a0614d16e1/tasks/" />
```

```
      <link rel="down"
        href="https://localhost:
8281/vco/api/workflows/CF80808080808080808080808080808080DA8080800130866682360l4a0614d16e1/icon/" />
   </relations>
   <input-parameters>
      <parameter name="name" type="string" />
   </input-parameters>
   <output-parameters>
      <parameter name="message" type="string" />
   </output-parameters>
   <name>Send Hello</name>
      <description></description>
</workflow>
```

2 在擁有工作流程執行物件的 URL 提出 POST 要求：

```
POST https://localhost:
8281/vco/api/workflows/CF80808080808080808080808080808080DA8080800130866682360l4a0614d16e1/execution
s/
```

在要求本體的 execution-context 元素中，傳遞輸入參數的數值：

```
<execution-context  xmlns="http://www.vmware.com/vco">
   <parameters>
      <parameter name="name" type="string">
         <string>John Smith</string>
       </parameter>
    </parameters>
</execution-context>
```

針對工作流程呈現驗證其輸入參數後執行工作流程

工作流程簡報可定義數值限制，讓您可傳遞至工作流程的輸入參數，如預先定義的數值清單或特定範圍的數值。為確保工作流程順利執行，您必須按照工作流程簡報定義，驗證傳遞至工作流程輸入參數的數值。

當您在自訂應用程式中整合工作流程時，您可能需要建立精靈，以便在執行工作流程時輸入用於工作流程輸入參數的數值。您可以透過使用工作流程呈現服務，具現化工作流程呈現，並針對與不同精靈畫面對應的部分輸入參數傳遞數值。您可針對在工作流程簡報中定義的限制，驗證傳遞至輸入參數的數值。

先決條件

請確認您已在 Orchestrator 中匯入範例工作流程套件。套件包含在 Orchestrator 範例應用程式 ZIP 檔案中，您可從 Orchestrator 說明文件頁面下載此檔案。

程序

1 在包含工作流程定義的 URL 提出 GET 要求，擷取您要執行工作流程的定義：

```
GET https://{orchestrator_host}:{port}/vco/api/workflows/{workflowID}/
```

您會在要求的回應本體中收到工作流程的定義。在工作流程定義中，您可檢視工作流程的輸入參數、工作流程描述及其他資訊。

2 在工作流程簡報的 URL 提出 GET 要求，擷取工作流程簡報定義：

```
GET https://{orchestrator_host}:{port}/vco/api/workflows/{workflowID}/presentation/
```

3 在要求的回應本體中，檢查工作流程簡報的定義是否有任何可傳遞至輸入參數的數值限制。

例如：輸入參數可擁有預先定義的數值清單供人選擇。

4 在簡報執行個體的 URL 提出 POST 要求，舉例說明工作流程簡報：

```
POST https://{orchestrator_host}:{port}/vco/api/workflows/{workflowID}/presentation/instances/
```

5 在要求本體中提供 `execution-context` 元素，舉例說明簡報。

您可以針對部分輸入參數只傳遞空的 `execution-context` 或傳遞含有數值的 `execution-context`。

6 若要傳遞數值至部分輸入參數，請視需要在擁有簡報執行個體的 URL 提出多數 POST 或 PUT 要求：

```
PUT https://{orchestrator_host}:
{port}/vco/api/workflows/{workflowID}/presentation/instances/{executionID}/
```

7 檢閱您提出的 POST 或 PUT 要求回應本體。

若您傳遞至輸入參數的數值有效，則會在 `execution` 標籤中發現 `valid="true"` 屬性。若簡報有效，您可取得列於回應 `out-parameters` 元素中的數值，並在您執行工作流程時當成數值傳遞至輸入參數。

8 若輸入參數的數值有效，請在擁有工作流程執行的 URL 提出 POST 要求以執行工作流程：

```
POST https://{orchestrator_host}:{port}/vco/api/workflows/{workflowID}/executions/
```

9 提供有效的數值至 `execution-context` 元素中的工作流程輸入參數。

範例 2-5. 驗證工作流程的輸入參數，執行 Send Hello 工作流程

您可根據工作流程簡報的定義驗證工作流程的輸入參數，執行 Send Hello 工作流程。

1 在擁有 Send Hello 工作流程定義的 URL 提出 GET 要求：

```
GET https://localhost:
8281/vco/api/workflows/CF8080808080808080808080808080DA8080800130866682360l4a0614d16e1/
```

您會在要求的回應本體中，收到工作流程定義：

```
<xml version="1.0" encoding="UTF-8" standalone="yes">
<workflow xmlns="http://www.vmware.com/vco" customized-icon="false"
   href="https://localhost:
8281/vco/api/workflows/CF80808080808080808080808080808080DA8080800130866682360l4a0614d16e1/">
   <relations>
      <link rel="up"
          href="https://localhost:8281/vco/api/inventory/System/Workflows/Samples/HelloWorld/" />
      <link rel="add"
          href="https://localhost:
8281/vco/api/workflows/CF80808080808080808080808080808080DA8080800130866682360l4a0614d16e1/execution
s/" />
      <link rel="down"
          href="https://localhost:
8281/vco/api/workflows/CF80808080808080808080808080808080DA8080800130866682360l4a0614d16e1/execution
s/" />
      <link rel="down"
          href="https://localhost:
8281/vco/api/workflows/CF80808080808080808080808080808080DA8080800130866682360l4a0614d16e1/presentati
on/" />
      <link rel="down"
             href="https://localhost:
8281/vco/api/workflows/CF80808080808080808080808080808080DA8080800130866682360l4a0614d16e1/tasks/" />
      <link rel="down"
          href="https://localhost:
8281/vco/api/workflows/CF80808080808080808080808080808080DA8080800130866682360l4a0614d16e1/icon/" />
   </relations>
   <input-parameters>
      <parameter name="name" type="string" />
   </input-parameters>
   <output-parameters>
      <parameter name="message" type="string" />
   </output-parameters>
   <name>Send Hello</name>
      <description></description>
</workflow>
```

2 在擁有工作流程簡報定義的 URL 提出 GET 要求：

```
GET https://localhost:
8281/vco/api/workflows/CF80808080808080808080808080808080DA8080800130866682360l4a0614d16e1/presentati
on/
```

3 在擁有工作流程簡報執行個體的 URL 提出 POST 要求：

```
POST https://localhost:
8281/vco/api/workflows/CF80808080808080808080808080808080DA8080800130866682360l4a0614d16e1/presentati
on/instances/
```

提供空的 `execution-context`，以便在不提供任何輸入參數數值的情況下舉例說明簡報。

```
<execution-context xmlns="http://www.vmware.com/vco"/>
```

回應本體包含附加至各欄位的錯誤訊息，說明用於輸入參數的該數值無效。

```
.......
<fields>
  <field type="string" hidden="false" id="name">
    <display-name>name</display-name>
    <description>name</description>
    <messages>
       <message severity="ERROR" code="VCO-CNS0002">
          <Summary>
                The minimum number of characters allowed for this field is 3.0
          </Summary>
       </message>
    </messages>
    <constraints>
       <number-range max="15.0" min="3.0" />
    </constraints>
.......
```

4 在擁有特定簡報執行個體的 URL 提出 POST 要求：

```
POST https://localhost:
8281/vco/api/workflows/CF8080808080808080808080808080DA8080800130866682360l4a0614d16e1/presentati
on/instances/888080808080808080808080808080803F8080800132145338690643f66a027ec/
```

在要求本體中，提供用於輸入參數的數值：

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
 <execution-context xmlns="http://www.vmware.com/vco">
  <parameters>
    <parameter name="name" type="string">
      <string>John Smith</string>
    </parameter>
  </parameters>
</execution-context>
```

在要求回應本體中，您可檢查輸入參數的數值是否有效：

```
<execution started-by="vcoadmin" .... valid="true".....>
```

5 若簡報有效，請在擁有工作流程執行的 URL 提出 POST 要求以執行工作流程：

```
POST https://localhost:
8281/vco/api/workflows/CF8080808080808080808080808080DA8080800130866682360l4a0614d16e1/executions/
```

在要求本體中，傳遞數值至工作流程的輸入參數。使用當成工作流程簡報輸出參數傳回的相同數值，或直接使用上次您在工作流程簡報提出 POST 要求的要求本體。

在工作流程執行時與其互動

Orchestrator REST API 允許您在執行工作流程時執行含工作流程的各種作業。您可取得執行中工作流程的狀態、回覆等待中的使用者互動，以及取消工作流程執行。

取得工作流程執行物件並檢查工作流程狀態

您可取得有關工作流程執行的資訊，如開始與結束日期、執行狀態和輸入參數數值。您也可取得工作流程執行產生的記錄。

先決條件

請確認您已在 Orchestrator 中匯入範例工作流程套件。套件包含在 Orchestrator 範例應用程式 ZIP 檔案中，您可從 Orchestrator 說明文件頁面下載此檔案。

程序

1 在工作流程的 URL 提出 GET 要求，擷取您要檢查的工作流程狀態定義：

```
GET https://{orchestrator_host}:{port}/vco/api/workflows/{workflowID}/
```

您會在要求的回應本體中收到工作流程的定義。工作流程定義包含工作流程執行個體的連結。

2 在工作流程執行個體的 URL 提出 GET 要求，擷取可用的工作流程執行個體：

```
GET https://{orchestrator_host}:{port}/vco/api/workflows/{workflowID}/executions/
```

要求的回應本體列出可用的工作流程執行個體，您可在此檢視各工作流程執行的開始與結束日期，以及其狀態和啟動器。

3 (選擇性) 若要瞭解更多有關特定工作流程執行的詳細資訊，請在該執行的 URL 提出 GET 要求：

```
GET https://{orchestrator_host}:{port}/vco/api/workflows/{workflowID}/executions/{executionID}/
```

在要求的回應本體中，您會收到特定工作流程的 XML 表示法。您可檢查為此工作流程傳遞的輸入參數數值，啟動執行的使用者、開始與結束日期以及執行狀態。

4 (選擇性) 若要擷取工作流程執行產生的記錄，請在擁有記錄的 URL 提出 GET 要求：

```
GET https://{orchestrator_host}:
{port}/vco/api/workflows/{workflowID}/executions/{executionID}/logs/
```

5 (選擇性) 若要擷取其他有關執行狀態的資訊，請在擁有工作流程狀態的 URL 提出 GET 要求：

```
GET https://{orchestrator_host}:
{port}/vco/api/workflows/{workflowID}/executions/{executionID}/state/
```

範例 2-6. 取得執行 Send Hello 工作流程及檢查特定執行的狀態

若您已執行 Send Hello 工作流程，可取得執行物件及檢查詳細資訊。

1 在擁有定義的 URL 提出 GET 要求，取得 Send Hello 工作流程的定義：

```
GET https://localhost:
8281/vco/api/workflows/CF80808080808080808080808080808080DA8080800130866682360l4a0614d16e1/
```

2 在擁有工作流程執行物件的 URL 提出 GET 要求，取得可執行的工作流程：

```
GET https://localhost:
8281/vco/api/workflows/CF80808080808080808080808080808080DA8080800130866682360l4a0614d16e1/execution
s/
```

3 在要求的回應本體中選擇工作流程並提出 GET 要求以擷取：

```
GET https://localhost:
8281/vco/api/workflows/CF80808080808080808080808080808080DA8080800130866682360l4a0614d16e1/execution
s/88808080808080808080808080808080803A8080800132145338690643f66a027ec/
```

回應本體包含有指定識別碼的工作流程 XML 表示法，您可在此檢查有關該執行的詳細資訊：

```
.......
<input-parameters>
   <parameter name="name" type="string">
      <string>John Smith</string>
   </parameter>
</input-parameters>
<output-parameters>
   <parameter name="message" type="string">
      <string>Hello, John Smith!</string>
   </parameter>
</output-parameters>
<start-date>2012-01-31T14:28:40.223+03:00</start-date>
<end-date>2012-01-31T14:28:40.410+03:00</end-date>
<started-by>vcoadmin</started-by>
<name>Send Hello</name>
......
```

回應等待中的使用者互動

您可利用 Orchestrator REST API 回應等待中的工作流程使用者互動。

先決條件

請確認您已在 Orchestrator 中匯入範例工作流程套件。套件包含在 Orchestrator 範例應用程式 ZIP 檔案中，您可從 Orchestrator 說明文件頁面下載此檔案。

程序

1 在擁有可用使用者互動物件的 URL 提出 GET 要求，或僅篩選等待中的使用者互動，以擷取所有使用者互動物件清單。

URL	說明
https://*orchestrator_host*:*port*/vco/api/catalog/System/UserInteraction	在 Orchestrator 擁有可用的使用者互動物件
https://*orchestrator_host*:*port*/vco/api/catalog/System/UserInteraction?status=0	僅篩選等待中的使用者互動物件。

您會收到可用使用者互動物件清單。等待中的使用者互動含有名稱為 `state` 且數值為 `waiting` 的屬性。

2 在您要回應並擁有等待中使用者互動詳細目錄項目的 URL 提出 GET 要求：

```
GET https://{orchestrator_host}:{port}/vco/api/catalog/System/UserInteraction/{userInteractionID}/
```

詳細目錄項目包含使用者互動執行個體連結。使用者互動執行個體與特定工作流程相關聯。

3 在用於特定工作流程執行的使用者互動執行個體 URL 提出 POST 要求：

```
POST https://{orchestrator_host}:
{port}/vco/api/workflows/{workflowID}/executions/{executionID}/interaction/
```

4 在要求本體的 `execution-context` 元素中提供使用者互動輸入參數的數值。

REST API 在您成功回應使用者互動時傳回 204 狀態。

範例 2-7. 回應互動式 Hello World 工作流程的使用者互動

您可執行互動式 Hello World 範本工作流程並回應其使用者互動。

1 在用於目錄服務使用者互動物件的端點提出 GET 要求，搜尋工作流程的等待中使用者互動：

```
GET https://localhost:8281/vco/api/catalog/System/UserInteraction?status=0
```

2 找出用於互動式 Hello World 工作流程的使用者互動詳細目錄位置，並在其 URL 上提出 GET 要求：

```
GET https://localhost:
8281/vco/api/catalog/System/UserInteraction/8880808080808080808080808080805A8080800132145338690643f66a027ec/
```

3 在用於目前工作流程執行的使用者互動物件 URL 提出 POST 要求：

```
POST https://localhost:
8281/vco/api/workflows/CF8080808080808080808080808080E68080800130866682360l4a0614d16e1/executions/88808080808080808080808080808080578080800132145338690643f66a027ec/interaction/
```

在要求本體中，提供輸入參數的數值：

```
<execution-context xmlns="http://www.vmware.com/vco">
   <parameters>
     <parameter name="name" type="string">
       <string>John Smith</string>
     </parameter>
   </parameters>
</execution-context>
```

回應驗證輸入參數後的使用者互動

使用者互動的呈現可定義您可傳遞至工作流程輸入參數的數值限制。當您回應使用者互動時，可對照在使用者互動呈現中所定義的限制，驗證傳遞至輸入參數的數值。

先決條件

請確認您已在 Orchestrator 中匯入範例工作流程套件。套件包含在 Orchestrator 範例應用程式 ZIP 檔案中，您可從 Orchestrator 說明文件頁面下載此檔案。

程序

1 在擁有可用使用者互動物件的 URL 提出 GET 要求，或僅篩選等待中的使用者互動，以擷取所有使用者互動物件清單。

URL	說明
https://*orchestrator_host*:*port*/vco/api/catalog/System/UserInteraction	在 Orchestrator 擁有可用的使用者互動物件。
https://*orchestrator_host*:*port*/vco/api/catalog/System/UserInteraction?status=0	僅篩選等待中的使用者互動物件。

您會收到可用使用者互動物件清單。等待中的使用者互動含有名稱為 `state` 且數值為 `waiting` 的屬性。

2 在您要回應並擁有等待中使用者互動詳細目錄項目的 URL 提出 GET 要求：

```
GET https://{orchestrator_host}:{port}/vco/api/catalog/System/UserInteraction/{userInteractionID}/
```

回應主體包含使用者互動執行個體的連結。使用者互動執行個體與特定的工作流程執行相關聯。

3 在使用者互動執行個體的 URL 提出 GET 要求：

```
GET https://{orchestrator_host}:
{port}/vco/api/workflows/{workflowID}/executions/{executionID}/interaction/
```

在回應主體中，您會發現使用者互動呈現的下行連結。

4 在使用者互動簡報的 URL 提出 GET 要求：

```
GET https://{orchestrator_host}:
{port}/vco/api/workflows/{workflowID}/executions/{executionID}/interaction/presentation/
```

您在回應本體收到使用者互動簡報的定義。

5 在簡報定義中，檢查您可傳遞至輸入參數的數值限制。

6 在有簡報執行個體的 URL 提出 POST 要求以執行使用者互動簡報：

```
POST https://{orchestrator_host}:
{port}/vco/api/workflows/{workflowID}/executions/{executionID}/interaction/presentation/instances/
```

7 在要求本體中，提供 `execution-context` 元素中用於輸入參數的數值。

在回應本體中，您會收到使用者互動簡報的執行個體。若您傳遞至輸入參數的數值有效，則會在 `execution` 元素中發現 `valid="true"` 屬性。在 `output-parameters` 元素中，您會發現可用來回應使用者互動輸入參數的有效數值。

8 在有使用者互動執行個體的 URL 提出 POST 要求後，即可回應使用者互動。

```
POST https://{orchestrator_host}:
{port}/vco/api/workflows/{workflowID}/executions/{executionID}/interaction/
```

9 在要求本體中，傳遞含有輸入參數數值的 `execution-context` 內容。

您可利用相同要求的本體作為在用於使用者互動簡報 URL 提出的 POST 要求。

若上次要求成功，您會收到狀態碼 204 及空的回應本體。

範例 2-8. 透過驗證輸入參數回應互動式 Hello World 工作流程的使用者互動

您可透過針對使用者互動簡報定義限制驗證輸入參數數值，回應互動式 Hello World 工作流程的使用者互動。

1 在用於目錄服務使用者互動物件的端點提出 GET 要求，以搜尋工作流程的等待中使用者互動：

```
GET https://localhost:8281/vco/api/catalog/System/UserInteraction?status=0
```

2 找出用於互動式 Hello World 工作流程的使用者互動詳細目錄位置，並在其 URL 上提出 GET 要求：

```
GET https://localhost:
8281/vco/api/catalog/System/UserInteraction/88808080808080808080808080808080805A8080800132145338690643f66a027ec/
```

3 在使用者互動執行個體的 URL 提出 GET 要求：

```
GET https://localhost:
8281/vco/api/catalog/System/UserInteraction/88808080808080808080808080808080805A8080800132145338690643f66a027ec/interaction/
```

4 在使用者互動呈現的 URL 提出 GET 要求：

```
GET https://localhost:
8281/vco/api/catalog/System/UserInteraction/88808080808080808080808080808080805A8080800132145338690643f66a027ec/interaction/presentation/
```

簡報定義輸入參數為強制性，並包含可傳遞字串的長度限制。

5 在擁有使用者互動簡報的執行個體 URL 提出 POST 要求：

```
POST https://localhost:
8281/vco/api/catalog/System/UserInteraction/88808080808080808080808080808080805A8080800132145338690643f66a027ec/interaction/presentation/instances/
```

在要求本體中，提供輸入參數的數值：

```
<execution-context xmlns="http://www.vmware.com/vco">
   <parameters>
     <parameter name="name" type="string">
       <string>John Smith</string>
     </parameter>
   </parameters>
</execution-context>
```

回應本體的 execution 元素含有 valid="true" 屬性，表示針對使用者互動簡報中的限制，該輸入參數數值為有效。有效數值列在 output-parameters 元素中：

```
.............
<output-parameters>
  <parameter name="name" type="string">
     <string>John Smith</string>
   </parameter>
</output-parameters>
.............
```

6 透過傳遞相同要求本體作為步驟 5 中的 POST 要求，以在使用者互動執行個體的 URL 提出 POST 要求。

```
POST https://localhost:
8281/vco/api/catalog/System/UserInteraction/88808080808080808080808080808080805A8080800132145338690643f66a027ec/interaction/
```

取消工作流程執行

您可使用 Orchestrator REST API 取消執行工作流程。

程序

1 在工作流程定義的 URL 提出 GET 要求，以擷取工作流程的定義：

```
GET https://{orchestrator_host}:{port}/vco/api/workflows/{workflowID}/
```

工作流程定義包含工作流程可用執行物件的連結。

2 在擁有工作流程可用執行物件的 URL 提出 GET 要求，以取得可執行的工作流程：

```
GET https://{orchestrator_host}:{port}/vco/api/workflows/{workflowID}/executions/
```

3 在可用工作流程的執行清單中，選擇一項您要取消的工作流程，然後在 URL 提出 DELETE 要求：

```
DELETE https://{orchestrator_host}:
{port}/vco/api/workflows/{workflowID}/executions/{executionID}/state
```

擷取工作流程的互動

您可使用 Orchestrator REST API 擷取工作流程所有使用者互動的清單。

程序

1 在工作流程定義的 URL 提出 GET 要求，以擷取工作流程的定義：

```
GET https://{orchestrator_host}:{port}/vco/api/workflows/{workflowID}/
```

2 在工作流程互動 URL 提出 GET 要求，取得工作流程互動清單：

```
GET https://{orchestrator_host}:{port}/vco/api/workflows/{workflowID}/interactions/
```

若 GET 要求成功，您會收到狀態碼 200 及該工作流程中所有使用者互動的清單。

存取工作流程配置

您可使用 Orchestrator REST API 存取工作流程的配置映像。

程序

1 在工作流程定義的 URL 提出 GET 要求，以擷取工作流程的定義：

```
GET https://{orchestrator_host}:{port}/vco/api/workflows/{workflowID}/
```

2 提出 GET 要求至工作流程配置 URL ，取得工作流程的配置映像：

```
GET https://{orchestrator_host}:{port}/vco/api/workflows/{workflowID}/schema/
```

若 GET 要求成功，您會收到狀態碼 200 及代表工作流程配置的映像二進位資料。例如：回應內容型別設為正確的媒體型別 `Content-Type:image/png`。

使用工作

您可利用 Orchestrator REST API 的工作服務，執行任何與管理 Orchestrator 工作相關的作業。您可建立用來排程工作流程的工作、修改已存在工作的屬性、刪除工作等。

Orchestrator 支援的排程工作上限為 50。

建立工作

您可使用 Orchestrator REST API 建立排程工作流程的工作。

先決條件

請確認您已在 Orchestrator 中匯入範例工作流程套件。套件包含在 Orchestrator 範例應用程式 ZIP 檔案中，您可從 Orchestrator 說明文件頁面下載此檔案。

程序

1 在工作流程的 URL 提出 GET 要求，擷取您要建立工作的工作流程定義：

```
GET https://{orchestrator_host}:{port}/vco/api/workflows/{workflowID}/
```

您可在工作流程定義中檢視工作流程名稱與識別碼，以及其輸入參數。

2 若要為工作流程建立新工作，請在工作服務的 URL 提出 POST 要求：

```
POST https://{orchestrator_host}:{port}/vco/api/tasks/
```

3 在要求本體中，提供 `task` 元素中用於新工作的參數。

若要求成功，API 會回應狀態碼 202 及空的回應本體。

範例 2-9. 建立用於 Send Hello 工作流程的工作

您可建立工作，排程 Send Hello 工作流程在指定日期的每小時開始時執行 15 分鐘。

1 在 Send Hello 工作流程的 URL 提出 GET 要求以擷取其定義：

```
GET https://localhost:
8281/vco/api/workflows/CF8080808080808080808080808080DA8080800130866682360l4a0614d16e1/
```

2 在要求本體中提供新工作參數，並在工作服務的 URL 提出 POST 要求：

```
POST https://localhost:8281/vco/api/tasks/
```

```
<task xmlns="http://www.vmware.com/vco">
    <name>Send Hello Task</name>
    <recurrence-cycle>every-hours</recurrence-cycle>
    <recurrence-start-date>2012-01-31T11:00:00+00:00</recurrence-start-date>
    <recurrence-end-date>2012-02-05T11:00:00+00:00</recurrence-end-date>
    <recurrence-pattern>15:15</recurrence-pattern>
    <input-parameters>
        <parameter name="name" type="string">
            <string>John Smith</string>
        </parameter>
    </input-parameters>
    <workflow href="https://localhost:
8281/vco/api/workflows/CF8080808080808080808080808080DA8080800130866682360l4a0614d16e1/">
        <name>Send Hello</name>
    </workflow>
    <start-mode>normal</start-mode>
</task>
```

修改工作

您可使用 Orchestrator REST API 變更現有工作的內容。

您只能新增排程內容至工作或變更已存在內容的數值。若您想要更換工作的排程內容，您必須刪除工作並建立新工作。

先決條件

請確認您已在 Orchestrator 中匯入範例工作流程套件。套件包含在 Orchestrator 範例應用程式 ZIP 檔案中，您可從 Orchestrator 說明文件頁面下載此檔案。

程序

1 在您要修改工作的 URL 提出 GET 要求：

```
GET https://{orchestrator_host}:{port}/vco/api/tasks/{task ID}/
```

2 在要求回應本體中檢查工作內容。

3 若要修改工作，請透過在要求本體的 task-data 元素提供工作的新內容並在工作的 URL 提出 POST 要求。

若 POST 要求成功，API 會在回應本體中傳回狀態碼 200 及更新工作。

範例 2-10. 更新 Send Hello 範例工作

您可更新工作的開始與結束日期。您可修改在 建立工作 中導入的範例工作。您必須透過在要求本體中提供新的開始與結束日期並在工作的 URL 提出 POST 要求：

```
<?xml version="1.0" encoding="utf-8" standalone="yes"?>
<task-data xmlns="http://www.vmware.com/vco">
   <recurrence-start-date>2012-02-01T14:00:00+02:00</recurrence-start-date>
   <recurrence-end-date>2012-02-05T14:00:00+02:00</recurrence-end-date>
</task-data>
```

檢查工作狀態

您可檢查目前可用工作的狀態，或檢查特定工作的所有執行個體狀態。

先決條件

請確認您已在 Orchestrator 中匯入範例工作流程套件。套件包含在 Orchestrator 範例應用程式 ZIP 檔案中，您可從 Orchestrator 說明文件頁面下載此檔案。

程序

- 若要檢查所有目前可用工作的狀態，請在工作服務的 URL 提出 GET 要求：

  ```
  GET https://{orchestrator_host}:{port}/vco/api/tasks/
  ```

  回應本體包含 Orchestrator 中目前可用工作的定義。各工作的狀態皆可在名稱為 `state` 的 `attribute` 元素中使用。此元素的數值可分別為 `finished`、`pending`、`running` 等。

- 若要檢查所有特定工作的執行狀態，請在有執行工作的 URL 提出 GET 要求：

  ```
  GET https://{orchestrator_host}:{port}/vco/api/tasks/{taskID}/executions/
  ```

  您收到回應本體中可執行工作的清單。在工作執行物件的 `state` 元素中，可使用各執行狀態。

在 Orchestrator 詳細目錄中尋找物件

您可使用目錄或詳細目錄服務尋找 Orchestrator 詳細目錄中的任何物件。您只能在提出 HTTP 要求的 URL 結尾套用篩選器參數，以存取特定物件的子集。

您可使用目錄服務尋找 Orchestrator 詳細目錄中屬於特定類型的物件，或按照其類型和識別碼擷取特定物件。例如：您可擷取屬於型別 `workflow` 或 `action` 的所有物件，或可擷取指定的工作流程或動作。

詳細目錄服務允許您按照父系-子系關係瀏覽 Orchestrator 詳細目錄。您可使用詳細目錄服務存取 Orchestrator 詳細目錄中指定位置的可用物件。例如：您可以在 Orchestrator 詳細目錄中瀏覽至資料中心管理的位置，即 `Library/vCenter/Datacenter` ，擷取所有資料中心管理的工作流程。

每項來自 Orchestrator REST API 的服務皆支援篩選器參數，因此您可在提出 HTTP 要求時於 URL 結尾新增。您可以使用篩選器參數縮減在特定 URL 上要求的回應主體中收到的結果範圍。

按型別及識別碼尋找物件

您可使用 REST API 的目錄服務，按類型及識別碼尋找 Orchestrator 中的物件。

先決條件

請確認您已在 Orchestrator 中匯入範例工作流程套件。套件包含在 Orchestrator 範例應用程式 ZIP 檔案中，您可從 Orchestrator 說明文件頁面下載此檔案。

程序

1 在類別目錄服務的 URL 提出 GET 要求：

```
GET https://{orchestrator_host}:{port}/vco/api/catalog/
```

要求的回應本體包含外掛程式類別目錄輸入點下方連結，可公開 Orchestrator 詳細目錄以及 Orchestrator 中的系統物件：

- ```
  https://{orchestrator_host}:{port}/vco/api/catalog/{plug-in namespace}/
  ```
- ```
  https://{orchestrator_host}:{port}/vco/api/catalog/System/
  ```

2 若要存取外掛程式公開物件或 Orchestrator 中的系統物件，請在外掛程式目錄輸入點的 URL 或在 Orchestrator 中存在系統物件的 URL 提出 GET 要求。

要求的回應本體包含公開的物件型別連結。

3 在您要存取的物件類型的 URL 上提出 GET 要求：

```
GET https://{orchestrator_host}:{port}/vco/api/catalog/{namespace}/{objectType}/
```

4 在您要尋找指定物件的 URL 提出 GET 要求：

```
GET https://{orchestrator_host}:{port}/vco/api/catalog/{namespace}/{objectType}/{objectID}/
```

範例 2-11. 尋找 Send Hello 工作流程

您可使用類別目錄服務尋找範本 Send Hello 工作流程。

1 在類別目錄服務的 URL 提出 GET 要求：

```
GET https://localhost:8281/vco/api/catalog/
```

2 在找出 Orchestrator 中所有系統物件的 URL 提出 GET 要求：

```
GET https://localhost:8281/vco/api/catalog/System/
```

3 在有所有工作流程的 URL 提出 GET 要求：

```
GET https://localhost:8281/vco/api/catalog/Workflow/
```

4 在 Send Hello 工作流程的 URL 提出 GET 要求：

```
GET https://localhost:
8281/vco/api/catalog/Workflow/CF80808080808080808080808080808080DA8080800130866682360l4a0614d16e1/
```

按關係尋找物件

您可使用 Orchestrator REST 的詳細目錄服務將 Orchestrator 及外掛程式詳細目錄當成階層瀏覽。

先決條件

請確認您已在 Orchestrator 中匯入範例工作流程套件。套件包含在 Orchestrator 範例應用程式 ZIP 檔案中，您可從 Orchestrator 說明文件頁面下載此檔案。

程序

1 在詳細目錄服務的 URL 提出 GET 要求：

```
GET https://{orchestrator_host}:{port}/vco/api/inventory/
```

回應本體包含已安裝外掛程式登錄的詳細目錄下方連結，以及 System 下 Orchestrator 系統物件的下方連結。

2 在您要存取詳細目錄的下方連結提出 GET 要求：

3 在詳細目錄項目的上方及下方連結提出 GET 要求，直到找到要尋找的物件為止。

範例 2-12. 尋找 Send Hello 工作流程

您可瀏覽 Orchestrator 詳細目錄，尋找 Send Hello 工作流程。

1 在詳細目錄服務的 URL 提出 GET 要求：

```
GET https://localhost:8281/vco/api/inventory/
```

2 在 Orchestrator 中有系統物件的 URL 提出 GET 要求：

```
GET https://localhost:8281/vco/api/inventory/System/
```

3 在 Orchestrator 中有所有工作流程的 URL 提出 GET 要求：

```
GET https://localhost:8281/vco/api/inventory/System/Workflows/
```

4 在範本工作流程類別的 URL 提出 GET 要求：

```
GET https://localhost:8281/vco/api/inventory/System/Workflows/Samples/
```

5 使用 Hello World 工作流程類別的下方連結找出 Send Hello 工作流程。

套用篩選器

Orchestrator REST API 的服務支援其他 URL 參數，允許您縮減 API 傳回 HTTP 要求的物件。

您可透過 REST API 存取各資源 URL 支援的不同查詢參數。若要瞭解適用 URL 的查詢參數資訊，請參閱 *vRealize Orchestrator REST API* 參考說明文件。

程序

- 若要縮減特定 URL 要求的結果範圍，請在 URL 結尾套用篩選器：

  URL? *filter_1*& *filter_2*&*filter_3*&....&*filter_N*.各篩選器皆含適用於相關 URL 的查詢參數。如需有關各 URL 有效查詢參數的資訊，請參閱 Orchestrator REST API 參考說明文件。

範例 2-13. 篩選器工作流程

若您尋找名稱中包含指定文字的工作流程，例如資料存放區，您可在類別服務的要求中套用下列篩選器：

```
GET https://localhost:8281/vco/api/catalog/System/Workflow?conditions=name~datastore
```

若要限制傳回特定數字的工作流程數量，例如五，請套用其他篩選器至要求：

```
GET https://localhost:8281/vco/api/catalog/System/Workflow?conditions=name~datastore&maxResult=5
```

匯入及匯出 Orchestrator 物件

Orchestrator REST API 提供讓您可用來匯入與匯出工作流程、動作、套件、資源及組態元素的網路服務。

匯入工作流程

您可使用 Orchestrator REST API 匯入工作流程。

視您的 REST 用戶端應用程式程式庫而定，您可使用定義工作流程內容的自訂程式碼。

先決條件

工作流程二進位內容應可作為多部分內容使用。如需詳細資訊，請參閱 RFC 2387。

程序

1 在 REST 用戶端應用程式中新增要求標頭，以定義您要匯入工作流程的內容。

2 在工作流程物件的 URL 提出 POST 要求：

```
POST http://{orchestrator_host}:{port}/vco/api/workflows/
```

若 POST 要求成功，您會收到狀態碼 202。

匯出工作流程

您可利用 Orchestrator REST API 匯出工作流程並將工作流程當成檔案下載。

程序

1 在 REST 用戶端應用程式中，新增含有下列數值的要求標頭。

- **名稱：接受**
- **數值：應用程式/zip**

2 在您要匯出工作流程的 URL 提出 GET 要求：

```
GET http://{orchestrator_host}:{port}/vco/api/workflows/{workflowID}/
```

若 GET 要求成功，您會收到狀態碼 200。工作流程二進位內容可當成含預設檔案名稱 *workflow_name*.workflow 的附件使用。您可儲存含有 REST 用戶端應用程式的檔案。

匯入動作

您可使用 Orchestrator REST API 匯入動作。

視您的 REST 用戶端應用程式程式庫而定，您可使用定義動作內容的自訂程式碼。

先決條件

動作二進位內容應可作為多部分內容使用。如需詳細資訊，請參閱 RFC 2387。

程序

1 在 REST 用戶端應用程式中新增要求標頭，以定義您要匯入動作的內容。

2 在動作物件的 URL 提出 POST 要求：

```
POST http://{orchestrator_host}:{port}/vco/api/actions/
```

若 POST 要求成功，您會收到狀態碼 202。

匯出動作

您可利用 Orchestrator REST API 匯出動作並將動作當成檔案下載。

程序

1 在 REST 用戶端應用程式中，新增含有下列數值的要求標頭。

- **名稱：接受**
- **數值：應用程式/zip**

2 在您要匯出動作的 URL 提出 GET 要求：

```
GET http://{orchestrator_host}:{port}/vco/api/actions/{actionID}/
```

若 GET 要求成功，您會收到狀態碼 200。動作二進位內容可當成含預設檔案名稱 *action_name*.action 的附件使用。您可儲存含有 REST 用戶端應用程式的檔案。

匯入套件

您可使用 Orchestrator REST API 匯入套件。

視您 REST 用戶端應用程式程式庫而定，您可使用定義套件內容的自訂程式碼。

依預設，若您匯入含重複名稱的 Orchestrator 套件，將不會覆寫現有套件。您可以在要求中使用參數，指定是否覆寫現有套件。

依預設，將匯入含組態元素屬性數值的 Orchestrator 套件。您可在要求中使用參數，匯入沒有屬性數值的套件。

依預設，Orchestrator 套件中的標籤為匯入，但若 Orchestrator 伺服器上已存在相同的標籤，將保留現有標籤的數值。您可以在要求中使用參數，指定是否保留現有標籤數值。

先決條件

套件二進位內容應可作為多部分內容使用。如需詳細資訊，請參閱 RFC 2387。

程序

1 在 REST 用戶端應用程式中新增要求標頭，以定義您要匯入套件的內容。

2 在套件物件的 URL 提出 POST 要求：

```
POST http://{orchestrator_host}:{port}/vco/api/packages/
```

3 (選擇性) 若要匯入套件並以相同名稱覆寫現有套件，請在 POST 要求中使用 overwrite 參數。

```
POST http://{orchestrator_host}:{port}/vco/api/packages/?overwrite=true
```

4 (選擇性) 若要匯入沒有組態元素屬性數值的套件，請使用 POST 要求中的 importConfigurationAttributeValues 參數：

```
POST http://{orchestrator_host}:{port}/vco/api/packages/?importConfigurationAttributeValues=false
```

5 (選擇性) 若要匯入沒有包含標籤的套件，請使用 POST 要求中的 tagImportMode 參數：

```
POST http://{orchestrator_host}:{port}/vco/api/packages/?tagImportMode=DoNotImport
```

6 (選擇性) 若要匯入有包含標籤及覆寫現有標籤數值的套件，請使用 POST 要求中的 tagImportMode 參數：

```
POST http://{orchestrator_host}:{port}/vco/api/packages/?
tagImportMode=ImportAndOverwriteExistingValue
```

若 POST 要求成功，您會收到狀態碼 202。

匯出套件

您可利用 Orchestrator REST API 匯出套件並將套件當成檔案下載。

依預設，將匯出含組態元素及全域標籤屬性數值的 Orchestrator 套件。您可在要求中使用參數，匯出沒有屬性數值或全域標籤的套件。您也可為下載的套件檔案指定自訂名稱。

程序

1 在 REST 用戶端應用程式中，新增含有下列數值的要求標頭。

- **名稱：接受**
- **數值：應用程式/zip**

2 在您要匯出套件的 URL 提出 GET 要求：

```
GET http://{orchestrator_host}:{port}/vco/api/packages/{package_name}/
```

3 (選擇性) 若要為匯出的套件設定自訂名稱，請使用 GET 要求中的 packageName 參數：

```
GET http://{orchestrator_host}:{port}/vco/api/packages/{package_name}/?packageName={custom_name}
```

4 (選擇性) 若要匯出沒有組態元素屬性數值的套件，請使用 GET 要求中的 exportConfigurationAttributeValues 參數：

```
GET http://{orchestrator_host}:{port}/vco/api/packages/{package_name}/?
exportConfigurationAttributeValues=false
```

5 (選擇性) 若要匯出沒有全域標籤的套件，請使用 GET 要求中的 exportGlobalTags 參數：

```
GET http://{orchestrator_host}:{port}/vco/api/packages/{package_name}/?exportGlobalTags=false
```

若 GET 要求成功，您會收到狀態碼 200。套件二進位內容可當成含預設檔案名稱 *package_name*.package 的附件使用。您可儲存含有 REST 用戶端應用程式的檔案。

匯入資源

您可使用 Orchestrator REST API 匯入資源。

視您 REST 用戶端應用程式程式庫而定，您可使用定義資源內容的自訂程式碼。

先決條件

資源二進位內容應可作為多部分內容使用。如需詳細資訊，請參閱 RFC 2387。

程序

1 在 REST 用戶端應用程式中新增要求標頭，以定義您要匯入資源的內容。

2 在資源物件的 URL 提出 POST 要求：

```
POST http://{orchestrator_host}:{port}/vco/api/resources/
```

若 POST 要求成功，您會收到狀態碼 202。

匯出資源

您可使用 Orchestrator REST API 匯出資源。

程序

1 在 REST 用戶端應用程式中，新增含有下列數值的要求標頭。

- **名稱：接受**
- **數值：應用程式/octet 字串**

2 在您要匯出資源的 URL 提出 GET 要求：

```
GET http://{orchestrator_host}:{port}/vco/api/resources/{resourceID}/
```

若 GET 要求成功，您會收到狀態碼 200。在回應本體中可使用資源內容。

匯入組態元素

您可使用 Orchestrator REST API 匯入組態元素。

視您 REST 用戶端應用程式程式庫而定，您可使用定義組態元素內容的自訂程式碼。

先決條件

組態元素二進位內容應可作為多部分內容使用。如需詳細資訊，請參閱 RFC 2387。

程序

1 在 REST 用戶端應用程式中新增要求標頭，以定義您要匯入組態元素的內容。

2 在組態元素物件的 URL 提出 POST 要求：

```
POST http://{orchestrator_host}:{port}/vco/api/configurations/
```

若 POST 要求成功，您會收到狀態碼 202。

匯出組態元素

您可使用 Orchestrator REST API 匯出組態元素。

程序

1 在 REST 用戶端應用程式中，新增含有下列數值的要求標頭。

- **名稱：接受**
- **數值：應用程式/vcoobject+xml**

2 在您要匯出組態元素的 URL 提出 GET 要求：

```
GET http://{orchestrator_host}:{port}/vco/api/configurations/{configuration_elementID}/
```

若 GET 要求成功，您會收到狀態碼 200。在回應本體中可使用組態元素內容。

刪除 Orchestrator 物件

Orchestrator REST API 提供讓您可用來刪除工作流程、動作、套件、資源、組態元素的網路服務。

刪除工作流程

您可使用 Orchestrator REST API 刪除工作流程。

程序

1 發出 GET 申請，然後從傳回的工作流程清單中擷取工作流程的識別碼：

```
GET http://{orchestrator_host}:{port}/vco/api/workflows/
```

2 在工作流程的 URL 提出 DELETE 要求：

```
DELETE http://{orchestrator_host}:{port}/vco/api/workflows/{workflowID}/
```

若 DELETE 要求成功，您會收到狀態碼 200，且回應本體是空的。

刪除動作

您可使用 Orchestrator REST API 刪除動作。

程序

1 發出 GET 申請，然後從傳回的動作清單中擷取動作的識別碼：

```
GET http://{orchestrator_host}:{port}/vco/api/actions/
```

2 在動作的 URL 提出 DELETE 要求：

```
DELETE http://{orchestrator_host}:{port}/vco/api/actions/{actionID}/
```

若 DELETE 要求成功，您會收到狀態碼 200，且回應本體是空的。

刪除套件

您可使用 Orchestrator REST API 刪除套件。

刪除套件時，不會刪除套件中的元素。若您想要刪除套件內容，您必須提供選項參數。

程序

1 發出 GET 申請，然後從傳回的套件清單中擷取套件的名稱：

```
GET http://{orchestrator_host}:{port}/vco/api/packages/
```

2 在套件的 URL 提出 DELETE 要求，且若您想要刪除套件中的元素，請在要求結束時提供選項參數：

```
DELETE http://{orchestrator_host}:{port}/vco/api/packages/{package_name}/?option={parameter}
```

參數	說明
deletePackage	只刪除套件，將同時保留其內容。
deletePackageWithContent	刪除套件及其所有內容。若其他套件與刪除的套件共用元素，將刪除其他套件中的共用元素。
deletePackageKeepingShared	套件及未共用的內容都將刪除。與其他套件共用的元素不會刪除。

若您不提供選項參數，將使用預設 deletePackage 參數。

若 DELETE 要求成功，您會收到狀態碼 200，且回應本體是空的。

刪除資源

您可使用 Orchestrator REST API 刪除資源。

程序

1 發出 GET 申請，然後從傳回的資源清單中擷取資源的識別碼：

```
GET http://{orchestrator_host}:{port}/vco/api/resources/
```

2 在資源的 URL 提出 DELETE 要求：

```
DELETE http://{orchestrator_host}:{port}/vco/api/resources/{resourceID}/
```

若 DELETE 要求成功，您會收到狀態碼 200，且回應本體是空的。

刪除組態元素

您可使用 Orchestrator REST API 刪除組態元素。

程序

1 發出 GET 申請，然後從傳回的組態元素清單中擷取組態元素的識別碼：

```
GET http://{orchestrator_host}:{port}/vco/api/configurations/
```

2 在組態元素的 URL 提出 DELETE 要求：

```
DELETE http://{orchestrator_host}:{port}/vco/api/configurations/{configuration_elementID}/
```

若 DELETE 要求成功，您會收到狀態碼 200，且回應本體是空的。

設定 Orchestrator 物件的權限

您可使用 Orchestrator REST API 設定 Orchestrator 物件的權限。若要設定權限，必須在物件權限的 URL 提出 POST 要求並在要求本體中定義權限。

您也可使用 Orchestrator REST API 擷取有關物件權限的資訊，或刪除現有權限。

REST API 權限

當您使用 Orchestrator REST API 設定權限時，您必須使用字元組合設定權限。

您可在 POST 要求的要求本體 <rights> 索引標籤中加入一系列字元，以定義元素權限。

您可用來設定 Orchestrator REST API 權限的字元含有特定意義。

表格 2-1. Orchestrator REST API 權限字元組合

字元	說明
r	賦予檢視權限。
x	賦予執行權限。
i	賦予檢查權限。
c	賦予編輯權限。
a	賦予管理權限。

範例 2-14. 設定權限的語法

您可在 Orchestrator 元素權限 URL 的 POST 要求的要求本體中使用下列範例語法。

```
<permissions xmlns="http://www.vmware.com/vco">
    <permission>
        <principal>cn=vcousers,ou=vco,dc=appliance</principal>
        <rights>ric</rights>
    </permission>
</permissions>
```

在要求本體的 <rights> 索引標籤設定 ric 權限後，您即可讓 vcousers 使用者群組的成員檢視、檢查及編輯 Orchestrator 元素。

擷取工作流程的權限

您可使用 Orchestrator REST API 擷取工作流程權限的資訊。

程序

1 發出 GET 申請，然後從傳回的工作流程清單中擷取工作流程的識別碼：

```
GET http://{orchestrator_host}:{port}/vco/api/workflows/
```

2 在工作流程權限的 URL 提出 GET 要求：

```
GET http://{orchestrator_host}:{port}/vco/api/workflows/{workflowID}/permissions/
```

若 GET 要求成功，您會收到狀態碼 200。在回應本體中，可使用有關工作流程權限的資訊。

刪除工作流程的權限

您可使用 Orchestrator REST API 刪除工作流程的權限。您可先刪除工作流程的現有權限，再設定新權限。

程序

1 發出 GET 申請，然後從傳回的工作流程清單中擷取工作流程的識別碼：

```
GET http://{orchestrator_host}:{port}/vco/api/workflows/
```

2 在工作流程權限的 URL 提出 DELETE 要求：

```
DELETE http://{orchestrator_host}:{port}/vco/api/workflows/{workflowID}/permissions/
```

若 DELETE 要求成功，您會收到狀態碼 204，且回應本體是空的。

設定工作流程的權限

您可使用 Orchestrator REST API 設定工作流程的權限。

先決條件

檢閱您可設定權限的型別，以及您可在要求本體中使用的語法。請參閱 REST API 權限。

程序

1 發出 GET 申請，然後從傳回的工作流程清單中擷取工作流程的識別碼：

```
GET http://{orchestrator_host}:{port}/vco/api/workflows/
```

2 在 REST 用戶端應用程式中新增要求標頭，以定義您要設定權限的工作流程內容。

3 在要求本體中，指定您要設定的權限。

4 在工作流程權限的 URL 提出 POST 要求：

```
POST http://{orchestrator_host}:{port}/vco/api/workflows/{workflowID}/permissions/
```

若 POST 要求成功，您會收到狀態碼 201。在回應本體中，可使用有關工作流程權限的資訊。

擷取動作的權限

您可使用 Orchestrator REST API 擷取動作權限的資訊。

程序

1 發出 GET 申請，然後從傳回的動作清單中擷取動作的識別碼：

```
GET http://{orchestrator_host}:{port}/vco/api/actions/
```

2 在動作權限的 URL 提出 GET 要求：

```
GET http://{orchestrator_host}:{port}/vco/api/actions/{actionID}/permissions/
```

若 GET 要求成功，您會收到狀態碼 200。在回應本體中，可使用有關動作權限的資訊。

刪除動作的權限

您可使用 Orchestrator REST API 刪除動作的權限。您可先刪除動作的現有權限，再設定新權限。

程序

1 發出 GET 申請，然後從傳回的動作清單中擷取動作的識別碼：

```
GET http://{orchestrator_host}:{port}/vco/api/actions/
```

2 在動作權限的 URL 提出 DELETE 要求：

```
DELETE http://{orchestrator_host}:{port}/vco/api/actions/{actionID}/permissions/
```

若 DELETE 要求成功，您會收到狀態碼 204，且回應本體是空的。

設定動作的權限

您可使用 Orchestrator REST API 設定動作的權限。

先決條件

檢閱您可設定權限的型別，以及您可在要求本體中使用的語法。請參閱 REST API 權限。

程序

1 發出 GET 申請，然後從傳回的動作清單中擷取動作的識別碼：

```
GET http://{orchestrator_host}:{port}/vco/api/actions/
```

2 在 REST 用戶端應用程式中新增要求標頭，以定義您要設定權限的動作內容。

3 在要求本體中，指定您要設定的權限。

4 在動作權限的 URL 提出 POST 要求：

```
POST http://{orchestrator_host}:{port}/vco/api/actions/{actionID}/permissions/
```

若 POST 要求成功，您會收到狀態碼 201。在回應本體中，可使用有關動作權限的資訊。

擷取套件的權限

您可使用 Orchestrator REST API 擷取套件權限的資訊。

程序

1 發出 GET 申請，然後從傳回的套件清單中擷取套件的名稱：

```
GET http://{orchestrator_host}:{port}/vco/api/packages/
```

2 在套件權限的 URL 提出 GET 要求：

```
GET http://{orchestrator_host}:{port}/vco/api/packages/{package_name}/permissions/
```

若 GET 要求成功，您會收到狀態碼 200。在回應本體中，可使用有關套件權限的資訊。

刪除套件的權限

您可使用 Orchestrator REST API 刪除套件的權限。您可先刪除套件的現有權限，再設定新權限。

程序

1 發出 GET 申請，然後從傳回的套件清單中擷取套件的名稱：

```
GET http://{orchestrator_host}:{port}/vco/api/packages/
```

2 在套件權限的 URL 提出 DELETE 要求：

```
DELETE http://{orchestrator_host}:{port}/vco/api/packages/{package_name}/permissions/
```

若 DELETE 要求成功，您會收到狀態碼 204，且回應本體是空的。

設定套件的權限

您可使用 Orchestrator REST API 設定套件的權限。

先決條件

檢閱您可設定權限的型別，以及您可在要求本體中使用的語法。請參閱 REST API 權限。

程序

1 發出 GET 申請，然後從傳回的套件清單中擷取套件的名稱：

```
GET http://{orchestrator_host}:{port}/vco/api/packages/
```

2 在 REST 用戶端應用程式中新增要求標頭，以定義您要設定權限的套件內容。

3 在要求本體中，指定您要設定的權限。

4 在套件權限的 URL 提出 POST 要求：

```
POST http://{orchestrator_host}:{port}/vco/api/packages/{package_name}/permissions/
```

若 POST 要求成功，您會收到狀態碼 201。在回應本體中，可使用有關套件權限的資訊。

擷取資源的權限

您可使用 Orchestrator REST API 擷取資源權限的資訊。

程序

1 發出 GET 申請，然後從傳回的資源清單中擷取資源的識別碼：

```
GET http://{orchestrator_host}:{port}/vco/api/resources/
```

2 在資源權限的 URL 提出 GET 要求：

```
GET http://{orchestrator_host}:{port}/vco/api/resources/{resourceID}/permissions/
```

若 GET 要求成功，您會收到狀態碼 200。在回應本體中，可使用有關資源權限的資訊。

刪除資源的權限

您可使用 Orchestrator REST API 刪除資源的權限。您可先刪除資源的現有權限，再設定新權限。

程序

1 發出 GET 申請，然後從傳回的資源清單中擷取資源的識別碼：

```
GET http://{orchestrator_host}:{port}/vco/api/resources/
```

2 在資源權限的 URL 提出 DELETE 要求：

```
DELETE http://{orchestrator_host}:{port}/vco/api/resources/{resourceID}/permissions/
```

若 DELETE 要求成功，您會收到狀態碼 204，且回應本體是空的。

設定資源的權限

您可使用 Orchestrator REST API 設定資源的權限。

先決條件

檢閱您可設定權限的型別，以及您可在要求本體中使用的語法。請參閱 REST API 權限。

程序

1 發出 GET 申請，然後從傳回的資源清單中擷取資源的識別碼：

```
GET http://{orchestrator_host}:{port}/vco/api/resources/
```

2 在 REST 用戶端應用程式中新增要求標頭，以定義您要設定權限的資源內容。

3 在要求本體中，指定您要設定的權限。

4 在資源權限的 URL 提出 POST 要求：

```
POST http://{orchestrator_host}:{port}/vco/api/resources/{resourceID}/permissions/
```

若 POST 要求成功，您會收到狀態碼 201。在回應本體中，可使用有關資源權限的資訊。

擷取組態元素的權限

您可使用 Orchestrator REST API 擷取有關組態元素權限的資訊。

程序

1 發出 GET 申請，然後從傳回的組態元素清單中擷取組態元素的識別碼：

```
GET http://{orchestrator_host}:{port}/vco/api/configurations/
```

2 在組態元素權限的 URL 提出 GET 要求：

```
GET http://{orchestrator_host}:{port}/vco/api/configurations/{configuration_elementID}/permissions/
```

若 GET 要求成功，您會收到狀態碼 200。在回應本體中，可使用有關組態元素權限的資訊。

刪除組態元素的權限

您可使用 Orchestrator REST API 刪除組態元素的權限。您可先刪除組態元素的現有權限，再設定新權限。

程序

1 發出 GET 申請，然後從傳回的組態元素清單中擷取組態元素的識別碼：

```
GET http://{orchestrator_host}:{port}/vco/api/configurations/
```

2 在組態元素權限的 URL 提出 DELETE 要求：

```
DELETE http://{orchestrator_host}:
{port}/vco/api/configurations/{configuration_elementID}/permissions/
```

若 DELETE 要求成功，您會收到狀態碼 204，且回應本體是空的。

設定組態元素的權限

您可使用 Orchestrator REST API 設定組態元素的權限。

先決條件

檢閱您可設定權限的型別，以及您可在要求本體中使用的語法。請參閱 REST API 權限。

程序

1 發出 GET 申請，然後從傳回的組態元素清單中擷取組態元素的識別碼：

```
GET http://{orchestrator_host}:{port}/vco/api/configurations/
```

2 在 REST 用戶端應用程式中新增要求標頭，以定義您要設定權限的組態元素內容。

3 在要求本體中，指定您要設定的權限。

4 在組態元素權限的 URL 提出 POST 要求：

```
POST http://{orchestrator_host}:
{port}/vco/api/configurations/{configuration_elementID}/permissions/
```

若 POST 要求成功，您會收到狀態碼 201。在回應本體中，可使用有關組態元素權限的資訊。

執行含外掛程式的作業

Orchestrator REST API 提供的網路服務可讓您用來執行各種含外掛程式的作業。

擷取有關外掛程式的資訊

您可利用 Orchestrator REST API 擷取所有安裝外掛程式的中繼資料資訊。

程序

1 在 REST 用戶端應用程式中新增要求標頭，以定義外掛程式的內容。

2 在外掛程式物件的 URL 提出 GET 要求：

```
GET http://{orchestrator_host}:{port}/vco/api/plugins/
```

若 GET 要求成功，您會收到狀態碼 200。

匯入外掛程式

您可使用 Orchestrator REST API 匯入外掛程式。

視您 REST 用戶端應用程式程式庫而定，您可使用定義外掛程式內容的自訂程式碼。

備註 若已安裝相同名稱的外掛程式，則您無法匯入外掛程式。

先決條件

外掛程式二進位內容應可作為多部分內容使用。如需詳細資訊，請參閱 RFC 2387。

程序

1 在 REST 用戶端應用程式中新增要求標頭，以定義您要匯入外掛程式的內容。

2 在外掛程式物件的 URL 提出 POST 要求：

```
POST http://{orchestrator_host}:{port}/vco/api/plugins/
```

若 POST 要求成功，您會收到狀態碼 200。

匯出外掛程式

您可使用 Orchestrator REST API 匯出外掛程式。

程序

1 在 REST 用戶端應用程式中，新增含有下列數值的要求標頭。

- **名稱：接受**
- **數值：應用程式/dar**

2 在您要匯出外掛程式的 URL 提出 GET 要求：

```
GET http://{orchestrator_host}:{port}/vco/api/plugins/{plug-in_name}/
```

若 GET 要求成功，您會收到狀態碼 200。在回應本體中可使用外掛程式內容。

啟用或停用外掛程式

您可使用 Orchestrator REST API 啟用或停用外掛程式。

您可在外掛程式的 URL 提出 PUT 要求，將啟用的外掛程式狀態變更為停用，或從停用變為啟用。您可擷取有關 Orchestrator 外掛程式的資訊，檢查外掛程式的目前狀態。請參閱擷取有關外掛程式的資訊。

先決條件

外掛程式二進位內容應可作為多部分內容使用。如需詳細資訊，請參閱 RFC 2387。

程序

1 在 REST 用戶端應用程式中新增要求標頭，以定義您要啟用或停用外掛程式的內容。

2 在您要啟用或停用外掛程式的 URL 提出 PUT 要求：

```
PUT http://{orchestrator_host}:{port}/vco/api/plugins/{plug-in_name}/state/
```

若 PUT 要求成功，您會收到狀態碼 200。

執行伺服器組態作業

Orchestrator REST API 提供的網路服務可讓您用來執行相關 Orchestrator 伺服器組態的各種作業。

擷取有關 Orchestrator 伺服器組態的資訊

您可使用 Orchestrator REST API 擷取有關 Orchestrator 伺服器組態的資訊。

程序

1 在 REST 用戶端應用程式中新增要求標頭，以定義您要擷取資訊的伺服器內容。

2 在外掛程式物件的 URL 提出 GET 要求：

```
GET http://{orchestrator_host}:{port}/vco/api/server-configuration/
```

若 GET 要求成功，您會收到狀態碼 200。

匯入 Orchestrator 伺服器組態

您可使用 Orchestrator REST API 匯入儲存的組態。

先決條件

組態二進位內容應可作為多部分內容使用。如需詳細資訊，請參閱 RFC 2387。

程序

1 在 REST 用戶端應用程式中，新增含有下列數值的要求標頭。

- **名稱：內容型別**
- **數值：多部分 / 表單資料**

2 在伺服器組態的 URL 提出 POST 要求：

```
POST http://{orchestrator_host}:{port}/vco/api/server-configuration/
```

若 POST 要求成功，您會收到狀態碼 200。

匯出 Orchestrator 伺服器組態

您可使用 Orchestrator REST API 匯出伺服器組態。

先決條件

組態二進位內容應可作為多部分內容使用。如需詳細資訊，請參閱 RFC 2387。

程序

1 在 REST 用戶端應用程式中，新增含有下列數值的要求標頭。

- **名稱：內容型別**
- **數值：多部分 / 表單資料**

2 新增其他含有下列數值的要求標頭。

- **名稱：接受**
- **數值：*/***

3 在伺服器組態的 URL 提出 POST 要求：

```
POST http://{orchestrator_host}:{port}/vco/api/server-configuration/
```

若 POST 要求成功，您會收到狀態碼 200。

執行標記作業

Orchestrator REST API 提供網路服務，讓您可用於執行各種作業以利用 Orchestrator 中的標籤讓物件更容易搜尋。

您可透過附加標籤讓物件更容易搜尋。標籤為長度介於 3 至 64 字元的字串，且不得包含任何空白字元。

您可新增全域或私人標籤。所有 Orchestrator 使用者都能看見全域標籤，而私人標籤只有建立該標籤的使用者才看得見。全域標籤只能由擁有管理權限的使用者建立與移除。

標記物件

您可使用 Orchestrator REST API 指派標記至物件。

您可建立私人與全域標籤您可在要求本體中，指定標籤為私人或全域。

備註 若要建立全域索引標籤，您必須以具有管理權限的使用者身分登入。

您也可指派數值至您建立的標籤。數值屬於選用參數，您可用來篩選標籤。

程序

1 使用下列語法定義要求本體。

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<tag-instance xmlns="http://www.vmware.com/vco" global="false">
  <name>tag_name</name>
  <value>tag_value</value>
</tag-instance>
```

備註 您可設定 **全域** 變數為「**true**」建立全域標籤。

2 在物件的 URL 提出 POST 要求：

```
POST http://{orchestrator_host}:{port}/vco/api/catalog/{namespace}/{objectType}/{objectId}/tags
```

若 POST 要求成功，您會收到狀態碼 200。

取消標記物件

您可使用 Orchestrator REST API 移除指派至物件的標籤。

您可移除私人與全域標籤。

備註 若要移除全域索引標籤，您必須以具有管理權限的使用者身分登入。

程序

- 提出 DELETE 要求，移除私人或全域標籤。
  - 若要移除私人標籤，請利用下列語法在物件的 URL 提出 DELETE 要求。

    ```
    DELETE http://{orchestrator_host}:
    {port}/vco/api/catalog/{namespace}/{objectType}/{objectId}/tag/{tag_name}
    ```

  - 若要移除全域標籤，請利用下列語法在物件的 URL 提出 DELETE 要求。

    ```
    DELETE http://{orchestrator_host}:
    {port}/vco/api/catalog/{namespace}/{objectType}/{objectId}/tag/:{tag_name}
    ```

若 DELETE 要求成功，您會收到狀態碼 200。

列出物件索引標籤

您可使用 Orchestrator REST API 擷取指派至物件的索引標籤清單。

程序

- 在物件的 URL 提出 GET 要求：

  ```
  GET http://{orchestrator_host}:{port}/vco/api/catalog/{namespace}/{objectType}/{objectId}/tags
  ```

若 GET 要求成功，您會收到狀態碼 200。

按型別列出標記物件

您可使用 Orchestrator REST API 擷取標記有指定索引標籤的物件清單，並透過物件型別篩選。

程序

- 在物件型別的 URL 提出 GET 要求：

  ```
  GET http://{orchestrator_host}:{port}/vco/api/catalog/{namespace}/{objectType}/?
  tags=tag1&tags=:tag2=value
  ```

若 GET 要求成功，您會收到狀態碼 200。

列出索引標籤擁有者

您可使用 Orchestrator REST API 擷取索引標籤擁有者的清單。索引標籤擁有者屬於建立至少一個索引標籤的使用者。

程序

- 在下列 URL 提出 GET 要求：

```
GET http://{orchestrator_host}:{port}/vco/api/tags
```

若 GET 要求成功，您會收到狀態碼 200。您擷取的清單包含已建立至少一個索引標籤的使用者。全域列在系統使用者名稱下 __GLOBAL__。

按使用者列出索引標籤

您可使用 Orchestrator REST API 擷取指定使用者建立的索引標籤清單。

您也可擷取全域索引標籤。全域列在系統使用者名稱下 __GLOBAL__。

程序

- 在使用者的 URL 提出 GET 要求：
  - 若要擷取指定使用者建立的索引標籤清單，請透過使用下列語法提出 GET 要求：

```
GET http://{orchestrator_host}:{port}/vco/api/tags/{user_name}
```

  - 若要擷取全域索引標籤清單，請透過使用下列語法提出 GET 要求：

```
GET http://{orchestrator_host}:{port}/vco/api/tags/__GLOBAL__
```

若 GET 要求成功，您會收到狀態碼 200。

列出按索引標籤名稱篩選的使用者索引標籤

您可使用 Orchestrator REST API 擷取指定使用者建立的索引標籤清單執行個體，並透過索引標籤名稱篩選索引標籤。

您也可擷取全域索引標籤執行個體。全域列在系統使用者名稱下 __GLOBAL__。

程序

- 在使用者的 URL 提出 GET 要求：
  - 若要擷取指定使用者建立的篩選索引標籤清單執行個體，請透過使用下列語法提出 GET 要求：

```
GET http://{orchestrator_host}:{port}/vco/api/tags/{user_name}/{tag_name}
```

  - 若要擷取全域索引標籤篩選清單執行個體，請透過使用下列語法提出 GET 要求：

```
GET http://{orchestrator_host}:{port}/vco/api/tags/__GLOBAL__/{tag_name}
```

若 GET 要求成功，您會收到狀態碼 200。您擷取的資訊包含標記物件參考、索引標籤名稱、索引標籤數值及指示索引標籤執行個體是否為全域或私人。

按使用者移除索引標籤

您可使用 Orchestrator REST API 移除指定使用者建立的所有索引標籤。

您也可移除全域索引標籤。全域列在系統使用者名稱下 __GLOBAL__。

備註 若要移除全域索引標籤，您必須以具有管理權限的使用者身分登入。

程序

- 在使用者的 URL 提出 DELETE 要求：
  - 若要移除指定使用者建立的索引標籤，請透過使用下列語法提出 DELETE 要求：

    ```
    DELETE http://{orchestrator_host}:{port}/vco/api/tags/{user_name}
    ```

  - 若要移除全域索引標籤，請透過使用下列語法提出 DELETE 要求：

    ```
    DELETE http://{orchestrator_host}:{port}/vco/api/tags/__GLOBAL__
    ```

若 DELETE 要求成功，您會收到狀態碼 204。