

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

vRealize Orchestrator 7.2

本文件支援所列的每個產品版本，並支援後續版本直到新版本的文件取代本文件為止。若要查看本文件的最新版本，請參閱 <http://www.vmware.com/tw/support/pubs>。

ZH_TW-002392-00

vmware[®]

您可以在 VMware 網站上找到最新的技術說明文件，網址為：

<http://www.vmware.com/tw/support/>

VMware 網站還提供了最新的產品更新。

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

Copyright © 2008–2016 VMware, Inc. 版權所有。 [版權和商標資訊](#)。

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 5

1 開發 Web 服務用戶端 7

2 使用 vRealize Orchestrator REST API 9

 針對 Orchestrator 及第三方系統進行驗證 10

 透過 Orchestrator REST API 使用 vCenter Single Sign-On 驗證 10

 使用含 Orchestrator REST API 的 LDAP 驗證 12

 存取 Orchestrator REST API 的參考說明文件 12

 使用 Java REST SDK 12

 含工作流程的作業 13

 尋找工作流程並擷取其定義 13

 執行工作流程 16

 針對工作流程呈現驗證其輸入參數後執行工作流程 18

 在工作流程執行時與其互動 21

 擷取工作流程的互動 25

 存取工作流程配置 26

 使用工作 26

 建立工作 26

 修改工作 27

 检查工作狀態 28

 在 Orchestrator 詳細目錄中尋找物件 28

 按型別及識別碼尋找物件 28

 按關係尋找物件 29

 套用篩選器 30

 匯入及匯出 Orchestrator 物件 30

 匯入工作流程 30

 匯出工作流程 31

 匯入動作 31

 匯出動作 31

 匯入套件 32

 匯出套件 32

 匯入資源 33

 匯出資源 33

 匯入組態元素 34

 匯出組態元素 34

 刪除 Orchestrator 物件 34

 刪除工作流程 34

 刪除動作 35

刪除套件	35
刪除資源	35
刪除組態元素	36
設定 Orchestrator 物件的權限	36
REST API 權限	36
擷取工作流程的權限	37
刪除工作流程的權限	37
設定工作流程的權限	37
擷取動作的權限	38
刪除動作的權限	38
設定動作的權限	38
擷取套件的權限	39
刪除套件的權限	39
設定套件的權限	39
擷取資源的權限	40
刪除資源的權限	40
設定資源的權限	40
擷取組態元素的權限	41
刪除組態元素的權限	41
設定組態元素的權限	41
執行含外掛程式的作業	42
擷取有關外掛程式的資訊	42
匯入外掛程式	42
匯出外掛程式	42
啟用或停用外掛程式	43
執行伺服器組態作業	43
擷取有關 Orchestrator 伺服器組態的資訊	43
匯入 Orchestrator 伺服器組態	43
匯出 Orchestrator 伺服器組態	44
執行標記作業	44
標記物件	44
取消標記物件	45
列出物件索引標籤	45
按型別列出標記物件	45
列出索引標籤擁有人	46
按使用者列出索引標籤	46
列出按索引標籤名稱篩選的使用者索引標籤	46
按使用者移除索引標籤	47

索引 49

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

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

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：

- [“針對 Orchestrator 及第三方系統進行驗證,”](#) 第 10 頁
- [“存取 Orchestrator REST API 的參考說明文件,”](#) 第 12 頁
- [“使用 Java REST SDK,”](#) 第 12 頁
- [“含工作流程的作業,”](#) 第 13 頁
- [“使用工作,”](#) 第 26 頁
- [“在 Orchestrator 詳細目錄中尋找物件,”](#) 第 28 頁
- [“匯入及匯出 Orchestrator 物件,”](#) 第 30 頁
- [“刪除 Orchestrator 物件,”](#) 第 34 頁
- [“設定 Orchestrator 物件的權限,”](#) 第 36 頁
- [“執行含外掛程式的作業,”](#) 第 42 頁
- [“執行伺服器組態作業,”](#) 第 43 頁
- [“執行標記作業,”](#) 第 44 頁

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，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。

視您是否設定含 LDAP 或含 vCenter Single Sign-On 的 Orchestrator 而定，Orchestrator REST API 的驗證配置也會不同。若 Orchestrator 使用 LDAP，您必須使用有效的認證來驗證。若 Orchestrator 使用 vCenter Single Sign-On，則視您組態而定，可使用基本驗證或 vCenter Single Sign-On 伺服器發行的金鑰持有者 Token 進行驗證。

備註 LDAP 驗證已過時，未來的版本將不支援。

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 預設連接埠號碼是 8281。

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

若設定 Orchestrator 停用 vCenter Single Sign-On 伺服器和基本驗證，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 伺服器與基本驗證，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。
- **VCOAuthzation**。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(ssoUri, sessionFactory, 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" solution-user="Orchestrator-133acc26ff78e5695b102146326"
admin-rights="true"/>
```

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。

使用含 Orchestrator REST API 的 LDAP 驗證

若透過 LDAP 設定 Orchestrator，或者若您使用 Orchestrator 伺服器存取使用 LDAP 設定的第三方系統，您必須套用基本 HTTP 驗證方案。

基本 HTTP 驗證方案允許您在 API 呼叫中加入 Authorization 標頭，針對 Orchestrator 或第三方系統驗證。您必須在 Authorization 標頭中提供 base64 編碼的認證。Orchestrator 使用相同的認證代表您針對透過 LDAP 設定的第三方系統驗證。

如需有關基本 HTTP 驗證的詳細資訊，請參閱 RFC 2617。

範例 2-2. 在 LDAP 模式中取得工作階段

下列範例代碼將取得 LDAP 模式中的工作階段。

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

//provide LDAP credentials
Authentication auth = new UsernamePasswordAuthentication("username", "password");

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

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

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 2-3. 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 2-4. 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/CF808080808080808080808080808080E6808080013086668236014a
0614d16e1/">
    <attributes>
      <attribute name="id"
value="CF808080808080808080808080808080E6808080013086668236014a0614d16e1"/>
      <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/CF808080808080808080808080808080DA808080013086668236014a
0614d16e1/">
    <attributes>
      <attribute name="id"
value="CF808080808080808080808080808080DA808080013086668236014a0614d16e1"/>
      <attribute name="canExecute" value="true" />
      <attribute name="description" value="" />
```


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 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 2-14. 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
<code>deletePackage</code>	只刪除套件，將同時保留其內容。
<code>deletePackageWithContent</code>	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。
<code>deletePackageKeepingShared</code>	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 2-15. 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](#)”第 36 頁。

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](#),” 第 36 頁。

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](#)”第 36 頁。

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,”](#) 第 36 頁。

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](#)”第 36 頁。

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](#)”第 42 頁。

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。

索引

字母

API 說明文件 12
Java REST SDK 12
LDAP
 vCenter Single Sign-On 10
 驗證 10
Orchestrator API 9
REST API, 權限 36
SDK, Java REST 12
Web 服務 7

三劃

工作
 state 28
 建立 26
 修改 27
 排程工作流程 26
 檢查狀態 28
工作流程
 下載 31
 上傳 30
 互動 21
 回應使用者互動 22
 存取配置 26
 刪除 25, 34
 刪除權限 37
 使用者互動 22, 23
 取消 25
 取得權限 37
 執行 16, 18, 21, 25
 執行時互動 21
 尋找 13
 匯入 30
 匯出 31
 篩選器 13
 擷取互動 25
 擷取權限 37
 驗證 18
 驗證並執行 18

五劃

外掛程式
 作業 42
 停用 43

啟用 43
匯入 42
匯出 42
資訊擷取 42
擷取資訊 42

七劃

伺服器組態
 作業 43
 匯入 43
 匯出 44
 擷取資訊 43
刪除物件 34

八劃

物件, 尋找 28

十劃

套件
 下載 32
 上傳 32
 刪除 35
 刪除權限 39
 取得權限 39
 匯入 32
 匯出 32
 擷取權限 39

十一劃

動作
 下載 31
 上傳 31
 刪除 35
 刪除權限 38
 取得權限 38
 匯入 31
 匯出 31
 擷取權限 38
參考 12
排程工作流程 26
組態元素
 下載 34
 上傳 34
 刪除 36
 刪除權限 41

- 取得權限 41
- 匯入 34
- 匯出 34
- 擷取權限 41

- vCenter Single Sign-On 10
- 解決方案使用者名稱 11

十二劃

尋找

- 物件 29
- 按照型別 28
- 按照識別碼 28
- 尋找物件 28

十三劃

- 匯入 30
- 匯出 30
- 詳細目錄, 搜尋 28
- 資源
 - 下載 33
 - 上傳 33
 - 刪除 35
 - 刪除權限 40
 - 取得權限 40
 - 匯入 33
 - 匯出 33
 - 擷取權限 40

十四劃

- 對象 5

十五劃

標記

- 列出索引標籤 45
- 列出索引標籤擁有者 46
- 作業 44
- 取消標記 45
- 按使用者列出索引標籤 46
- 按使用者列出篩選的索引標籤 46
- 按使用者移除索引標籤 47
- 按型別列出標記物件 45
- 標籤 44

十六劃

- 篩選器 30

二十二劃

- 權限 36

二十三劃

驗證

- LDAP 12
- Orchestrator 解決方案使用者名稱 11