

# Desarrollo con VMware vRealize Orchestrator

vRealize Orchestrator 7.6



vmware®

Puede encontrar la documentación técnica más actualizada en el sitio web de VMware:

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

Si tiene comentarios relacionados con esta documentación, envíelos a:

[docfeedback@vmware.com](mailto:docfeedback@vmware.com)

**VMware, Inc.**  
3401 Hillview Ave.  
Palo Alto, CA 94304  
[www.vmware.com](http://www.vmware.com)

**VMware Spain, S.L.**  
Calle Rafael Boti 26  
2.ª planta  
Madrid 28023  
Tel.: +34 914125000  
[www.vmware.com/es](http://www.vmware.com/es)

Copyright © 2008-2019 VMware, Inc. Todos los derechos reservados. [Información sobre el copyright y la marca comercial.](#)

# Contenido

## Desarrollar con VMware vRealize Orchestrator 10

### 1 Desarrollar flujos de trabajo 11

Conceptos clave de los flujos de trabajo	14
Parámetros de flujo de trabajo	14
Atributos de los flujos de trabajo	14
Esquema de flujo de trabajo	15
Presentación del flujo de trabajo	15
Tokens de flujo de trabajo	15
Fases del proceso de desarrollo de flujos de trabajo	15
Recomendaciones para desarrollar flujos de trabajo	16
Derechos de acceso para el cliente de Orchestrator	16
Probar flujos de trabajo durante el desarrollo	16
Crear y editar un flujo de trabajo	17
Crear un flujo de trabajo	17
Editar un flujo de trabajo	18
Editar un flujo de trabajo de la biblioteca estándar	18
Pestañas del editor de flujos de trabajo	19
Proporcionar información de flujo de trabajo general	20
Definir atributos y parámetros	21
Definir parámetros de flujo de trabajo	22
Definir atributos de flujo de trabajo	23
Restricciones de nomenclatura de atributos y parámetros	23
Esquema de flujo de trabajo	24
Ver el esquema de los flujos de trabajo	25
Crear un flujo de trabajo en el esquema del flujo de trabajo	26
Elementos de esquema	29
Propiedades de elementos de esquema	34
Enlaces y vínculos	37
Decisiones	44
Control de excepciones	47
Utilizar gestores de errores	48
Elementos Foreach y tipos compuestos	50
Añadir actividad de switch a un flujo de trabajo	53
Desarrollar complementos	54
Descripción general de los complementos	54
Contenido y estructura de un complemento	64
Referencia de API del complemento de Orchestrator	69

Elementos del archivo de definición del complemento vso.xml	80
Recomendaciones para el desarrollo de complementos de Orchestrator	99
Obtener parámetros de entrada de los usuarios cuando se inicia un flujo de trabajo	114
Crear el cuadro de diálogo Parámetros de entrada en la pestaña Presentación	115
Establecer propiedades de parámetros	117
Interacciones de usuario requeridas durante la ejecución de un flujo de trabajo	119
Añadir una interacción del usuario a un flujo de trabajo	120
Configurar el atributo security.group para la interacción del usuario	121
Establecimiento del atributo timeout.date en una fecha absoluta	122
Cálculo del tiempo de espera relativo para las interacciones del usuario	123
Establecer el atributo timeout.date en una fecha relativa	125
Definir las entradas externas para una interacción del usuario	125
Definir el comportamiento de excepciones de la interacción del usuario	126
Crear el cuadro de diálogo de parámetros de entrada para la interacción del usuario	128
Responder a una solicitud de interacción del usuario	129
Llamar a flujos de trabajo desde flujos de trabajo	130
Elementos de los flujos de trabajo que llaman a flujos de trabajo	131
Llamar a un flujo de trabajo de forma sincrónica	134
Llamar a un flujo de trabajo de forma asincrónica	135
Programar un flujo de trabajo	136
Requisitos previos para llamar a un flujo de trabajo remoto desde otro flujo de trabajo	137
Llamar a varios flujos de trabajo simultáneamente	138
Ejecutar un flujo de trabajo en una selección de objetos	139
Implementar los flujos de trabajo Iniciar flujos de trabajo en serie e Iniciar flujos de trabajo en paralelo	140
Desarrollar flujos de trabajo de larga ejecución	142
Establecer una fecha y una hora relativas para flujos de trabajo basados en temporizador	143
Crear un flujo de trabajo de larga ejecución basado en temporizador	144
Crear un objeto activador	146
Crear un flujo de trabajo de larga ejecución basado en activador	148
Elementos de configuración	149
Crear un elemento de configuración	149
Permisos de usuario de un flujo de trabajo	151
Establecer permisos de usuario en un flujo de trabajo	151
Validar flujos de trabajo	152
Validar un flujo de trabajo y corregir los errores de validación	152
Depurar flujos de trabajo	153
Depurar un flujo de trabajo	154
Depurar un flujo de trabajo de ejemplo	155
Flujos de trabajo en ejecución	156
Ejecutar un flujo de trabajo en el Editor de flujos de trabajo	156
Ejecutar un flujo de trabajo	157

Reanudar la ejecución de un flujo de trabajo fallido	159
Comportamiento para reanudar una ejecución de flujo de trabajo fallida	159
Definición de propiedades personalizadas para reanudar las ejecuciones de flujos de trabajo que han fallado	160
Reanudar la ejecución de un flujo de trabajo fallido	160
Generar documentación de flujo de trabajo	161
Historial de versiones de flujos de trabajo	161
Restaurar flujos de trabajo eliminados	162
Desarrollar un flujo de trabajo simple de ejemplo	163
Crear el ejemplo de flujo de trabajo simple	165
Crear el esquema del ejemplo de flujo de trabajo simple	166
Crear las zonas de ejemplo de flujo de trabajo simple	168
Definir los parámetro del ejemplo de flujo de trabajo simple	170
Definir los enlaces de decisiones del ejemplo de flujo de trabajo simple	171
Enlazar los elementos de acción del ejemplo de flujo de trabajo simple	172
Enlazar los elementos de tarea de scripts de ejemplo de flujo de trabajo simple	176
Definir los enlaces de excepciones del ejemplo de flujo de trabajo simple	184
Establecer las propiedades de lectura-escritura para atributos del ejemplo de flujo de trabajo simple	185
Definir las propiedades de parámetro del ejemplo de flujo de trabajo simple	186
Establecer el diseño del cuadro de diálogo de parámetros de entrada del ejemplo de flujo de trabajo simple	188
Validar y ejecutar el ejemplo de flujo de trabajo simple	190
Desarrollar un flujo de trabajo complejo	191
Crear el ejemplo de flujo de trabajo complejo	193
Crear una acción personalizada para el ejemplo de flujo de trabajo completo	194
Crear el esquema del ejemplo de flujo de trabajo complejo	195
Crear zonas de ejemplo de flujo de trabajo complejo	197
Definir los parámetros del ejemplo de flujo de trabajo complejo	199
Definir los enlaces para el ejemplo de flujo de trabajo complejo	200
Establecer las propiedades de los atributos del ejemplo de flujo de trabajo complejo	211
Crear el diseño de los parámetros de entrada del ejemplo de flujo de trabajo complejo	212
Validar y ejecutar el ejemplo de flujo de trabajo complejo	213

## 2 Crear scripts 215

Elementos de Orchestrator que requieren creación de scripts	215
Limitaciones de la implementación de Rhino de Mozilla en Orchestrator	216
Uso de la API de creación de scripts de Orchestrator	217
Acceder al motor de creación de scripts desde el Editor de flujos de trabajo	218
Acceder al motor de creación de scripts desde el editor de acciones o de políticas	218
Acceder al Explorador de API de Orchestrator	219
Utilizar el Explorador de API de Orchestrator	219

Escribir scripts	221
Añadir parámetros a scripts	222
Acceder al sistema de archivos del servidor de Orchestrator desde JavaScript y flujos de trabajo	223
Acceder a las clases de Java desde JavaScript	224
Acceder a los comandos del sistema operativo desde JavaScript	224
Usar expresiones XPath con el complemento vCenter Server	224
Usar expresiones XPath con el complemento vCenter Server	225
Directrices para el control de excepciones	226
Ejemplos de JavaScript de Orchestrator	227
Ejemplos básicos de creación de scripts	228
Ejemplos de creación de scripts de correo electrónico	230
Ejemplos de creación de scripts del sistema de archivos	231
Ejemplos de creación de scripts de LDAP	232
Ejemplos de creación de scripts de registro	232
Ejemplos de creación de scripts de redes	232
Ejemplos de creación de scripts de flujos de trabajo	233
<b>3 Desarrollar acciones</b>	<b>235</b>
Reutilizar acciones	235
Acceder a la vista Acciones	236
Componentes de la vista Acciones	236
Crear acciones	236
Crear una acción	237
Buscar elementos que implementan una acción	238
Directrices de codificación de acciones	238
Historial de versiones de acciones	239
Restaurar acciones eliminadas	240
<b>4 Creación de elementos de recursos</b>	<b>241</b>
Visualizar un elemento de recursos	241
Importar un objeto externo para utilizar como elemento de recursos	242
Editar la información de los elementos de los recursos y los derechos de acceso	242
Guardar un elemento de recursos en un archivo	243
Actualizar un elemento de recursos	244
Añadir un elemento de recursos a un flujo de trabajo	244
<b>5 Crear paquetes</b>	<b>246</b>
Crear un paquete	246
Establecer permisos de usuario en un paquete	247
<b>6 Desarrollar complementos</b>	<b>249</b>

Descripción general de los complementos	249
Estructura de un complemento de Orchestrator	250
Exponer una API externa a Orchestrator	252
Componentes de un complemento	252
Función del archivo vso.xml	254
Funciones del adaptador de complemento	254
Funciones de la fábrica del complemento	255
Función de los objetos de buscador	256
Función de los objetos de creación de scripts	257
Función de los gestores de eventos	257
Contenido y estructura de un complemento	258
Definir la asignación de aplicaciones en el archivo vso.xml	259
Formato del archivo de definición del complemento vso.xml	260
Designar objetos de complemento	261
Convenciones de nomenclatura de objetos de complemento	262
Estructura de archivos del complemento	263
Referencia de API del complemento de Orchestrator	264
Interfaz IAop	264
Interfaz IDynamicFinder	265
Interfaz IPluginAdaptor	265
Interfaz IPluginEventPublisher	266
Interfaz IPluginFactory	267
Interfaz IPluginNotificationHandler	268
Interfaz IPluginPublisher	269
Interfaz WebConfigurationAdaptor	269
Clase PluginTrigger	270
Clase PluginWatcher	271
Clase QueryResult	271
Clase SDKFinderProperty	272
Clase PluginExecutionException	273
Clase PluginOperationException	274
Enumeración HasChildrenResult	274
Tipo de anotación ScriptingAttribute	275
Tipo de anotación ScriptingFunction	276
Tipo de anotación ScriptingParameter	276
Elementos del archivo de definición del complemento vso.xml	276
Elemento module	276
Elemento description	277
Elemento deprecated	278
Elemento url	278
Elemento installation	278

Elemento action	279
Elemento finder-datasources	279
Elemento finder-datasource	280
Elemento inventory	281
Elemento finders	281
Elemento finder	282
Elemento properties	283
Elemento property	283
Elemento relations	284
Elemento relation	284
Elemento id	285
Elemento inventory-children	285
Elemento relation-link	286
Elemento events	286
Elemento trigger	286
Elemento trigger-properties	287
Elemento trigger-property	287
Elemento gauge	287
Elemento scripting-objects	288
Elemento object	288
Elemento constructors	289
Elemento constructor	289
Elemento parameters del constructor	289
Elemento parameter del constructor	289
Elemento attributes	290
Elemento attribute	290
Elemento methods	291
Elemento method	291
Elemento de ejemplo	292
Elemento code	293
Elemento method parameters	293
Elemento method parameter	293
Elemento singleton	293
Elemento enumerations	294
Elemento enumeration	294
Elemento entries	295
Elemento entry	295
Recomendaciones para el desarrollo de complementos de Orchestrator	296
Enfoques para crear complementos de Orchestrator	296
Tipos de complementos de Orchestrator	298
Implementación de complementos	302



[Recomendaciones para el desarrollo de complementos de Orchestrator](#) 307

[Documentación de API y cadenas de interfaz de usuario para complementos](#) 310

## **7 Creación de complementos mediante Maven** 312

[Crear un complemento de Orchestrator con Maven a partir de un arquetipo](#) 312

[Arquetipos de Maven](#) 313

[Procedimientos recomendados para el desarrollo de complementos basados en Maven](#) 314

# Desarrollar con VMware vRealize Orchestrator

*Desarrollar con VMware vRealize Orchestrator* proporciona información e instrucciones para desarrollar flujos de trabajo y acciones personalizadas de VMware® vRealize Orchestrator.

Además, la documentación contiene información acerca de los elementos de Orchestrator que requieren creación de scripts e incluye ejemplos de JavaScript. *Desarrollar con VMware vRealize Orchestrator* también facilita instrucciones para crear recursos y paquetes.

## Público objetivo

Esta información está dirigida a desarrolladores que quieren crear acciones y flujos de trabajo personalizados de Orchestrator así como bloques de creación personalizados.

---

**Nota** Los procedimientos descritos en esta guía se basan en la interfaz de usuario del cliente heredado de vRealize Orchestrator.

---

# Desarrollar flujos de trabajo

Los flujos de trabajo se desarrollan en la interfaz del cliente de Orchestrator. El desarrollo de un flujo de trabajo requiere el uso del Editor de flujos de trabajo, el motor de creación de scripts integrado de JavaScript Rhino de Mozilla, así como las API de Orchestrator y de vCenter Server.

- **Conceptos clave de los flujos de trabajo**

Los flujos de trabajo están compuestos por un esquema, atributos y parámetros. El esquema de flujo de trabajo es el componente principal de un flujo de trabajo, ya que define todos los elementos del flujo de trabajo y las conexiones lógicas entre ellos. Los parámetros y los atributos de flujo de trabajo son las variables que los flujos de trabajo utilizan para transferir datos. Orchestrator guarda un token de flujo de trabajo cada vez que se ejecuta un flujo de trabajo y registra los detalles de dicha ejecución del flujo de trabajo.

- **Fases del proceso de desarrollo de flujos de trabajo**

El proceso para desarrollar un flujo de trabajo implica una serie de fases. Puede seguir una secuencia de fases distinta u omitir una fase, según el tipo de flujo de trabajo que esté desarrollando. Por ejemplo, puede crear un flujo de trabajo sin la creación de scripts personalizada.

- **Recomendaciones para desarrollar flujos de trabajo**

VMware recomienda varias prácticas para el desarrollo de flujos de trabajo de Orchestrator a cargo de varios usuarios y en un entorno de clúster.

- **Derechos de acceso para el cliente de Orchestrator**

De forma predeterminada, los miembros del grupo LDAP del administrador de Orchestrator son los únicos que pueden acceder al cliente de Orchestrator.

- **Probar flujos de trabajo durante el desarrollo**

Los flujos de trabajo se pueden probar en cualquier momento de su desarrollo, incluso si el flujo de trabajo todavía no se ha completado o incluido en un elemento Fin.

- **Crear y editar un flujo de trabajo**

Cree flujos de trabajo en el cliente de Orchestrator y modifíquelos en el Editor de flujos de trabajo. El Editor de flujos de trabajo es el IDE del cliente de Orchestrator para desarrollar flujos de trabajo.

- **Proporcionar información de flujo de trabajo general**

En la pestaña **General** del Editor de flujos de trabajo, se proporciona el nombre y la descripción del flujo de trabajo, se definen los atributos y determinados aspectos del comportamiento del flujo de trabajo, se establece el número de versión, se comprueba la firma y se establecen los permisos de usuario.

- **Definir atributos y parámetros**

Después de crear un flujo de trabajo, debe definir los atributos globales, los parámetros de entrada y de salida del flujo de trabajo.

- **Esquema de flujo de trabajo**

Un esquema de un flujo de trabajo es una representación gráfica del flujo de trabajo que muestra el flujo de trabajo como un diagrama de flujo con elementos del flujo de trabajo interconectados. El esquema de un flujo de trabajo es el flujo lógico de ese flujo de trabajo.

- **Desarrollar complementos**

Orchestrator permite la integración con soluciones de administración mediante su arquitectura de complementos abierta. Puede utilizar el cliente de Orchestrator para ejecutar y crear flujos de trabajo de complementos y acceder a la API de complementos.

- **Obtener parámetros de entrada de los usuarios cuando se inicia un flujo de trabajo**

Si un flujo de trabajo precisa de parámetros de entrada, se abre un cuadro de diálogo en el que los usuarios proporcionan los valores de parámetros de entrada necesarios cuando se ejecuta. El contenido y el diseño, o la presentación, de este cuadro de diálogo se pueden organizar en la pestaña **Presentación** en el Editor de flujos de trabajo.

- **(opcional) Interacciones de usuario requeridas durante la ejecución de un flujo de trabajo**

En ocasiones, un flujo de trabajo puede requerir parámetros de entrada adicionales de una fuente externa durante la ejecución. Estos parámetros de entrada pueden proceder de otra aplicación u otro flujo de trabajo, o los puede proporcionar el usuario directamente.

- **Llamar a flujos de trabajo desde flujos de trabajo**

Los flujos de trabajo pueden llamar a otros flujos de trabajo durante su ejecución. Un flujo de trabajo puede iniciar otro flujo de trabajo porque requiere el resultado del otro flujo de trabajo como parámetro de entrada para su propia ejecución, o bien puede iniciar un flujo de trabajo y dejar que prosiga su proceso de ejecución de forma independiente. Los flujos de trabajo también pueden iniciar un flujo de trabajo en un momento dado del futuro o bien iniciar varios flujos de trabajo de manera simultánea.

- **Ejecutar un flujo de trabajo en una selección de objetos**

Puede automatizar las tareas repetitivas ejecutando un flujo de trabajo en una selección de objetos. Por ejemplo, puede crear un flujo de trabajo que tome una instantánea de todas las máquinas virtuales en una carpeta de máquinas virtuales, o bien puede crear un flujo de trabajo que desactive todas las máquinas virtuales de un host concreto.

- **Desarrollar flujos de trabajo de larga ejecución**

Un flujo de trabajo en espera consume recursos del sistema porque de manera continuada sondea el objeto del que requiere una respuesta. Si sabe que un flujo de trabajo puede terminar esperando un intervalo de tiempo prolongado antes de recibir la respuesta que necesita, puede añadir elementos de flujos de trabajo de larga ejecución al flujo de trabajo.

- **Elementos de configuración**

Un elemento de configuración es una lista de atributos que puede utilizar para configurar constantes en toda una implementación del servidor de Orchestrator.

- **Permisos de usuario de un flujo de trabajo**

Orchestrator define niveles de permisos que puede aplicar a grupos para permitirles o prohibirles el acceso a los flujos de trabajo.

- **Validar flujos de trabajo**

Orchestrator proporciona una herramienta para la validación de flujos de trabajo. Al validar un flujo de trabajo, se pueden identificar errores en el flujo de trabajo y comprobar que los datos fluyan correctamente de un elemento al siguiente.

- **Depurar flujos de trabajo**

Orchestrator proporciona una herramienta para la depuración de flujos de trabajo. Puede depurar un flujo de trabajo para inspeccionar los parámetros de entrada y de salida, así como los atributos al comienzo de cualquier actividad, reemplazar valores de parámetros o de atributos durante la ejecución de un flujo de trabajo en modo de edición, y reanudar un flujo de trabajo a partir de la última actividad que no se pudo realizar.

- **Flujos de trabajo en ejecución**

Un flujo de trabajo de Orchestrator se ejecuta de acuerdo con un flujo lógico de eventos.

- **Reanudar la ejecución de un flujo de trabajo fallido**

Si falla un flujo de trabajo, Orchestrator proporciona una opción para reanudar la ejecución del flujo de trabajo a partir de la última actividad fallida.

- **Generar documentación de flujo de trabajo**

Puede exportar documentación en formato PDF acerca de un flujo de trabajo o una carpeta de flujos de trabajo que seleccione en cualquier momento.

- **Historial de versiones de flujos de trabajo**

Puede utilizar el historial de versiones para revertir un flujo de trabajo a un estado guardado previamente. Puede revertir el estado del flujo de trabajo a una versión anterior o una posterior. También puede comparar las diferencias entre el estado actual del flujo de trabajo y una versión guardada de este.

- **Restaurar flujos de trabajo eliminados**

Puede restaurar flujos de trabajo que se han eliminado de la biblioteca.

- **Desarrollar un flujo de trabajo simple de ejemplo**

El desarrollo de un flujo de trabajo simple de ejemplo demuestra los pasos más habituales del proceso de desarrollo de flujos de trabajo.

- **Desarrollar un flujo de trabajo complejo**

El desarrollo de un flujo de trabajo complejo de ejemplo demuestra los pasos más habituales del proceso de desarrollo de flujos de trabajo, así como opciones más avanzadas, como la creación de decisiones personalizadas y de bucles.

## Conceptos clave de los flujos de trabajo

Los flujos de trabajo están compuestos por un esquema, atributos y parámetros. El esquema de flujo de trabajo es el componente principal de un flujo de trabajo, ya que define todos los elementos del flujo de trabajo y las conexiones lógicas entre ellos. Los parámetros y los atributos de flujo de trabajo son las variables que los flujos de trabajo utilizan para transferir datos. Orchestrator guarda un token de flujo de trabajo cada vez que se ejecuta un flujo de trabajo y registra los detalles de dicha ejecución del flujo de trabajo.

### Parámetros de flujo de trabajo

Los flujos de trabajo reciben parámetros de entrada y generan parámetros de salida cuando se ejecutan.

#### Parámetros de entrada

La mayoría de los flujos de trabajo requieren un determinado conjunto de parámetros de entrada para ejecutarse. Un parámetro de entrada es un argumento que el flujo de trabajo procesa cuando se inicia. El usuario, la aplicación, otro flujo de trabajo o una acción pasan los parámetros de entrada a otro flujo de trabajo para que los procesen cuando se inicie.

Por ejemplo, si un flujo de trabajo restablece una máquina virtual, el flujo de trabajo requiere el nombre de la máquina virtual como parámetro de entrada.

#### Parámetros de salida

Los parámetros de salida de un flujo de trabajo representan el resultado de la ejecución del flujo de trabajo. Los parámetros de salida pueden cambiar cuando se ejecuta un flujo de trabajo o un elemento del flujo de trabajo. Mientras los flujos de trabajo se están ejecutando, pueden recibir los parámetros de salida de otros flujos de trabajo como parámetros de entrada.

Por ejemplo, si un flujo de trabajo crea una instantánea de una máquina virtual, el parámetro de salida para el flujo de trabajo es la instantánea resultante.

### Atributos de los flujos de trabajo

Los elementos de los flujos de trabajo procesan datos que reciben como parámetro de entrada y definen los datos resultantes como atributos de flujos de trabajo o parámetros de salida.

Los atributos de solo lectura de los flujos de trabajo actúan como constantes globales de un flujo de trabajo. Los atributos de escritura actúan como variables globales de un flujo de trabajo.

Puede utilizar atributos para transferir datos entre los elementos de un flujo de trabajo. Puede obtener atributos de las siguientes maneras:

- Definiendo atributos cuando crea un flujo de trabajo
- Estableciendo el parámetro de salida de un atributo del flujo de trabajo como atributo del flujo de trabajo
- Heredando atributos de un elemento de configuración

## Esquema de flujo de trabajo

Un esquema de un flujo de trabajo es una representación gráfica que muestra el flujo de trabajo como un diagrama de flujo con elementos del flujo de trabajo interconectados. El esquema de un flujo de trabajo es el elemento más importante del flujo de trabajo, ya que determina su lógica.

## Presentación del flujo de trabajo

Cuando los usuarios ejecutan un flujo de trabajo, proporcionan los valores para los parámetros de entrada del flujo de trabajo en la presentación del flujo de trabajo. Cuando organice la presentación del flujo de trabajo, tenga en cuenta el tipo y el número de parámetros de entrada del flujo de trabajo.

## Tokens de flujo de trabajo

Un token de flujo de trabajo representa un flujo de trabajo que está en ejecución o que se ha ejecutado.

Un flujo de trabajo es una descripción abstracta de un proceso que define una secuencia genérica de pasos y un conjunto genérico de los parámetros de entrada necesarios. Cuando ejecuta un flujo de trabajo con un conjunto de parámetros de entrada reales, recibe una instancia de este flujo de trabajo abstracto que se comporta de acuerdo con los parámetros de entrada específicos asignados. Esta instancia concreta de un flujo de trabajo en ejecución o completado se denomina token de flujo de trabajo.

## Atributos de token de flujo de trabajo

Los atributos de token de flujo de trabajo son los parámetros específicos con los que se ejecuta un token de flujo de trabajo. Los atributos de token de flujo de trabajo son una agregación de los atributos globales del flujo de trabajo y los parámetros de salida específicos con los que ha ejecutado el token de flujo de trabajo.

## Fases del proceso de desarrollo de flujos de trabajo

El proceso para desarrollar un flujo de trabajo implica una serie de fases. Puede seguir una secuencia de fases distinta u omitir una fase, según el tipo de flujo de trabajo que esté desarrollando. Por ejemplo, puede crear un flujo de trabajo sin la creación de scripts personalizada.

En general, un flujo de trabajo se desarrolla a través de las fases siguientes.

- 1 Crear un flujo de trabajo nuevo o un duplicado de uno que ya exista en la biblioteca estándar.
- 2 Proporcionar información general acerca del flujo de trabajo.
- 3 Definir los parámetros de entrada del flujo de trabajo.
- 4 Diseñar y vincular el esquema de flujo de trabajo para determinar el flujo lógico del flujo de trabajo.
- 5 Enlazar los parámetros de entrada y de salida de cada elemento de esquema con los atributos de flujo de trabajo.
- 6 Escribir los scripts necesarios para los elementos de tarea de scripts o los elementos de decisión personalizada.

- 7 Crear la presentación del flujo de trabajo para definir el diseño del cuadro de diálogo de parámetros de entrada que ven los usuarios al ejecutar el flujo de trabajo.
- 8 Validar el flujo de trabajo.

## Recomendaciones para desarrollar flujos de trabajo

VMware recomienda varias prácticas para el desarrollo de flujos de trabajo de Orchestrator a cargo de varios usuarios y en un entorno de clúster.

- Cada desarrollador posee una instancia de Orchestrator independiente de prueba específica para crear y desarrollar flujos de trabajo.
- Los flujos de trabajo se guardan como proyectos de Maven en un sistema de control de código fuente compartido.
- Para garantizar un rendimiento óptimo de la implementación de producción de Orchestrator, la mejor opción es importar los flujos de trabajo en un intervalo de tiempo programado.
- Cuando importe los flujos de trabajo en un clúster de Orchestrator, conecte el cliente de Orchestrator a uno de los nodos mediante su nombre de host local o la dirección IP, en lugar de la dirección del servidor virtual del equilibrador de carga.

---

**Nota** Las modificaciones de un flujo de trabajo se aplican la siguiente vez que se ejecuta el flujo de trabajo.

---

## Derechos de acceso para el cliente de Orchestrator

De forma predeterminada, los miembros del grupo LDAP del administrador de Orchestrator son los únicos que pueden acceder al cliente de Orchestrator.

El administrador de Orchestrator puede otorgar acceso al cliente de Orchestrator a otros grupos de usuarios estableciendo como mínimo el permiso **Ver**.

Para permitirle el acceso al cliente de Orchestrator, el administrador debe añadirle al grupo LDAP del administrador de Orchestrator, o bien establecer los permisos **Ver**, **Inspeccionar**, **Editar**, **Ejecutar** o **Administrar** para un grupo del cual sea miembro.

## Probar flujos de trabajo durante el desarrollo

Los flujos de trabajo se pueden probar en cualquier momento de su desarrollo, incluso si el flujo de trabajo todavía no se ha completado o incluido en un elemento Fin.

De forma predeterminada, Orchestrator comprueba que un flujo de trabajo sea válido antes de que un usuario lo pueda ejecutar. La validación automática se puede desactivar durante el desarrollo de los flujos de trabajo para ejecutar flujos de trabajo parciales a modo de prueba.

---

**Nota** Tras finalizar el desarrollo del flujo de trabajo es importante volver a activar la validación automática.

---



## Procedimiento

- 1 En el menú del cliente de Orchestrator, haga clic en **Herramientas > Preferencias del usuario**.
- 2 Haga clic en la pestaña **Flujos de trabajo**.
- 3 Anule la selección de la casilla de verificación **Validar un flujo de trabajo antes de ejecutarlo**.

Ha desactivado la validación automática de flujos de trabajo.

## Crear y editar un flujo de trabajo

Cree flujos de trabajo en el cliente de Orchestrator y modifíquelos en el Editor de flujos de trabajo. El Editor de flujos de trabajo es el IDE del cliente de Orchestrator para desarrollar flujos de trabajo.

El Editor de flujos de trabajo se abre al modificar un flujo de trabajo.

### ■ [Crear un flujo de trabajo](#)

Puede crear flujos de trabajo en la lista jerárquica de flujos de trabajo del cliente de Orchestrator.

### ■ [Editar un flujo de trabajo](#)

Edite un flujo de trabajo para efectuar cambios en él o para desarrollar uno nuevo que esté vacío.

### ■ [Editar un flujo de trabajo de la biblioteca estándar](#)

Orchestrator proporciona una biblioteca estándar de flujos de trabajo que puede usar para automatizar las operaciones de la infraestructura virtual. Los flujos de trabajo de la biblioteca estándar están bloqueados como solo lectura.

### ■ [Pestañas del editor de flujos de trabajo](#)

El editor de flujos de trabajo está estructurado en pestañas en las que puede editar los componentes de los flujos de trabajo.

## Crear un flujo de trabajo

Puede crear flujos de trabajo en la lista jerárquica de flujos de trabajo del cliente de Orchestrator.

## Procedimiento

- 1 Seleccione **Diseño** en el menú desplegable del cliente de Orchestrator.
- 2 Haga clic en la vista **Flujos de trabajo**.
- 3 (opcional) Haga clic con el botón secundario en la raíz de la lista jerárquica de flujos de trabajo o en una carpeta de la lista, y seleccione **Añadir carpeta** para crear una carpeta de flujos de trabajo.
- 4 (opcional) Escriba el nombre de la nueva carpeta.
- 5 Haga clic con el botón secundario en la nueva carpeta o en una carpeta que ya exista, y seleccione **Nuevo flujo de trabajo**.
- 6 Asigne un nombre al nuevo flujo de trabajo y haga clic en **Aceptar**.

Se crea un nuevo flujo de trabajo vacío en la carpeta que se seleccione.

## Pasos siguientes

El flujo de trabajo puede editarse.

## Editar un flujo de trabajo

Edite un flujo de trabajo para efectuar cambios en él o para desarrollar uno nuevo que esté vacío.

### Procedimiento

- 1 Seleccione **Diseño** en el menú desplegable del cliente de Orchestrator.
- 2 Haga clic en la vista **Flujos de trabajo**.
- 3 Expanda la lista jerárquica de flujos de trabajo para acceder al flujo de trabajo que desea editar.
- 4 Para abrir el flujo de trabajo y editarlo, haga clic con el botón secundario en el flujo de trabajo y seleccione **Editar**.

Se abre el Editor de flujos de trabajo para poder editar.

## Editar un flujo de trabajo de la biblioteca estándar

Orchestrator proporciona una biblioteca estándar de flujos de trabajo que puede usar para automatizar las operaciones de la infraestructura virtual. Los flujos de trabajo de la biblioteca estándar están bloqueados como solo lectura.

Para editar un flujo de trabajo de la biblioteca estándar debe crear un duplicado de ese flujo de trabajo. Puede editar flujos de trabajo duplicados o flujos de trabajo personalizados.

### Procedimiento

- 1 Seleccione **Diseño** en el menú desplegable del cliente de Orchestrator.
- 2 Haga clic en la vista **Flujos de trabajo**.
- 3 (opcional) Haga clic con el botón derecho en la raíz de la lista jerárquica de carpetas de flujos de trabajo y seleccione **Nueva carpeta** para crear una carpeta que contenga el flujo de trabajo que se va a editar.
- 4 Expanda la lista jerárquica **Biblioteca** de los flujos de trabajo estándar para navegar hasta el flujo de trabajo que se va a editar.
- 5 Haga clic con el botón derecho en el flujo de trabajo que se va a editar.  
La opción **Editar** está atenuada. El flujo de trabajo es de solo lectura.
- 6 Haga clic con el botón derecho en el flujo de trabajo y seleccione **Flujo de trabajo duplicado**.
- 7 Especifique un nombre para el flujo de trabajo duplicado.

De manera predeterminada, Orchestrator nombra el flujo de trabajo duplicado como *Copia de nombre\_flujo\_trabajo*.

- 8 Haga clic en el valor **Carpeta de flujo de trabajo** para buscar una carpeta en la que guardar el flujo de trabajo duplicado.

Seleccione la carpeta que haya creado en [Paso 3](#). Si no creó ninguna carpeta, seleccione una carpeta que no se encuentre en la biblioteca de flujos de trabajo estándar.

- 9 Haga clic en **Sí** o **No** para copiar el historial de las versiones del flujo de trabajo en el duplicado.

Opción	Descripción
<b>Sí</b>	El historial de las versiones del flujo de trabajo original se replica en el duplicado.
<b>No</b>	La versión del duplicado se revierte a 0.0.0.

- 10 Haga clic en **Duplicar** para duplicar el flujo de trabajo.

- 11 Haga clic con el botón derecho en el flujo de trabajo duplicado y seleccione **Editar**.

El Editor de flujos de trabajo se abre. Ahora puede editar el flujo de trabajo duplicado.

Así es como se duplica un flujo de trabajo de la biblioteca estándar. Ahora puede editar el flujo de trabajo duplicado.

## Pestañas del editor de flujos de trabajo

El editor de flujos de trabajo está estructurado en pestañas en las que puede editar los componentes de los flujos de trabajo.

**Tabla 1-1. Pestañas del editor de flujos de trabajo**

Pestaña	Descripción
<b>General</b>	Edite el nombre del flujo de trabajo, proporcione una descripción de lo que hace el flujo de trabajo, indique el número de versión, compruebe los permisos de los usuarios, defina el comportamiento del flujo de trabajo si el servidor de Orchestrator se reinicia y defina los atributos globales del flujo de trabajo.
<b>Entradas</b>	Defina los parámetros que requiere el flujo de trabajo cuando se ejecuta. Estos parámetros de entrada son los datos que procesa el flujo de trabajo. El comportamiento del flujo de trabajo cambia en función de estos parámetros.
<b>Salidas</b>	Defina los valores que el flujo de trabajo genera cuando completa su ejecución. Otros flujos de trabajo u otras acciones pueden utilizar estos valores cuando se ejecutan.
<b>Esquema</b>	Cree el flujo de trabajo. Cree flujos de trabajo arrastrando elementos de esquema de flujos de trabajo desde la paleta de flujos de trabajo en la parte izquierda de la pestaña <b>Esquema</b> . Al hacer clic en un elemento del diagrama del esquema, puede definir y editar el comportamiento del elemento en la parte inferior de la pestaña <b>Esquema</b> .

Tabla 1-1. Pestañas del editor de flujos de trabajo (continuación)

Pestaña	Descripción
<b>Presentación</b>	Defina el diseño del cuadro de diálogo de entrada del usuario que aparece cuando los usuarios ejecutan un flujo de trabajo. Ordene los parámetros y los atributos en pasos y grupos de presentación para facilitar la identificación de los parámetros en el cuadro de diálogo de parámetros de entrada. También puede definir las restricciones sobre los parámetros de entrada que los usuarios pueden proporcionar en la presentación ajustando las propiedades de los parámetros.
<b>Referencias de los parámetros</b>	Compruebe qué elementos del flujo de trabajo consumen los atributos y los parámetros del flujo lógico del flujo de trabajo. Esta pestaña también le muestra las restricciones sobre estos parámetros y atributos que ya definió en la pestaña <b>Presentación</b> .
<b>Tokens de flujos de trabajo</b>	Compruebe los detalles de cada ejecución del flujo de trabajo. Esta información incluye el estado del flujo de trabajo, el usuario que lo ejecutó, el estado empresarial del elemento actual, así como la fecha y la hora en que se inició y finalizó el flujo de trabajo.
<b>Eventos</b>	Acceda a información sobre cada evento concreto que se produce cuando se ejecuta el flujo de trabajo. Esta información incluye una descripción del evento, el usuario que lo inició, el tipo y el origen del evento, así como la fecha y la hora en que ocurrió.
<b>Permisos</b>	Defina los permisos para interactuar con el flujo de trabajo para usuarios o grupos de usuarios.

## Proporcionar información de flujo de trabajo general

En la pestaña **General** del Editor de flujos de trabajo, se proporciona el nombre y la descripción del flujo de trabajo, se definen los atributos y determinados aspectos del comportamiento del flujo de trabajo, se establece el número de versión, se comprueba la firma y se establecen los permisos de usuario.

### Requisitos previos

Abra un flujo de trabajo para editar en el Editor de flujos de trabajo.

### Procedimiento

- 1 Haga clic en la pestaña **General** en el Editor de flujos de trabajo.
- 2 Haga clic en los dígitos de **Versión** para establecer un número de versión para el flujo de trabajo.  
Se abre el cuadro de diálogo **Comentario de versión**.
- 3 Escriba un comentario para esta versión del flujo de trabajo y haga clic en **Aceptar**.  
Por ejemplo, escriba **Creación inicial** si acaba de crear el flujo de trabajo.

Se crea una nueva versión del flujo de trabajo. Más adelante puede revertir el estado del flujo de trabajo a esta versión.

- 4 Defina el comportamiento del flujo de trabajo si el servidor de Orchestrator se reinicia estableciendo el valor **Comportamiento de reinicio del servidor**.

- Deje el valor predeterminado de **Reanudar ejecución de flujo de trabajo** para que el flujo de trabajo se reanude en el punto en que se interrumpió cuando se detuvo el servidor.
- Haga clic en **Reanudar ejecución de flujo de trabajo** y seleccione **No reanudar ejecución de flujo de trabajo (establecer como ERRÓNEA)** para evitar que el flujo de trabajo se reinicie si se reinicia el servidor de Orchestrator.

Impida que el flujo de trabajo se reinicie si este depende del entorno en el que se ejecuta. Por ejemplo, si un flujo de trabajo requiere una instancia determinada de vCenter Server y se vuelve a configurar Orchestrator para que se conecte a otra instancia de vCenter Server, al reiniciar el flujo de trabajo después de reiniciar el servidor de Orchestrator, falla el flujo de trabajo.

- 5 Escriba una descripción detallada del flujo de trabajo en el cuadro de texto **Descripción**.

- 6 Haga clic en **Guardar** en la parte inferior del Editor de flujos de trabajo.

Un mensaje verde en la parte inferior izquierda del Editor de flujos de trabajo confirma que se han guardado los cambios.

Ha definido aspectos del comportamiento del flujo de trabajo, establecido la versión y definido las operaciones que pueden llevar a cabo los usuarios en el flujo de trabajo.

### Pasos siguientes

Debe definir los atributos y los parámetros del flujo de trabajo.

## Definir atributos y parámetros

Después de crear un flujo de trabajo, debe definir los atributos globales, los parámetros de entrada y de salida del flujo de trabajo.

Los atributos de flujo de trabajo almacenan los datos que el flujo de trabajo procesa internamente. Los parámetros de entrada del flujo de trabajo son datos proporcionados por una fuente externa, por ejemplo un usuario u otro flujo de trabajo. Los parámetros de salida del flujo de trabajo son datos que el flujo de trabajo entrega al término de su ejecución.

- [Definir parámetros de flujo de trabajo](#)

Puede usar parámetros de entrada y de salida para pasar del flujo de trabajo y al flujo de trabajo.

- [Definir atributos de flujo de trabajo](#)

Los atributos de flujo de trabajo son los datos que procesa el flujo de trabajo.

## ■ Restricciones de nomenclatura de atributos y parámetros

Puede utilizar expresiones de OGNL para determinar los parámetros de entrada dinámicamente cuando se ejecuta un flujo de trabajo. El analizador de OGNL de Orchestrator utiliza determinadas palabras clave durante el procesamiento de OGNL que no puede utilizar en los nombres de atributos o en los parámetros del flujo de trabajo.

## Definir parámetros de flujo de trabajo

Puede usar parámetros de entrada y de salida para pasar del flujo de trabajo y al flujo de trabajo.

Los parámetros de un flujo de trabajo se pueden definir en el Editor de flujos de trabajo. Los parámetros de entrada son los datos iniciales necesarios para la ejecución del flujo de trabajo. Los usuarios proporcionan los valores de los parámetros de entrada cuando ejecutan el flujo de trabajo. Los parámetros de salida son los datos devueltos por el flujo de trabajo al término de su ejecución.

### Requisitos previos

Abra un flujo de trabajo para editar en el Editor de flujos de trabajo.

### Procedimiento

- 1 Haga clic en la pestaña correspondiente en el Editor de flujos de trabajo.
  - Haga clic en **Entradas** para crear parámetros de entrada.
  - Haga clic en **Salidas** para crear parámetros de salida.
- 2 Haga clic con el botón secundario dentro de la pestaña Parámetros y seleccione **Añadir parámetro**.
- 3 Haga clic en el nombre del parámetro para cambiárselo.
 

El nombre predeterminado es `arg_in_X` para parámetros de entrada y `arg_out_X` para parámetros de salida, donde X es un número.
- 4 (opcional) Para cambiar el valor del tipo de parámetro, haga clic en el valor y seleccione uno en la lista de valores.
 

Cadena es el tipo de valor predeterminado del parámetro.
- 5 En el cuadro de texto **Descripción**, escriba una descripción del parámetro.
- 6 (opcional) Si decide que el parámetro debe ser un atributo en lugar de un parámetro, haga clic con el botón secundario en el parámetro y seleccione **Mover como atributo** para cambiar el parámetro a un atributo.

Ha definido un parámetro de entrada o de salida del flujo de trabajo.

### Pasos siguientes

Tras definir los parámetros del flujo de trabajo, cree el esquema de flujo de trabajo.

## Definir atributos de flujo de trabajo

Los atributos de flujo de trabajo son los datos que procesa el flujo de trabajo.

---

**Nota** También puede definir atributos de flujo de trabajo en los elementos del esquema de flujo de trabajo cuando crea el esquema de flujo de trabajo. A menudo, resulta más fácil definir un atributo cuando se crea el elemento del esquema de flujo de trabajo que lo procesa.

---

### Requisitos previos

Abra un flujo de trabajo para editar en el Editor de flujos de trabajo.

### Procedimiento

- 1 Haga clic en la pestaña **General** en el Editor de flujos de trabajo.

El panel Atributos aparece en la mitad inferior de la pestaña **General**.

- 2 Haga clic con el botón secundario en el panel Atributos y seleccione **Añadir atributo**.

En la lista de atributos, aparece un atributo nuevo cuyo tipo predeterminado es Cadena.

- 3 Haga clic en el atributo para cambiarle el nombre.

El nombre predeterminado es attX, donde X es un número.

---

**Nota** Los atributos de flujo de trabajo no deben tener el mismo nombre que ninguno de los parámetros del flujo de trabajo.

---

- 4 Haga clic en el tipo de atributo para seleccionar un tipo nuevo en una lista de valores.

El tipo de atributo predeterminado es Cadena.

- 5 Haga clic en el valor del atributo para establecer o seleccionar un valor conforme al tipo de atributo.

- 6 En el cuadro de texto **Descripción**, escriba una descripción del atributo.

- 7 Si el atributo es una constante en lugar de una variable, seleccione la casilla de verificación a la izquierda del atributo para que su valor sea de solo lectura.

El icono de candado identifica la columna de casillas de verificación de solo lectura.

- 8 (opcional) Si decide que el atributo debe ser un parámetro de entrada o de salida en lugar de un atributo, haga clic con el botón secundario en el atributo y seleccione **Mover como parámetro de entrada/Mover como parámetro de salida** para cambiar el atributo a un parámetro.

Ha definido un atributo del flujo de trabajo.

### Pasos siguientes

Puede definir los parámetros de entrada y salida del flujo de trabajo.

## Restricciones de nomenclatura de atributos y parámetros

Puede utilizar expresiones de OGNL para determinar los parámetros de entrada dinámicamente cuando se ejecuta un flujo de trabajo. El analizador de OGNL de Orchestrator utiliza determinadas palabras clave

durante el procesamiento de OGNL que no puede utilizar en los nombres de atributos o en los parámetros del flujo de trabajo.

El uso de una palabra clave de OGNL reservada como prefijo de un nombre de atributo no interrumpe el procesamiento de OGNL. Por ejemplo, puede asignar el nombre `trueParameter` a un parámetro. Las palabras clave reservadas no distinguen entre mayúsculas y minúsculas.

No puede utilizar las palabras clave siguientes en nombres de atributos y parámetros de flujos de trabajo.

**Tabla 1-2. Palabras clave prohibidas en nombres de atributos y parámetros**

Palabra clave prohibida	Palabra clave prohibida	Palabra clave prohibida
■ <code>abstract</code>	■ <code>eof</code>	■ <code>_memberAccess</code>
■ <code>back_char_esc</code>	■ <code>esc</code>	■ <code>native</code>
■ <code>back_char_literal</code>	■ <code>exponent</code>	■ <code>package</code>
■ <code>boolean</code>	■ <code>export</code>	■ <code>private</code>
■ <code>byte</code>	■ <code>extends</code>	■ <code>public</code>
■ <code>char</code>	■ <code>false</code>	■ <code>root</code>
■ <code>char_literal</code>	■ <code>final</code>	■ <code>short</code>
■ <code>class</code>	■ <code>flt_literal</code>	■ <code>static</code>
■ <code>_classResolver</code>	■ <code>flt_suff</code>	■ <code>string_esc</code>
■ <code>const</code>	■ <code>ident</code>	■ <code>string_literal</code>
■ <code>context</code>	■ <code>implements</code>	■ <code>synchronized</code>
■ <code>debugger</code>	■ <code>import</code>	■ <code>this</code>
■ <code>dec_digits</code>	■ <code>in</code>	■ <code>_traceEvaluations</code>
■ <code>dec_flt</code>	■ <code>int</code>	■ <code>true</code>
■ <code>default</code>	■ <code>int_literal</code>	■ <code>_typeConverter</code>
■ <code>delete</code>	■ <code>interface</code>	■ <code>volatil</code>
■ <code>digit</code>	■ <code>_keepLastEvaluation</code>	■ <code>with</code>
■ <code>double</code>	■ <code>_lastEvaluation</code>	■ <code>WithinBackCharLiteral</code>
■ <code>dynamic_subscript</code>	■ <code>letter</code>	■ <code>WithinCharLiteral</code>
■ <code>enum</code>	■ <code>long</code>	■ <code>WithinStringLiteral</code>

## Esquema de flujo de trabajo

Un esquema de un flujo de trabajo es una representación gráfica del flujo de trabajo que muestra el flujo de trabajo como un diagrama de flujo con elementos del flujo de trabajo interconectados. El esquema de un flujo de trabajo es el flujo lógico de ese flujo de trabajo.

### ■ [Ver el esquema de los flujos de trabajo](#)

Puede ver el esquema de un flujo de trabajo en la pestaña **Esquema** de este flujo de trabajo en el cliente de Orchestrator.

### ■ [Crear un flujo de trabajo en el esquema del flujo de trabajo](#)

Los esquemas de flujo de trabajo se componen de una secuencia de elementos de esquema. Los elementos de esquema de flujo de trabajo son los bloques de creación del flujo de trabajo; pueden representar decisiones, tareas con scripts, acciones, controlares de excepciones o, incluso, otros flujos de trabajo.



- **Elementos de esquema**

El Editor de flujos de trabajo presenta los elementos de esquema de flujo de trabajo en menús en la pestaña **Esquema**. Los elementos de esquema de la pestaña **Esquema** se pueden utilizar para crear un flujo de trabajo.

- **Propiedades de elementos de esquema**

Los elementos de esquema tienen propiedades que puede definir y editar en la pestaña **Esquema** de la paleta de flujos de trabajo.

- **Enlaces y vínculos**

Los vínculos entre elementos determinan el flujo lógico del flujo de trabajo. Los enlaces rellenan los elementos con datos de otros elementos enlazando parámetros de entrada y de salida a los atributos de flujos de trabajo.

- **Decisiones**

Los flujos de trabajo pueden implementar funciones de decisión que definen diferentes tipos de acciones conforme a una instrucción booleana `true` o `false`.

- **Control de excepciones**

El control de excepciones detecta los errores que se producen durante la ejecución de un elemento de esquema. El control de excepciones define el modo en el que se comporta el elemento de esquema cuando se produce el error.

- **Utilizar gestores de errores**

Puede utilizar un gestor de errores estándar para definir el comportamiento si se produce un error en un determinado elemento de esquema de flujo de trabajo. Puede utilizar un gestor de errores global para definir el comportamiento si se producen errores no detectados por los gestores de errores estándar.

- **Elementos Foreach y tipos compuestos**

Puede insertar un elemento Foreach en el flujo de trabajo que desarrolle para ejecutar un flujo de trabajo secundario que se itere sobre las matrices de parámetros o atributos. Para mejorar la comprensión y la legibilidad del flujo de trabajo, puede agrupar varios parámetros de flujo de trabajo de diferentes tipos que estén conectados lógicamente en un solo tipo conocido como tipo compuesto.

- **Añadir actividad de switch a un flujo de trabajo**

Puede añadir actividad de switch básica a un esquema de flujo de trabajo que defina los casos de switch según los parámetros o los atributos del flujo de trabajo.

## Ver el esquema de los flujos de trabajo

Puede ver el esquema de un flujo de trabajo en la pestaña **Esquema** de este flujo de trabajo en el cliente de Orchestrator.

### Procedimiento

- 1 Seleccione **Diseño** en el menú desplegable del cliente de Orchestrator.

2 Vaya a un flujo de trabajo en la lista jerárquica de flujos de trabajo.

3 Haga clic en el flujo de trabajo.

Aparecerá la información de ese flujo de trabajo en el panel de la derecha.

4 Seleccione la pestaña **Esquema** en el panel de la derecha.

Verá la representación gráfica del flujo de trabajo.

## Crear un flujo de trabajo en el esquema del flujo de trabajo

Los esquemas de flujo de trabajo se componen de una secuencia de elementos de esquema. Los elementos de esquema de flujo de trabajo son los bloques de creación del flujo de trabajo; pueden representar decisiones, tareas con scripts, acciones, controlares de excepciones o, incluso, otros flujos de trabajo.

Genere flujos de trabajo en el Editor de flujos de trabajo arrastrando elementos de esquema de la paleta de flujos de trabajo en la parte izquierda del Editor de flujos de trabajo al diagrama del esquema de flujo de trabajo.

## Editar un esquema de flujo de trabajo

Un flujo de trabajo se genera creando una secuencia de elementos de esquema que definen el flujo lógico del flujo de trabajo.

De forma predeterminada, todos los elementos de esquema de flujo de trabajo están vinculados. Los vínculos entre los elementos se representan como flechas. Al añadir un elemento nuevo al esquema de flujo de trabajo, debe arrastrarlo a una flecha o a un elemento de flujo de trabajo que no esté vinculado a un próximo elemento. Después de añadir elementos de flujo de trabajo al esquema, puede eliminar los vínculos creados y crear otros para definir el flujo lógico del flujo de trabajo.

Puede copiar un elemento o una selección de elementos del esquema de un flujo de trabajo existente al esquema de flujo de trabajo que está editando. Consulte [Copia de elementos de esquema de flujo de trabajo](#).

Un esquema de flujo de trabajo debe tener como mínimo un elemento **Finalizar flujo de trabajo**, pero puede tener varios.

### Requisitos previos

Abra un flujo de trabajo para editar en el Editor de flujos de trabajo.

### Procedimiento

1 Haga clic en la pestaña **Esquema** en el Editor de flujos de trabajo.

2 Arrastre un elemento de esquema del menú **Genérico** al panel izquierdo del esquema de flujo de trabajo.

3 Haga doble clic en el elemento que ha arrastrado al esquema de flujo de trabajo, asígnele un nombre y pulse Entrar.

Debe proporcionar elementos con nombres exclusivos en el contexto del flujo de trabajo.

No puede cambiar el nombre de los elementos **Temporizador de espera**, **Evento en espera**, **Finalizar flujo de trabajo** ni **Lanzar excepción**.

- 4 (opcional) Haga clic con el botón secundario en un elemento del esquema y seleccione **Copiar**.
- 5 (opcional) Haga clic con el botón secundario en una posición adecuada del esquema y seleccione **Pegar**.

Copiar y pegar elementos de esquema es un modo rápido de añadir elementos similares al esquema. Todas las opciones de configuración del elemento copiado aparecen en el elemento pegado excepto el estado empresarial. Ajuste la configuración del elemento pegado como corresponda.

- 6 Arrastre elementos de esquema de los menús **Básico**, **Registro** o **Red** al esquema de flujo de trabajo.

Puede editar los nombres de los elementos en los menús **Básico**, **Registro** o **Red**. No puede editar su creación de scripts.

- 7 Arrastre elementos de esquema del menú **Genérico** al esquema de flujo de trabajo.

Al arrastrar acciones o flujos de trabajo al esquema de flujo de trabajo, aparece un cuadro de diálogo en el que se puede buscar la acción o el flujo de trabajo que insertar.

- 8 En el cuadro de texto **Filtro**, escriba el nombre completo o parcial de la acción o del flujo de trabajo que desea insertar en el flujo de trabajo.

En el cuadro de diálogo, aparecen los flujos de trabajo o las acciones que coinciden con la búsqueda.

- 9 Haga doble clic en un flujo de trabajo o en una acción para seleccionarlos.

Ha insertado la acción o el flujo de trabajo en el esquema de flujo de trabajo.

- 10 Repita este procedimiento hasta haber añadido todos los elementos de esquema necesarios al esquema de flujo de trabajo.

### Pasos siguientes

Defina las propiedades de los elementos que ha añadido al esquema de flujo de trabajo, vincúlelas y enlázelas.

## Copia de elementos de esquema de flujo de trabajo

Puede copiar un elemento o una selección de elementos del esquema de un flujo de trabajo existente al esquema del flujo de trabajo que está editando.

### Requisitos previos

Abra un flujo de trabajo para editar en el Editor de flujos de trabajo.

### Procedimiento

- 1 Haga clic en la pestaña **Esquema** en el Editor de flujos de trabajo.

- 2 En el panel izquierdo, seleccione el flujo de trabajo cuyos elementos de esquema desea copiar.
  - Haga clic en **Todos los flujos de trabajo** y seleccione el flujo de trabajo en la lista jerárquica.
  - Escriba el nombre del flujo de trabajo en el cuadro de texto de búsqueda y pulse Entrar.
- 3 Haga clic con el botón derecho en el flujo de trabajo seleccionado y elija **Abrir**.  
Se abre una ventana con las propiedades del flujo de trabajo.
- 4 En la ventana del flujo de trabajo, haga clic en la pestaña **Esquema**.
- 5 Seleccione uno o varios elementos de esquema de flujo de trabajo; a continuación, haga clic con el botón derecho en la selección y elija **Copiar**.
- 6 En la pestaña **Esquema** del flujo de trabajo que está editando, haga clic con el botón derecho y seleccione **Pegar**.

Se han copiado los elementos de esquema de un flujo de trabajo a otro.

### Pasos siguientes

Debe enlazar los elementos de esquema copiados con el esquema de flujo de trabajo existente.

## Promocionar parámetros de entrada y de salida

Puede promocionar los parámetros de entrada y de salida de un elemento secundario al flujo de trabajo principal.

Puede promocionar un atributo personalizado definido en la pestaña **General** del Editor de flujos de trabajo. Puede promocionar atributos predefinidos simplemente reemplazando un parámetro de entrada por un atributo de un tipo correspondiente.

---

**Nota** Si promociona un atributo predefinido y le asigna un valor personalizado, se crea un atributo duplicado para evitar sobrescribir el valor del atributo original. El atributo duplicado conserva el nombre del atributo original con un valor numérico incrementado al final del nombre del atributo.

---

### Requisitos previos

Abra un flujo de trabajo para editar en el Editor de flujos de trabajo.

### Procedimiento

- 1 Haga clic en la pestaña **Esquema** en el Editor de flujos de trabajo.
- 2 Añada un elemento de flujo de trabajo o de acción al esquema del flujo de trabajo.  
En la parte superior del panel de esquema aparece la siguiente notificación.  
¿Desea añadir los parámetros de la actividad como entrada/salida del flujo de trabajo actual?
- 3 En la notificación, haga clic en **Configuración**.  
Se abrirá una ventana con las opciones disponibles.

#### 4 Seleccione el tipo de asignación para cada parámetro de entrada.

Opción	Descripción
<b>Entrada</b>	El argumento se asigna al parámetro de flujo de trabajo de entrada.
<b>Omitir</b>	El argumento se asigna a un valor NULL.
<b>Valor</b>	El argumento se asigna a un atributo con un valor que puede configurar en la columna Valor.

#### 5 Seleccione el tipo de asignación para cada parámetro de salida.

Opción	Descripción
<b>Salida</b>	El argumento se asigna al parámetro de flujo de trabajo de salida.
<b>Omitir</b>	El argumento se asigna a un valor NULL.
<b>Variable local</b>	El argumento se asigna a un atributo.

#### 6 Haga clic en **Promocionar**.

Los parámetros han quedado promocionados al flujo de trabajo principal.

## Modificar resultados de búsqueda

Utilice el cuadro de texto **Buscar** para buscar elementos como flujos de trabajo o acciones. Si una búsqueda devuelve un resultado parcial, puede modificar el número de resultados que se devuelven.

Cuando utiliza la búsqueda de un elemento, un cuadro de mensaje verde indica que la búsqueda muestra todos los resultados. Un cuadro de mensaje amarillo indica que la búsqueda solo muestra resultados parciales.

### Procedimiento

- 1 (opcional) Si edita un flujo de trabajo en el Editor de flujos de trabajo, haga clic en **Guardar y cerrar** para salir del editor.
- 2 En el menú del cliente de Orchestrator, seleccione **Herramientas > Preferencias del usuario**.
- 3 Haga clic en la pestaña **General**.
- 4 Escriba el número de resultados que devolverán las búsquedas en el cuadro de texto **Tamaño máximo del buscador**.
- 5 Haga clic en **Guardar y cerrar** en el cuadro de diálogo Preferencias del usuario.

Ha modificado el número de resultados que devuelven las búsquedas.

## Elementos de esquema

El Editor de flujos de trabajo presenta los elementos de esquema de flujo de trabajo en menús en la pestaña **Esquema**. Los elementos de esquema de la pestaña **Esquema** se pueden utilizar para crear un flujo de trabajo.

Tabla 1-3. Elementos e iconos de esquema






Nombre de elemento de esquema	Descripción	Icono	Ubicación en el Editor de flujos de trabajo
<b>Iniciar flujo de trabajo</b>	Punto inicial del flujo de trabajo. Todos los flujos de trabajo contienen este elemento. Un flujo de trabajo solo puede tener un elemento Iniciar. Los elementos Iniciar tienen una salida, no tienen ninguna entrada y no se pueden quitar del esquema de flujo de trabajo.		Siempre presente en la pestaña <b>Esquema</b>
<b>Tarea de scripts</b>	Tareas de propósito general definidas por el usuario. Se escriben funciones de JavaScript en este elemento.		Paleta de flujos de trabajo <b>Genérico</b>
<b>Decisión</b>	Función booleana. Un elemento Decisión toma un parámetro de entrada y devuelve true o false. El tipo de decisión tomada por el elemento depende del tipo de parámetro de entrada. El elemento Decisión permite que el flujo de trabajo se bifurque en direcciones diferentes según el parámetro de entrada recibido por el elemento Decisión. Si el parámetro de entrada recibido corresponde a un valor previsto, el flujo de trabajo continúa por una determinada ruta. Si la entrada no es el valor previsto, el flujo de trabajo continúa por una ruta alternativa.		Paleta de flujos de trabajo <b>Genérico</b>
<b>Decisión personalizada</b>	Función booleana. Las decisiones personalizadas pueden tomar varios parámetros de entrada y procesarlos conforme a los scripts personalizados. Devuelve true o false.		Paleta de flujos de trabajo <b>Genérico</b>
<b>Actividad de decisión</b>	Función booleana. Una actividad de decisión ejecuta un flujo de trabajo y enlaza sus parámetros de salida a una ruta de true o de false.		Paleta de flujos de trabajo <b>Genérico</b>

Tabla 1-3. Elementos e iconos de esquema (continuación)





Nombre de elemento de esquema	Descripción	Icono	Ubicación en el Editor de flujos de trabajo
<b>Interacción del usuario</b>	Permite que los usuarios pasen parámetros de entrada nuevos al flujo de trabajo. Puede establecer el modo en el que el elemento Interacción del usuario presenta la solicitud de parámetros de entrada y coloca restricciones en los parámetros que los usuarios pueden proporcionar. Puede establecer permisos para determinar los usuarios que pueden proporcionar los parámetros de entrada. Cuando un flujo de trabajo en ejecución llega a un elemento Interacción del usuario, accede a un estado pasivo y solicita la entrada al usuario. Puede establecer un periodo de tiempo de espera en el que los usuarios deben proporcionar la entrada. El flujo de trabajo se reanuda teniendo en cuenta los datos pasados por el usuario o devuelve una excepción si caduca el periodo de tiempo de espera. Mientras espera la respuesta del usuario, el token de flujo de trabajo se encuentra en estado <code>waiting</code> .		Paleta de flujos de trabajo <b>Genérico</b>
<b>Temporizador de espera</b>	Lo utilizan flujos de trabajo de larga ejecución. Cuando un flujo de trabajo en ejecución llega a un elemento Temporizador de espera, accede a un estado pasivo. Establezca la fecha absoluta en la que se reanuda la ejecución del flujo de trabajo. Mientras espera la fecha, el token de flujo de trabajo se encuentra en estado <code>waiting-signal</code> .		Paleta de flujos de trabajo <b>Genérico</b>
<b>Evento de espera</b>	Lo utilizan los flujos de trabajo de larga ejecución. Cuando un flujo de trabajo en ejecución llega a un elemento Evento de espera, accede a un estado pasivo. Defina un evento de activador que el flujo de trabajo espera antes de reanudar la ejecución. Mientras espera el evento, el token de flujo de trabajo se encuentra en estado <code>waiting-signal</code> .		Paleta de flujos de trabajo <b>Genérico</b>
<b>Finalizar flujo de trabajo</b>	Endpoint de un flujo de trabajo. Puede haber varios elementos Finalizar en un esquema para representar los diferentes resultados posibles de un flujo de trabajo. Los elementos Finalizar flujo de trabajo tienen una entrada sin salida. Cuando un flujo de trabajo llega al elemento Finalizar flujo de trabajo, el token de flujo de trabajo pasa al estado <code>completed</code> .		Paleta de flujos de trabajo <b>Genérico</b>

Tabla 1-3. Elementos e iconos de esquema (continuación)








Nombre de elemento de esquema	Descripción	Icono	Ubicación en el Editor de flujos de trabajo
<b>Lanzar excepción</b>	Crea una excepción y detiene el flujo de trabajo. En el esquema de flujo de trabajo, puede haber varias ocurrencias de este elemento. Los elementos Excepción tienen un parámetro de entrada, que solo puede ser del tipo cadena, y no tienen parámetros de salida. Cuando un flujo de trabajo llega al elemento Excepción, el token de flujo de trabajo pasa al estado <code>failed</code> .		Paleta de flujos de trabajo <b>Genérico</b>
<b>Nota de flujo de trabajo</b>	Permite anotar secciones del flujo de trabajo. Puede ajustar notas para delinear secciones del flujo de trabajo. Puede cambiar el color de fondo de las notas para diferenciar zonas del flujo de trabajo. Las notas de flujo de trabajo solo aportan información visual para facilitar la comprensión del esquema.		Paleta de flujos de trabajo <b>Genérico</b>
<b>Elemento de acción</b>	Llama a una acción desde las bibliotecas de acciones de Orchestrator. Cuando un flujo de trabajo llega a un elemento de acción, lo llama y ejecuta dicha acción.		Paleta de flujos de trabajo <b>Genérico</b>
<b>Elemento Flujo de trabajo</b>	Inicia otro flujo de trabajo de forma sincrónica. Cuando un flujo de trabajo llega a un elemento Flujo de trabajo en su esquema, ejecuta dicho flujo de trabajo como parte de su propio proceso. El flujo de trabajo original continúa únicamente tras el término de la ejecución del flujo de trabajo llamado.		Paleta de flujos de trabajo <b>Genérico</b>
<b>Elemento Foreach</b>	Ejecuta un flujo de trabajo en cada elemento de una matriz. Por ejemplo, puede ejecutar el flujo de trabajo Cambiar nombre de máquina virtual en todas las máquinas virtuales de una carpeta.		Paleta de flujos de trabajo <b>Genérico</b>
<b>Flujo de trabajo asincrónico</b>	Inicia un flujo de trabajo de forma asincrónica. Cuando un flujo de trabajo llega al elemento Flujo de trabajo asincrónico, inicia dicho flujo de trabajo y continúa su propia ejecución. El flujo de trabajo original no espera la finalización del flujo de trabajo llamado.		Paleta de flujos de trabajo <b>Genérico</b>
<b>Programar flujo de trabajo</b>	Crea una tarea para ejecutar el flujo de trabajo a una hora establecida; después, el flujo de trabajo continúa su ejecución.		Paleta de flujos de trabajo <b>Genérico</b>



Tabla 1-3. Elementos e iconos de esquema (continuación)





Nombre de elemento de esquema	Descripción	Icono	Ubicación en el Editor de flujos de trabajo
<b>Flujos de trabajo anidados</b>	Inicia varios flujos de trabajo simultáneamente. Puede elegir anidar flujos de trabajo locales y remotos que están en otro servidor de Orchestrator. También puede ejecutar flujos de trabajo con credenciales diferentes. El flujo de trabajo espera la conclusión de todos los flujos de trabajo anidados antes de continuar ejecutándose.		Paleta de flujos de trabajo <b>Genérico</b>
<b>Gestor de errores</b>	Gestiona un error de un determinado elemento de flujo de trabajo. El flujo de trabajo puede gestionar el error creando una excepción, llamando a otro flujo de trabajo o ejecutando un script personalizado.		Paleta de flujos de trabajo <b>Genérico</b>
<b>Gestor de errores predeterminado</b>	Gestiona los errores de flujo de trabajo no detectados por los gestores de errores estándar. Puede utilizar cualquier elemento de esquema disponible para gestionar errores.		Paleta de flujos de trabajo <b>Genérico</b>

Tabla 1-3. Elementos e iconos de esquema (continuación)

Nombre de elemento de esquema	Descripción	Icono	Ubicación en el Editor de flujos de trabajo
Switch	Cambia a rutas alternativas de flujo de trabajo según un atributo o un parámetro de flujo de trabajo.		Paleta de flujos de trabajo <b>Genérico</b>
Tarea predefinida	<p>Elementos de script no editables que efectúan tareas estándar utilizadas habitualmente por flujos de trabajo. Están predefinidas las tareas siguientes:</p> <p><b>Básico</b></p> <ul style="list-style-type: none"> <li>■ Reposo</li> <li>■ Cambiar credencial</li> <li>■ Esperar hasta fecha</li> <li>■ Esperar evento personalizado</li> <li>■ Enviar evento personalizado</li> <li>■ Incrementar contador</li> <li>■ Disminuir contador</li> </ul> <p><b>Registro</b></p> <ul style="list-style-type: none"> <li>■ Log del sistema</li> <li>■ Advertencia del sistema</li> <li>■ Error del sistema</li> <li>■ Log del servidor</li> <li>■ Advertencia del servidor</li> <li>■ Error del servidor</li> <li>■ Log del sistema+servidor</li> <li>■ Advertencia del sistema+servidor</li> <li>■ Error del sistema+servidor</li> </ul> <p><b>Red</b></p> <ul style="list-style-type: none"> <li>■ HTTP post</li> <li>■ HTTP get</li> </ul>		Paletas de flujos de trabajo <b>Básico, Log y Red</b>

## Propiedades de elementos de esquema

Los elementos de esquema tienen propiedades que puede definir y editar en la pestaña **Esquema** de la paleta de flujos de trabajo.

### Editar las propiedades globales de un elemento de esquema

Las propiedades globales de un elemento de esquema se definen en la pestaña Información del elemento.

#### Requisitos previos

Verifique que la pestaña **Esquema** del Editor de flujos de trabajo contenga elementos.

#### Procedimiento

- 1 Haga clic en la pestaña **Esquema** en el Editor de flujos de trabajo.

- 2 Seleccione un elemento que editar haciendo clic en el icono **Editar** (✎).

Aparece un cuadro de diálogo con las propiedades del elemento.

- 3 Haga clic en la pestaña **Información**.

- 4 En el cuadro de texto **Nombre**, escriba un nombre para el elemento de esquema.

Este será el nombre que aparece en el elemento de esquema en el diagrama del esquema de flujo de trabajo.

- 5 En el menú desplegable **Interacción**, seleccione una descripción.

La propiedad **Interacción** permite seleccionar entre descripciones estándar el modo en el que este elemento interactúa con objetos fuera del flujo de trabajo. Esta propiedad solo es para información.

- 6 (opcional) En el cuadro de texto **Estado empresarial**, proporciona una descripción del estado empresarial.

La propiedad **Estado empresarial** es una descripción resumida de lo que hace este elemento. Cuando un flujo de trabajo está en ejecución, el token de flujo de trabajo muestra el estado empresarial de cada elemento a medida que se ejecuta. Esta función es útil para realizar el seguimiento del estado del flujo de trabajo.

- 7 (opcional) En el cuadro de texto **Descripción**, escriba una descripción del elemento de esquema.

## Pestañas de propiedades de elementos de esquema

Para acceder a las propiedades de un elemento de esquema, haga clic en un elemento que haya arrastrado al esquema de flujo de trabajo. Las propiedades del elemento aparecen en las pestañas en la parte inferior del Editor de flujos de trabajo.

Los distintos elementos de esquema tienen diferentes pestañas de propiedades.

**Tabla 1-4. Pestañas de propiedades por elemento de esquema**

Pestaña de propiedad de elemento de esquema	Descripción	Se aplica al tipo de elemento de esquema
<b>Atributos</b>	Atributos que requieren los elementos de un origen externo, por ejemplo el usuario, un evento o un temporizador. Los atributos pueden ser un límite de tiempo de espera, una fecha y una hora, un activador o credenciales de usuario.	<ul style="list-style-type: none"> <li>■ Interacción del usuario</li> <li>■ Evento de espera</li> <li>■ Temporizador de espera</li> </ul>
<b>Decisión</b>	Define la instrucción de decisión. El parámetro de entrada que recibe el elemento de decisión puede coincidir o no con la instrucción de decisión, de modo que existen dos posibles cursos de acción.	Decisión

Tabla 1-4. Pestañas de propiedades por elemento de esquema (continuación)

Pestaña de propiedad de elemento de esquema	Descripción	Se aplica al tipo de elemento de esquema
<b>Finalizar flujo de trabajo</b>	Finaliza el flujo de trabajo, porque se ha completado correctamente o porque se ha detectado un error y se ha devuelto una excepción.	<ul style="list-style-type: none"> <li>■ Fin</li> <li>■ Excepción</li> </ul>
<b>Excepción</b>	Comportamiento de este elemento de esquema en caso de excepción.	<ul style="list-style-type: none"> <li>■ Acción</li> <li>■ Flujo de trabajo asíncrono</li> <li>■ Excepción</li> <li>■ Flujos de trabajo anidados</li> <li>■ Tarea predefinida</li> <li>■ Flujo de trabajo de programación</li> <li>■ Tarea de scripts</li> <li>■ Interacción del usuario</li> <li>■ Evento de espera</li> <li>■ Temporizador de espera</li> <li>■ Flujo de trabajo</li> </ul>
<b>Entradas externas</b>	Parámetros de entrada que debe proporcionar el usuario en un momento determinado mientras se ejecuta el flujo de trabajo.	Interacción del usuario
<b>IN</b>	El enlace IN de este elemento. El enlace IN define el modo en el que el elemento de esquema recibe la entrada del elemento que lo precede en el flujo de trabajo.	<ul style="list-style-type: none"> <li>■ Acción</li> <li>■ Flujo de trabajo asíncrono</li> <li>■ Decisión personalizada</li> <li>■ Tarea predefinida</li> <li>■ Flujo de trabajo de programación</li> <li>■ Tarea de scripts</li> <li>■ Flujo de trabajo</li> </ul>
<b>Información</b>	Las propiedades generales y una descripción del elemento de esquema. La información que aparece en la pestaña <b>Información</b> depende del tipo de elemento de esquema.	<ul style="list-style-type: none"> <li>■ Acción</li> <li>■ Flujo de trabajo asíncrono</li> <li>■ Decisión personalizada</li> <li>■ Decisión</li> <li>■ Flujos de trabajo anidados</li> <li>■ Nota</li> <li>■ Tarea predefinida</li> <li>■ Flujo de trabajo de programación</li> <li>■ Tarea de scripts</li> <li>■ Interacción del usuario</li> <li>■ Evento de espera</li> <li>■ Temporizador de espera</li> <li>■ Flujo de trabajo</li> </ul>

Tabla 1-4. Pestañas de propiedades por elemento de esquema (continuación)

Pestaña de propiedad de elemento de esquema	Descripción	Se aplica al tipo de elemento de esquema
<b>OUT</b>	El enlace OUT de este elemento. El enlace OUT define el modo en el que el elemento de esquema enlaza los parámetros de salida con los atributos de flujo de trabajo o los parámetros de salida del flujo de trabajo.	<ul style="list-style-type: none"> <li>■ Acción</li> <li>■ Flujo de trabajo asincrónico</li> <li>■ Tarea predefinida</li> <li>■ Flujo de trabajo de programación</li> <li>■ Tarea de scripts</li> <li>■ Flujo de trabajo</li> </ul>
<b>Presentación</b>	Define el diseño del cuadro de diálogo de parámetros de entrada que ve el usuario si el flujo de trabajo necesita entrada del usuario mientras se ejecuta.	Interacción del usuario
<b>Crear scripts</b>	Muestra la función de JavaScript que define el comportamiento de este elemento de esquema. En Flujo de trabajo asincrónico, Programar flujo de trabajo y Elementos de acción, estos scripts son de solo lectura. En las tareas de scripts y los elementos de decisión personalizados, se edita JavaScript en esta pestaña.	<ul style="list-style-type: none"> <li>■ Acción</li> <li>■ Flujo de trabajo asincrónico</li> <li>■ Decisión personalizada</li> <li>■ Tarea predefinida</li> <li>■ Flujo de trabajo de programación</li> <li>■ Tarea de scripts</li> </ul>
<b>Enlace visual</b>	Muestra una representación gráfica de cómo se enlazan los parámetros y los atributos de este elemento de esquema con los parámetros y los atributos de los elementos que hay antes y después en el flujo de trabajo. Es otra representación de los enlaces IN y OUT del elemento.	<ul style="list-style-type: none"> <li>■ Acción</li> <li>■ Flujo de trabajo asincrónico</li> <li>■ Tarea predefinida</li> <li>■ Flujo de trabajo de programación</li> <li>■ Tarea de scripts</li> <li>■ Flujo de trabajo</li> </ul>
<b>Flujos de trabajo</b>	Selecciona los flujos de trabajo que se anidarán.	Flujos de trabajo anidados

## Enlaces y vínculos

Los vínculos entre elementos determinan el flujo lógico del flujo de trabajo. Los enlaces rellenan los elementos con datos de otros elementos enlazando parámetros de entrada y de salida a los atributos de flujos de trabajo.

Para comprender los enlaces y los vínculos, debe conocerse la diferencia entre el flujo lógico y el flujo de datos de un flujo de trabajo.

## Flujo lógico de un flujo de trabajo

El flujo lógico de un flujo de trabajo es el progreso del flujo de trabajo de un elemento al siguiente en el esquema durante la ejecución del flujo de trabajo. El flujo lógico del flujo de trabajo se define vinculando elementos en el esquema.

La ruta estándar es la ruta del flujo de trabajo a través del flujo lógico si todos los elementos se ejecutan del modo esperado. La ruta de excepción es la ruta del flujo de trabajo a través del flujo lógico si un elemento no se ejecuta del modo esperado.

Diferentes estilos de flecha en el esquema del flujo de trabajo indican distintas rutas que puede tomar el flujo lógico del flujo de trabajo.

- Una flecha azul indica la ruta estándar que toma el flujo de trabajo de un elemento al siguiente.
- Una flecha verde indica la ruta que toma el flujo de trabajo si un elemento de decisión booleano devuelve `true`.
- Una flecha de puntos rojos indica la ruta que toma el flujo de trabajo si un elemento de decisión booleano devuelve `false`.
- Una flecha roja discontinua indica la ruta de excepción que toma el flujo de trabajo si un elemento del flujo de trabajo no se ejecuta correctamente.

La figura siguiente muestra un esquema de flujo de trabajo de ejemplo que demuestra las diferentes rutas que pueden tomar los flujos de trabajo.

**Figura 1-1. Distintas rutas del flujo lógico del flujo de trabajo**



Este flujo de trabajo de ejemplo puede tomar las rutas siguientes mediante su flujo lógico.

- Ruta estándar, resultado de decisión `true`, sin excepciones.
  - a El elemento de decisión devuelve `true`.
  - b El flujo de trabajo `SnapVMsInResourcePool` se ejecuta correctamente.
  - c La acción `sendHtmlEmail` se ejecuta correctamente.
  - d El flujo de trabajo finaliza correctamente en el estado `completed`.
- Ruta estándar, resultado de decisión `false`, sin excepciones.
  - a El elemento de decisión devuelve `false`.
  - b La operación que define el elemento de tarea de scripts se ejecuta correctamente.
  - c La acción `sendHtmlEmail` se ejecuta correctamente.
  - d El flujo de trabajo finaliza correctamente en el estado `completed`.
- Resultado de decisión `true`, excepción.
  - a El elemento de decisión devuelve `true`.
  - b El flujo de trabajo `SnapVMsInResourcePool` detecta un error.
  - c El flujo de trabajo devuelve una excepción y se detiene en el estado `failed`.
- Resultado de decisión `false`, excepción.
  - a El elemento de decisión devuelve `false`.
  - b La operación que define el elemento de tarea de scripts detecta un error.
  - c El flujo de trabajo devuelve una excepción y se detiene en el estado `failed`.

## Vínculos de elementos

Los vínculos conectan elementos de esquema y definen el flujo lógico del flujo de trabajo de un elemento al siguiente.

En general, los elementos solo pueden establecer un vínculo saliente a otro elemento del flujo de trabajo y un vínculo de excepción a un elemento que define su comportamiento de excepciones. El vínculo saliente define la ruta estándar del flujo de trabajo. El vínculo de excepción define la ruta de excepción del flujo de trabajo. En la mayoría de los casos, un solo elemento de esquema puede recibir vínculos de ruta entrantes de varios elementos.

Los elementos siguientes son excepciones a las afirmaciones anteriores.

- El elemento `Iniciar Flujo de trabajo` no puede recibir vínculos entrantes ni tiene vínculo de excepción.
- Los elementos de excepción pueden recibir varios vínculos de excepción entrantes y no tienen vínculos salientes ni de excepción.
- Los elementos de decisión tienen dos vínculos salientes que definen las rutas que sigue el flujo de trabajo dependiendo de si el resultado de la decisión es `true` o `false`. Las decisiones no tienen vínculo de excepción.

- Los elementos Finalizar flujo de trabajo no pueden tener vínculos salientes ni de excepción.

## Crear vínculos de ruta estándar

Los vínculos de ruta estándar determinan la ejecución normal del flujo de trabajo.

Cuando vincula un elemento con otro, siempre vincula los elementos en el orden en el que se ejecutan en el flujo de trabajo. Para crear un vínculo entre dos elementos, siempre se empieza por el que se ejecuta primero.

### Requisitos previos

- Abra un flujo de trabajo para editar en el Editor de flujos de trabajo.
- Verifique que la pestaña **Esquema** del Editor de flujos de trabajo contenga elementos.

### Procedimiento

- 1 Coloque el puntero sobre el elemento que desea conectar a otro elemento.  
A la derecha del elemento, aparecen una flecha azul y una roja.
- 2 Coloque el puntero sobre la flecha azul.  
Dicha flecha se hace más grande.
- 3 Haga clic con el botón izquierdo del ratón en la flecha azul y manténgalo pulsado; a continuación, mueva el puntero al elemento de destino.  
Aparece una flecha azul entre los dos elementos y un rectángulo verde alrededor del elemento de destino.
- 4 Suelte el botón izquierdo del ratón.  
La flecha azul permanece entre los dos elementos.

Ahora una ruta estándar vincula los elementos.

### Pasos siguientes

Los elementos están unidos, pero no se ha definido el flujo de datos. Debe definir los enlaces de IN y OUT para enlazar los datos de entrada y salida con los atributos del flujo de trabajo.

## Flujo de datos de un flujo de trabajo

El flujo de datos de un flujo de trabajo es la forma en que un elemento de flujo de trabajo y los parámetros de entrada y de salida se enlazan a atributos de flujo de trabajo cuando se ejecuta cada elemento del flujo de trabajo. El flujo de datos de un flujo de trabajo se define mediante enlaces de elementos de esquema.

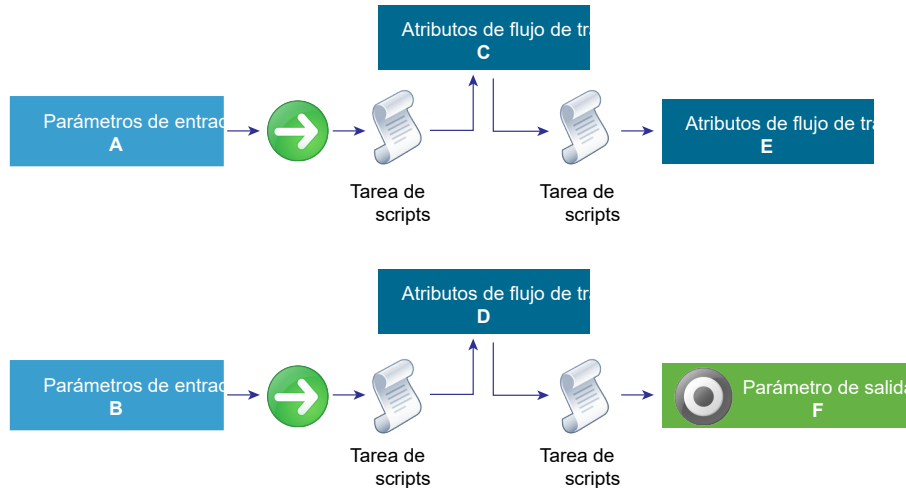
Cuando se ejecuta un elemento del esquema de flujo de trabajo, requiere datos en forma de parámetros de entrada. Toma los datos para sus parámetros de entrada enlazando a un atributo de flujo de trabajo establecido al crear el flujo de trabajo o bien enlazando a un atributo establecido por un elemento precedente del flujo de trabajo cuando se ejecutó.



El elemento procesa los datos, posiblemente los transforma y genera los resultados de su ejecución en forma de parámetros de salida. El elemento enlaza sus parámetros de salida resultantes a los nuevos atributos de flujo de trabajo que crea. Otros elementos del esquema se pueden enlazar a estos atributos nuevos de flujo de trabajo como sus parámetros de entrada. El flujo de trabajo puede generar los atributos como sus parámetros de salida al final de su ejecución.

La figura siguiente ilustra un flujo de trabajo muy sencillo. La flecha azul representa el elemento que enlaza y el flujo lógico del flujo de trabajo. Las líneas rojas muestran el flujo de datos del flujo de trabajo.

**Figura 1-2. Ejemplo de flujo de datos de flujo de trabajo**



Los datos fluyen por el flujo de trabajo como se indica a continuación.

- 1 El flujo de trabajo empieza por los parámetros de entrada a y b.
- 2 El primer elemento procesa el parámetro a y enlaza el resultado de procesarlo al atributo c del flujo de trabajo.
- 3 El primer elemento procesa el parámetro b y enlaza el resultado de procesarlo al atributo d del flujo de trabajo.
- 4 El segundo elemento toma el atributo c del flujo de trabajo como parámetro de entrada, lo procesa y enlaza el parámetro de salida resultante al atributo e del flujo de trabajo.
- 5 El segundo elemento toma el atributo d del flujo de trabajo como parámetro de entrada, lo procesa y genera el parámetro de salida f.
- 6 El flujo de trabajo concluye y genera el atributo f del flujo de trabajo como su parámetro de salida, el resultado de su ejecución.

## Enlaces de elementos

Debe enlazar todos los parámetros de entrada y de salida del elemento del flujo de trabajo a atributos de flujo de trabajo. Los enlaces establecen datos en los elementos, además de definir su comportamiento de salida y de excepciones. Los vínculos definen el flujo lógico del flujo de trabajo, mientras que los enlaces definen el flujo de datos.

Para establecer datos en un elemento, generar parámetros de salida a partir del elemento tras el procesamiento y controlar todos los errores que pudieran producirse en las ejecuciones del elemento, debe definir los enlaces del elemento.

<b>Enlaces IN</b>	Establece datos de entrada del elemento de esquema. Enlaza los parámetros de entrada locales del elemento a atributos de flujo de trabajo de origen. En la pestaña <b>IN</b> , figuran los parámetros de entrada del elemento en la columna Parámetro local. En la pestaña <b>IN</b> , figuran los atributos de flujo de trabajo a los que se enlaza el parámetro local en la columna Parámetro de origen. Asimismo, en la pestaña aparecen el tipo de parámetro y una descripción de este.
<b>Enlaces OUT</b>	Cambie atributos de flujo de trabajo y genere parámetros de salida cuando un elemento termina de ejecutarse. En la pestaña <b>OUT</b> , figuran los parámetros de salida del elemento en la columna Parámetro local. En la pestaña <b>OUT</b> , figuran los atributos de flujo de trabajo a los que se enlaza el parámetro local en la columna Parámetro de origen. Asimismo, en la pestaña aparecen el tipo de parámetro y una descripción de este.
<b>Enlaces de excepción</b>	Víncule a controladores de excepciones si el elemento detecta una excepción cuando se ejecuta.

Los enlaces IN leen valores del parámetro de origen enlazado. Los enlaces OUT escriben valores en el parámetro de origen enlazado.

Debe utilizar los enlaces IN para vincular cada atributo o parámetro de entrada que utilice en un elemento de esquema a un atributo del flujo de trabajo. Si el elemento cambia los valores de los parámetros de entrada que recibe cuando se ejecuta, debe enlazarlos a un atributo de flujo de trabajo mediante un enlace OUT. Al enlazar los parámetros de salida del elemento a elementos de flujo de trabajo, otros elementos que lo siguen en el esquema de flujo de trabajo pueden tomar esos parámetros de salida como sus parámetros de entrada.

Al crear flujos de trabajo, es un error habitual no enlazar valores de parámetros de salida para reflejar los cambios que el elemento efectúa en los atributos de flujo de trabajo.

---

**Importante** Al añadir un elemento que requiere parámetros de entrada y de salida de un tipo que ya ha definido en el flujo de trabajo, Orchestrator establece los enlaces a estos parámetros. Debe verificar que los parámetros enlazados por Orchestrator sean correctos, en caso de que el flujo de trabajo defina parámetros diferentes del mismo tipo al que el elemento puede enlazar.

---

## Definir enlaces de elementos

Tras vincular elementos para crear el flujo lógico del flujo de trabajo, se definen enlaces de elementos para determinar el modo en el que cada elemento procesa los datos que recibe y que genera.

### Requisitos previos

En la pestaña **Esquema** del Editor de flujos de trabajo, verifique que tenga un esquema de flujo de trabajo y que haya creado vínculos entre los elementos.

## Procedimiento

- 1 Haga clic en el icono **Editar** (✎) del elemento en el que desea establecer los enlaces.

Aparece un cuadro de diálogo con las propiedades del elemento.

- 2 Haga clic en la pestaña **IN**.

El contenido de la pestaña **IN** depende del tipo de elemento que seleccione.

- Si ha seleccionado una tarea, un flujo de trabajo o un elemento de acción predefinidos, la pestaña **IN** muestra los posibles parámetros de entrada locales para ese tipo de elemento, pero el enlace no está establecido.
- Si ha seleccionado otro tipo de elemento, puede elegir en una lista de parámetros de entrada y atributos que ya haya definido para el flujo de trabajo haciendo clic con el botón secundario en la pestaña **IN** y, a continuación, seleccionando **Enlazar a parámetro o atributo de flujo de trabajo**.
- Si el atributo requerido todavía no existe, puede crearlo haciendo clic con el botón secundario en la pestaña **IN** y, a continuación, seleccionando **Enlazar a parámetro o atributo de flujo de trabajo > Crear parámetro o atributo en flujo de trabajo**.

- 3 Si existe un parámetro adecuado, elija un parámetro de entrada que enlazar; a continuación, haga clic en el botón **No establecido** en el cuadro de texto **Parámetro de origen**.

Aparece una lista con posibles atributos y parámetros de origen que enlazar.

- 4 Elija un parámetro de origen para enlazar al parámetro de entrada local de la lista propuesta.
- 5 (opcional) Si no ha definido el parámetro de origen al que enlazar, créelo haciendo clic en el vínculo **Crear atributo o parámetro en flujo de trabajo** en el cuadro de diálogo de selección de parámetros.

- 6 Haga clic en la pestaña **OUT**.

El contenido de la pestaña **OUT** depende del tipo de elemento que seleccione.

- Si ha seleccionado una tarea, un flujo de trabajo o un elemento de acción predefinidos, la pestaña **OUT** muestra los posibles parámetros de salida locales para ese tipo de elemento, pero el enlace no está establecido.
- Si ha seleccionado otro tipo de elemento, puede elegir de una lista de parámetros de salida y atributos que ya haya definido para el flujo de trabajo haciendo clic con el botón secundario en la pestaña **OUT** y, a continuación, seleccionando **Enlazar a parámetro o atributo de flujo de trabajo**.
- Si el atributo requerido no existe, puede crearlo haciendo clic con el botón secundario en la pestaña **IN** y, a continuación, seleccionando **Enlazar a parámetro o atributo de flujo de trabajo > Crear parámetro o atributo en flujo de trabajo**.

- 7 Elija parámetro que enlazar.

- 8 Haga clic en el botón **Parámetro de origen > No establecido**.

- 9 Elija un parámetro de origen para enlazar al parámetro de entrada.
- 10 (opcional) Si no ha definido el parámetro al que enlazar, créelo haciendo clic en el vínculo **Crear atributo o parámetro en flujo de trabajo** en el cuadro de diálogo de selección de parámetros.

Ha definido los parámetros de entrada que recibe el elemento y los parámetros de salida que genera, y los ha enlazado a parámetros y atributos de flujo de trabajo.

### Pasos siguientes

Puede crear bifurcaciones en la ruta del flujo de trabajo definiendo decisiones.

## Decisiones

Los flujos de trabajo pueden implementar funciones de decisión que definen diferentes tipos de acciones conforme a una instrucción booleana `true` o `false`.

Las decisiones son bifurcaciones del flujo de trabajo. Las decisiones de flujo de trabajo se efectúan conforme a las entradas realizadas por un usuario, otros flujos de trabajo, aplicaciones o el entorno en el que se ejecuta el flujo de trabajo. El valor del parámetro de entrada recibido por elemento de decisión determina la rama que toma la bifurcación del flujo de trabajo. Por ejemplo, una decisión de flujo de trabajo podría recibir el estado de conexión de una determinada máquina virtual como su entrada. Si la máquina virtual está encendida, el flujo de trabajo sigue una determinada ruta a través de su flujo lógico. Si la máquina virtual está apagada, el flujo de trabajo sigue una ruta diferente.

Las decisiones siempre son funciones booleanas. Los únicos resultados posibles de cada decisión son `true` o `false`.

### Decisiones personalizadas

Las decisiones personalizadas se diferencian de las estándar en el hecho de que la instrucción de decisión se define en un script. Las decisiones personalizadas devuelven `true` o `false` según la instrucción que se defina, como ilustra el ejemplo siguiente.

```
if (decision_statement){
    return true;
}else{
    return false;
}
```

### Crear vínculos de elementos de decisión

Los elementos de decisión son diferentes de los demás elementos en un flujo de trabajo. Solo tienen los parámetros de salida `true` o `false`. Los elementos de decisión no tienen vínculos de excepción.

### Requisitos previos

Compruebe que la pestaña **Esquema** del Editor de flujos de trabajo contenga elementos, como mínimo un elemento de decisión que no esté vinculado con otros elementos.

## Procedimiento

- 1 Coloque el puntero del ratón en un elemento de decisión para vincularlo con otros dos elementos que definan dos posibles ramas en el flujo de trabajo.

A la derecha del elemento, aparecen una flecha azul y una flecha roja.

- 2 Coloque el puntero en la flecha azul y, con el botón izquierdo del ratón pulsado, mueva el puntero al elemento de destino.

Aparece una flecha verde entre los dos elementos y el elemento de destino se vuelve de color verde. La flecha verde representa la ruta `true` que toma el flujo de trabajo si el atributo o el parámetro de entrada recibidos por el elemento de decisión coinciden con la instrucción de decisión.

- 3 Suelte el botón izquierdo del ratón.

La flecha verde permanece entre los dos elementos. Ha definido la ruta que toma el flujo de trabajo cuando el elemento de decisión recibe el valor esperado.

- 4 Coloque el puntero en el elemento de decisión, mantenga pulsado el botón izquierdo del ratón y mueva el puntero al elemento de destino.

Aparece una flecha de puntos rojos entre los dos elementos y el elemento de destino se vuelve de color verde. La flecha roja representa la ruta `false` que toma el flujo de trabajo si el atributo o el parámetro de entrada recibidos por el elemento de decisión no coinciden con la instrucción de decisión.

- 5 Suelte el botón izquierdo del ratón.

La flecha de puntos verdes permanece entre los dos elementos. Ha definido la ruta que toma el flujo de trabajo cuando el elemento de decisión recibe una entrada inesperada.

Ha definido las rutas `true` o `false` posibles que toma el flujo de trabajo según el atributo o el parámetro de entrada que recibe el elemento de decisión.

## Pasos siguientes

Defina la instrucción de decisión. Consulte [Crear ramas de flujos de trabajo mediante decisiones](#).

## Eliminar un elemento de decisión vinculado

Al eliminar un elemento de decisión vinculado de un esquema de flujo de trabajo, debe especificar las rutas del flujo de trabajo que se deben eliminar.

## Requisitos previos

Verifique que la pestaña **Esquema** del Editor de flujos de trabajo contiene elementos, como mínimo un elemento de decisión con rutas `true` y `false`.

## Procedimiento

- 1 Seleccione el elemento de decisión y pulse Suprimir.

Se muestra un cuadro de diálogo con opciones.

## 2 Seleccione la rama de decisión que eliminar.

Opción	Descripción
<b>Rama correcta</b>	El elemento de decisión y todos los elementos que siguen la ruta de decisión <code>true</code> se eliminan del esquema de flujo de trabajo.
<b>Rama incorrecta</b>	El elemento de decisión y todos los elementos que siguen la ruta de decisión <code>false</code> se eliminan del esquema de flujo de trabajo.
<b>Ambas ramas</b>	El elemento de decisión y todos los elementos que siguen ambas rutas de decisión se eliminan del esquema de flujo de trabajo.
<b>Ninguno</b>	Solo se eliminan del esquema de flujo de trabajo el elemento de decisión y sus vínculos. Todos los elementos que siguen ambas rutas de decisión se mantienen en el esquema de flujo de trabajo.

## 3 Haga clic en **Aceptar**.

### Crear ramas de flujos de trabajo mediante decisiones

Los elementos de decisión son meras funciones booleanas que se utilizan para crear ramas en flujos de trabajo. Los elementos de decisión determinan si la entrada recibida coincide con la instrucción de decisión que se ha establecido. Como función de esta decisión, el flujo de trabajo sigue su curso por una de dos rutas posibles.

#### Requisitos previos

Verifique que tenga un elemento de decisión vinculado a otros dos elementos en el esquema en el Editor de flujos de trabajo antes de definir la decisión.

#### Procedimiento

- 1 Haga clic en el icono **Editar** (✎) del elemento de decisión.

Aparece un cuadro de diálogo con las propiedades del elemento de decisión.

- 2 En el cuadro de diálogo, haga clic en la pestaña **Decisión**.

- 3 Haga clic en el vínculo **No establecido (nulo)** a fin de seleccionar el parámetro de entrada de origen para esta decisión.

Aparece un cuadro de diálogo con todos los atributos y parámetros de entrada definidos en este flujo de trabajo.

- 4 Seleccione un parámetro de entrada en la lista haciendo doble clic en él.

- 5 Si no ha definido el parámetro de origen al que enlazar, créelo haciendo clic en el vínculo **Crear atributo o parámetro en flujo de trabajo** en el cuadro de diálogo de selección de parámetros.

- 6 Seleccione una instrucción de decisión en el menú desplegable.

Las instrucciones propuestas en el menú son contextuales y difieren según el tipo de parámetro de entrada que se seleccione.

## 7 Añada un valor que deba coincidir con la instrucción de decisión.

Según el tipo de entrada y la instrucción que se seleccione, en el cuadro de texto del valor, podría aparecer un vínculo **No establecido (nulo)**. Si se hace clic en este vínculo, se obtiene una serie predefinida de valores. De lo contrario, por ejemplo para Cadenas, es un cuadro de texto en el que se proporciona un valor.

Ha definido una instrucción para el elemento de decisión. Cuando el elemento de decisión recibe el parámetro de entrada, compara el valor del parámetro de entrada con el de la instrucción y determina si esta es verdadera (true) o falsa (false).

### Pasos siguientes

Debe establecer el modo en el que el flujo de trabajo controla las excepciones.

## Control de excepciones

El control de excepciones detecta los errores que se producen durante la ejecución de un elemento de esquema. El control de excepciones define el modo en el que se comporta el elemento de esquema cuando se produce el error.

Todos los elementos de un flujo de trabajo, salvo las decisiones y los elementos de inicio y fin, contienen un tipo de parámetro de salida específico que sirve solo para el control de excepciones. Si un elemento detecta un error durante su ejecución, puede enviar una señal de error a un controlador de excepciones. Los controladores de excepciones detectan el error y actúan en función de los errores que reciban. Si los controladores de excepciones que define no pueden controlar un error concreto, puede enlazar el parámetro de salida de excepción de un elemento con un elemento Exception, que finaliza la ejecución del flujo de trabajo en el estado erróneo.

Las excepciones actúan como una secuencia try y catch dentro de un elemento de flujo de trabajo. Si no necesita controlar una excepción específica en un elemento, no es necesario enlazar el parámetro de salida de excepción del elemento.

El tipo de parámetro de salida para las excepciones siempre es un objeto errorCode.

## Crear enlaces de excepción

Los elementos pueden establecer enlaces que definan el comportamiento del flujo de trabajo en caso de error en los elementos.

### Requisitos previos

Verifique que la pestaña **Esquema** del Editor de flujos de trabajo contenga elementos.




### Procedimiento

#### 1 Coloque el puntero en el elemento para el que desea definir los enlaces de excepción.

Aparece una flecha roja a la derecha del elemento.

- 2 Coloque el puntero en la flecha roja hasta que se haga más grande; mantenga pulsado el botón izquierdo del ratón y arrastre la flecha roja hasta el elemento de destino.

Una flecha roja discontinua vincula los dos elementos. El elemento de destino define el comportamiento del flujo de trabajo si el elemento que se vincula a él detecta un error.

- 3 Haga clic en el icono **Editar** () del elemento que vincula al elemento de control de excepciones.
- 4 Haga clic en la pestaña **Excepción** en las pestañas de propiedades del elemento de esquema.
- 5 Para definir el valor de **Enlace de excepciones de salida**, haga clic en **No establecido**.
  - Seleccione un parámetro para enlazar al parámetro de salida de excepción desde el cuadro de diálogo de enlace de atributos de excepción; a continuación, haga clic en **Seleccionar**.
  - Haga clic en **Crear parámetro o atributo en el flujo de trabajo** para crear un parámetro de salida de excepción.
- 6 Haga clic en el elemento de destino que define el comportamiento de control de excepciones.
- 7 Haga clic en la pestaña **IN** en las pestañas de propiedades del elemento de esquema.
- 8 Haga clic en el icono de **Enlazar a parámetro o atributo de flujo de trabajo** ().  
Se abre el cuadro de diálogo para seleccionar el parámetro de entrada.
- 9 Seleccione el parámetro de salida de excepción y haga clic en **Seleccionar**.
- 10 Haga clic en la pestaña **OUT** para el elemento de control de excepciones en las pestañas de propiedades del elemento de esquema.
- 11 Defina el comportamiento del elemento de control de excepciones.
  - Haga clic en el icono de **Enlazar a parámetro o atributo de flujo de trabajo** () para seleccionar un parámetro de salida para el elemento de control de excepciones que se generará.
  - Haga clic en la pestaña **Creación de scripts** y utilice JavaScript para definir el comportamiento del elemento de control de excepciones.

Ha definido la forma en que el elemento controla las excepciones.

### Pasos siguientes

Debe definir cómo obtener los parámetros de entrada de los usuarios cuando ejecutan el flujo de trabajo.

## Utilizar gestores de errores

Puede utilizar un gestor de errores estándar para definir el comportamiento si se produce un error en un determinado elemento de esquema de flujo de trabajo. Puede utilizar un gestor de errores global para definir el comportamiento si se producen errores no detectados por los gestores de errores estándar.

### Añadir un gestor de errores a un flujo de trabajo

Para definir cómo se gestionan los errores de un elemento de flujo de trabajo específico durante la ejecución de un flujo de trabajo, añada un gestor de errores al elemento de flujo de trabajo. Puede añadir



un gestor de errores solo a los elementos de flujo de trabajo que no tengan especificada una ruta de errores.

---

**Importante** Los flujos de trabajo que contienen un elemento **Gestor de errores** son incompatibles con Orchestrator 5.5.x y versiones anteriores.

---

#### Requisitos previos

- Cree un flujo de trabajo.
- Abra el flujo de trabajo para editar en el Editor de flujos de trabajo.
- Añada algunos elementos al esquema del flujo de trabajo.

#### Procedimiento

- 1 Arrastre un elemento **Gestor de errores** al elemento correspondiente del esquema de flujo de trabajo.

Se muestra un cuadro de diálogo.

- 2 En el menú desplegable del cuadro de diálogo, seleccione cómo se deben gestionar los errores.

Opción	Descripción
Lanzar excepción	Cuando se produce un error, se lanza una excepción. Puede modificar el enlace de excepciones.
Llamar a un flujo de trabajo	Cuando se produce un error, se ejecuta el flujo de trabajo seleccionado.
Script personalizado	Cuando se produce un error, se ejecuta un script personalizado.

- 3 Haga clic en **Seleccionar**.

Se ha añadido un gestor de errores a un flujo de trabajo. Cuando el flujo de trabajo alcanza este elemento, lleva a cabo la acción seleccionada antes de finalizar su ejecución.

### Añadir un gestor de errores global a un flujo de trabajo

Para definir cómo se gestionan los errores que no se detectan mediante los gestores de errores estándar durante la ejecución de un flujo de trabajo, añada un gestor de errores global al esquema del flujo de trabajo. Es posible añadir un gestor de errores global a un esquema de flujo de trabajo.

---

**Importante** Los flujos de trabajo que contienen un elemento **Gestor de errores predeterminado** son incompatibles con Orchestrator 5.5.x y versiones anteriores.

---

#### Requisitos previos

- Cree un flujo de trabajo.
- Abra el flujo de trabajo para editar en el Editor de flujos de trabajo.
- Añada algunos elementos al esquema del flujo de trabajo.

## Procedimiento

- 1 Arrastre un elemento **Gestor de errores predeterminado** al esquema del flujo de trabajo.
- 2 (opcional) Añada elementos de esquema entre el elemento **Gestor de errores predeterminado** y el elemento **Lanzar excepción** para especificar cómo se gestionan los errores de flujo de trabajo globales.

Se ha añadido un gestor de errores global en un flujo de trabajo. Cuando se produce un error no detectado por los gestores de errores estándar en el flujo de trabajo, el gestor de errores global realiza las acciones especificadas antes de finalizar la ejecución del flujo de trabajo.

## Elementos Foreach y tipos compuestos

Puede insertar un elemento Foreach en el flujo de trabajo que desarrolle para ejecutar un flujo de trabajo secundario que se itere sobre las matrices de parámetros o atributos. Para mejorar la comprensión y la legibilidad del flujo de trabajo, puede agrupar varios parámetros de flujo de trabajo de diferentes tipos que estén conectados lógicamente en un solo tipo conocido como tipo compuesto.

### Uso de elementos Foreach

Un elemento Foreach ejecuta un flujo de trabajo secundario de forma iterativa sobre una matriz de parámetros o atributos de entrada. Puede seleccionar las matrices sobre las que se ejecuta el flujo de trabajo secundario, así como transferir los valores para los elementos de dicha matriz cuando ejecute el flujo de trabajo. El flujo de trabajo secundario se ejecuta tantas veces como elementos se haya definido en la matriz.

Si tiene un elemento de configuración que contiene una matriz de atributos, puede ejecutar un flujo de trabajo que se itere sobre esos atributos en un elemento Foreach.

Por ejemplo, supongamos que tiene diez máquinas virtuales en una carpeta cuyo nombre desea cambiar. Para ello, debe insertar un elemento Foreach en un flujo de trabajo y definir el flujo de trabajo Cambiar nombre de máquina virtual como flujo de trabajo secundario en el elemento. El flujo de trabajo Cambiar nombre de máquina virtual toma dos parámetros de entrada, una máquina virtual y su nuevo nombre. Puede promocionar estos parámetros como entrada en el flujo de trabajo actual para que se conviertan en matrices sobre las que se iterará el flujo de trabajo Cambiar nombre de máquina virtual. Cuando ejecuta el flujo de trabajo, puede especificar las diez máquinas virtuales en la carpeta y sus nuevos nombres. Cada vez que se ejecuta el flujo de trabajo, toma un elemento de la matriz de máquinas virtuales y un elemento de la matriz de nombres nuevos para las máquinas virtuales.

### Utilizar tipos compuestos

Un tipo compuesto es un grupo de más de un parámetro o atributo de entrada que están conectados lógicamente, pero que son de tipos diferentes. En un elemento Foreach, puede enlazar un grupo de parámetros como valor compuesto. De este modo, el elemento Foreach toma los valores de los parámetros agrupados a la vez en cada ejecución subsiguiente del flujo de trabajo.

Por ejemplo, supongamos que va a cambiar el nombre de una máquina virtual. Necesita el objeto de máquina virtual y su nombre nuevo. Si debe cambiar el nombre de varias máquinas virtuales, necesita dos matrices, una para las máquinas virtuales y otra para sus nombres. Estas dos matrices no están conectadas de forma explícita. Un tipo compuesto permite tener una matriz en la que cada elemento contenga la máquina virtual y su nombre nuevo. De este modo, la conexión entre esos dos parámetros en caso de haber varios valores se especifica de forma explícita y no se deduce del esquema del flujo de trabajo.

---

**Nota** No es posible ejecutar un flujo de trabajo que contenga tipos compuestos desde vSphere Web Client.

---

## Definir un elemento Foreach

Si desea ejecutar un flujo de trabajo secundario varias veces pasando valores diferentes para sus parámetros o atributos en cada ejecución subsiguiente, puede insertar un elemento Foreach en el flujo de trabajo principal.

Al insertar un elemento Foreach, al menos se debe seleccionar una matriz en la cual se itera el elemento Foreach. Un elemento de matriz puede tener valores diferentes para cada ejecución de flujo de trabajo subsiguiente.

Si el flujo de trabajo secundario tiene parámetros de salida, debe seleccionar los parámetros de salida del elemento Foreach en el que acumular salidas de flujo de trabajo, con el fin de que el flujo de trabajo secundario también se pueda iterar en ellas.

### Requisitos previos

Abra un flujo de trabajo para editar en el Editor de flujos de trabajo.

### Procedimiento

- 1 En el Editor de flujos de trabajo, seleccione la pestaña **Esquema**.
- 2 En el menú **Genérico**, arrastre un elemento Foreach al esquema de flujo de trabajo.
- 3 Seleccione un flujo de trabajo en el cuadro de diálogo Selector.

En la parte superior del panel de esquema aparece la siguiente notificación.

¿Desea añadir los parámetros de la actividad como entrada/salida del flujo de trabajo actual?

- 4 En la notificación, haga clic en **Configuración**.

Se abrirá una ventana con las opciones disponibles.

5 Seleccione el tipo de asignación para cada parámetro de entrada.

Opción	Descripción
<b>Entrada</b>	El argumento se asigna al parámetro de flujo de trabajo de entrada.
<b>Omitir</b>	El argumento se asigna a un valor NULL.
<b>Valor</b>	El argumento se asigna a un atributo con un valor que puede configurar en la columna Valor.

6 Seleccione el tipo de asignación para cada parámetro de salida.

Opción	Descripción
<b>Salida</b>	El argumento se asigna al parámetro de flujo de trabajo de salida.
<b>Omitir</b>	El argumento se asigna a un valor NULL.
<b>Variable local</b>	El argumento se asigna a un atributo.

7 Haga clic en **Promocionar**.

8 Haga clic con el botón derecho en el elemento Foreach y seleccione **Sincronizar > Sincronizar presentación**.

Se muestra un cuadro de diálogo de confirmación.

9 Haga clic en **Aceptar** para propagar la presentación del elemento Foreach al flujo de trabajo actual.

En un cuadro de diálogo, se muestra información sobre el resultado de la operación.

10 En la pestaña **Entradas**, verifique que los parámetros del flujo de trabajo secundario se hayan añadido como elementos de matriz de tipo.

11 En la pestaña **Salidas**, verifique que los parámetros del flujo de trabajo secundario se hayan añadido como elementos de matriz de tipo.

Ha definido elemento Foreach en el flujo de trabajo. El elemento Foreach ejecuta un flujo de trabajo que toma como parámetros cada elemento de la matriz de parámetros o de atributos que haya definido.

En lo concerniente a los parámetros o los atributos que no estén definidos como matrices, el flujo de trabajo toma el mismo valor en cada ejecución subsiguiente.

### Ejemplo: Cambiar el nombre de máquinas virtuales con un elemento Foreach

Un elemento Foreach permite cambiar el nombre de varias máquinas virtuales de manera simultánea. Se debe insertar un elemento Foreach en un flujo de trabajo, y promover los parámetros `vm` y `newName` como entrada en el flujo de trabajo actual. De este modo, al ejecutar el flujo de trabajo, se especifican las máquinas virtuales cuyos nombres se deben cambiar y los nombres nuevos que tendrán. Las máquinas virtuales se incluyen como elementos en la matriz que se ha creado para el parámetro `vm`. Los nombres nuevos de las máquinas virtuales se incluyen en la matriz que se ha creado para el parámetro `newName`.

## Definición de un tipo compuesto en un elemento Foreach

Puede agrupar varios parámetros de flujo de trabajo que estén conectados lógicamente en un tipo nuevo denominado "tipo compuesto". Puede utilizar un elemento Foreach para enlazar un grupo de parámetros como valor compuesto para conectar varias matrices de parámetros en una sola matriz.

### Requisitos previos

- Abra un flujo de trabajo para editar en el Editor de flujos de trabajo.
- Compruebe que tenga un elemento Foreach en el flujo de trabajo.

### Procedimiento

- 1 Seleccione la pestaña **IN** o **OUT** del elemento Foreach.
- 2 Seleccione un parámetro local que desee agrupar con otros parámetros locales en un tipo compuesto.
- 3 Haga clic en **Enlazar un grupo de parámetros como valor compuesto** en la parte superior de la pestaña **IN** o **OUT**.
- 4 En el panel Enlaces, seleccione los parámetros que desee agrupar como tipo compuesto.
- 5 Seleccione **Enlazar como iterador**.

Ha establecido el elemento Foreach para que se itere en una matriz del tipo compuesto.

- 6 Haga clic en **Aceptar**.

Ha definido un tipo compuesto y se ha asegurado de que el flujo de trabajo se itere en una matriz de este tipo compuesto. Los parámetros agrupados como tipo compuesto se denominan *composite\_type\_name.parameter\_name*. Por ejemplo, si crea un tipo compuesto snapshots, los parámetros que forman un grupo en el tipo pueden ser `snapshots.vm[in-parameter]` o `snapshots.name[in-parameter]`. Cada elemento de la matriz del tipo compuesto contiene una sola instancia de cada parámetro que se ha agrupado en el tipo compuesto.

### Ejemplo: Cambio de nombre de las máquinas virtuales

Supongamos que desea cambiar el nombre de diez máquinas virtuales a la vez. Para ello, inserte un elemento Foreach en un flujo de trabajo y seleccione el flujo de trabajo Cambiar nombre de máquina virtual en el elemento. Crea un tipo compuesto para conectar los parámetros `vm` y `newName` de forma explícita. Enlaza el tipo compuesto como iterador, de modo que se crea una sola matriz que contiene los parámetros `vm` y `newName`.

## Añadir actividad de switch a un flujo de trabajo

Puede añadir actividad de switch básica a un esquema de flujo de trabajo que defina los casos de switch según los parámetros o los atributos del flujo de trabajo.

Toda actividad de switch puede tener varios casos de switch. Cualquier actividad de switch se define con una condición relativa a un atributo o un parámetro. Si se cumple la condición, el flujo de trabajo ejecuta switches para el elemento de flujo de trabajo correspondiente que se defina. Si no se cumple ninguna de las condiciones especificadas, la ejecución del flujo de trabajo cambia a un elemento de flujo de trabajo predeterminado que se defina.

---

**Importante** Los flujos de trabajo que contienen un elemento **Switch** son incompatibles con Orchestrator 5.5.x y versiones anteriores.

---

### Requisitos previos

Verifique que la pestaña **Esquema** del Editor de flujos de trabajo contenga elementos.

### Procedimiento

- 1 Arrastre un elemento **Switch** al elemento pertinente del esquema del flujo de trabajo.
- 2 Haga clic en el icono **Editar** (✎) del elemento **Switch**.
- 3 En la pestaña **Casos**, añada o elimine casos de switch.  
La prioridad de los casos de switch puede modificarse.
- 4 Defina la condición de cada caso de switch.
- 5 Seleccione el elemento de flujo de trabajo correspondiente para cada caso de switch.
- 6 Seleccione el elemento de flujo de trabajo predeterminado al que cambiar.
- 7 Haga clic en **Cerrar**.
- 8 Haga clic en **Guardar**.

Se han definido las condiciones de caso de switch y las rutas del flujo de trabajo.

## Desarrollar complementos

Orchestrator permite la integración con soluciones de administración mediante su arquitectura de complementos abierta. Puede utilizar el cliente de Orchestrator para ejecutar y crear flujos de trabajo de complementos y acceder a la API de complementos.

### Descripción general de los complementos

Los complementos de Orchestrator deben incluir un conjunto estándar de componentes y seguir una arquitectura estándar. Estas prácticas facilitan la creación de complementos para la mayor variedad posible de tecnologías externas.

#### ■ Estructura de un complemento de Orchestrator

Los complementos de Orchestrator tienen una estructura común compuesta de varios tipos de capa que implementan funciones específicas.

- **Exponer una API externa a Orchestrator**

Puede exponer una API de un producto externo a la plataforma Orchestrator creando un complemento de Orchestrator. Puede crear un complemento para cualquier tecnología que exponga una API que pueda asignar a objetos Javascript que use Orchestrator.

- **Componentes de un complemento**

Los complementos se componen de un conjunto estándar de componentes que exponen los objetos en la tecnología conectada a la plataforma de Orchestrator.

- **Función del archivo vso.xml**

El archivo `vso.xml` se utiliza para asignar los objetos, las clases, los métodos y los atributos de la tecnología conectada a los objetos de inventario y los atributos, así como a los tipos, las clases y los métodos de creación de scripts de Orchestrator. El archivo `vso.xml` también define la configuración y el comportamiento de inicio del complemento.

- **Funciones del adaptador de complemento**

El adaptador de complemento es el punto de entrada del complemento al servidor de Orchestrator. El adaptador de complemento sirve como almacén de datos para la tecnología conectada en el servidor de Orchestrator, crea la fábrica del complemento y administra los eventos que tienen lugar en la tecnología conectada.

- **Funciones de la fábrica del complemento**

La fábrica del complemento define el modo en el que Orchestrator busca objetos en la tecnología conectada y lleva a cabo operaciones en los objetos.

- **Función de los objetos de buscador**

Los objetos de buscador identifican y localizan determinadas instancias de los tipos de objetos administrados en la tecnología conectada. Orchestrator puede modificar e interactuar con los objetos que encuentre en la tecnología conectada ejecutando flujos de trabajo en los objetos de buscador.

- **Función de los objetos de creación de scripts**

Los objetos de creación de scripts son representaciones de JavaScript de objetos de la tecnología conectada. Los objetos de creación de scripts de los complementos aparecen en la API de JavaScript de Orchestrator, y se pueden usar en los elementos con script en los flujos de trabajo y las acciones.

- **Función de los gestores de eventos**

Los eventos son cambios en los estados o los atributos de los objetos que Orchestrator detecta en la tecnología conectada. Orchestrator supervisa los eventos implementando gestores de eventos.

## Estructura de un complemento de Orchestrator

Los complementos de Orchestrator tienen una estructura común compuesta de varios tipos de capa que implementan funciones específicas.

Las tres capas inferiores de un complemento de Orchestrator, que son clases de infraestructura, clases de ajuste y objetos de creación de scripts, implementan la conexión entre la tecnología conectada y Orchestrator.

Las partes de un complemento de Orchestrator que son visibles para el usuario son las tres capas superiores: acciones, bloques de creación y flujos de trabajo de alto nivel.

**Figura 1-3. Estructura de un complemento de Orchestrator**



**Clases de infraestructura**

Un conjunto de clases que proporciona la conexión entre la tecnología conectada y Orchestrator. Entre las clases de infraestructura están las utilizadas para implementaciones según la definición del complemento, por ejemplo fábrica de complementos, adaptador de complementos, etc. Las clases de infraestructura también incluyen las que proporcionan funciones para objetos y tareas comunes, por ejemplo para ayuda, almacenamiento en caché o inventario.

**Clases de ajuste**

Un conjunto de clases que adapta el modelo de objetos de la tecnología conectada al modelo de objetos que quiere exponer dentro de Orchestrator.

**Objetos de creación de scripts**

Son tipos de objeto de JavaScript que proporcionan acceso a las clases de ajuste, los métodos y los atributos en la tecnología conectada. En el archivo `vso.xml`, debe definir qué clases de ajuste, atributos y métodos de tecnología conectada se expondrán a Orchestrator.

**Acciones**

Un conjunto de funciones de JavaScript que puede usar directamente en flujos de trabajo y tareas de creación de scripts. Las acciones admiten múltiples parámetros de entrada y tienen un solo valor de devolución.

**Flujos de trabajo de bloques de creación**

Un conjunto de flujos de trabajo que cubren todas las funciones genéricas que quiera proporcionar con el complemento. Un flujo de trabajo de bloques de creación suele representar una operación en la interfaz de



usuario de la tecnología orquestada. Los flujos de trabajo de bloques de creación se pueden usar directamente o se pueden incluir en flujos de trabajo de alto nivel.

### **Flujos de trabajo de alto nivel**

Un conjunto de flujos de trabajo que cubren funciones específicas del complemento. Puede proporcionar flujos de trabajo de alto nivel para cumplir requisitos concretos o mostrar ejemplos complejos del uso de complementos.

## **Exponer una API externa a Orchestrator**

Puede exponer una API de un producto externo a la plataforma Orchestrator creando un complemento de Orchestrator. Puede crear un complemento para cualquier tecnología que exponga una API que pueda asignar a objetos Javascript que use Orchestrator.

Los complementos asignan objetos y métodos Java que añaden a la API de creación de scripts de Orchestrator. Si una tecnología externa expone una API de Java, puede asignar la API directamente a JavaScript para que Orchestrator la use en los flujos de trabajo y en las acciones.

Puede crear complementos para aplicaciones que exponen una API en un lenguaje distinto a Java utilizando WSDL (Web service definition language), REST (Representational state transfer) o un servicio de mensajería para integrar la API expuesta con objetos Java. A continuación, puede asignar los objetos Java integrados a JavaScript para que Orchestrator pueda usarlos.

La tecnología conectada es independiente de Orchestrator. Puede crear complementos de Orchestrator para productos externos aunque solo tenga acceso al código binario, por ejemplo en archivos Java (archivos JAR), en lugar de código fuente.

## **Componentes de un complemento**

Los complementos se componen de un conjunto estándar de componentes que exponen los objetos en la tecnología conectada a la plataforma de Orchestrator.

Los principales componentes de un complemento son el adaptador de complemento, la fábrica y las implementaciones de eventos. Los objetos y las operaciones que se han definido en el adaptador, la fábrica y las implementaciones de eventos se asignan a los objetos de Orchestrator en un archivo de definición XML denominado `vso.xml`. El archivo `vso.xml` asigna los objetos y las funciones de la tecnología conectada a los objetos de creación de scripts de JavaScript que aparecen en la API de JavaScript de Orchestrator. El archivo `vso.xml` también asigna tipos de objetos de la tecnología conectada a los buscadores, que aparecen en la pestaña **Inventario** de Orchestrator.

Los componentes incluyen los componentes que se indican a continuación.

### **Módulo de complemento**

El complemento propiamente dicho, como se define mediante un conjunto de clases de Java, un archivo `vso.xml` y paquetes de los flujos de trabajo

	y las acciones que interactúan con los objetos a los que se accede con el complemento. El módulo de complemento es obligatorio.
<b>Adaptador de complemento</b>	Define la interfaz entre la tecnología conectada y el servidor de Orchestrator. El adaptador es el punto de entrada del complemento a la plataforma de Orchestrator. El adaptador crea la fábrica del complemento, administra la carga y descarga del complemento, y gestiona los eventos que tienen lugar en los objetos en la tecnología conectada. El adaptador de complemento es obligatorio.
<b>Fábrica de complemento</b>	Define el modo en el que Orchestrator busca los objetos en la tecnología conectada y lleva a cabo operaciones con ellos. El adaptador crea una fábrica para la sesión de cliente que se abre entre Orchestrator y una tecnología conectada. La fábrica permite compartir una sesión entre todas las conexiones de cliente o abrir una sesión por conexión de cliente. La fábrica de complemento es obligatoria.
<b>Configuración</b>	Orchestrator no define un método estándar para que el complemento almacene su configuración. Puede guardar la información de configuración utilizando registros de Windows, archivos de configuración estáticos, bases de datos o archivos XML. Los complementos de Orchestrator se pueden configurar ejecutando flujos de trabajo de configuración en el cliente de Orchestrator.
<b>Buscadores</b>	Reglas de interacción que definen el modo en el que Orchestrator localiza y representa los objetos en la tecnología conectada. Los buscadores recuperan objetos del conjunto de objetos que expone la tecnología conectada a Orchestrator. En el archivo <code>vso.xml</code> , se definen las relaciones entre los objetos para permitir la navegación por la red de objetos. Orchestrator representa el modelo de objetos de la tecnología conectada en la pestaña <b>Inventario</b> . Los buscadores son obligatorios si se desea exponer objetos en la tecnología conectada a Orchestrator.
<b>Objetos de creación de scripts</b>	Tipos de objetos de JavaScript que proporcionan acceso a los objetos, las operaciones y los atributos en la tecnología conectada. Los objetos de creación de scripts definen la forma en la que Orchestrator accede al modelo de objeto de la tecnología conectada mediante JavaScript. Las clases y los métodos de la tecnología conectada se asignan a los objetos de JavaScript en el archivo <code>vso.xml</code> . Puede acceder a los objetos de JavaScript en la API de creación de scripts de Orchestrator e integrarlos en los flujos de trabajo, las acciones y las tareas de scripts de Orchestrator. Los objetos de creación de scripts son obligatorios si se desea añadir métodos, clases y tipos de creación de scripts a la API de JavaScript de Orchestrator.
<b>Inventario</b>	Las instancias de los objetos en la tecnología conectada que Orchestrator localiza utilizando buscadores aparecen en la vista <b>Inventario</b> en el cliente

de Orchestrator. Puede realizar operaciones en los objetos del inventario ejecutando flujos de trabajo en ellos. El inventario es opcional. Puede crear un complemento que solo añada clases y tipos de creación de scripts a la API de JavaScript de Orchestrator y no exponga ninguna instancia de objetos en el inventario.

## Eventos

Cambios en el estado de un objeto en la tecnología conectada. Orchestrator puede escuchar de forma pasiva los eventos que ocurren en la tecnología conectada. Orchestrator también puede activar eventos en la tecnología conectada. Los eventos son opcionales.

## Función del archivo vso.xml

El archivo `vso.xml` se utiliza para asignar los objetos, las clases, los métodos y los atributos de la tecnología conectada a los objetos de inventario y los atributos, así como a los tipos, las clases y los métodos de creación de scripts de Orchestrator. El archivo `vso.xml` también define la configuración y el comportamiento de inicio del complemento.

El archivo `vso.xml` realiza las funciones principales siguientes.

<b>Comportamiento de inicio y de configuración</b>	Define el modo en que se inicia el complemento y localiza las implementaciones de configuración que define el complemento. Carga el adaptador del complemento.
<b>Objetos de inventario</b>	Define los tipos de objetos a los que accede el complemento en la tecnología conectada. Los métodos de buscador de la implementación de fábrica del complemento localizan las instancias de estos objetos y las muestran en el inventario de Orchestrator.
<b>Tipos de creación de scripts</b>	Añade tipos de creación de scripts a la API de JavaScript de Orchestrator para representar los distintos tipos de objetos en el inventario. Puede utilizar estos tipos de creación de scripts como parámetros en los flujos de trabajo.
<b>Clases de creación de scripts</b>	Añade clases a la API de JavaScript de Orchestrator que puede utilizar en los elementos con script de los flujos de trabajo, las acciones, las políticas, etc.
<b>Métodos de creación de scripts</b>	Añade métodos a la API de JavaScript de Orchestrator que puede utilizar en elementos con scripts en los flujos de trabajo, las acciones, las políticas, etc.
<b>Atributos de creación de scripts</b>	Añade los atributos de los objetos en la tecnología conectada a la API de JavaScript de Orchestrator que puede utilizar en los elementos con scripts en los flujos de trabajo, las acciones, las políticas, etc.

## Funciones del adaptador de complemento

El adaptador de complemento es el punto de entrada del complemento al servidor de Orchestrator. El adaptador de complemento sirve como almacén de datos para la tecnología conectada en el servidor de Orchestrator, crea la fábrica del complemento y administra los eventos que tienen lugar en la tecnología conectada.

Para crear un adaptador de complemento, cree una clase de Java que implemente la interfaz de `IPluginAdaptor`.

La clase del adaptador de complemento que cree administra la fábrica, los eventos y los activadores del complemento en la tecnología conectada. La interfaz `IPluginAdaptor` proporciona métodos que se utilizan para llevar a cabo estas tareas.

El adaptador de complemento desempeña las funciones principales siguientes.

<b>Crea una fábrica</b>	La función más importante del adaptador de complemento es cargar y descargar una instancia de fábrica de complemento para cada conexión de Orchestrator con la tecnología conectada. La clase del adaptador de complemento llama al método <code>IPluginAdaptor.createPluginFactory()</code> para crear una instancia de una clase que implementa la interfaz de <code>IPluginFactory</code> .
<b>Administra eventos</b>	El adaptador de complemento es la interfaz entre el servidor de Orchestrator y la tecnología conectada. El adaptador de complemento administra los eventos que realiza o supervisa Orchestrator en los objetos de la tecnología conectada. El adaptador administra eventos a través de los publicadores de eventos. Los publicadores de eventos son instancias de la interfaz de <code>IPluginEventPublisher</code> que crea el adaptador llamando al método <code>IPluginAdaptor.registerEventPublisher()</code> . Los publicadores de eventos establecen activadores y medidores en los objetos de la tecnología conectada, para permitir a Orchestrator lanzar acciones definidas si se producen determinados eventos en el objeto o si los valores del objeto superan ciertos umbrales. Asimismo, puede definir las instancias de <code>PluginTrigger</code> y de <code>PluginWatcher</code> que definen eventos que esperan los elementos Evento de espera en flujos de trabajo de larga ejecución.
<b>Establece el nombre del complemento</b>	El nombre del complemento se indica en el archivo <code>vso.xml</code> . El adaptador de complemento obtiene este nombre del archivo <code>vso.xml</code> y lo publica en la vista <b>Inventario</b> en el cliente de Orchestrator.
<b>Instala licencias</b>	Puede llamar a métodos para instalar los archivos de licencia que requiera la tecnología conectada en la implementación del adaptador.

Para obtener más información sobre la interfaz de `IPluginAdaptor`, todos sus métodos y todas las demás clases de la API del complemento, consulte [Referencia de API del complemento de Orchestrator](#).

## Funciones de la fábrica del complemento

La fábrica del complemento define el modo en el que Orchestrator busca objetos en la tecnología conectada y lleva a cabo operaciones en los objetos.

Para crear la fábrica del complemento, debe implementar y extender la interfaz de `IPluginFactory` desde la API del complemento de Orchestrator. La clase de fábrica del complemento que cree define las funciones de buscador que Orchestrator utiliza para acceder a objetos en la tecnología conectada. La fábrica permite al servidor de Orchestrator buscar objetos por su ID, su relación con otros objetos o una cadena de consulta.

La fábrica de complemento desempeña las funciones principales siguientes.

<b>Busca objetos</b>	Puede crear funciones que busquen objetos según su nombre y su tipo. Para buscar objetos por nombre y tipo, utilice el método <code>IPluginFactory.find()</code> .
<b>Buscar objetos en relación con otros objetos</b>	Puede crear funciones para buscar objetos que se relacionen con un objeto determinado mediante un determinado tipo de relación. Las relaciones se definen en el archivo <code>vso.xml</code> . También puede crear buscadores para buscar objetos secundarios dependientes que se relacionan con todos los objetos principales mediante un determinado tipo de relación. El método <code>IPluginFactory.findRelation()</code> permite buscar objetos que se relacionen con un determinado objeto principal mediante un tipo concreto de relación. El método <code>IPluginFactory.hasChildrenInRelation()</code> permite descubrir si existe al menos un objeto secundario en una instancia principal.
<b>Definir consultas para buscar objetos según sus propios criterios</b>	Puede crear buscadores de objetos que implementen reglas de consulta que se definan. El método <code>IPluginFactory.findAll()</code> permite buscar todos los objetos que cumplan las reglas de consulta que se definan cuando la fábrica llame a este método. Los resultados del método <code>findAll()</code> se obtienen en un objeto <code>QueryResult</code> que contiene una lista de todos los objetos encontrados que coincidan con las reglas de consulta definidas.

Para obtener más información sobre la interfaz de `IPluginFactory`, todos sus métodos y todas las demás clases de la API del complemento, consulte [Referencia de API del complemento de Orchestrator](#).

## Función de los objetos de buscador

Los objetos de buscador identifican y localizan determinadas instancias de los tipos de objetos administrados en la tecnología conectada. Orchestrator puede modificar e interactuar con los objetos que encuentre en la tecnología conectada ejecutando flujos de trabajo en los objetos de buscador.

Cualquier instancia de un determinado tipo de objeto administrado en la tecnología conectada debe tener un identificador único para que los objetos de buscador de Orchestrator la puedan encontrar. La tecnología conectada proporciona los identificadores únicos para las instancias de objetos como cadenas. Cuando se ejecuta un flujo de trabajo, Orchestrator establece los identificadores únicos de los objetos que encuentra como valores de atributo de flujo de trabajo. Los flujos de trabajo que requieren un objeto de un tipo concreto como parámetro de entrada se ejecutan en una instancia específica de ese tipo de objeto.

Los objetos de buscador que añaden los complementos a la API de JavaScript de Orchestrator tienen el nombre de complemento como prefijo. Por ejemplo, el tipo de objeto administrado `VirtualMachine` de la API de vCenter Server aparece en Orchestrator como tipo `VC:VirtualMachine` de JavaScript.

Por ejemplo, Orchestrator accede a una instancia `VC:VirtualMachine` específica a través del complemento de vCenter Server implementando un objeto de buscador que utiliza el atributo `id` de la máquina virtual como identificador único. Puede transferir esta instancia de objeto a los elementos de flujo de trabajo como valores de atributo.

Un complemento de Orchestrator asigna los objetos de la tecnología conectada a objetos de buscador equivalentes de Orchestrator en los elementos `<finder>` del archivo `vso.xml`. Los elementos `<finder>` identifican el método o la función de la tecnología conectada que obtiene el identificador único para una instancia determinada de un objeto. Los elementos `<finder>` también definen relaciones entre los objetos, para buscar los objetos por la forma en que se relacionan entre ellos.

Los objetos de buscador aparecen en la pestaña **Inventario** de Orchestrator, bajo el complemento que los contiene.

## Función de los objetos de creación de scripts

Los objetos de creación de scripts son representaciones de JavaScript de objetos de la tecnología conectada. Los objetos de creación de scripts de los complementos aparecen en la API de JavaScript de Orchestrator, y se pueden usar en los elementos con script en los flujos de trabajo y las acciones.

Los objetos de creación de scripts de los complementos aparecen en la API de JavaScript de Orchestrator como módulos, tipos y clases de JavaScript. La mayoría de los objetos de buscador tienen una representación de objeto de creación de scripts. Las clases de JavaScript pueden añadir métodos y atributos a la API de JavaScript de Orchestrator que representan los métodos y los atributos de los objetos de la API de la tecnología conectada. La tecnología conectada proporciona las implementaciones de los objetos, los tipos, las clases, los atributos y los métodos, independientemente de Orchestrator. Por ejemplo, el complemento de vCenter Server representa todos los objetos de la API de vCenter Server como objetos JavaScript en la API de JavaScript de Orchestrator, con representaciones JavaScript de todas las clases, los métodos y los atributos que define la API de vCenter Server. Puede utilizar las clases de creación de scripts de vCenter Server, así como los métodos y los atributos que definen en las funciones con scripts de Orchestrator.

Por ejemplo, el tipo de objeto administrado `VirtualMachine` de la API de vCenter Server se detecta mediante el buscador `VC:VirtualMachine` y aparece en la API de JavaScript de Orchestrator como clase `VcVirtualMachine` de JavaScript. La clase `VcVirtualMachine` de JavaScript en la API de JavaScript de Orchestrator define los mismos métodos y atributos que el objeto administrado `VirtualMachine` de la API de vCenter Server.

Un complemento de Orchestrator asigna los objetos, los tipos, las clases, los atributos y los métodos de la tecnología conectada a los objetos, los tipos, las clases, los atributos y los métodos JavaScript de Orchestrator en el elemento `<scripting-objects>` en el archivo `vso.xml`.

## Función de los gestores de eventos

Los eventos son cambios en los estados o los atributos de los objetos que Orchestrator detecta en la tecnología conectada. Orchestrator supervisa los eventos implementando gestores de eventos.

Los complementos de Orchestrator permiten supervisar eventos en una tecnología conectada de distintas formas. La API del complemento de Orchestrator permite crear los siguientes tipos de gestores de eventos para supervisar los eventos en una tecnología conectada.

### Escuchas

Supervise de forma pasiva los objetos de la tecnología conectada para detectar cambios en su estado. La implementación del complemento o DE la tecnología conectada define los eventos supervisados por las escuchas. Las escuchas no inician eventos, pero notifican a Orchestrator cuando se producen los eventos. Las escuchas detectan los eventos sondeando la tecnología conectada o recibiendo notificaciones de ella. Cuando se producen los eventos, las políticas o los flujos de trabajo de Orchestrator que esperan el evento pueden reaccionar iniciando operaciones en el servidor de Orchestrator. Los componentes de escucha son opcionales.

### Políticas

Supervise determinados elementos en la tecnología conectada e inicie operaciones en el servidor de Orchestrator si se producen los eventos. Las políticas pueden supervisar los activadores y los medidores de políticas. Los activadores de políticas definen un evento en la tecnología conectada que, cuando se produce, provoca que una política en ejecución inicie una operación en el servidor de Orchestrator, por ejemplo la ejecución de un flujo de trabajo. Los medidores de políticas definen los rangos de valores para los atributos de un objeto en la tecnología conectada; cuando se superan, hacen que Orchestrator inicie una operación. Las políticas son opcionales.

### Activadores de flujo de trabajo

Si un flujo de trabajo en ejecución contiene un elemento Evento de espera, cuando alcanza dicho elemento, suspende su ejecución y espera a que se produzca un evento en la tecnología conectada. Los activadores de flujo de trabajo definen los eventos en la tecnología conectada a los que esperan

los elementos Evento de espera en los flujos de trabajo. Los activadores de flujo de trabajo se registran con monitores. Además, son opcionales.

## Monitores

Supervise los activadores de flujo de trabajo para un determinado evento en la tecnología conectada, en nombre de un elemento Evento de espera en un flujo de trabajo. Cuando tiene lugar el evento, los monitores lo notifican a los flujos de trabajo que esperan dicho evento. Los monitores son opcionales.

## Contenido y estructura de un complemento

Los complementos de Orchestrator deben contener un conjunto estándar de componentes y seguir una estructura de archivos estándar. Para que un complemento siga la estructura de archivos estándar, debe incluir carpetas y archivos específicos.

Para crear un complemento de Orchestrator, defina el modo en que Orchestrator accede e interactúa con los objetos en la tecnología conectada. Asigne también todos los objetos y las funciones de la tecnología conectada a los objetos y las funciones correspondientes de Orchestrator en el archivo `vso.xml`.

El archivo `vso.xml` debe incluir una referencia a todos los tipos de objeto u operaciones que se exponen a Orchestrator. Se debe proporcionar un identificador único para cada objeto que encuentre el complemento en la tecnología conectada. Defina los nombres de objeto en los elementos `finder` y en los elementos de objeto en el archivo `vso.xml`.

Un complemento se puede entregar como archivo de Java estándar (JAR) o como archivo ZIP; en cualquier caso, debe cambiarse su extensión a `.dar`.

---

**Nota** Puede utilizar el centro de control de Orchestrator para importar un archivo DAR al servidor de Orchestrator.

---

- **Definir la asignación de aplicaciones en el archivo `vso.xml`**

Los objetos que incluye en el archivo `vso.xml` aparecen como objetos de creación de scripts en la API de creación de scripts de Orchestrator, o como objetos de buscador en la pestaña **Inventario** de Orchestrator.

- **Formato del archivo de definición del complemento `vso.xml`**

El archivo `vso.xml` define la manera que el servidor de Orchestrator tiene de interactuar con la tecnología conectada. Debe incluir una referencia a cada tipo de objeto o de operación para exponer a Orchestrator en el archivo `vso.xml`.

- **Designar objetos de complemento**

Se debe proporcionar un identificador exclusivo para cada objeto que encuentre el complemento en la tecnología conectada. Los nombres de objetos se definen en los elementos `<finder>` y `<object>` del archivo `vso.xml`.

- **Convenciones de nomenclatura de objetos de complemento**

Al asignar nombres a objetos de complementos, debe seguir las convenciones de las clases de Java.



## ■ Estructura de archivos del complemento

Un complemento debe seguir una estructura de archivos estándar, así como incluir determinados archivos y carpetas. Un complemento se entrega como archivo Java (JAR) o ZIP estándar, cuyo nombre se debe cambiar con la extensión `.dar`.

## Definir la asignación de aplicaciones en el archivo `vso.xml`

Los objetos que incluye en el archivo `vso.xml` aparecen como objetos de creación de scripts en la API de creación de scripts de Orchestrator, o como objetos de buscador en la pestaña **Inventario** de Orchestrator.

El archivo `vso.xml` proporciona la información siguiente al servidor de Orchestrator:

- Un nombre de versión y la descripción del complemento
- Referencias a las clases de la tecnología conectada y al adaptador de complementos asociado
- Inicializa el complemento cuando se inicia el servidor de Orchestrator
- Tipos de creación de scripts para representar los tipos de objetos en la tecnología conectada
- Relaciones entre tipos de objetos para definir el modo en el que los objetos se visualizan en el inventario de Orchestrator
- Clases de creación de scripts que asignan los objetos y las operaciones de la tecnología conectada a funciones y tipos de objetos en la API de JavaScript de Orchestrator
- Enumeraciones para definir una lista de valores constantes que se aplican a todos los objetos de un tipo concreto
- Eventos que Orchestrator supervisa en la tecnología conectada

El archivo `vso.xml` debe seguir la definición del esquema XML de los complementos de Orchestrator. Se puede acceder a la definición del esquema en el sitio de soporte de VMware.

```
http://www.vmware.com/support/orchestrator/plugin-4-1.xsd
```

Para obtener descripciones de todos los elementos del archivo `vso.xml`, consulte [Elementos del archivo de definición del complemento `vso.xml`](#).

## Formato del archivo de definición del complemento `vso.xml`

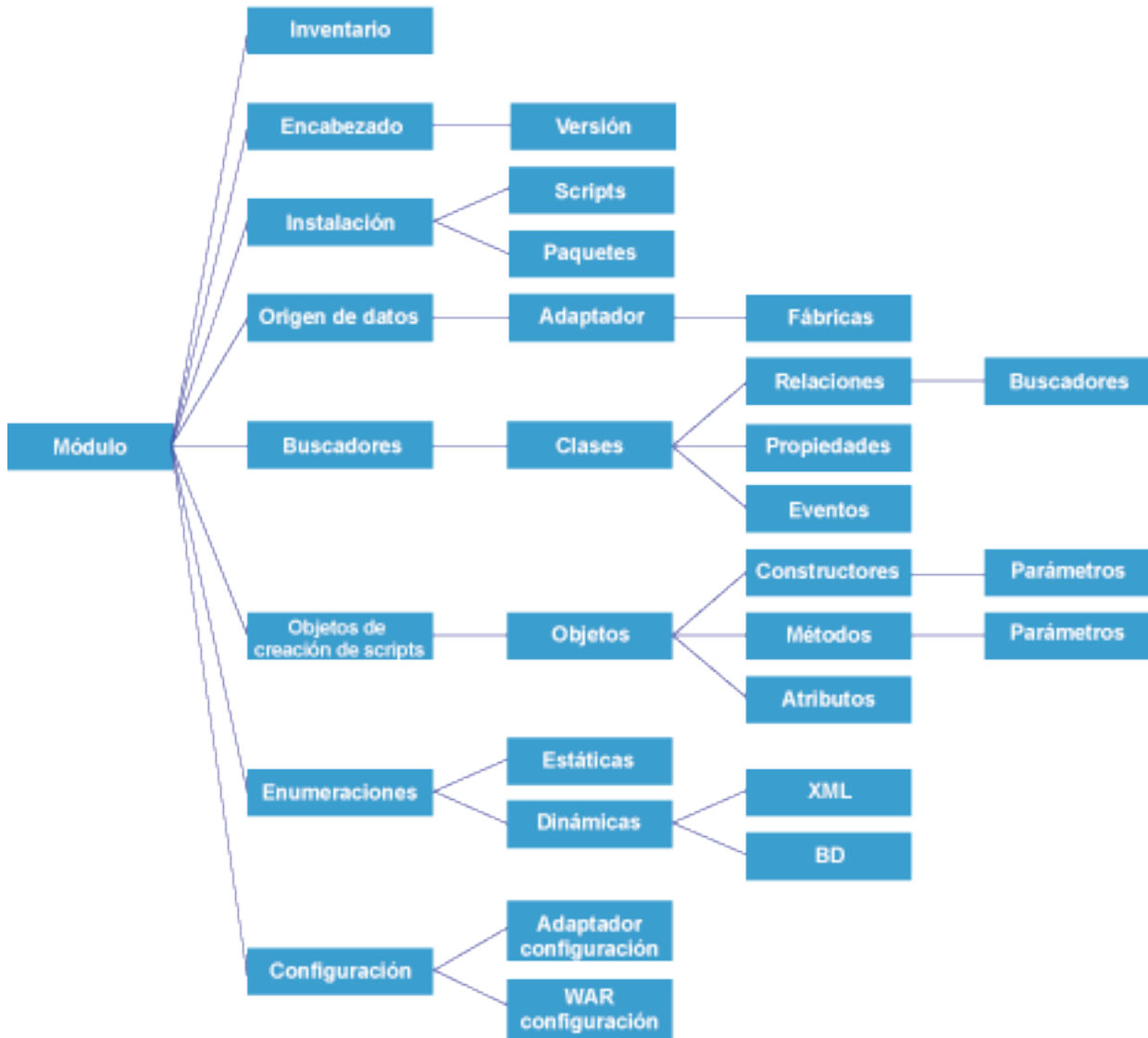
El archivo `vso.xml` define la manera que el servidor de Orchestrator tiene de interactuar con la tecnología conectada. Debe incluir una referencia a cada tipo de objeto o de operación para exponer a Orchestrator en el archivo `vso.xml`.

Los objetos que incluye en el archivo `vso.xml` aparecen como objetos de creación de scripts en la API de creación de scripts de Orchestrator o como objetos de buscador en la pestaña **Inventario** de Orchestrator.

Como parte de la arquitectura abierta y la implementación estandarizada de complementos, el archivo `vso.xml` debe tener un formato estándar.

El diagrama siguiente muestra el formato del archivo de definición del complemento `vso.xml` y cómo se anidan los elementos.

**Figura 1-4. Formato del archivo de definición del complemento `vso.xml`**



## Designar objetos de complemento

Se debe proporcionar un identificador exclusivo para cada objeto que encuentre el complemento en la tecnología conectada. Los nombres de objetos se definen en los elementos `<finder>` y `<object>` del archivo `vso.xml`.

Las operaciones del buscador que se definen en la implementación de fábrica buscan objetos en la tecnología conectada. Cuando el complemento encuentra objetos, puede utilizarlos en los flujos de trabajo de Orchestrator y transferirlos de un elemento de flujo de trabajo a otro. Los identificadores únicos que proporciona para los objetos permiten transferirlos entre los elementos de un flujo de trabajo.

El servidor de Orchestrator guarda solo el tipo y el identificador de cada objeto que procesa; no almacena ningún dato sobre el lugar o el modo en el que Orchestrator ha obtenido el objeto. Debe designar los objetos de forma coherente en la implementación del complemento para poder realizar un seguimiento de los objetos que se obtienen de los complementos.

Si el servidor de Orchestrator se detiene mientras los flujos de trabajo se ejecutan, cuando reinicia el servidor, los flujos de trabajo se reanudan en el elemento del flujo de trabajo que se estaba ejecutando cuando se detuvo el servidor. El flujo de trabajo utiliza identificadores para recuperar los objetos que el elemento estaba procesando cuando se detuvo el servidor.

## Convenciones de nomenclatura de objetos de complemento

Al asignar nombres a objetos de complementos, debe seguir las convenciones de las clases de Java.

---

**Importante** Debido al modo en que el motor de flujos de trabajo realiza la serialización de datos, no use las secuencias citadas a continuación en nombres de objeto. Si se usan esas secuencias de caracteres en los identificadores de objeto, el motor de flujos de trabajo analizará los flujos incorrectamente y la ejecución de flujos podría tener resultados imprevistos.

- # ; #
  - # , #
  - # = #
- 

Use estas directrices al asignar nombres a objetos de complementos:

- Use una inicial mayúscula para cada palabra del nombre.
- No use espacios para separar palabras.
- Como letras, use únicamente los caracteres estándar A - Z y a - z.
- No use caracteres especiales (por ejemplo, acentos).
- No use un número como primer carácter de un nombre.
- Si es posible, use menos de 10 caracteres.

La [Tabla 1-5. Reglas de nomenclatura de objetos de complemento](#) muestra reglas aplicables a tipos de objeto individuales.

**Tabla 1-5. Reglas de nomenclatura de objetos de complemento**

Tipo de objeto	Reglas de nomenclatura
Complemento	<ul style="list-style-type: none"> <li>■ Se define en el elemento <code>&lt;module&gt;</code> del archivo <code>vso.xml</code>.</li> <li>■ Debe ser conforme a las convenciones de nomenclatura de las clases de Java.</li> <li>■ Debe ser un nombre exclusivo. No se pueden ejecutar dos complementos con el mismo nombre en un servidor de Orchestrator.</li> </ul>
Objeto de buscador	<ul style="list-style-type: none"> <li>■ Se define en los elementos <code>&lt;finder&gt;</code> del archivo <code>vso.xml</code>.</li> <li>■ Debe ser conforme a las convenciones de nomenclatura de las clases de Java.</li> <li>■ Debe ser un nombre exclusivo en el complemento.</li> </ul> <p>Orchestrator añade el nombre del complemento y el signo de dos puntos a los nombres de objeto de buscador en los tipos de objeto de buscador de la API de creación de scripts de Orchestrator. Por ejemplo, el tipo de objeto <code>VirtualMachine</code> del complemento de vCenter Server aparece en la API de creación de scripts de Orchestrator como <code>VC:VirtualMachine</code>.</p>
Objeto de creación de scripts	<ul style="list-style-type: none"> <li>■ Se define en los elementos <code>&lt;scripting-object&gt;</code> del archivo <code>vso.xml</code>.</li> <li>■ Debe ser conforme a las convenciones de nomenclatura de las clases de Java.</li> <li>■ Debe ser un nombre exclusivo en el servidor de Orchestrator.</li> <li>■ Para evitar confundir objetos de script con objetos de buscador que tengan el mismo nombre, o con objetos de script de otros complementos, añada siempre como prefijo el nombre del complemento al nombre del objeto, pero sin un signo de dos puntos. Por ejemplo, la clase <code>VirtualMachine</code> del complemento de vCenter Server aparece en la API de creación de scripts de Orchestrator como la clase <code>VcVirtualMachine</code>.</li> </ul>

## Estructura de archivos del complemento

Un complemento debe seguir una estructura de archivos estándar, así como incluir determinados archivos y carpetas. Un complemento se entrega como archivo Java (JAR) o ZIP estándar, cuyo nombre se debe cambiar con la extensión `.dar`.

El contenido del archivo DAR debe seguir las convenciones siguientes de nomenclatura y de estructura de archivos.

**Tabla 1-6. Estructura del archivo DAR**

<b>Carpetas</b>	<b>Descripción</b>
<code>plug-in_name\VS0-INF\</code>	<p>Contiene el archivo <code>vso.xml</code> que define la asignación de los objetos en la tecnología conectada a objetos de Orchestrator. La carpeta <code>VS0-INF</code> y el archivo <code>vso.xml</code> son obligatorios.</p>
<code>plug-in_name\lib\</code>	<p>Contiene los archivos JAR con los archivos binarios de la tecnología conectada. También contiene archivos JAR con las implementaciones del adaptador, fábrica, controladores de notificaciones y otras interfaces del complemento. La carpeta <code>lib</code> y los archivos JAR son obligatorios.</p>
<code>plug-in_name\resources\</code>	<p>Contiene los archivos de recursos necesarios para el complemento. La carpeta <code>resources</code> puede incluir los tipos de elemento siguientes:</p> <ul style="list-style-type: none"> <li>■ Archivos de imagen, para representar los objetos del complemento en la pestaña <b>Inventario</b> de Orchestrator.</li> <li>■ Scripts, para definir el comportamiento de inicialización cuando se inicia el complemento.</li> <li>■ Paquetes de Orchestrator, que pueden contener acciones, flujos de trabajo personalizados y otros recursos que interactúan con los objetos a los que se accede mediante el complemento.</li> </ul> <p>Los recursos se pueden organizar en subcarpetas. Por ejemplo, <code>resources\images\</code>, <code>resources\scripts\</code> o <code>resources\packages\</code>.</p> <p>La carpeta <code>resources</code> es opcional.</p>

Utilice el centro de control de Orchestrator para importar un archivo DAR al servidor de Orchestrator.

## Referencia de API del complemento de Orchestrator

La API del complemento de Orchestrator define las clases y las interfaces de Java para implementar y extender cuando se desarrollan las implementaciones de `IPluginAdaptor` y `IPluginFactory` para crear un complemento.

Todas las clases están incluidas en el paquete `ch.dunes.vso.sdk.api`, a menos que se indique lo contrario.

### Interfaz `IAop`

La interfaz `IAop` proporciona métodos para obtener y definir propiedades sobre objetos en la tecnología conectada.

```
public interface IAop
```

La interfaz `IAop` define los siguientes métodos:

Método	Devuelve	Descripción
<code>get(java.lang.String propertyName, java.lang.Object object, java.lang.Object sdkObject)</code>	<code>java.lang.Object</code>	Obtiene una propiedad de un determinado objeto del complemento.
<code>set(java.lang.String propertyName, java.lang.String propertyValue, java.lang.Object object)</code>	Vacío	Define una propiedad de un determinado objeto del complemento.

## Interfaz IDynamicFinder

La interfaz IDynamicFinder devuelve el ID de las propiedades de un buscador mediante programación, en lugar de definir el ID y las propiedades en el archivo `vso.xml`.

La interfaz IDynamicFinder define los métodos siguientes:

Método	Devuelve	Descripción
<code>getIdAccessor(java.lang.String type)</code>	<code>java.lang.String</code>	Proporciona una expresión OGNL para obtener un ID de objeto mediante programación.
<code>getProperties(java.lang.String type)</code>	<code>java.util.List&lt;SDKFinderProperty&gt;</code>	Proporciona una lista de propiedades de objeto mediante programación.

## Interfaz IPluginAdaptor

La interfaz IPluginAdaptor se implementa para administrar fábricas, eventos y monitores del complemento. La interfaz IPluginAdaptor define un adaptador entre el complemento y el servidor de Orchestrator.

Las instancias de IPluginAdaptor se encargan de administrar las sesiones. La interfaz IPluginAdaptor define los métodos siguientes:

Método	Devuelve	Descripción
<code>addWatcher(PluginWatcher watcher)</code>	Vacío	Añade un monitor para supervisar un evento específico.
<code>createPluginFactory(java.lang.String sessionId, java.lang.String username, java.lang.String password, IPluginNotificationHandler notificationHandler)</code>	<code>IPluginFactory</code>	<p>Crea una instancia de <code>IPluginFactory</code>. El servidor de Orchestrator utiliza la fábrica para obtener objetos de la tecnología conectada por su ID, su relación con otros objetos, etcétera.</p> <p>El ID de sesión permite identificar una sesión que se está ejecutando. Por ejemplo, un usuario puede iniciar sesión en diferentes clientes de Orchestrator y ejecutar dos sesiones a la vez.</p> <p>De forma similar, el inicio de un flujo de trabajo crea una sesión independiente del cliente en el que se ha iniciado el flujo de trabajo. Un flujo de trabajo sigue ejecutándose aunque se cierre el cliente de Orchestrator.</p>

Método	Devuelve	Descripción
<code>installLicenses(PluginLicense[] licenses)</code>	Vacío	Instala la información de licencia de los complementos estándar que proporciona VMware.
<code>registerEventPublisher(java.lang.String type, java.lang.String id, IPluginEventPublisher publisher)</code>	Vacío	Establece los activadores y los medidores en un elemento del inventario.
<code>removeWatcher(java.lang.String watcherId)</code>	Vacío	Elimina un monitor.
<code>setPluginName(java.lang.String pluginName)</code>	Vacío	Obtiene el nombre del complemento del archivo <code>vso.xml</code> .
<code>setPluginPublisher(IPluginPublisher pluginPublisher)</code>	Vacío	Establece el publicador del complemento.
<code>uninstallPluginFactory(IPluginFactory plugin)</code>	Vacío	Desinstala una fábrica de complemento.
<code>unregisterEventPublisher(java.lang.String type, java.lang.String id, IPluginEventPublisher publisher)</code>	Vacío	Elimina los activadores y los medidores de un elemento del inventario.

## Interfaz IPluginEventPublisher

La interfaz `IPluginEventPublisher` publica medidores y activadores en un bus de notificación de eventos para la supervisión mediante políticas de Orchestrator.

Puede crear instancias de `IPluginEventPublisher` directamente en la implementación del adaptador de complementos o bien crearlas en clases de generadores de eventos independientes.

Puede implementar la interfaz `IPluginEventPublisher` para publicar eventos de la tecnología conectada en el motor de políticas de Orchestrator. Debe crear métodos para configurar medidores y activadores de políticas en objetos de la tecnología conectada y escuchas de eventos relacionados con esos objetos.

Las políticas pueden implementar medidores o activadores para supervisar objetos en la tecnología conectada. Los medidores de políticas supervisan los atributos de objetos e insertan un evento en el servidor de Orchestrator si los valores del objeto superan ciertos límites. Los activadores de políticas supervisan e insertan un evento en el servidor de Orchestrator si se produce un evento definido en el objeto. Debe registrar activadores y medidores de políticas con instancias `IPluginEventPublisher` para que las políticas de Orchestrator los supervisen.

La interfaz `IPluginEventPublisher` define los métodos siguientes:

Tipo	Devuelve	Descripción
<code>pushGauge(java.lang.String type, java.lang.String id, java.lang.String gaugeName, java.lang.String deviceName, java.lang.Double gaugeValue)</code>	Vacío	<p>Publica un medidor para la supervisión mediante políticas. Obtiene los parámetros siguientes:</p> <ul style="list-style-type: none"> <li>■ <code>type</code>: tipo de objeto que se supervisa.</li> <li>■ <code>id</code>: identificador del objeto que se supervisa.</li> <li>■ <code>gaugeName</code>: nombre de este medidor.</li> <li>■ <code>deviceName</code>: nombre del tipo de atributo que supervisa el medidor.</li> <li>■ <code>gaugeValue</code>: valor para el que el medidor supervisa el objeto.</li> </ul>
<code>pushTrigger(java.lang.String type, java.lang.String id, java.lang.String triggerName, java.util.Properties additionalProperties)</code>	Vacío	<p>Publica un activador para la supervisión mediante políticas. Obtiene los parámetros siguientes:</p> <ul style="list-style-type: none"> <li>■ <code>type</code>: tipo de objeto que se supervisa.</li> <li>■ <code>id</code>: identificador del objeto que se supervisa.</li> <li>■ <code>triggerName</code>: nombre de este activador.</li> <li>■ <code>additionalProperties</code>: cualquier propiedad adicional para la supervisión mediante el activador.</li> </ul>

## Interfaz IPluginFactory

IPluginAdaptor devuelve instancias de IPluginFactory. Las instancias de IPluginFactory ejecutan comandos en la aplicación conectada y busca objetos con los que realizar operaciones de Orchestrator.

La interfaz IPluginFactory define el campo siguiente:

```
static final java.lang.String RELATION_CHILDREN
```

La interfaz IPluginFactory define los métodos siguientes:

Método	Devuelve	Descripción
<code>executePluginCommand(java.lang.String cmd)</code>	Vacío	Utilice el complemento para ejecutar un comando. VMware recomienda no utilizar este método.
<code>find(java.lang.String type, java.lang.String id)</code>	<code>java.lang.Object</code>	Utilice un complemento para buscar un objeto. Identifique el objeto por su ID y su tipo.
<code>findAll(java.lang.String type, java.lang.String query)</code>	<code>QueryResult</code>	Utilice el complemento para buscar objetos de un tipo que coincidan con una cadena de consulta. La sintaxis de la consulta se define en la implementación IPluginFactory del complemento. Si no define la sintaxis de la consulta, <code>findAll()</code> devuelve todos los objetos del tipo especificado.



Método	Devuelve	Descripción
<code>findRelation(java.lang.String parentType, java.lang.String parentId, java.lang.String relationName)</code>	<code>java.util.List</code>	Determina si un objeto tiene elementos secundarios.
<code>hasChildrenInRelation(java.lang.String parentType, java.lang.String parentId, java.lang.String relationName)</code>	<code>HasChildrenResult</code>	Busca todos los elementos secundarios relativos a un determinado elemento principal mediante una relación concreta.
<code>invalidate(java.lang.String type, java.lang.String id)</code>	Vacío	Invalida los objetos por tipo y por ID.
<code>void invalidateAll()</code>	Vacío	Invalida todos los objetos de la caché.

## Interfaz IPluginNotificationHandler

IPluginNotificationHandler define métodos para notificar a Orchestrator los distintos tipos de evento que se producen en los objetos a los que accede Orchestrator a través del complemento.

La interfaz IPluginNotificationHandler define los métodos siguientes:

Método	Devuelve	Descripción
<code>getSessionID()</code>	<code>java.lang.String</code>	Devuelve el ID de sesión actual.
<code>notifyElementDeleted(java.lang.String type, java.lang.String id)</code>	Vacío	Notifica al sistema que se ha eliminado un objeto con el tipo y el ID especificados.
<code>notifyElementInvalidate(java.lang.String type, java.lang.String id)</code>	Vacío	Notifica al sistema que han cambiado las relaciones de un objeto. Puede utilizar el método <code>notifyElementInvalidate()</code> para notificar a Orchestrator todos los cambios en las relaciones entre objetos, no solo los cambios de relaciones que invalidan un objeto. Por ejemplo, cuando se añade un objeto secundario a uno principal, se produce un cambio en la relación entre los dos objetos.
<code>notifyElementUpdated(java.lang.String type, java.lang.String id)</code>	Vacío	Notifica al sistema que se han modificado los atributos de un objeto.
<code>notifyMessage(ch.dunes.vso.sdk.api.ErrorLevel severity, java.lang.String type, java.lang.String id, java.lang.String message)</code>	Vacío	Publica un mensaje de error en relación con el módulo actual.

## Interfaz IPluginPublisher

La interfaz IPluginPublisher publica un evento de monitor en un bus de notificación de eventos para la supervisión de los elementos Evento de espera de un flujo de trabajo de larga ejecución.

Cuando un activador de flujo de trabajo inicia un evento en la tecnología conectada, un monitor de complemento que supervisa el activador y está registrado con una instancia de `IPluginPublisher` notifica a cualquier flujo de trabajo en espera que el evento ha ocurrido.

La interfaz `IPluginPublisher` define el método siguiente:

Tipo	Valor	Descripción
<code>pushWatcherEvent(java.lang.String id, java.util.Properties properties)</code>	Vacío	Publica un evento de monitor en un bus de notificación de eventos.

## Interfaz `WebConfigurationAdaptor`

La interfaz `WebConfigurationAdaptor` implementa `IConfigurationAdaptor`, y define métodos para localizar e instalar una aplicación web en la pestaña de configuración de un complemento.

**Nota** La interfaz `WebConfigurationAdaptor` se ha dejado de usar desde Orchestrator 4.1. Para añadir una aplicación web a la configuración, implemente `IConfigurationAdaptor` y utilice el atributo `configuration-war` en el archivo `vso.xml` para identificar la aplicación web.

La interfaz `WebConfigurationAdaptor` define los siguientes métodos:

Método	Devuelve	Descripción
<code>getWebAppContext()</code>	String	Localiza el archivo WAR de la aplicación web para la pestaña de configuración. Proporciona el nombre y la ruta al archivo WAR desde el directorio <code>/webapps</code> del archivo DAR como cadena.
<code>setWebConfiguration(boolean webConfiguration)</code>	Booleano	Determina si el contenido de la pestaña de configuración se define a través de una aplicación web.

## Clase `PluginTrigger`

La clase `PluginTrigger` crea un módulo activador que obtiene información sobre los objetos y los eventos para supervisar en la tecnología conectada, en nombre de un elemento Evento de espera en un flujo de trabajo.

La clase `PluginTrigger` define métodos para obtener o configurar el tipo y el nombre del objeto que supervisar, la naturaleza del evento y un periodo de tiempo de espera.

Debe crear implementaciones de la clase `PluginTrigger` exclusivamente para los elementos Evento de espera de flujos de trabajo. Debe definir activadores de políticas para políticas de Orchestrator en clases que definen eventos e implementar el método `IPluginEventPublisher.pushTrigger()`.

```
public class PluginTrigger
extends java.lang.Object
implements java.io.Serializable
```

La clase `PluginTrigger` define los métodos siguientes:

Método	Devuelve	Descripción
getModuleName()	java.lang.String	Obtiene el nombre del módulo activador.
getProperties()	java.util.Properties	Obtiene una lista de las propiedades del activador.
getSdkId()	java.lang.String	Obtiene el ID del objeto para supervisar en la tecnología conectada.
getSdkType()	java.lang.String	Obtiene el tipo de objeto para supervisar en la tecnología conectada.
getTimeout()	Completo	Obtiene el periodo de tiempo de espera del activador.
setModuleName(java.lang.String moduleName)	Vacío	Establece el nombre del módulo activador.
setProperties(java.util.Properties properties)	Vacío	Establece una lista de las propiedades del activador.
setSdkId(java.lang.String sdkId)	Vacío	Establece el ID del objeto para supervisar en la tecnología conectada.
setSdkType(java.lang.String sdkType)	Vacío	Establece el tipo de objeto para supervisar en la tecnología conectada.
setTimeout(long timeout)	Vacío	Establece un periodo de tiempo de espera en segundos. Un valor negativo desactiva el tiempo de espera.

## Constructores

- PluginTrigger()
- PluginTrigger(java.lang.String moduleName, long timeout, java.lang.String sdkType, java.lang.String sdkId)

## Clase PluginWatcher

La clase PluginWatcher supervisa un módulo activador para un evento definido en la tecnología conectada en nombre de un elemento Evento de espera de un flujo de trabajo de larga ejecución.

La clase PluginWatcher define un constructor que usted puede usar para crear instancias de monitor del complemento. La clase PluginWatcher define métodos para obtener o configurar el nombre del activador de flujo de trabajo que supervisar y un periodo de tiempo de espera.

```
public class PluginWatcher
extends java.lang.Object
implements java.io.Serializable
```

La clase PluginWatcher define los métodos siguientes:

Método	Devuelve	Descripción
getId()	java.lang.String	Obtiene el ID del activador.
getModuleName()	java.lang.String	Obtiene el nombre del módulo activador.

Método	Devuelve	Descripción
getTimeoutDate()	Completo	Obtiene la fecha de tiempo de espera del activador.
getTrigger()	Vacío	Obtiene un activador.
setId(java.lang.String id)	Vacío	Establece el ID del activador.
setTimeoutDate()	Vacío	Establece la fecha de tiempo de espera del activador.

## Constructor

PluginWatcher(PluginTrigger trigger)

## Clase QueryResult

La clase QueryResult contiene los resultados de una consulta find realizada en los objetos a los que accede Orchestrator a través del complemento.

```
public class QueryResult
extends java.lang.Object
implements java.io.Serializable
```

El valor totalCount puede ser mayor que el número de elementos que devuelve QueryResult, si el número total de resultados encontrados es mayor que el número de resultados que devuelve la consulta. El número de resultados que devuelve la consulta se define en la sintaxis de consulta en el archivo vso.xml.

La clase QueryResult define los métodos siguientes:

Método	Devuelve	Descripción
addElement(java.lang.Object element)	Vacío	Añade un elemento a QueryResult.
addElements(java.util.List elements)	Vacío	Añade una lista de elementos a QueryResult.
getElements()	java.util.List	Obtiene elementos de la aplicación conectada.
getTotalCount()	Completo	Obtiene un recuento de todos los elementos disponibles en la tecnología conectada.
isPartialResult()	Booleano	Determina si el resultado obtenido está completo.
removeElement(java.lang.Object element)	Vacío	Elimina un elemento de la tecnología conectada.
setElements(java.util.List elements)	Vacío	Establece los elementos en la tecnología conectada.
setTotalCount(long totalCount)	Vacío	Establece el número total de elementos disponibles en la tecnología conectada.

## Constructores

- `QueryResult()`
- `QueryResult(java.util.List ret)`
- `QueryResult(java.util.List elements, long totalCount)`

## Clase SDKFinderProperty

La clase `SDKFinderProperty` define métodos para obtener y establecer propiedades en los objetos encontrados por los objetos de buscador de Orchestrator en la tecnología conectada. El método `IDynamicFinder.getProperties` devuelve objetos `SDKFinderProperty`.

```
public class SDKFinderProperty
extends java.lang.Object
```

La clase `SDKFinderProperty` define los siguientes métodos:

Método	Devuelve	Descripción
<code>getAttributeName()</code>	<code>java.lang.String</code>	Obtiene un nombre de atributo de objeto
<code>getBeanProperty()</code>	<code>java.lang.String</code>	Obtiene propiedades de un bean de Java
<code>getDescription()</code>	<code>java.lang.String</code>	Obtiene una descripción de objeto
<code>getDisplayName()</code>	<code>java.lang.String</code>	Obtiene un nombre para mostrar de objeto
<code>getPossibleResultType()</code>	<code>java.lang.String</code>	Obtiene los tipos posibles de resultado que devuelve el buscador
<code>getPropertyAccessor()</code>	<code>java.lang.String</code>	Obtiene un descriptor de propiedades de objeto
<code>getPropertyAccessorTree()</code>	<code>java.lang.Object</code>	Obtiene un árbol de descriptor de propiedades de objeto
<code>isHidden()</code>	Booleano	Muestra u oculta el objeto
<code>isShowInColumn()</code>	Booleano	Muestra u oculta el objeto en la columna de base de datos
<code>isShowInDescription()</code>	Booleano	Muestra u oculta la descripción del objeto
<code>setAttributeName(java.lang.String attributeName)</code>	Vacío	Establece un nombre de atributo de objeto
<code>setBeanProperty(java.lang.String beanProperty)</code>	Vacío	Establece propiedades en un bean de Java
<code>setDescription(java.lang.String description)</code>	Vacío	Establece una descripción de objeto
<code>setDisplayName(java.lang.String displayName)</code>	Vacío	Establece un nombre para mostrar de objeto
<code>setHidden(boolean hidden)</code>	Vacío	Muestra u oculta el objeto
<code>setPossibleResultType(java.lang.String possibleResultType)</code>	Vacío	Establece los tipos posibles de resultado que devuelve el buscador

Método	Devuelve	Descripción
setPropertyAccessor(java.lang.String propertyAccessor)	Vacío	Establece un descriptor de propiedades de objeto
setPropertyAccessorTree(java.lang.Object propertyAccessorTree)	Vacío	Establece un árbol de descriptor de propiedades de objeto
setShowInColumn(boolean showInTable)	Vacío	Muestra u oculta el objeto en la columna de base de datos
setShowInDescription(boolean showInDescription)	Vacío	Muestra u oculta la descripción del objeto

## Constructor

SDKFinderProperty(java.lang.String attributeName, java.lang.String displayName, java.lang.String beanProperty, java.lang.String propertyAccessor)

## Clase PluginExecutionException

La clase PluginExecutionException devuelve un mensaje de error si el complemento detecta una excepción cuando ejecuta una operación.

```
public class PluginExecutionException
extends java.lang.Exception
implements java.io.Serializable
```

La clase PluginExecutionException hereda los métodos siguientes de class java.lang.Throwable:

fillInStackTrace, getCause, getLocalizedMessage, getMessage, getStackTrace, initCause, printStackTrace, printStackTrace, printStackTrace, setStackTrace, toString, fillInStackTrace, getCause, getLocalizedMessage, getMessage, getStackTrace, initCause, printStackTrace

## Constructor

PluginExecutionException(java.lang.String message)

## Clase PluginOperationException

La clase PluginOperationException controla los errores detectados durante el funcionamiento de un complemento.

```
public class PluginOperationException
extends java.lang.RuntimeException
implements java.io.Serializable
```

La clase PluginOperationException hereda los métodos siguientes de class java.lang.Throwable:

fillInStackTrace, getCause, getLocalizedMessage, getMessage, getStackTrace, initCause, printStackTrace, printStackTrace, printStackTrace, setStackTrace, toString

## Constructor

PluginOperationException(java.lang.String message)

## Enumeración HasChildrenResult

La enumeración `HasChildrenResult` declara si un determinado elemento principal tiene elementos secundarios. El método `IPluginFactory.hasChildrenInRelation` devuelve objetos `HasChildrenResult`.

```
public enum HasChildrenResult
extends java.lang.Enum<HasChildrenResult>
implements java.io.Serializable
```

La enumeración `HasChildrenResult` define las constantes siguientes:

- `public static final HasChildrenResult Yes`
- `public static final HasChildrenResult No`
- `public static final HasChildrenResult Unknown`

La enumeración `HasChildrenResult` define los métodos siguientes:

Método	Devuelve	Descripción
<code>getValue()</code>	<code>int</code>	Devuelve uno de los valores siguientes: <div> <div><b>1</b></div>El elemento principal tiene elementos secundarios           </div> <div> <div><b>-1</b></div>El elemento principal no tiene elementos secundarios           </div> <div> <div><b>0</b></div>Parámetro desconocido o no válido           </div>
<code>valueOf(java.lang.String name)</code>	<code>static HasChildrenResult</code>	Devuelve una constante de enumeración de este tipo con el nombre especificado. La cadena debe coincidir exactamente con el identificador utilizado para declarar una constante de enumeración de este tipo. No utilice caracteres de espacio en blanco en el nombre de la enumeración.
<code>values()</code>	<code>static HasChildrenResult[]</code>	Devuelve una matriz que incluye las constantes de este tipo de enumeración, en el orden en que se declaran. Este método se puede iterar en constantes, como se indica a continuación: <div> <pre>for (HasChildrenResult c : HasChildrenResult.values()) System.out.println(c);</pre> </div>

La enumeración `HasChildrenResult` hereda los métodos siguientes de class `java.lang.Enum`:

`clone`, `compareTo`, `equals`, `finalize`, `getDeclaringClass`, `hashCode`, `name`, `ordinal`, `toString`, `valueOf`

## Tipo de anotación ScriptingAttribute

El tipo de anotación `ScriptingAttribute` anota un atributo de un objeto en la tecnología conectada para utilizar como propiedad en la creación de scripts.

```
@Retention(value=RUNTIME)
@Target(value={METHOD, FIELD})
public @interface ScriptingAttribute
```

El tipo de anotación `ScriptingAttribute` tiene el valor siguiente:

```
public abstract java.lang.String value
```

## Tipo de anotación ScriptingFunction

El tipo de anotación `ScriptingFunction` anota un método para utilizar como propiedad en la creación de scripts.

```
@Retention(value=RUNTIME)
@Target(value={METHOD, CONSTRUCTOR})
public @interface ScriptingFunction
```

El tipo de anotación `ScriptingFunction` tiene el valor siguiente:

```
public abstract java.lang.String value
```

## Tipo de anotación ScriptingParameter

El tipo de anotación `ScriptingParameter` anota un parámetro para utilizar como propiedad en la creación de scripts.

```
@Retention(value=RUNTIME)
@Target(value=PARAMETER)
public @interface ScriptingParameter
```

El tipo de anotación `ScriptingParameter` tiene el valor siguiente:

```
public abstract java.lang.String value
```

## Elementos del archivo de definición del complemento vso.xml

El archivo `vso.xml` contiene un conjunto de flujos de trabajo estándar. Algunos de los elementos son obligatorios y otros son opcionales. Cada elemento tiene atributos que definen valores para los objetos y las operaciones que el usuario asigna a objetos y operaciones de Orchestrator.

Además, los elementos pueden tener varios elementos secundarios o ninguno. Un elemento secundario define el elemento principal con más detalle. El mismo elemento secundario puede aparecer en diferentes elementos principales. Por ejemplo, el elemento `description` carece de elementos secundarios, pero aparece como elemento secundario en muchos elementos principales: `module`, `example`, `trigger`, `gauge`, `finder`, `constructor`, `method`, `object` y `enumeration`.



Cada definición de elemento que sigue enumera sus atributos, elementos superiores y secundarios.

## Elemento module

Un módulo describe un conjunto de objetos de complemento que se ponen a disposición para Orchestrator.

El módulo contiene información sobre el modo en el que los datos de la tecnología conectada se asignan a las clases de Java, las versiones, el modo en el que se implementa el módulo y el modo en el que aparece el complemento en el inventario de Orchestrator.

El elemento `<module>` es opcional. El elemento `<module>` cuenta con los atributos siguientes:

Atributos	Valor	Descripción
name	String	Define el tipo de todos los elementos <code>&lt;finder&gt;</code> en el complemento. Atributo obligatorio.
version	Número	Número de versión del complemento; se utiliza cuando se vuelven a cargar los paquetes en una nueva versión del complemento. Atributo obligatorio.
build-number	Número	Número de compilación del complemento; se utiliza cuando se vuelven a cargar los paquetes en una nueva versión del complemento. Atributo obligatorio.
image	Archivo de imagen	Icono que se mostrará en el inventario de Orchestrator. Atributo obligatorio.
display-name	String	Nombre que aparece en el inventario de Orchestrator. Atributo opcional.
interface-mapping-allowed	true o false	VMware desaconseja la asignación de interfaces. Atributo opcional.

**Tabla 1-7. Jerarquía de elementos**

Elemento principal	Elementos secundarios
Ninguno	<ul style="list-style-type: none"> <li>■ <code>&lt;description&gt;</code></li> <li>■ <code>&lt;installation&gt;</code></li> <li>■ <code>&lt;configuration&gt;</code></li> <li>■ <code>&lt;finder-datasources&gt;</code></li> <li>■ <code>&lt;inventory&gt;</code></li> <li>■ <code>&lt;finders&gt;</code></li> <li>■ <code>&lt;scripting-objects&gt;</code></li> <li>■ <code>&lt;enumerations&gt;</code></li> </ul>

## Elemento description

El elemento `<description>` proporciona descripciones de los elementos del complemento que aparece en la documentación del Explorador de API.

Debe añadir el texto que aparece en la documentación del Explorador de API entre las etiquetas `<description>` y `</description>`.

El elemento `<description>` es opcional. El elemento `<description>` no tiene atributos.

**Tabla 1-8. Jerarquía de elementos**

Elementos principales	Elementos secundarios
<ul style="list-style-type: none"> <li>■ <code>&lt;module&gt;</code></li> <li>■ <code>&lt;example&gt;</code></li> <li>■ <code>&lt;trigger&gt;</code></li> <li>■ <code>&lt;gauge&gt;</code></li> <li>■ <code>&lt;finder&gt;</code></li> <li>■ <code>&lt;constructor&gt;</code></li> <li>■ <code>&lt;method&gt;</code></li> <li>■ <code>&lt;object&gt;</code></li> <li>■ <code>&lt;enumeration&gt;</code></li> </ul>	Ninguno

## Elemento deprecated

El elemento `<deprecated>` marca objetos y métodos que han quedado obsoletos en la documentación del Explorador de API.

Debe añadir el texto que aparece en la documentación del Explorador de API entre las etiquetas `<deprecated>` y `</deprecated>`.

El elemento `<deprecated>` es opcional. El elemento `<deprecated>` no tiene atributos.

**Tabla 1-9. Jerarquía de elementos**

Elementos principales	Elementos secundarios
<ul style="list-style-type: none"> <li>■ <code>&lt;method&gt;</code></li> <li>■ <code>&lt;object&gt;</code></li> </ul>	Ninguno

## Elemento url

El elemento `<url>` proporciona una URL que apunta a documentación externa sobre un objeto o una enumeración.

La URL se proporciona entre las etiquetas `<url>` y `</url>`.

El elemento `<url>` es opcional. El elemento `<url>` no tiene atributos.

**Tabla 1-10. Jerarquía de elementos**

Elementos principales	Elementos secundarios
<ul style="list-style-type: none"> <li>■ <code>&lt;enumeration&gt;</code></li> <li>■ <code>&lt;object&gt;</code></li> </ul>	Ninguno

## Elemento installation

El elemento `<installation>` permite instalar un paquete o ejecutar un script cuando se inicia el servidor.

El elemento `<installation>` es opcional. El elemento `<installation>` tiene los atributos siguientes:

Atributos	Valor	Descripción
mode	always, never o version	La configuración del valor mode produce el comportamiento siguiente cuando se inicia el servidor de Orchestrator: <ul style="list-style-type: none"> <li>■ Se ejecuta la acción always</li> <li>■ Se ejecuta la acción never</li> <li>■ La acción se ejecuta cuando el servidor detecta una versión más reciente del complemento</li> </ul> Atributo obligatorio.

**Tabla 1-11. Jerarquía de elementos**

Elemento principal	Elemento secundario
<code>&lt;module&gt;</code>	<code>&lt;action&gt;</code>

## Elemento action

El elemento `<action>` especifica la acción que se ejecuta al iniciar el servidor de Orchestrator.

Los atributos del elemento `<action>` proporcionan la ruta del paquete de Orchestrator que define el comportamiento del complemento cuando se inicia.

El elemento `<action>` es opcional. Un complemento puede tener un número ilimitado de elementos `<action>`. El elemento `<action>` tiene los siguientes atributos.

Atributos	Valor	Descripción
resource	String	La ruta del script o del paquete de Java desde la raíz del archivo dar. Atributo obligatorio.
type	install-package o execute-script	Instala el paquete de Orchestrator indicado en el servidor de Orchestrator o bien ejecuta el script especificado. Atributo obligatorio.

**Tabla 1-12. Jerarquía de elementos**

Elemento principal	Elementos secundarios
<code>&lt;installation&gt;</code>	Ninguno

## Elemento finder-datasources

El elemento `<finder-datasources>` es el contenedor de los elementos `<finder-datasource>`.

El elemento `<finder-datasources>` es opcional. El elemento `<finder-datasources>` no tiene atributos.

Tabla 1-13. Jerarquía de elementos

Elemento principal	Elementos secundarios
<module>	<finder-datasource>

## Elemento finder-datasource

El elemento <finder-datasource> apunta al archivo de clase de Java de la implementación IPluginAdaptor que creó para el complemento.

Puede definir cómo Orchestrator accede a los objetos de la tecnología conectada en el elemento <finder-datasource>. El elemento <finder-datasource> identifica la clase de Java del adaptador de complementos que ha creado. La clase del adaptador de complementos crea instancias de la fábrica de complementos que ha creado. La fábrica de complementos define métodos que buscan objetos en la tecnología conectada. Puede definir tiempos de espera en el elemento <finder-datasource> para las llamadas de método de buscador que realiza la fábrica. En la interfaz IPluginFactory se aplican distintos tiempos de espera para los distintos métodos de buscador.

El elemento <finder-datasource> es opcional. Un complemento puede tener un número ilimitado de elementos <finder-datasources>. El elemento <finder-datasource> tiene los siguientes atributos.

Atributos	Valor	Descripción
name	String	Identifica la fuente de los datos del elemento <finder>, atributos datasource. Equivalente a un id de XML. Atributo obligatorio.
adaptor-class	Clase de Java	Apunta a la implementación IPluginAdaptor que usted define para crear el adaptador de complementos, por ejemplo <code>com.vmware.plugins.sample.Adaptor</code> . Atributo obligatorio.
concurrent-call	true (predeterminado) o false	Permite a varios usuarios acceder al adaptador al mismo tiempo. Debe configurar concurrent-call como false si el complemento no admite llamadas simultáneas. Atributo opcional.
invoker-mode	direct (predeterminado) o timeout	Establece un tiempo de espera en la función buscador. Si se configura como direct, el tiempo de espera de las llamadas a las funciones de buscador nunca se agota. Si se configura como timeout, el servidor de Orchestrator aplica el periodo de tiempo de espera que corresponde al método de buscador. Atributo opcional.
anonymous-login-mode	never (predeterminado) o always	Pasa (o no) el nombre y la contraseña del usuario al complemento. Atributo opcional.

Atributos	Valor	Descripción
timeout-fetch-relation	Número; de manera predeterminada, 30 segundos	Se aplica a las llamadas de findRelation(). Atributo opcional.
timeout-find-all	Número; de manera predeterminada, 60 segundos	Se aplica a las llamadas de findAll(). Atributo opcional.
timeout-find	Número; de manera predeterminada, 60 segundos	Se aplica a las llamadas de find(). Atributo opcional.
timeout-has-children-in-relation	Número; de manera predeterminada, 2 segundos	Se aplica a las llamadas de findChildrenInRelation(). Atributo opcional.
timeout-execute-plugin-command	Número; de manera predeterminada, 30 segundos	Se aplica a las llamadas de executePluginCommand(). Atributo opcional.

Tabla 1-14. Jerarquía de elementos

Elemento principal	Elementos secundarios
<finder-datasources>	Ninguno

## Elemento inventory

El elemento <inventory> define la raíz de la lista jerárquica del complemento que aparece en los cuadros de diálogo de selección de objetos y en la vista **Inventario** del cliente de Orchestrator.

El elemento <inventory> no representa un objeto en la aplicación conectada, sino el propio complemento como objeto de la API de creación de scripts de Orchestrator.

El elemento <inventory> es opcional. El elemento <inventory> tiene el atributo siguiente.

Atributos	Valor	Descripción
type	Un tipo de objeto de Orchestrator	El tipo de elemento <finder> que representa la raíz de la jerarquía de objetos. Atributo obligatorio.

Tabla 1-15. Jerarquía de elementos

Elemento principal	Elementos secundarios
<module>	Ninguno

## Elemento finders

El elemento <finders> es el contenedor de todos los elementos <finder>.

El elemento <finders> es opcional. El elemento <finders> no tiene atributos.

Tabla 1-16. Jerarquía de elementos

Elemento principal	Elemento secundario
<module>	<finder>

## Elemento finder

El elemento `<finder>` representa en el cliente de Orchestrator un tipo de objeto encontrado a través del complemento.

El elemento `<finder>` identifica la clase de Java que define el objeto que representa el buscador de objetos. El elemento `<finder>` define el modo en el que el objeto aparece en la interfaz del cliente de Orchestrator. También identifica el objeto de creación de scripts que define la API de creación de scripts de Orchestrator para representar este objeto.

Los buscadores actúan como interfaz entre formatos de objeto utilizados por diferentes tipos de tecnologías conectadas.

El elemento `<finder>` es opcional. Un complemento puede tener un número ilimitado de elementos `<finder>`. El elemento `<finder>` define los atributos siguientes:

Atributos	Valor	Descripción
<code>type</code>	Un tipo de objeto de Orchestrator	Tipo de objeto representado por el buscador. Atributo obligatorio.
<code>datasource</code>	Atributo <code>&lt;finder-datasource name&gt;</code>	Identifica la clase de Java que define el objeto utilizando el origen de datos <code>refid</code> . Atributo obligatorio.
<code>dynamic-finder</code>	Método de Java	Define el método de buscador personalizado que se implementa en una instancia de <code>IDynamicFinder</code> , para devolver el ID y las propiedades de un buscador mediante programación, en lugar de definirlo en el archivo <code>vso.xml</code> . Atributo opcional.
<code>hidden</code>	<code>true</code> o <code>false</code> (predeterminado)	Si es <code>true</code> , oculta el buscador en el cliente de Orchestrator. Atributo opcional.
<code>image</code>	Ruta a un archivo gráfico	Un icono de 16x16 para representar el buscador en listas jerárquicas en el cliente de Orchestrator. Atributo opcional.
<code>java-class</code>	Nombre de una clase de Java	La clase de Java que define el objeto que el buscador busca y asigna a un objeto de creación de scripts. Atributo opcional.
<code>script-object</code>	Atributo <code>&lt;scripting-object type&gt;</code>	El tipo <code>&lt;scripting-object&gt;</code> , si lo hay, al que se asigna este buscador. Atributo opcional.

Tabla 1-17. Jerarquía de elementos

Elemento principal	Elementos secundarios
<finders>	<ul style="list-style-type: none"> <li>■ &lt;id&gt;</li> <li>■ &lt;description&gt;</li> <li>■ &lt;properties&gt;</li> <li>■ &lt;default-sorting&gt;</li> <li>■ &lt;inventory-children&gt;</li> <li>■ &lt;relations&gt;</li> <li>■ &lt;inventory-tabs&gt;</li> <li>■ &lt;events&gt;</li> </ul>

## Elemento properties

El elemento <properties> es el contenedor de los elementos <finder><property>.

El elemento <properties> es opcional. El elemento <properties> no tiene atributos.

Tabla 1-18. Jerarquía de elementos

Elemento principal	Elemento secundario
<finder>	<property>

## Elemento property

El elemento <property> asigna las propiedades del objeto encontrado a las llamadas de método o las propiedades de Java.

Puede llamar a los métodos de la clase SDKFinderProperty cuando implementa la fábrica de complementos para obtener propiedades para el procesamiento de la implementación de la fábrica de complementos.

En las vistas del cliente de Orchestrator, puede mostrar u ocultar las propiedades de los objetos. Asimismo, puede utilizar enumeraciones para definir propiedades de objetos.

El elemento <property> es opcional. Un complemento puede tener un número ilimitado de elementos <property>. El elemento <property> tiene los atributos siguientes.

Atributos	Valor	Descripción
name	Nombre de buscador	El nombre que utiliza FinderResult para almacenar el elemento. Atributo obligatorio.
display-name	Nombre de buscador	El nombre de propiedad que se muestra. Atributo opcional.

Atributos	Valor	Descripción
bean-property	Nombre de propiedad	El atributo bean-property permite identificar una propiedad que se obtiene mediante las operaciones get y set. Si identifica una propiedad llamada MyProperty, el complemento define las operaciones getMyProperty y setMyProperty.  Puede establecer bean-property o property-accessor, pero no las dos a la vez. Atributo opcional.
property-accessor	El método que obtiene un valor de propiedad de un objeto	El atributo property-accessor permite definir una expresión OGNL para validar las propiedades de un objeto.  Puede establecer bean-property o property-accessor, pero no las dos a la vez. Atributo opcional.
show-in-column	true (predeterminado) o false	Si es true, esta propiedad aparece en la tabla de resultados del cliente de Orchestrator. Atributo opcional.
show-in-description	true (predeterminado) o false	Si es true, esta propiedad aparece en la descripción del objeto. Atributo opcional.
hidden	true o false (predeterminado)	Si es true, esta propiedad se oculta en todos los casos. Atributo opcional.
linked-enumeration	Nombre de enumeración	Vincula una propiedad de buscador con una enumeración. Atributo opcional.

Tabla 1-19. Jerarquía de elementos

Elemento principal	Elementos secundarios
<properties>	Elementos secundarios

## Elemento relations

El elemento <relations> es el contenedor de los elementos <finder><relation>.

El elemento <relations> es opcional. El elemento <relations> no tiene atributos.

Tabla 1-20. Jerarquía de elementos

Elemento principal	Elemento secundario
<finder>	<relation>

## Elemento relation

El elemento <relation> define el modo en el que los objetos se relacionan con otros objetos.

En el elemento <relation> se define el nombre de la relación.



El elemento `<relation>` es opcional. Un complemento puede tener un número ilimitado de elementos `<relation>`. El elemento `<relation>` tiene los atributos siguientes.

Atributos	Valor	Descripción
name	Nombre de relación	Nombre de esta relación. Atributo obligatorio.
type	Tipo de objeto de Orchestrator	Tipo de objeto que se relaciona con otro objeto mediante esta relación. Atributo obligatorio.
cardinality	to-one o to-many	Define la relación entre los objetos como de uno a uno o de uno a varios. Atributo opcional.

**Tabla 1-21. Jerarquía de elementos**

Elemento principal	Elementos secundarios
<code>&lt;relations&gt;</code>	Ninguno

## Elemento id

El elemento `<id>` define un método para obtener el ID exclusivo del objeto identificado por el buscador.

El elemento `<id>` es opcional. El elemento `<id>` tiene los atributos siguientes.

Atributos	Valor	Descripción
accessor	Nombre de método	El atributo <code>accessor</code> permite definir una expresión OGNL para validar las propiedades de un objeto. Atributo obligatorio.

**Tabla 1-22. Jerarquía de elementos**

Elemento principal	Elementos secundarios
<code>&lt;finder&gt;</code>	Ninguno

## Elemento inventory-children

El elemento `<inventory-children>` define la jerarquía de las listas que muestran los objetos en los cuadros de selección de objetos y en la vista **Inventario** del cliente de Orchestrator.

El elemento `<inventory-children>` es opcional. El elemento `<inventory-children>` no tiene atributos.

**Tabla 1-23. Jerarquía de elementos**

Elemento principal	Elemento secundario
<code>&lt;finder&gt;</code>	<code>&lt;relation-link&gt;</code>

## Elemento relation-link

El elemento `<relation-link>` define las jerarquías entre los objetos principales y secundarios en la pestaña **Inventario**.

El elemento `<relation-link>` es opcional. Un complemento puede tener un número ilimitado de elementos `<relation-link>`. El elemento `<relation-link>` tiene el atributo siguiente.

Tipo	Valor	Descripción
name	Nombre de relación	refid para un nombre de relación. Atributo obligatorio.

**Tabla 1-24. Jerarquía de elementos**

Elemento principal	Elementos secundarios
<code>&lt;inventory-children&gt;</code>	Ninguno

## Elemento events

El elemento `<events>` es el contenedor de los elementos `<trigger>` y `<gauge>`.

El elemento `<events>` puede contener un número ilimitado de activadores o de medidores.

El elemento `<events>` es opcional. El elemento `<events>` no tiene atributos.

**Tabla 1-25. Jerarquía de elementos**

Elemento principal	Elementos secundarios
<code>&lt;finder&gt;</code>	<ul style="list-style-type: none"> <li>■ <code>&lt;trigger&gt;</code></li> <li>■ <code>&lt;gauge&gt;</code></li> </ul>

## Elemento trigger

El elemento `<trigger>` declara los activadores que se pueden utilizar para este buscador. Debe implementar los métodos `registerEventPublisher()` y `unregisterEventPublisher()` de `IPluginAdaptor` para establecer activadores.

El elemento `<trigger>` es opcional. El elemento `<trigger>` tiene el atributo siguiente.

Tipo	Valor	Descripción
name	Nombre del activador	Nombre para este activador. Atributo obligatorio.

**Tabla 1-26. Jerarquía de elementos**

Elemento principal	Elementos secundarios
<code>&lt;events&gt;</code>	<ul style="list-style-type: none"> <li>■ <code>&lt;description&gt;</code></li> <li>■ <code>&lt;trigger-properties&gt;</code></li> </ul>

## Elemento trigger-properties

El elemento `<trigger-properties>` es el contenedor de los elementos `<trigger-property>`.

El elemento `<trigger-properties>` es opcional. El elemento `<trigger-properties>` no tiene atributos.

Tabla 1-27. Jerarquía de elementos

Elemento principal	Elemento secundario
<trigger>	<trigger-property>

## Elemento trigger-property

El elemento <trigger-property> define las propiedades que identifican un objeto activador.

El elemento <trigger-property> es opcional. Un complemento puede tener un número ilimitado de elementos <trigger-property>. El elemento <trigger-property> tiene los atributos siguientes.

Tipo	Valor	Descripción
name	Nombre del activador	Nombre del activador. Atributo opcional.
display-name	Nombre del activador	Nombre que se muestra en el cliente de Orchestrator. Atributo opcional.
type	Tipo de activador	Tipo de objeto que define el activador. Atributo obligatorio.

Tabla 1-28. Jerarquía de elementos

Elemento principal	Elementos secundarios
<trigger-properties>	Ninguno

## Elemento gauge

El elemento <gauge> define los medidores que puede utilizar para este buscador. Debe implementar los métodos `registerEventPublisher()` y `unregisterEventPublisher()` de `IPluginAdaptor` para establecer medidores.

El elemento <gauge> es opcional. Un complemento puede tener un número ilimitado de elementos <gauge>. El elemento <gauge> tiene los atributos siguientes.

Tipo	Valor	Descripción
name	Nombre de medidor	Nombre del medidor. Atributo obligatorio.
min-value	Número	Límite mínimo. Atributo opcional.
max-value	Número	Límite máximo. Atributo opcional.
unit	Tipo de objeto	Tipo de objeto que define el medidor. Atributo obligatorio.
format	Cadena	Formato del valor supervisado. Atributo opcional.

Tabla 1-29. Jerarquía de elementos

Elemento principal	Elemento secundario
<events>	<description>

## Elemento scripting-objects

El elemento `<scripting-objects>` es el contenedor de los elementos `<object>`.

El elemento `<scripting-objects>` es opcional. El elemento `<scripting-objects>` no tiene atributos.

**Tabla 1-30. Jerarquía de elementos**

Elemento principal	Elemento secundario
<code>&lt;module&gt;</code>	<code>&lt;object&gt;</code>

## Elemento object

El elemento `<object>` asigna los constructores, los atributos y los métodos de la tecnología conectada a tipos de objetos de JavaScript que expone la API de creación de scripts de Orchestrator.

Consulte [Designar objetos de complemento](#) para obtener información sobre las convenciones de nomenclatura de los objetos.

El elemento `<object>` es opcional. Un complemento puede tener un número ilimitado de elementos `<object>`. El elemento `<object>` tiene los atributos siguientes.

Tipo	Valor	Descripción
<code>script-name</code>	Nombre de JavaScript	Nombre de creación de scripts de la clase. Debe ser único globalmente. Atributo obligatorio.
<code>java-class</code>	Clase de Java	Clase de Java incluida por esta clase de JavaScript. Atributo obligatorio.
<code>create</code>	<code>true</code> (predeterminado) o <code>false</code>	Si es <code>true</code> , puede crear una nueva instancia de esta clase. Atributo opcional.
<code>strict</code>	<code>true</code> o <code>false</code> (predeterminado)	Si es <code>true</code> , solo puede llamar a métodos que anote o declare en el archivo <code>vso.xml</code> . Atributo opcional.
<code>is-deprecated</code>	<code>true</code> o <code>false</code> (predeterminado)	Si es <code>true</code> , el objeto asigna una clase de Java obsoleta. Atributo opcional.
<code>since-version</code>	Cadena	Versión a partir de la cual la clase de Java queda obsoleta. Atributo opcional.

**Tabla 1-31. Jerarquía de elementos**

Elemento principal	Elementos secundarios
<code>&lt;scripting-objects&gt;</code>	<ul style="list-style-type: none"> <li>■ <code>&lt;description&gt;</code></li> <li>■ <code>&lt;deprecated&gt;</code></li> <li>■ <code>&lt;url&gt;</code></li> <li>■ <code>&lt;constructors&gt;</code></li> <li>■ <code>&lt;attributes&gt;</code></li> <li>■ <code>&lt;methods&gt;</code></li> <li>■ <code>&lt;singleton&gt;</code></li> </ul>

## Elemento constructors

El elemento `<constructors>` es el contenedor de los elementos `<object><constructor>`.

El elemento `<constructors>` es opcional. El elemento `<constructors>` no tiene atributos.

**Tabla 1-32. Jerarquía de elementos**

Elemento principal	Elemento secundario
<code>&lt;object&gt;</code>	<code>&lt;constructor&gt;</code>

## Elemento constructor

El elemento `<constructor>` define un método de constructor. El método de `<constructor>` produce documentación en el Explorador de API.

El elemento `<constructor>` es opcional. Un complemento puede tener un número ilimitado de elementos `<constructor>`. El elemento `<constructor>` no tiene atributos.

**Tabla 1-33. Jerarquía de elementos**

Elemento principal	Elementos secundarios
<code>&lt;constructors&gt;</code>	<ul style="list-style-type: none"> <li>■ <code>&lt;description&gt;</code></li> <li>■ <code>&lt;parameters&gt;</code></li> </ul>

## Elemento parameters del constructor

El elemento `<parameters>` es el contenedor de los elementos `<constructor><parameter>`.

El elemento `<parameters>` es opcional. El elemento `<parameters>` no tiene atributos.

**Tabla 1-34. Jerarquía de elementos**

Elemento principal	Elemento secundario
<code>&lt;constructor&gt;</code>	<code>&lt;parameter&gt;</code>

## Elemento parameter del constructor

El elemento `<parameter>` define los parámetros del constructor.

El elemento `<parameter>` es opcional. Un complemento puede tener un número ilimitado de elementos `<parameter>`. El elemento `<parameter>` tiene los atributos siguientes.

Tipo	Valor	Descripción
name	Cadena	Nombre de parámetro que se utiliza en la documentación de API. Atributo obligatorio.
type	Tipo de parámetro de Orchestrator	Tipo de parámetro que se utiliza en la documentación de API. Atributo obligatorio.

Tipo	Valor	Descripción
is-optional	true o false	Si está establecido como true, el valor puede ser nulo. Atributo opcional.
since-version	Cadena	Versión de método. Atributo opcional.

Tabla 1-35. Jerarquía de elementos

Elemento principal	Elementos secundarios
<parameters>	Ninguno

## Elemento attributes

El elemento <attributes> es el contenedor de los elementos <object><attribute>.

El elemento <attributes> es opcional. El elemento <attributes> no tiene atributos.

Tabla 1-36. Jerarquía de elementos

Elemento principal	Elemento secundario
<object>	<attribute>

## Elemento attribute

El elemento <attribute> asigna los atributos de una clase de Java de la tecnología conectada a atributos disponibles mediante el motor de JavaScript de Orchestrator.

El elemento <attribute> es opcional. Un complemento puede tener un número ilimitado de elementos <attribute>. El elemento <attribute> tiene los siguientes atributos.

Tipo	Valor	Descripción
java-name	Atributo de Java	Nombre del atributo de Java. Atributo obligatorio.
script-name	Objeto de JavaScript	Nombre del objeto de JavaScript correspondiente. Atributo obligatorio.
return-type	Cadena	Tipo de objeto devuelto por este atributo. Aparece en la documentación del Explorador de API. Atributo opcional.  <b>Nota</b> Si el tipo de valor devuelto de JavaScript es Properties, las implementaciones de Java subyacentes admitidas son java.util.HashMap y java.util.Hashtable.
read-only	true o false	Si es true, no puede modificar este atributo. Atributo opcional.
is-optional	true o false	Si es true, este campo puede ser nulo. Atributo opcional.

Tipo	Valor	Descripción
show-in-api	true o false	Si es false, este atributo no aparece en la documentación de la API. Atributo opcional.
is-deprecated	true o false	Si es true, el objeto asigna un atributo obsoleto. Atributo opcional.
since-version	Número	Versión en la que el atributo quedó obsoleto. Atributo opcional.

Tabla 1-37. Jerarquía de elementos

Elemento principal	Elementos secundarios
<attributes>	Ninguno

## Elemento methods

El elemento <methods> es el contenedor de los elementos <object><method>.

El elemento <methods> es opcional. El elemento <methods> no tiene atributos.

Tabla 1-38. Jerarquía de elementos

Elemento principal	Elemento secundario
<object>	<method>

## Elemento method

El elemento <method> asigna un método de Java de la tecnología conectada a un método de JavaScript que exponga el motor de JavaScript de Orchestrator.

El elemento <method> es opcional. Un complemento puede tener un número ilimitado de elementos <method>. El elemento <method> tiene los atributos siguientes.

Tipo	Valor	Descripción
java-name	Método de Java	Nombre de la firma del método de Java con tipos de argumento entre paréntesis, por ejemplo getVms(DataStore). Atributo obligatorio.
script-name	Método de JavaScript	Nombre del método de JavaScript correspondiente. Atributo obligatorio.
return-type	Tipo de objeto de Java	El tipo que obtiene este método. Atributo opcional.  <b>Nota</b> Si el tipo de valor devuelto de JavaScript es Properties, las implementaciones de Java subyacentes admitidas son java.util.HashMap y java.util.Hashtable.
static	true o false	Si es true, este método es estático. Atributo opcional.

Tipo	Valor	Descripción
show-in-api	true o false	Si es false, este método no aparece en la documentación de la API. Atributo opcional.
is-deprecated	true o false	Si es true, el objeto asigna un método en desuso. Atributo opcional.
since-version	Número	La versión en la que el método queda en desuso. Atributo opcional.

Tabla 1-39. Jerarquía de elementos

Elemento principal	Elementos secundarios
<methods>	<ul style="list-style-type: none"> <li>■ &lt;deprecated&gt;</li> <li>■ &lt;description&gt;</li> <li>■ &lt;example&gt;</li> <li>■ &lt;parameters&gt;</li> </ul>

## Elemento de ejemplo

El elemento <example> le permite añadir ejemplos de código a los métodos de Javascript que aparecen en la documentación del Explorador de API.

El elemento <example> es opcional. El elemento <example> no tiene atributos.

Tabla 1-40. Jerarquía de elementos

Elemento principal	Elementos secundarios
<method>	<ul style="list-style-type: none"> <li>■ &lt;code&gt;</li> <li>■ &lt;description&gt;</li> </ul>

## Elemento code

El elemento <code> proporciona código de ejemplo que aparece en la documentación del Explorador de API.

El ejemplo de código se proporciona entre las etiquetas <code> y </code>. El elemento <code> es opcional. El elemento <code> no tiene atributos.

Tabla 1-41. Jerarquía de elementos

Elemento principal	Elementos secundarios
<example>	Ninguno

## Elemento method parameters

El elemento <parameters> es el contenedor de los elementos <method><parameter>.

El elemento <parameters> es opcional. El elemento <parameters> no tiene atributos.



Tabla 1-42.

Elemento principal	Elemento secundario
<method>	<parameter>

## Elemento method parameter

El elemento <parameter> define los parámetros de entrada del método.

El elemento <parameter> es opcional. Un complemento puede tener un número ilimitado de elementos <parameter>. El elemento <parameter> tiene los atributos siguientes.

Tipo	Valor	Descripción
name	Cadena	Nombre del parámetro. Atributo obligatorio.
type	Tipo de parámetro de Orchestrator	Tipo de parámetro. Atributo obligatorio.
is-optional	true o false	Si está establecido como true, el valor puede ser nulo. Atributo opcional.
since-version	Cadena	Versión de método. Atributo opcional.

Tabla 1-43. Jerarquía de elementos

Elemento principal	Elemento secundario
<parameters>	Ninguno

## Elemento singleton

El elemento <singleton> crea un objeto de creación de scripts de JavaScript como instancia de singleton.

Un objeto singleton se comporta del mismo modo que una clase estática de Java. Los objetos singleton definen objetos genéricos para su uso por parte del complemento, en lugar de definir instancias específicas de objetos a las que accede Orchestrator en la tecnología conectada. Por ejemplo, puede utilizar un objeto singleton para establecer la conexión con la tecnología conectada.

El elemento <singleton> es opcional. El elemento <singleton> tiene los atributos siguientes.

Tipo	Valor	Descripción
script-name	Objeto de JavaScript	Nombre del objeto de JavaScript correspondiente. Atributo obligatorio.
datasource	Objeto de Java	El objeto de origen de Java para este objeto de JavaScript. Atributo obligatorio.

Tabla 1-44. Jerarquía de elementos

Elemento principal	Elemento secundario
<object>	Ninguno

## Elemento enumerations

El elemento `<enumerations>` es el contenedor de los elementos `<enumeration>`.

El elemento `<enumerations>` es opcional. El elemento `<enumerations>` no tiene atributos.

**Tabla 1-45. Jerarquía de elementos**

Elemento principal	Elemento secundario
<code>&lt;module&gt;</code>	<code>&lt;enumeration&gt;</code>

## Elemento enumeration

El elemento `<enumeration>` define valores comunes que se aplican a todos los objetos de un tipo concreto.

Si todos los objetos de un tipo determinado requieren un cierto atributo, y si el rango de valores de dicho atributo es limitado, puede definir diferentes valores como entradas de enumeración. Por ejemplo, si un tipo de objeto requiere un atributo `color`, y si los únicos colores disponibles son el rojo, el azul y el verde, puede definir tres entradas de enumeración para establecer esos tres valores de color. Las entradas se definen como elementos secundarios del elemento de enumeración.

El elemento `<enumeration>` es opcional. Un complemento puede tener un número ilimitado de elementos `<enumeration>`. El elemento `<enumeration>` tiene el atributo siguiente.

Tipo	Valor	Descripción
type	Tipo de objeto de Orchestrator	Tipo de enumeración. Atributo obligatorio.

**Tabla 1-46. Jerarquía de elementos**

Elemento principal	Elementos secundarios
<code>&lt;enumerations&gt;</code>	<ul style="list-style-type: none"> <li>■ <code>&lt;url&gt;</code></li> <li>■ <code>&lt;description&gt;</code></li> <li>■ <code>&lt;entries&gt;</code></li> </ul>

## Elemento entries

El elemento `<entries>` es el contenedor de los elementos `<enumeration><entry>`.

El elemento `<entries>` es opcional. El elemento `<entries>` no tiene atributos.

**Tabla 1-47. Jerarquía de elementos**

Elemento principal	Elemento secundario
<code>&lt;enumeration&gt;</code>	<code>&lt;entry&gt;</code>

## Elemento entry

El elemento `<entry>` proporciona un valor para un atributo de enumeración.

El elemento `<entry>` es opcional. Un complemento puede tener un número ilimitado de elementos `<entry>`. El elemento `<entry>` tiene los atributos siguientes.

Tipo	Valor	Descripción
id	Texto	El identificador que utilizan los objetos para establecer la entrada de enumeración como atributo. Atributo obligatorio.
name	Texto	El nombre de la entrada. Atributo obligatorio.

Tabla 1-48. Jerarquía de elementos

Elemento principal	Elementos secundarios
<entries>	Ninguno

## Recomendaciones para el desarrollo de complementos de Orchestrator

Podrá mejorar determinados aspectos de los complementos de Orchestrator que desarrolle si comprende la estructura y el contenido de los complementos, así como la forma de evitar determinados problemas.

### ■ Enfoques para crear complementos de Orchestrator

Para crear complementos de Orchestrator se pueden utilizar diferentes enfoques. Se puede empezar por crear un complemento capa a capa, o se puede empezar por crear todas las capas del complemento al mismo tiempo.

### ■ Tipos de complementos de Orchestrator

Mediante el uso de complementos, es posible integrar bibliotecas o utilidades generales como XML o SSH, así como sistemas completos como vCloud Director, con Orchestrator. Según la tecnología que se integre con Orchestrator, los módulos se pueden categorizar como complementos para los servicios, o bien como complementos generales o complementos para sistemas.

### ■ Implementación de complementos

Puede usar técnicas y recomendaciones útiles para estructurar sus complementos, implementar las clases de Java y los objetos JavaScript requeridos, desarrollar los flujos de trabajo y las acciones del complemento y proporcionar la presentación del flujo de trabajo.

### ■ Recomendaciones para el desarrollo de complementos de Orchestrator

El cumplimiento de ciertas recomendaciones durante el desarrollo de diversos componentes de los complementos de Orchestrator contribuye a una mejor calidad de los complementos.

### ■ Documentación de API y cadenas de interfaz de usuario para complementos

Cuando escriba cadenas de la interfaz de usuario (IU) para complementos de Orchestrator y la documentación de las API relacionadas, siga las reglas de estilo y formato aceptadas.

## Enfoques para crear complementos de Orchestrator

Para crear complementos de Orchestrator se pueden utilizar diferentes enfoques. Se puede empezar por crear un complemento capa a capa, o se puede empezar por crear todas las capas del complemento al mismo tiempo.

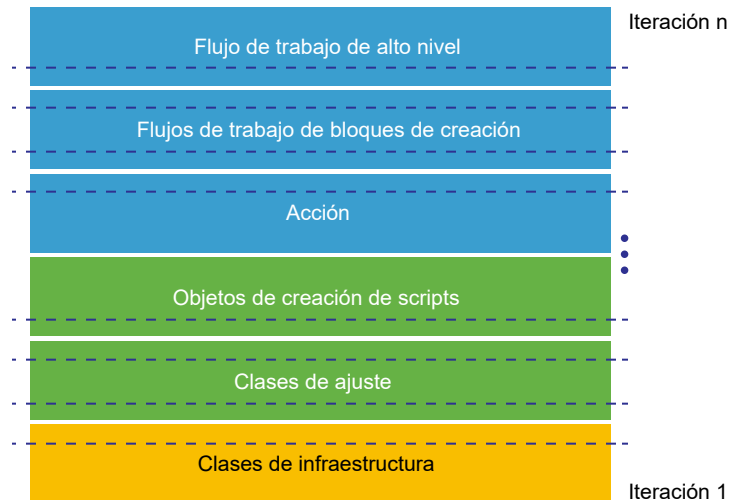
Para obtener información sobre las capas de los complementos, consulte [Estructura de un complemento de Orchestrator](#).

### Desarrollo de complementos de abajo arriba

Un complemento se puede crear capa por capa mediante el enfoque de desarrollo de abajo arriba.

El enfoque de desarrollo de abajo arriba crea el complemento capa por capa desde las capas de nivel inferior hasta las de nivel superior. Cuando este enfoque se combina con un enfoque de desarrollo interactivo e iterativo, se crea toda la capa o parte de ella para cada iteración. Tras un número N de iteraciones, el complemento se termina.

**Figura 1-5. Desarrollo de complementos de abajo arriba**



Una ventaja del enfoque de desarrollo de abajo arriba es que el desarrollo se efectúa capa por capa.

Sin embargo, también es necesario tener en cuenta las desventajas siguientes del enfoque de desarrollo de complementos de abajo arriba.

- El progreso del desarrollo de complementos es difícil de ver hasta que no se completan ciertas inserciones.
- No encaja muy bien en las prácticas de desarrollo de Agile.

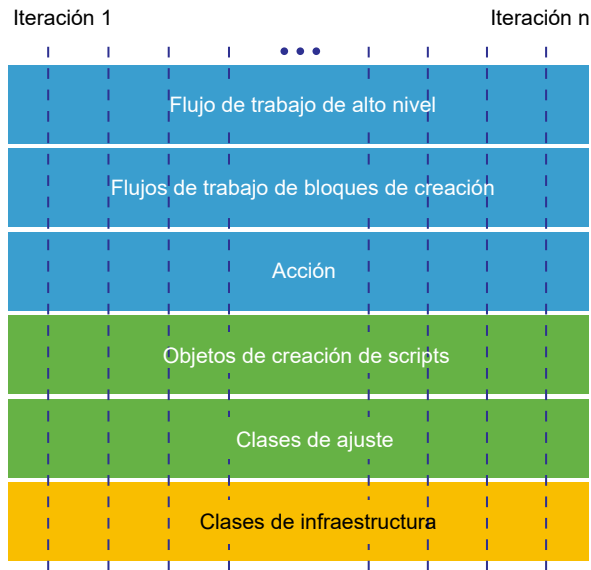
El proceso de desarrollo de abajo arriba se considera apto para complementos pequeños, que tengan pocos objetos de creación de scripts, clases de ajuste, acciones o flujos de trabajo, o bien que no los tengan.

## Desarrollo de complementos de arriba abajo

Un complemento se puede crear dividiéndolo en una funcionalidad de arriba abajo mediante el enfoque de desarrollo de arriba abajo.

Cuando el enfoque de arriba abajo se combina con un proceso de desarrollo de Agile, se ofrece nueva funcionalidad para cada iteración. Como resultado, al final de la iteración N, el complemento se implementa del todo.

**Figura 1-6. Desarrollo de complementos de arriba abajo**



El enfoque de desarrollo de complementos de arriba abajo presenta las ventajas siguiente.

- El progreso del desarrollo de complementos es fácil de mostrar desde la primera iteración, ya que la nueva funcionalidad se completa para cada iteración, y el complemento se puede lanzar y utilizar después de cada iteración.
- Completar una división vertical de la funcionalidad permite establecer criterios de éxito bien definidos y determinar lo que se ha efectuado. Además, posibilita una mejor comunicación entre desarrolladores, ingenieros de administración de productos e ingenieros de control de calidad (CC).
- Permite a los ingenieros de CC empezar las pruebas y la automatización desde el inicio del proceso de desarrollo. Gracias a este enfoque, se obtienen comentarios valiosos y se reduce el tiempo de entrega global del proyecto.

Una desventaja del enfoque de desarrollo de complementos de arriba abajo es que el desarrollo está activo en diferentes capas al mismo tiempo.

Es necesario aplicar el proceso de desarrollo de complementos de arriba abajo para la mayoría de los complementos. Resulta adecuado para los complementos con requisitos dinámicos.

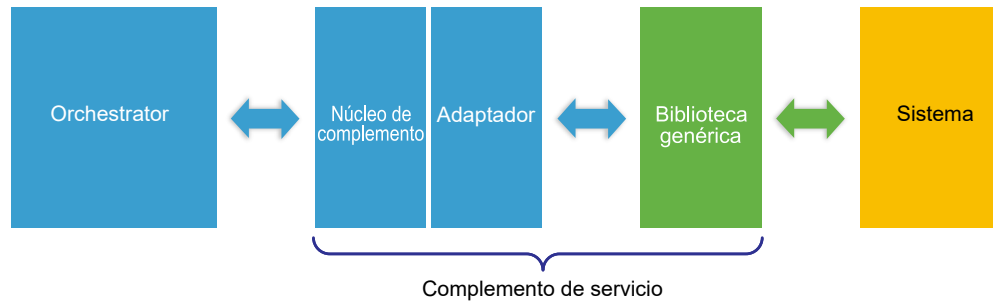
## Tipos de complementos de Orchestrator

Mediante el uso de complementos, es posible integrar bibliotecas o utilidades generales como XML o SSH, así como sistemas completos como vCloud Director, con Orchestrator. Según la tecnología que se integre con Orchestrator, los módulos se pueden categorizar como complementos para los servicios, o bien como complementos generales o complementos para sistemas.

### Complementos para servicios

Los complementos para servicios o complementos de propósito general ofrecen funciones que pueden considerarse como un servicio dentro de Orchestrator.

**Figura 1-7. Arquitectura de complementos para servicios**



Los complementos para servicios exponen a Orchestrator utilidades o bibliotecas genéricas tales como XML, SSH o SOAP. Por ejemplo, estos complementos disponibles en Orchestrator son complementos para servicios:

<b>Complemento de JDBC</b>	Le permite usar cualquier base de datos dentro de un flujo de trabajo.
<b>Complemento de correo</b>	Le permite enviar correo electrónico dentro de un flujo de trabajo.
<b>Complemento de SSH</b>	Le permite abrir conexiones SSH y ejecutar comandos dentro de un flujo de trabajo.
<b>Complemento de XML</b>	Le permite administrar documentos XML dentro de un flujo de trabajo.

Los complementos para servicios tienen estas características:

<b>Complejidad</b>	Los complementos para servicios tienen niveles de complejidad bajos o intermedios. Los complementos para servicios exponen una biblioteca específica, o parte de una, dentro de Orchestrator para proporcionar funciones concretas. Por ejemplo, el complemento de XML añade una implementación de analizador de modelo de objetos de documento (DOM) XML a la API de JavaScript de Orchestrator.
<b>Tamaño</b>	Los complementos para servicios suelen ser relativamente pequeños. Requieren el mismo conjunto de clases básico que los demás

complementos, y también necesitan otras clases que ofrezcan nuevos objetos de creación de scripts para añadir funciones nuevas.

## Inventario

Los complementos para servicios requieren un inventario de objetos pequeño o puede que no lo necesiten en absoluto. Los complementos para servicios tienen un modelo de objetos pequeño y genérico, que no necesitan mostrar dentro del inventario de Orchestrator.

## Complementos para sistemas

Los complementos para sistemas conectan el motor de flujos de trabajo de Orchestrator a un sistema externo, para permitirle orquestar este último.

Los siguientes son ejemplos de complementos para sistemas:

<b>Complemento de vCenter Server</b>	Le permite administrar instancias de vCenter Server mediante flujos de trabajo.
<b>Complemento de vCloud Director</b>	Le permite interactuar con una instalación de vCloud Director dentro de un flujo de trabajo.
<b>Complemento de Cisco UCSM</b>	Le permite interactuar con entidades Cisco dentro de un flujo de trabajo.

Estas son las principales características de los complementos para sistemas.

## Complejidad

Los complementos para sistemas tienen un nivel de complejidad mayor que el de los complementos con propósito general, porque las tecnologías que exponen son relativamente complejas. Los complementos para sistemas deben representar todos los elementos del sistema externo dentro de Orchestrator para interactuar con el sistema externo y ofrecer sus funciones en Orchestrator. Si el sistema externo proporciona un mecanismo de integración, puede usarlo para exponer las funciones del sistema en Orchestrator más fácilmente. No obstante, además de representar los elementos del sistema externo en Orchestrator, los complementos para sistemas también podrían tener que ofrecer amplia escalabilidad, proporcionar un mecanismo de almacenamiento en caché, gestionar eventos y notificaciones, etc.

## Tamaño

Los complementos para sistemas son de tamaño de mediano a grande. Los complementos para sistemas requieren muchas clases aparte de los conjuntos de clases básicos, ya que normalmente ofrecen un gran número de objetos de creación de scripts. Los complementos para sistemas podrían requerir otras clases auxiliares y de ayuda con las que interactuar.

## Inventario

Por lo general, los complementos para sistemas tienen un gran número de objetos, que usted debe exponer debidamente en el inventario para poder localizarlos y trabajar fácilmente con ellos en Orchestrator. Dado el elevado número de objetos que deben exponer los complementos para sistemas, le

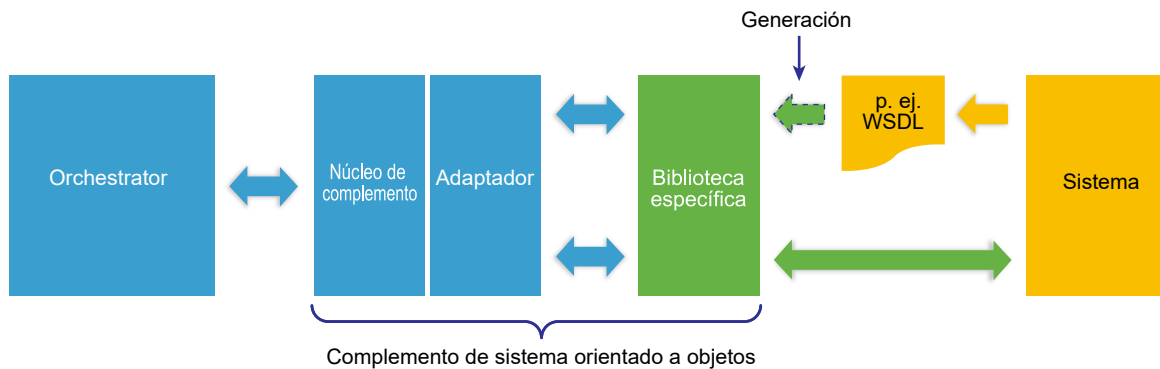
convendría crear una herramienta auxiliar o un proceso para generar automáticamente tanto código como sea posible para el complemento. Por ejemplo, el complemento de vCenter Server proporciona una herramienta de ese tipo.

## Complementos para sistemas orientados a objetos

Los sistemas orientados a objetos ofrecen un mecanismo de interacción basado en objetos y RPC.

El modelo más usado para un sistema orientado a objetos es el de un servicio web con SOAP. Los objetos dentro de este modelo tienen una serie de atributos relacionados con el estado de esos objetos y ofrecen un conjunto de métodos remotos que se invocan en el lado del sistema de destino.

### Figura 1-8. Complementos para sistemas orientados a objetos



Al implementar complementos para sistemas orientados a objetos, le conviene tener en cuenta lo siguiente:

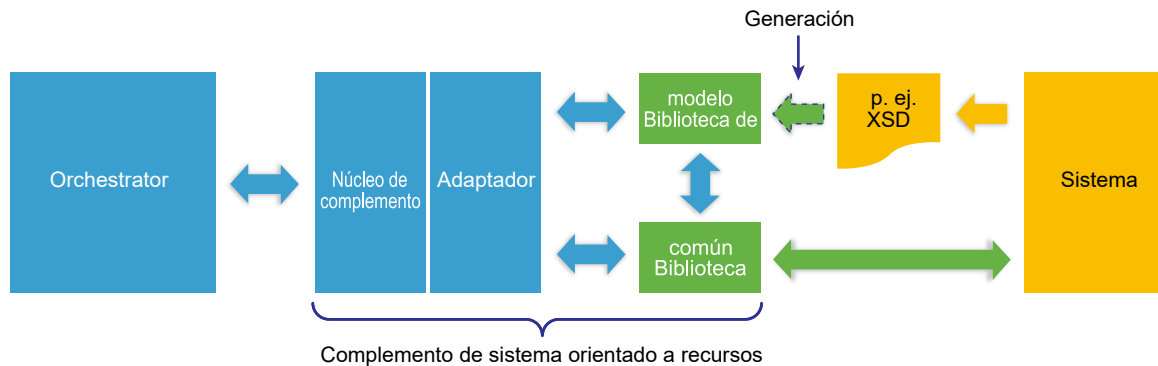
- Si usa SOAP, puede utilizar el archivo WSDL para generar un conjunto de clases que combine el modelo de objetos y el mecanismo de comunicación.
- Este modelo de objetos es casi lo único que tiene que exponer dentro de Orchestrator.

## Complementos para sistemas orientados a recursos

Los sistemas orientados a recursos proporcionan un mecanismo de interacción basado en recursos y operaciones sencillas que usan métodos HTTP.

El modelo más representativo para un sistema orientado a recursos es REST combinado, por ejemplo, con XML. Los objetos dentro de este modelo tienen una serie de atributos relacionados con el estado de esos objetos. Para invocar métodos en el sistema de destino (mecanismo de comunicación), debe usar el método HTTP estándar, por ejemplo GET, POST o PUT y seguir ciertas convenciones.



**Figura 1-9. Complementos para sistemas orientados a recursos**

Al desarrollar complementos para sistemas orientados a recursos, le conviene tener en cuenta lo siguiente:

- Si usa REST o únicamente HTTP con XML, obtiene uno o más archivos de esquema XML para leer y escribir mensajes. A partir de esos esquemas, puede generar una serie de clases que definen el modelo de objetos. Esta serie de clases solo define el estado de los objetos, ya que las operaciones se definen implícitamente con los métodos HTTP, por ejemplo según lo definido en el complemento de vCloud Director, o explícitamente con mensajes XML específicos, como es el caso en el complemento de Cisco UCSM.
- Debe implementar el mecanismo de comunicación en otro conjunto de clases. Este conjunto de clases define un nuevo modelo de objetos que interactúa con el modelo original. El modelo de objetos para el mecanismo de comunicación se compone únicamente de objetos y métodos.
- Puede exponer tanto el modelo de objetos original como el modelo de objetos para el mecanismo de comunicación dentro de Orchestrator. Esto podría aumentar la complejidad, según cómo se expongan ambos modelos de objetos y también dependiendo de si se van a combinar objetos relacionados de ambos lados (para simular un sistema orientado a objetos) o se van a mantener separados.

## Implementación de complementos

Puede usar técnicas y recomendaciones útiles para estructurar sus complementos, implementar las clases de Java y los objetos JavaScript requeridos, desarrollar los flujos de trabajo y las acciones del complemento y proporcionar la presentación del flujo de trabajo.

### ■ Estructura de los proyectos

Puede aplicar una estructura estándar para los proyectos de complementos de Orchestrator.

### ■ Elementos internos del proyecto

Al implementar un complemento puede aplicar diversos enfoques, como usar objetos de caché y objetos en el fondo, clonar objetos, etc. De esa forma se puede mejorar el rendimiento de los complementos, evitar problemas de simultaneidad y mejorar la respuesta del cliente de Orchestrator.

### ■ Elementos internos del flujo de trabajo

Puede implementar un flujo de trabajo para supervisar operaciones prolongadas realizadas por su complemento de Orchestrator.

### ■ Flujos de trabajo y acciones

Para facilitar el desarrollo y el uso del flujo de trabajo, puede seguir ciertas recomendaciones.

### ■ Presentación del flujo de trabajo

Al crear la presentación de un flujo de trabajo, debería aplicar una estructura y unas reglas determinadas.

## Estructura de los proyectos

Puede aplicar una estructura estándar para los proyectos de complementos de Orchestrator.

Puede utilizar una estructura Maven estándar con módulos para que los proyectos de complemento distingan dónde se encuentra cada función.

**Tabla 1-49. Estructura de un proyecto de complemento**

Módulo	Descripción
/myAwesomePlugin-plugin	Raíz del proyecto de complemento.
/o11nplugin-myAwesomePlugin	Módulo que compone el archivo DAR del complemento final.
/o11nplugin-myAwesomePlugin-config	Módulo que contiene la aplicación web de configuración del complemento. Genera un archivo WAR estándar.
/o11nplugin-myAwesomePlugin-core	Módulo que contiene todas las clases que implementan cualquier interfaz de módulo estándar de Orchestrator y otras clases auxiliares que utilicen. Genera un archivo JAR estándar.
/o11nplugin-myAwesomePlugin-model	Módulo que contiene todas las clases que permiten integrar tecnología de terceros con Orchestrator a través del complemento. Las clases no deben contener ninguna referencia directa a las API de complemento estándar de Orchestrator.
/o11nplugin-myAwesomePlugin-package	Módulo que importa un archivo de paquete externo de Orchestrator con acciones y flujos de trabajo para incluirlo dentro del archivo DAR del complemento final. El módulo es opcional.

## Elementos internos del proyecto

Al implementar un complemento puede aplicar diversos enfoques, como usar objetos de caché y objetos en el fondo, clonar objetos, etc. De esa forma se puede mejorar el rendimiento de los complementos, evitar problemas de simultaneidad y mejorar la respuesta del cliente de Orchestrator.

### Objetos de caché

El complemento puede interactuar con un servicio remoto, y esta interacción la proporcionan los objetos locales que representan a objetos remotos en el lado de servicio. Para lograr un buen rendimiento del complemento así como una buena respuesta de la interfaz de usuario de Orchestrator, puede poner en caché los objetos locales, en lugar de tener que obtenerlos de un servicio remoto cada vez que se

requieren. Puede decidir el ámbito de la memoria caché, por ejemplo: una caché para todos los clientes del complemento, una por cada usuario del complemento y una por cada usuario del servicio de terceros. Al implementar el mecanismo de caché, se integra con la interfaz del complemento para encontrar e invalidar objetos.

### Usar objetos en el fondo

Si tiene que mostrar listas de objetos largas en el inventario del complemento y no tiene un método rápido para recuperar esos objetos, puede usar objetos en el fondo. Para ello puede, por ejemplo, tener objetos con dos estados: `fake` y `loaded`. Teniendo en cuenta que los objetos `fake` son muy fáciles de crear y proporcionan la información mínima que mostrar en el inventario (por ejemplo, el nombre y el ID), sería posible devolver siempre objetos `fake`, y cuando se necesita toda la información (el objeto real), la entidad que lo requiere o el complemento pueden llamar a un método `load` automáticamente para obtener el objeto real. Incluso puede configurar el proceso de carga de objetos para que se inicie automáticamente después de que se devuelvan los objetos falsos, para prever las acciones de la entidad que los usa.

### Clonar objetos para evitar problemas de simultaneidad

Si usa una caché para su complemento, tiene que clonar objetos. Puede haber efectos imprevistos si usa una caché que siempre devuelve la misma instancia de un objeto para cada entidad que lo solicita. Por ejemplo, la entidad A solicita el objeto O, y la entidad visualiza el objeto en el inventario con todos sus atributos. Al mismo tiempo, la entidad B solicita el objeto O también y la entidad A ejecuta un flujo de trabajo que empieza a cambiar los atributos del objeto O. Al final de la ejecución, el flujo de trabajo llama al método `update` del objeto para actualizar el objeto en el lado del servidor. Si las entidades A y B obtienen la misma instancia del objeto O, la entidad A ve en el inventario todos los cambios que realice la entidad B, incluso antes de que los cambios se validen en el lado del servidor. Si la ejecución se realiza correctamente, no debería haber problemas; pero si la ejecución falla, no se revertirán los atributos del objeto O para la entidad A. De ocurrir eso, si la caché (las operaciones de `find` en el complemento) devuelve un clon del objeto en lugar de la misma instancia continuamente, cada entidad ve y modifica su propia copia, y se evitan problemas de simultaneidad, al menos dentro de Orchestrator.

### Notificar cambios a otros

Se pueden producir problemas si se usa una caché mientras se clonan objetos. El problema más grave es que el objeto que usa vistas de entidad podría no ser la versión más reciente disponible. Por ejemplo, si una entidad muestra el inventario, los objetos se cargan una vez pero, al mismo tiempo, si otra entidad está cambiando objetos, la primera entidad no ve esos cambios. Para evitar este problema puede usar los métodos `PluginWatcher` y `IPluginPublisher` de la API de complemento de Orchestrator para notificar que algo ha cambiado para permitir que otras instancias de clientes de Orchestrator vean los cambios. Esto también es aplicable a una sola instancia del cliente de Orchestrator cuando los cambios de un objeto del inventario afectan a otros objetos del inventario y también es necesario notificarlos. Las operaciones que más tienden a usar notificaciones son las de añadir, actualizar y eliminar objetos cuando estos, o algunas de sus propiedades, aparecen en el inventario.

### Activar la localización de cualquier objeto en cualquier momento

Debe implementar el método `find` de la interfaz `IPluginFactory` para encontrar objetos solo por tipo e ID. El método `find` se puede llamar directamente después del reinicio de Orchestrator y la reanudación de un flujo de trabajo.

## Simular un servicio de consultas si no existe uno

El cliente de Orchestrator puede requerir consultas para algunos objetos en casos específicos o que se muestren no en estructura de árbol sino como listas o tablas, por ejemplo. Esto significa que el complemento debe poder realizar consultas de conjuntos de objetos en cualquier momento. Si la tecnología de terceros ofrece un servicio de consultas, tendrá que adaptarse a usar este servicio. De no hacerlo así, tal vez pueda simular este tipo de servicio, aunque podría ser una solución más compleja o de rendimiento inferior.

## Los métodos de búsqueda no deberían devolver excepciones de tiempo de ejecución

Los métodos de la interfaz `IPluginFactory` que implementan las búsquedas dentro del complemento no deberían devolver excepciones de tiempo de ejecución, ni controladas ni no controladas. Esto podría causar extraños fallos de error de validación *error de validación* durante la ejecución de un flujo de trabajo. Por ejemplo, entre dos nodos de un flujo de trabajo, se llama al método `find` si una salida del primer nodo es una entrada del segundo nodo. En ese momento, si el objeto no se encuentra por haberse producido una excepción de tiempo de ejecución, puede que lo único que obtenga en el cliente de Orchestrator sea un *error de validación*. Después de eso, obtendrá más o menos información en los archivos de registro dependiendo de cómo registra excepciones el complemento.

## Elementos internos del flujo de trabajo

Puede implementar un flujo de trabajo para supervisar operaciones prolongadas realizadas por su complemento de Orchestrator.

Puede implementar un flujo de trabajo para supervisar operaciones largas, como la supervisión de tareas. Este flujo de trabajo se puede basar en eventos en espera y activadores de Orchestrator. Tenga en cuenta que un flujo de trabajo bloqueado a la espera de una tarea se puede reanudar en cuanto se inicia el servidor de Orchestrator. El complemento debe poder obtener toda la información requerida para que el proceso de supervisión se reanude correctamente.

El flujo de trabajo de supervisión o la tarea que puede usar internamente deberían proporcionar un mecanismo para especificar la tasa de sondeo y un posible periodo de tiempo de espera.

El proceso de depuración de una porción de código de script dentro de un flujo de trabajo no es una tarea simple, especialmente si el código no invoca ningún código Java. Por esa razón, a veces la única opción consiste en usar métodos de registro ofrecidos por los objetos de script predeterminados de Orchestrator.

## Flujos de trabajo y acciones

Para facilitar el desarrollo y el uso del flujo de trabajo, puede seguir ciertas recomendaciones.

### Iniciar el desarrollo de los flujos de trabajo como bloques de creación

Un bloque de creación puede ser un flujo de trabajo sencillo que requiere unos cuantos parámetros de entrada y devuelve una salida simple. Si tiene un conjunto amplio de bloques de creación, puede crear flujos de trabajo de alto nivel fácilmente, y ofrecer mejores herramientas para componer flujos de trabajo complejos.

## Crear flujos de trabajo de nivel más alto basados en componentes menores

Si tiene que desarrollar un flujo de trabajo complejo con varias entradas y pasos internos, puede dividirlo en acciones y flujos de trabajo con bloques de creación menores y más simples.

### Crear acciones siempre que sea posible

Puede crear acciones para tener más flexibilidad al desarrollar flujos de trabajo.

- Para crear fácilmente parámetros u objetos complejos para métodos de creación de scripts
- Para evitar repetir constantemente elementos comunes de código
- Para realizar validaciones de interfaz de usuario

### Los flujos de trabajo deberían invocar acciones siempre que sea posible

Las acciones pueden invocarse directamente como nodos dentro del esquema de flujo de trabajo. Esto puede contribuir a un esquema de flujo de trabajo más simple, al no haber necesidad de añadir bloques de código de script para invocar una sola acción.

### Rellenar la información prevista

Proporcione información para cada elemento de un flujo de trabajo o una acción.

- Proporcione una descripción del flujo de trabajo o la acción.
- Proporcione una descripción de los parámetros de entrada.
- Proporcione una descripción de las salidas.
- Proporcione una descripción de los atributos para los flujos de trabajo.

### Mantener actualizada la información de versión

Al crear versiones de complementos, añada comentarios relevantes con datos como actualizaciones importantes aplicadas al complemento, detalles de implementaciones, etc.

### Presentación del flujo de trabajo

Al crear la presentación de un flujo de trabajo, debería aplicar una estructura y unas reglas determinadas.

Use las siguientes propiedades para las entradas de flujo de trabajo en la presentación.

**Tabla 1-50. Propiedades para entradas de flujo de trabajo**

Propiedades	Uso
Show in Inventory	Use esta propiedad para ayudar al usuario a ejecutar un flujo de trabajo desde la vista de inventario.
Specify a root object to be shown in the chooser	Use esta propiedad para ayudar al usuario a seleccionar entradas. Si el objeto raíz se puede actualizar en la presentación, es un atributo o se recupera mediante un método de objetos, tendrá que crear o configurar una acción adecuada para actualizar el objeto en la presentación.
Maximum string length	Use esta propiedad para cadenas largas, como nombres, descripciones, rutas de archivo, etc.

**Tabla 1-50. Propiedades para entradas de flujo de trabajo (continuación)**

Propiedades	Uso
Minimum string length	Use esta propiedad para evitar cadenas vacías de herramientas de prueba.
Custom validation	Sirve para implementar validaciones que no sean simples con acciones.

Organice las entradas con pasos y un grupo de visualización. Dicha organización ayuda al usuario a identificar y distinguir todos los parámetros de entrada de un flujo de trabajo.

## Recomendaciones para el desarrollo de complementos de Orchestrator

El cumplimiento de ciertas recomendaciones durante el desarrollo de diversos componentes de los complementos de Orchestrator contribuye a una mejor calidad de los complementos.

**Tabla 1-51. Recomendaciones útiles para la implementación de complementos**

Componente	Elemento	Descripción
General	Acceso a API de terceros	Los complementos deberían proporcionar métodos simplificados para acceder a API de terceros siempre que sea posible.
	Interfaz	Los complementos deberían proporcionar una interfaz estándar para los usuarios, aunque la API no lo haga.
Acción	Objetos de creación de scripts	Debería crear acciones para cada creación, modificación, eliminación y los demás métodos disponibles para un objeto de creación de scripts.
	Descripción	La descripción de una acción debería describir qué hace la acción, no cómo funciona.
	Creación de scripts	Al usar la creación de scripts para obtener las propiedades o métodos de un objeto, puede comprobar si el valor del objeto es distinto de null o undefined.
	En desuso	Si una acción está en desuso, la instrucción <code>comment</code> o <code>throw</code> debería indicar la acción de reemplazo; o la acción debería llamar a una nueva acción de reemplazo para que no fallen las soluciones basadas en la versión en desuso.
Flujo de trabajo	Operaciones de interfaz de usuario en la tecnología orquestada	Debería crear un flujo de trabajo para cada operación disponible en la interfaz de usuario de la tecnología orquestada.
	Descripción	La descripción de un flujo de trabajo debería describir qué hace el flujo, no cómo funciona.
	Propiedad de presentación <code>mandatory input</code>	Debe configurar la propiedad <code>mandatory input</code> para todas las entradas de flujo de trabajo obligatorias.
	Propiedad de presentación <code>default value</code>	Si desarrolla un flujo de trabajo que configura una entidad, la presentación del flujo de trabajo debería cargar los valores de configuración predeterminados para esta entidad. Por ejemplo, si desarrolla un flujo de trabajo denominado Configuración de host, la presentación del flujo de trabajo debe cargar los valores predeterminados de la configuración de host.

**Tabla 1-51. Recomendaciones útiles para la implementación de complementos (continuación)**

Componente	Elemento	Descripción
	Propiedad de presentación <code>Show in inventory</code>	Debe configurar la propiedad <code>Show in inventory</code> para tener flujos de trabajo contextuales en objetos de inventario.
	Propiedad de presentación <code>specify a root parameter</code>	Debería usar esta propiedad en flujos de trabajo cuando no sea necesario examinar el inventario desde la raíz del árbol.
	Validación de flujos de trabajo	Debe validar los flujos de trabajo y corregir todos los errores.
	Creación de objetos	Todos los flujos de trabajo que crean un objeto nuevo deberían devolver este como parámetro de salida.
	En desuso	Si se deja de usar un flujo de trabajo, las instrucciones <code>comment</code> o <code>throw</code> deberían indicar el flujo de trabajo de reemplazo; o el flujo de trabajo debería llamar a un nuevo flujo de trabajo de reemplazo para que no fallen las soluciones basadas en la versión en desuso.
	Desconexión de host	Si el inventario contiene una conexión a un host y este deja de estar disponible, debería indicar que el host se ha desconectado. Puede hacerlo asignando otro nombre al objeto raíz, añadiéndole <code>- disconnected</code> , o quitando el árbol de objetos debajo de este, como lo hace el complemento de vCloud Director.
	Propiedad <code>Select value as list</code>	Un objeto de inventario debe estar seleccionable como <code>treeview</code> o como <code>list</code> .
	Administrador de host	Si el complemento implementa un objeto <code>host</code> para el sistema de origen, debería existir un objeto raíz <code>hostmanager</code> principal con propiedades para agregar, quitar y editar propiedades de host.
Inventario	Obtener o actualizar objetos	Si hay un servicio de consultas en ejecución en la tecnología orquestada, debería usarlo para obtener varios objetos.
	Detección de objetos secundarios	Si tiene que recuperar objetos secundarios uno por uno, el proceso de recuperación debe ser de tipo multiproceso y no bloqueable por un solo error.
	Cambio de objetos de Orchestrator	Todos los flujos de trabajo que pueden cambiar el estado de un elemento en el inventario deben actualizar este para evitar que haya objetos desincronizados.
	Cambio de objetos externos	Puede usar un mecanismo de notificación de cambios producidos en la tecnología orquestada como resultado de operaciones realizadas fuera de Orchestrator. Si esas operaciones provocan la desaparición de objetos de la tecnología orquestada, tendrá que actualizar el inventario según corresponda, para evitar fallos o pérdidas de datos. Por ejemplo, si una máquina virtual se elimina de vCenter Server, el complemento de vCenter Server actualiza el inventario para quitar el objeto de la máquina virtual eliminada.
	Objeto de buscador	Los objetos de buscador deberían tener propiedades utilizables para diversos objetos. Estas propiedades son las que suelen estar presentes en la interfaz de usuario.
Objeto de creación de scripts	Implementación	Se debe implementar el método <code>equals</code> para asegurar el funcionamiento de la operación <code>==</code> en el mismo objeto, ya que en ciertos casos podría haber dos instancias de un objeto.

**Tabla 1-51. Recomendaciones útiles para la implementación de complementos (continuación)**

Componente	Elemento	Descripción
	Propiedades de objetos de complementos	Los objetos que tienen objetos principales deberían implementar una propiedad <code>parent</code> .
	Propiedades de objetos de complementos	Los objetos que tienen objetos secundarios deberían implementar métodos <code>GET</code> que devuelvan matrices de objetos secundarios.
	Objetos de inventario	Los objetos de inventario deberían admitir búsquedas con <code>Server.find</code> .
		Todos los objetos de inventario deberían ser serializables para poder usarse como atributos de entrada o salida de un flujo de trabajo.
	Constructor y métodos	En la mayoría de los casos, los objetos de scripts deberían tener un constructor o ser devueltos por otros métodos o atributos de objeto.
	ID de objeto	Los objetos con un ID emitido por un sistema externo deberían usar un ID interno para asegurar que no hay duplicaciones de ID al orquestar más de un servidor.
	Buscar objetos	Los métodos <code>search</code> o <code>find</code> deberían implementar un filtro para encontrar el nombre o ID especificado, en lugar de todos los objetos. Por ejemplo, el servidor de Orchestrator tiene un método <code>Server.FindForId</code> que permite encontrar un objeto de complemento por su ID. Para ello, el método se debe implementar para cada objeto localizable en el complemento.
	Activador	De ser posible, debería haber activadores disponibles para objetos cambiantes, de modo que se activen políticas de Orchestrator al producirse diversos eventos. Por ejemplo, para determinar cuándo se agrega, enciende, apaga, etc. una máquina virtual, Orchestrator puede supervisar un activador o un evento en el complemento de vCenter del objeto Datacenter.
	Propiedades de objetos	Los objetos que residen en otros complementos deberían tener propiedades que faciliten la conversión de un objeto de complemento a otro. Por ejemplo, los objetos de máquina virtual deben tener un <code>moref</code> (ID de referencia de objeto administrado).
	Administrador de sesión	Si se va a conectar a un servidor remoto que puede tener otra sesión, el complemento debería implementar una sesión compartida y una sesión por usuario.
Activador	Activador	Todos los métodos de bloqueo y operaciones largas deberían poder iniciarse de forma asincrónica con una tarea devuelta, y generar un evento de activador al completarse.
Enumeraciones	Enumeraciones	Las enumeraciones para un tipo determinado deberían tener un objeto de inventario que permita seleccionar entre los diversos valores de la enumeración.
Registro	registros	Los métodos deberían implementar distintos niveles de registro.
Control de versiones	Versión de complemento	La versión de complemento debería ser acorde con los estándares y actualizarse junto con el complemento.



**Tabla 1-51. Recomendaciones útiles para la implementación de complementos (continuación)**

Componente	Elemento	Descripción
documentación de las API	Métodos	Los métodos descritos en la documentación de API nunca deberían devolver la excepción <code>xyz method / property</code> para un objeto. Deberían devolver <code>null</code> cuando no hay propiedades disponibles y documentarse con detalles cuando estas propiedades no están disponibles.
	<code>vso.xml</code>	Todos los objetos, métodos y propiedades deben documentarse en <code>vso.xml</code> .

## Documentación de API y cadenas de interfaz de usuario para complementos

Cuando escriba cadenas de la interfaz de usuario (IU) para complementos de Orchestrator y la documentación de las API relacionadas, siga las reglas de estilo y formato aceptadas.

### Recomendaciones generales

- Utilice los nombres oficiales de los productos de VMware que forman parte del complemento. Por ejemplo, utilice los nombres oficiales de los productos siguientes y la terminología de VMware.

Término correcto	No utilizar
vCenter Server	VC o vCenter
vCloud Director	vCloud

- Termine todas las descripciones de flujos de trabajo con un punto. Por ejemplo, `Creates a new Organization.` es una descripción de flujo de trabajo.
- Utilice un editor de texto con un corrector ortográfico para escribir las descripciones y luego páselas al complemento.
- Asegúrese de que el nombre del complemento coincida exactamente con el nombre de producto de terceros aprobado con el que esté asociado.

### Flujos de trabajo y acciones

- Escriba descripciones informativas. Una o dos frases son suficientes para la mayoría de las acciones y los flujos de trabajo.
- Los flujos de trabajo de nivel superior podrían incluir descripciones y comentarios más extensos.
- Inicie las descripciones con un verbo, por ejemplo, `Creates....` No utilice lenguaje que se haga referencia a sí mismo, como `This workflow creates.`
- Coloque un punto al final de las descripciones que sean frases completas.
- Describa lo que hacen un flujo de trabajo o una acción en lugar de cómo se implementan.
- Los flujos de trabajo y las acciones se suelen incluir en carpetas y paquetes. Incluya también una breve descripción de estas carpetas y paquetes. Por ejemplo, una carpeta de flujo de trabajo puede tener una descripción similar a `Set of workflows related to vApp Template management.`

## Parámetros de flujos de trabajo y acciones

- Inicie las descripciones de flujos de trabajo y acciones con un sintagma nominal descriptivo, por ejemplo `Name of`. No utilice frases del tipo `It's the name of`.
- No coloque punto al final de las descripciones de parámetros y acciones. No son frases completas.
- Los parámetros de entrada de los flujos de trabajo deben especificar una etiqueta con nombres apropiados en la vista de presentación. En muchos casos, puede combinar entradas relacionadas en un grupo de visualización. Por ejemplo, en lugar de tener dos entradas con las etiquetas `Nombre de empresa` y `Nombre completo de la empresa`, puede crear un grupo de visualización con la etiqueta `Empresa` y colocar las entradas `Nombre` y `Nombre completo` en el grupo `Empresa`.
- Para los pasos y los grupos de visualización, añada descripciones o comentarios que también aparezcan en la presentación del flujo de trabajo.

## API de complemento

- La documentación de la API hace referencia a toda la documentación del archivo `vso.xml` y los archivos de origen de Java.
- Para el archivo `vso.xml`, utilice las mismas reglas para las descripciones de los objetos de buscador y de creación de scripts con sus métodos que las que utiliza para los flujos de trabajo y las acciones. Las descripciones de los atributos de objetos y los parámetros de método utilizan las mismas reglas que los parámetros de flujos de trabajo y las acciones.
- Evite los caracteres especiales en el archivo `vso.xml` e incluya las descripciones dentro de una etiqueta `<![CDATA[insert your description here!]]>`.
- Utilice el estilo de Javadoc estándar para los archivos de origen de Java.

## Obtener parámetros de entrada de los usuarios cuando se inicia un flujo de trabajo

Si un flujo de trabajo precisa de parámetros de entrada, se abre un cuadro de diálogo en el que los usuarios proporcionan los valores de parámetros de entrada necesarios cuando se ejecuta. El contenido y el diseño, o la presentación, de este cuadro de diálogo se pueden organizar en la pestaña **Presentación** en el Editor de flujos de trabajo.

La forma de organizar parámetros en la pestaña **Presentación** se refleja en el cuadro de diálogo de parámetros de entrada cuando se ejecuta el flujo de trabajo.

Además, en la pestaña **Presentación** se pueden añadir descripciones de los parámetros de entrada para ayudar a los usuarios cuando proporcionan parámetros de entrada. También puede establecer propiedades y restricciones de parámetros en la pestaña **Presentación** a fin de limitar los parámetros que pueden proporcionar los usuarios. Si los parámetros proporcionados por el usuario no se ajustan a las restricciones establecidas en la pestaña **Presentación**, el flujo de trabajo no se ejecutará.

- **Crear el cuadro de diálogo Parámetros de entrada en la pestaña Presentación**

En la pestaña **Presentación** del Editor de flujos de trabajo, se define el diseño del cuadro de diálogo en el que los usuarios proporcionan parámetros de entrada cuando ejecutan un flujo de trabajo.

- **Establecer propiedades de parámetros**

Orchestrator permite definir propiedades para cualificar los valores de los parámetros de entrada que los usuarios proporcionan cuando ejecutan flujos de trabajo. Las propiedades de parámetros que define imponen límites en los tipos y los valores de los parámetros de entrada que proporcionan los usuarios.

## Crear el cuadro de diálogo Parámetros de entrada en la pestaña Presentación

En la pestaña **Presentación** del Editor de flujos de trabajo, se define el diseño del cuadro de diálogo en el que los usuarios proporcionan parámetros de entrada cuando ejecutan un flujo de trabajo.

En la pestaña **Presentación**, se pueden agrupar parámetros de entrada en categorías, además de definir el orden de aparición de dichas categorías en el cuadro de diálogo Parámetros de entrada.

### Descripciones de presentaciones

Puede añadir una descripción asociada para cada parámetro o grupo de parámetros, que aparece en el cuadro de diálogo Parámetros de entrada. Estas descripciones proporcionan información a los usuarios para que les resulte más fácil especificar los parámetros de entrada correctos. El diseño del texto de la descripción se puede mejorar mediante formato HTML.

### Definir pasos de entrada de la presentación

De forma predeterminada, el cuadro de diálogo Parámetros de entrada muestra todos los parámetros de entrada necesarios en una sola lista. Para ayudar a los usuarios a indicar los parámetros de entrada, puede definir nodos, denominados pasos de entrada, en la pestaña Presentación. Los pasos de entrada agrupan los parámetros de entrada de naturaleza similar. Los parámetros de entrada de un determinado paso de entrada aparecen en una sección distinta en el cuadro de diálogo Parámetros de entrada cuando se ejecuta el flujo de trabajo.

### Definir grupos de visualización de la presentación

Cada paso de entrada puede tener sus propios nodos, denominados grupos de visualización. Los grupos de visualización definen el orden de aparición de los cuadros de texto de entrada de parámetros en su sección del cuadro de diálogo Parámetros de entrada. Puede definir grupos de visualización independientemente de los pasos de entrada.

## Crear la presentación del cuadro de diálogo de parámetros de entrada

Cree la presentación del cuadro de diálogo en que los usuarios proporcionan los parámetros de entrada cuando ejecutan un flujo de trabajo en la pestaña **Presentación** en el Editor de flujos de trabajo.

### Requisitos previos

- Abra un flujo de trabajo para editar en el Editor de flujos de trabajo.
- Compruebe que el flujo de trabajo tenga una lista definida de parámetros de entrada.

### Procedimiento

- 1 En el Editor de flujos de trabajo, haga clic en la pestaña **Presentación**.

De forma predeterminada, todos los parámetros del flujo de trabajo aparecen bajo el nodo principal **Presentación** en el orden en el que se crean.

- 2 Haga clic con el botón secundario en el nodo **Presentación** y seleccione **Crear paso**.

Aparece el nodo **Nuevo paso** bajo el nodo **Presentación**.

- 3 Indique un nombre adecuado para el paso y pulse Entrar.

Este nombre aparece como encabezado de sección en el cuadro de diálogo de parámetros de entrada cuando se ejecuta el flujo de trabajo.

- 4 Haga clic en el paso de entrada y añada una descripción en la pestaña **General** en la mitad inferior de la pestaña **Presentación**.

Esta descripción aparece en el cuadro de diálogo de parámetros de entrada para ofrecer información a los usuarios que les ayude a proporcionar los parámetros de entrada correctos. El diseño del texto de la descripción se puede mejorar mediante formato HTML.

- 5 Haga clic con el botón secundario en el paso de entrada que ha creado y seleccione **Crear grupo de visualización**.

Aparece el nodo **Nuevo grupo** bajo el nodo del paso de entrada.

- 6 Indique un nombre adecuado para el grupo de visualización y pulse Entrar.

Este nombre aparece como encabezado de subsección en el cuadro de diálogo de parámetros de entrada cuando se ejecuta el flujo de trabajo.

- 7 Haga clic en el grupo de visualización y añada una descripción en la pestaña **General** en la mitad inferior de la pestaña **Presentación**.

Esta descripción aparece en el cuadro de diálogo de parámetros de entrada. El diseño del texto de la descripción se puede mejorar mediante formato HTML. Puede añadir un valor de parámetro a una descripción de grupo mediante una instrucción OGNL, como `${#param}`.

- 8 Repita los pasos anteriores hasta haber creado todos los pasos de entrada y los grupos de visualización que aparecerán en el cuadro de diálogo de parámetros de entrada cuando se ejecute el flujo de trabajo.

- 9 Arrastre los parámetros que hay bajo el nodo **Presentación** a los pasos y grupos que elija.

Ha creado el diseño del cuadro de diálogo de parámetros de entrada que permite a los usuarios proporcionar valores de parámetros de entrada cuando se ejecuta el flujo de trabajo.

### Pasos siguientes

Debe establecer las propiedades de los parámetros.

## Establecer propiedades de parámetros

Orchestrator permite definir propiedades para cualificar los valores de los parámetros de entrada que los usuarios proporcionan cuando ejecutan flujos de trabajo. Las propiedades de parámetros que define imponen límites en los tipos y los valores de los parámetros de entrada que proporcionan los usuarios.

Cada parámetro puede tener varias propiedades. Las propiedades de un parámetro de entrada se definen en la pestaña **Propiedades** de un determinado parámetro en la pestaña **Presentación**.

Las propiedades de parámetro validan los parámetros de entrada; asimismo, modifican el modo en el que los cuadros de texto aparecen en el cuadro de diálogo de parámetros de entrada. Algunas propiedades de parámetro pueden crear dependencias entre parámetros.

### Valores de propiedades de parámetro estáticas y dinámicas

Un valor de propiedad de parámetro puede ser estático o dinámico. Los valores de propiedad estáticos permanecen constantes. Si establece un valor de propiedad en estático, puede definir el valor de la propiedad en una lista generada por el Editor de flujos de trabajo conforme al tipo de parámetro.

Los valores de propiedad dinámicos dependen del valor de otro parámetro o atributo. Defina las funciones según las cuales las propiedades dinámicas obtienen valores mediante una expresión de OGNL. Si una propiedad dinámica de parámetro depende del valor de otra propiedad de parámetro y cambia el valor de la otra propiedad de parámetro, la expresión de OGNL vuelve a calcular y cambia el valor de la propiedad dinámica.

## Establecer propiedades de parámetros

Cuando se inicia un flujo de trabajo, valida valores de parámetros de entrada de usuarios en relación con cualquier propiedad de parámetros que se haya establecido.

### Requisitos previos

- Abra un flujo de trabajo para editar en el Editor de flujos de trabajo.
- Compruebe que el flujo de trabajo tenga una lista definida de parámetros de entrada.

### Procedimiento

- 1 En el Editor de flujos de trabajo, haga clic en la pestaña **Presentación**.
- 2 En la pestaña **Presentación**, haga clic en un parámetro.

Las pestañas **General** y **Propiedades** del parámetro aparecen en la parte inferior de la pestaña **Presentación**.

- 3 Haga clic en la pestaña **Propiedades** del parámetro.



- 4 Haga clic con el botón secundario en la pestaña **Propiedades** y seleccione **Añadir propiedad**.

Se abre un cuadro de diálogo con una lista de propiedades posibles para un parámetro del tipo que se ha seleccionado.

- 5 Seleccione una propiedad de la lista que aparece en el cuadro de diálogo y haga clic en **Aceptar**.

La propiedad aparece en la pestaña **Propiedades**.



- 6 En **Valor**, establezca la propiedad en estática o dinámica seleccionando el símbolo correspondiente en el menú desplegable.

Opción	Descripción
	Propiedad estática
	Propiedad dinámica

- 7 Si establece en estática el valor de la propiedad, seleccione un valor de propiedad conforme al tipo de parámetro para el que se establecen las propiedades.

- 8 Si establece en dinámica el valor de la propiedad, la función se define para obtener el valor de la propiedad de parámetro mediante una expresión OGNL.

El Editor de flujos de trabajo proporciona ayuda para escribir la expresión OGNL.

- Haga clic en  icono para obtener una lista de todos los atributos y parámetros definidos por el flujo de trabajo a los que puede llamar esta expresión.
- Haga clic en el  icono para obtener una lista de todas las acciones de la API de Orchestrator API que devuelven un parámetro de salida del tipo para el que se definen las propiedades.

Al hacer clic en los elementos de las listas propuestas de acciones y parámetros, se añaden a la expresión OGNL.

- 9 Haga clic en **Guardar** en la parte inferior del Editor de flujos de trabajo.

Ha definido las propiedades de los parámetros de entrada del flujo de trabajo.

### Pasos siguientes

Valide y depure el flujo de trabajo.

## Object Missing

This object is not available in the repository.

## Valores de constantes predefinidas para expresiones OGNL

Puede utilizar constantes predefinidas cuando cree expresiones OGNL para obtener valores de propiedades de parámetros dinámicos.

Orchestrator define las constantes siguientes para utilizar en expresiones OGNL.

**Tabla 1-52. Valores de constantes OGNL predefinidas**

Valor de constante	Descripción
<code>\${#__current}</code>	Valor actual de la propiedad de validación personalizada o de la propiedad de expresión coincidente.
<code>\${#__username}</code>	Nombre del usuario que ha iniciado el flujo de trabajo
<code>\${#__userdisplayname}</code>	Nombre para mostrar del usuario que ha iniciado el flujo de trabajo
<code>\${#__serverurl}</code>	URL que contiene la dirección IP del servidor desde el cual el usuario inicia el flujo de trabajo. La URL cuenta con la dirección IP del servidor y un puerto de búsqueda:  <code>{ServerIP}:{lookupPort}</code>
<code>\${#__datetime}</code>	Fecha y hora actuales.
<code>\${#__date}</code>	Fecha actual, con el tiempo establecido en 00:00:00.
<code>\${#__timezone}</code>	Zona horaria actual.

## Interacciones de usuario requeridas durante la ejecución de un flujo de trabajo

En ocasiones, un flujo de trabajo puede requerir parámetros de entrada adicionales de una fuente externa durante la ejecución. Estos parámetros de entrada pueden proceder de otra aplicación u otro flujo de trabajo, o los puede proporcionar el usuario directamente.

Por ejemplo, si se produce cierto evento mientras se ejecuta un flujo de trabajo, el flujo puede solicitar una interacción humana para decidir qué acciones llevar a cabo. El flujo de trabajo espera antes de continuar, bien hasta que el usuario responda a la solicitud de información o hasta que se agote el tiempo de espera, si se ha establecido uno. Si se agota el tiempo de espera, el flujo de trabajo devuelve una excepción.

Los atributos predeterminada para interacciones de usuario son `security.group` y `timeout.date`. Al configurar el atributo `security.group` para un grupo concreto de usuarios de LDAP, el permiso de respuesta a la solicitud de interacción del usuario se limita a los miembros de ese grupo de usuarios.

Al configurar el atributo `timeout.date`, debe definir la fecha y hora hasta las que esperará el flujo de trabajo para que el usuario proporcione la información. Puede configurar una fecha absoluta o crear un elemento de flujo de trabajo con script para calcular una hora relativa a la actual.

### Procedimiento

#### 1 (opcional) Añadir una interacción del usuario a un flujo de trabajo

Solicite parámetros de entrada de usuarios durante una ejecución de flujo de trabajo añadiendo un elemento de esquema **Interacción del usuario** al flujo de trabajo. Cuando un flujo de trabajo detecta un elemento **Interacción del usuario**, suspende su ejecución y espera a que el usuario proporcione los datos que necesita.

**2 (opcional) Configurar el atributo `security.group` para la interacción del usuario**

El atributo `security.group` de un elemento de interacción del usuario establece los usuarios o los grupos de usuarios que tienen permiso para responder a la interacción del usuario.

**3 (opcional) Establecimiento del atributo `timeout.date` en una fecha absoluta**

Establezca el atributo `timeout.date` para que una interacción del usuario determine durante cuánto espera el flujo de trabajo a que un usuario responda a una interacción.

**4 (opcional) Cálculo del tiempo de espera relativo para las interacciones del usuario**

Puede calcular en un objeto `Date` una fecha y hora relativas para el tiempo de espera de una interacción del usuario.

**5 (opcional) Establecer el atributo `timeout.date` en una fecha relativa**

Puede establecer el atributo `timeout.date` de un elemento **Interacción del usuario** en una fecha y hora relativas vinculándolo a un objeto `Date`. Se ha definido el objeto en una función con script.

**6 (opcional) Definir las entradas externas para una interacción del usuario**

Especifique la información que los usuarios deben facilitar durante la ejecución de un flujo de trabajo como parámetros de entrada de una interacción del usuario.

**7 (opcional) Definir el comportamiento de excepciones de la interacción del usuario**

Si un usuario no proporciona parámetros de entrada en el intervalo del tiempo de espera, la interacción del usuario devuelve una excepción. El comportamiento de las excepciones se puede definir en una función con scripts.

**8 (opcional) Crear el cuadro de diálogo de parámetros de entrada para la interacción del usuario**

Los usuarios proporcionan parámetros de entrada durante la ejecución de un flujo de trabajo en un cuadro de diálogo de parámetros de entrada, del mismo modo en el que proporcionan parámetros de entrada la primera vez que se inicia un flujo de trabajo.

**9 (opcional) Responder a una solicitud de interacción del usuario**

Los flujos de trabajo que requieren interacciones del usuario durante la ejecución se ponen en pausa hasta que el usuario proporciona la información requerida o hasta que se agota el tiempo de espera del flujo.

## Añadir una interacción del usuario a un flujo de trabajo

Solicite parámetros de entrada de usuarios durante una ejecución de flujo de trabajo añadiendo un elemento de esquema **Interacción del usuario** al flujo de trabajo. Cuando un flujo de trabajo detecta un elemento **Interacción del usuario**, suspende su ejecución y espera a que el usuario proporcione los datos que necesita.

### Requisitos previos

- Cree un flujo de trabajo.
- Abra el flujo de trabajo para editar en el Editor de flujos de trabajo.
- Añada algunos elementos al esquema del flujo de trabajo.



**Procedimiento**

- 1 Arrastre un elemento **Interacción del usuario** a la posición pertinente del esquema del flujo de trabajo.
- 2 Haga clic en el icono **Editar** (✎) del elemento **Interacción del usuario**.
- 3 Proporcione un nombre y una descripción para la interacción del usuario en la pestaña **Información**; a continuación, haga clic en **Cerrar**.
- 4 Haga clic en **Guardar**.

Ha añadido una interacción del usuario a un flujo de trabajo. Cuando el flujo de trabajo alcanza este elemento, espera la información del usuario antes de continuar el proceso de ejecución.

**Pasos siguientes**

Establezca el atributo `security.group` de la interacción del usuario para limitar el permiso a responder a la interacción de un usuario o de un grupo de usuarios. Consulte [Configurar el atributo `security.group` para la interacción del usuario](#).

## Configurar el atributo `security.group` para la interacción del usuario

El atributo `security.group` de un elemento de interacción del usuario establece los usuarios o los grupos de usuarios que tienen permiso para responder a la interacción del usuario.

**Requisitos previos**

- Cree un flujo de trabajo.
- Abra el flujo de trabajo para editar en el Editor de flujos de trabajo.
- Añada algunos elementos y una interacción de usuario al esquema de flujo de trabajo.
- Identifique un grupo de usuarios LDAP para responder a la solicitud de interacción del usuario.

**Procedimiento**

- 1 Haga clic en el icono **Editar** (✎) del elemento **Interacción del usuario** en el esquema del flujo de trabajo.
- 2 Haga clic en la pestaña **Atributos** para la interacción del usuario.
- 3 Haga clic en **No establecido** para que el parámetro de origen `security.group` establezca los usuarios que pueden responder a la interacción del usuario.
- 4 (opcional) Seleccione **NULL** para permitir que todos los usuarios respondan a la solicitud para la interacción del usuario.
- 5 A fin de limitar el permiso para responder a un usuario o un grupo de usuarios específicos, haga clic en **Crear parámetro/atributo en flujo de trabajo**.

Se abre el cuadro de diálogo **Información de parámetro**.

- 6 Asigne un nombre al parámetro.
- 7 Seleccione **Crear atributo de flujo de trabajo con el mismo nombre** para crear el atributo `LdapGroup` en el flujo de trabajo.
- 8 Haga clic en **No establecido** para que el valor de parámetro abra el cuadro de selección **LdapGroup**.
- 9 Escriba el nombre del grupo de usuarios LDAP en el cuadro de texto **Filtro**.
- 10 Seleccione el grupo de usuarios LDAP en la lista y haga clic en **Seleccionar**.

Por ejemplo, si selecciona el grupo **Administradores**, significa que solo los miembros de ese grupo pueden responder a esta solicitud para la interacción del usuario.

Se ha limitado el permiso para responder a la solicitud de interacción del usuario.

- 11 Haga clic en **Aceptar** para cerrar el cuadro de diálogo **Información de parámetro**.

Se ha establecido el atributo `security.group` para la interacción del usuario.

### Pasos siguientes

Configure el atributo `timer.date` para a fin de establecer el tiempo de espera para la interacción del usuario.

- Para establecer el tiempo de espera en una fecha y una hora absolutas, consulte [Establecimiento del atributo `timeout.date` en una fecha absoluta](#).
- Para crear una función para calcular un tiempo de espera relativo a la fecha y la hora actuales, consulte [Cálculo del tiempo de espera relativo para las interacciones del usuario](#).

## Establecimiento del atributo `timeout.date` en una fecha absoluta

Establezca el atributo `timeout.date` para que una interacción del usuario determine durante cuánto espera el flujo de trabajo a que un usuario responda a una interacción.

Establezca una fecha y una hora absolutas en el objeto `Date`. Cuando llegue la hora de la fecha establecida, finalizará el tiempo de espera del flujo de trabajo que espera una interacción del usuario en el estado `Failed`. Por ejemplo, puede establecer que el tiempo de espera de la interacción del usuario finalice a mediodía del 12 de febrero. Para calcular un tiempo de espera relativo a la fecha y la hora actuales, consulte [Cálculo del tiempo de espera relativo para las interacciones del usuario](#).

### Requisitos previos

- Abra un flujo de trabajo para editar en el Editor de flujos de trabajo.
- Añada un elemento de interacción del usuario al esquema del flujo de trabajo.
- Establezca el atributo `security.group` para la interacción del usuario.

### Procedimiento

- 1 Haga clic en el icono **Editar** (✎) del elemento **Interacción del usuario** en el esquema del flujo de trabajo.

- 2 Haga clic en la pestaña **Atributos** para la interacción del usuario.
- 3 Haga clic en **No establecido** para que el parámetro de origen `timeout.date` establezca el valor del parámetro de tiempo de espera.
- 4 (opcional) Seleccione **NULL** para permitir que la interacción del usuario establezca que el flujo de trabajo espere indefinidamente a que el usuario responda a la interacción.
- 5 Haga clic en **Crear parámetro o atributo en flujo de trabajo** para establecer que se produzca un error en el flujo de trabajo después de un periodo de tiempo de espera.  
Se abre el cuadro de diálogo **Información de parámetro**.
- 6 Asigne un nombre al parámetro.
- 7 Seleccione **Crear atributo de flujo de trabajo con el mismo nombre** para crear un atributo `Date` en el flujo de trabajo.
- 8 Haga clic en **No establecido** para el parámetro **Valor**.
- 9 Utilice el calendario para seleccionar una fecha y hora absolutas hasta la que el flujo de trabajo esperará a que responda el usuario.
- 10 Haga clic en **Aceptar** para cerrar el calendario.
- 11 Haga clic en **Aceptar** para cerrar el cuadro de diálogo **Información de parámetro**.

Ha establecido el atributo `timeout.date` en una fecha absoluta. El tiempo de espera del flujo de trabajo se agota si el usuario no responde a la interacción antes de esa fecha y esa hora.

#### Pasos siguientes

Defina los parámetros de entrada externa que la interacción del usuario requiere del usuario. Consulte [Definir las entradas externas para una interacción del usuario](#).

## Cálculo del tiempo de espera relativo para las interacciones del usuario



Puede calcular en un objeto `Date` una fecha y hora relativas para el tiempo de espera de una interacción del usuario.

Puede establecer una fecha y hora absolutas en un objeto `Date`. Cuando llegue la hora de la fecha establecida, finalizará el tiempo de espera de la solicitud de una interacción del usuario. También puede crear un elemento de flujo de trabajo que calcule y genere un objeto `Date` relativo de acuerdo con una función que se defina. Por ejemplo, puede crear un objeto `Date` relativo que añada 24 horas al tiempo actual.

#### Requisitos previos

- Abra un flujo de trabajo para editar en el Editor de flujos de trabajo.
- Añada un elemento de interacción del usuario al esquema del flujo de trabajo.
- Establezca el atributo `security.group` para la interacción del usuario.

## Procedimiento

- 1 Arrastre un elemento **Tarea de scripts** del menú **Genérico** al esquema de un flujo de trabajo, antes del elemento que requiere el objeto relativo Date para su atributo timeout.date.
- 2 Haga clic en el icono **Editar** () del elemento **Tarea de scripts** en el esquema del flujo de trabajo.
- 3 Proporcione un nombre y una descripción para el elemento de flujo de trabajo con script en la pestaña de propiedades **Información**.
- 4 Haga clic en la pestaña de propiedades **OUT**; a continuación, haga clic en el icono de **Enlazar a parámetro o atributo de flujo de trabajo** ()
- 5 Haga clic en **Crear parámetro o atributo en flujo de trabajo** para crear un atributo de flujo de trabajo.
  - a Asigne un nombre al atributo timerDate.
  - b Seleccione Date en la lista de tipos de atributo.
  - c Seleccione **Crear atributo de flujo de trabajo con el mismo nombre**.
  - d Deje el valor del atributo en **No establecido**, ya que una función con script proporcionará este valor.
  - e Haga clic en **Aceptar**.
- 6 Haga clic en la pestaña **Creación de scripts** para el elemento de flujo de trabajo con script.
- 7 Defina una función para calcular y generar un objeto Date denominado timerDate en la ventana de creación de scripts de la pestaña **Creación de scripts**.

Por ejemplo, puede crear un objeto Date implementando la función de JavaScript siguiente, en la que el periodo de tiempo de espera es un retraso relativo en milisegundos.

```
timerDate = new Date();
System.log( "Current date : '" + timerDate + "'" );
timerDate.setTime( timerDate.getTime() + (86400 * 1000) );
System.log( "Timer will expire at '" + timerDate + "'" );
```

La función de JavaScript de ejemplo anterior define un objeto Date que obtiene la fecha y la hora actuales con el método getTime, y añade 86.400.000 milisegundos o 24 horas. El elemento **Tarea de scripts** genera este valor como su parámetro de salida.

- 8 Haga clic en **Cerrar**.
- 9 Haga clic en **Guardar**.

Se ha creado una función que calcula una fecha y una hora relativas a la fecha y la hora actuales, y que genera un objeto Date. Un elemento **Interacción del usuario** puede recibir este objeto Date como parámetro de entrada para establecer el periodo de tiempo de espera para la entrada del usuario. Cuando el flujo de trabajo llega al elemento **Interacción del usuario**, suspende su ejecución y espera a que el usuario proporcione la información necesaria, o durante 24 horas para que finalice el tiempo de espera.

## Pasos siguientes

Debe enlazar el objeto `Date` con el parámetro `timeout.date` del elemento **Interacción del usuario**. Consulte [Establecer el atributo `timeout.date` en una fecha relativa](#).

## Establecer el atributo `timeout.date` en una fecha relativa

Puede establecer el atributo `timeout.date` de un elemento **Interacción del usuario** en una fecha y hora relativas vinculándolo a un objeto `Date`. Se ha definido el objeto en una función con script.

Si crea un objeto `Date` relativo en una función con script, puede vincular el atributo `timeout.date` de una interacción del usuario a este objeto `Date`. Por ejemplo, si vincula el atributo `timeout.date` a un objeto `Date` que añade 24 horas a la hora actual, el tiempo de espera de la interacción del usuario se agota transcurridas 24 horas.

## Requisitos previos

- Add a user interaction element to the workflow schema.
- Set the `security.group` attribute for the user interaction.
- Cree una función con script que calcule una fecha y una hora relativas, y que las atribuya a un objeto `Date` en el flujo de trabajo. Consulte [Cálculo del tiempo de espera relativo para las interacciones del usuario](#).

## Procedimiento

- 1 Haga clic en el icono **Editar** (✎) del elemento **Interacción del usuario** en el esquema del flujo de trabajo.
- 2 Haga clic en la pestaña **Atributos** para la interacción del usuario.
- 3 Haga clic en **No establecido** para que el parámetro de origen `timeout.date` establezca el valor del parámetro de tiempo de espera.
- 4 Seleccione el objeto `Date` que atribuya una fecha y hora relativas definidas en una función con script y haga clic en **Seleccionar**.

Establezca el atributo `timeout.date` en una fecha y hora relativas que calcule una función con script.

## Pasos siguientes

Defina los parámetros de entrada externa que la interacción del usuario requiere del usuario. Consulte [Definir las entradas externas para una interacción del usuario](#).

## Definir las entradas externas para una interacción del usuario



Especifique la información que los usuarios deben facilitar durante la ejecución de un flujo de trabajo como parámetros de entrada de una interacción del usuario.

Cuando un flujo de trabajo alcanza un elemento de interacción del usuario, espera hasta que un usuario proporciona la información que la interacción del usuario necesita como parámetros de entrada.

## Requisitos previos

- Añada un elemento de interacción del usuario al esquema del flujo de trabajo.
- Set the security.group attribute for the user interaction.
- Establezca el atributo timer.date para la interacción del usuario.

## Procedimiento

- 1 Haga clic en el icono **Editar** () del elemento **Interacción del usuario** en el esquema del flujo de trabajo.
- 2 Haga clic en la pestaña **Entradas externas**.
- 3 Haga clic en el icono de **Enlazar a parámetro o atributo de flujo de trabajo** () para definir los parámetros que el usuario debe proporcionar en la interacción del usuario.
- 4 (opcional) Si ya ha definido los parámetros de entrada en el flujo de trabajo, selecciónelos en la lista propuesta.
- 5 Haga clic en el atributo **Crear parámetro o atributo en flujo de trabajo** para crear un atributo de flujo de trabajo al que enlazar el parámetro de entrada que proporciona el usuario.
- 6 Asigne un nombre apropiado al parámetro.
- 7 Seleccione el tipo de parámetro de entrada en la lista de tipos; para ello, busque un tipo de objeto en el cuadro **Filtro**.  
  
Por ejemplo, si la interacción del usuario consiste en proporcionar una máquina virtual como parámetro de entrada, seleccione VC:VirtualMachine.
- 8 Seleccione **Crear atributo de flujo de trabajo con el mismo nombre** para enlazar el parámetro de entrada proporcionado por el usuario a un atributo nuevo en el flujo de trabajo.
- 9 Deje el valor del parámetro de entrada en **No establecido**.  
  
El usuario proporciona este valor cuando responden a la interacción del usuario durante la ejecución del flujo de trabajo.
- 10 Haga clic en **Aceptar** para cerrar el cuadro de diálogo **Información de parámetro**.

Ha definido los parámetros de entrada proporcionados por el usuario durante una interacción del usuario.

## Pasos siguientes

Defina el comportamiento de excepciones si la interacción del usuario detecta un error. Consulte [Definir el comportamiento de excepciones de la interacción del usuario](#).

## Definir el comportamiento de excepciones de la interacción del usuario

Si un usuario no proporciona parámetros de entrada en el intervalo del tiempo de espera, la interacción del usuario devuelve una excepción. El comportamiento de las excepciones se puede definir en una función con scripts.

Si no define la acción que debe realizar el flujo de trabajo si se agota el tiempo de espera de interacción del usuario, el flujo de trabajo finaliza con el estado `Failed`. Definir el comportamiento de las excepciones es un procedimiento recomendable para desarrollar flujos de trabajo.

### Requisitos previos

- Añada un elemento de interacción del usuario al esquema del flujo de trabajo.
- Establezca los atributos `security.group` y `timer.date` para la interacción del usuario.
- Defina los parámetros de entrada externos para la interacción del usuario.

### Procedimiento

- 1 Haga clic en el icono **Editar** (✎) del elemento **Interacción del usuario** en el esquema del flujo de trabajo.
- 2 Haga clic en la pestaña **Excepción**.
- 3 Haga clic en **No establecido** para el enlace de excepciones de salida.
- 4 Haga clic en **Crear parámetro o atributo en flujo de trabajo** para crear un atributo de excepción al que enlazar la interacción del usuario.

Se abre el cuadro de diálogo **Información de parámetro**.

- 5 Cree un atributo `errorCode`.

Un atributo `errorCode` tiene las propiedades predeterminadas siguientes:

- Nombre: **errorCode**
- Tipo: cadena
- Crear: **Crear atributo de flujo de trabajo con el mismo nombre**
- Valor: escriba el correspondiente mensaje de error.

- 6 Haga clic en **Aceptar** para cerrar el cuadro de diálogo **Información de parámetro**.
- 7 Arrastre un elemento de tarea de scripts sobre el elemento de interacción del usuario en el esquema de flujo de trabajo.

Una flecha con líneas discontinuas rojas, que representa el vínculo de excepción, aparece entre los dos elementos. El elemento de tarea de scripts se enlaza automáticamente al atributo `errorCode` desde la interacción del usuario.

- 8 Haga doble clic en el elemento de tarea de scripts y asígnele un nombre.

Por ejemplo, **Registrar tiempo de espera agotado**.

- 9 En la pestaña **Crear scripts** del elemento de tarea de scripts, escriba una función de JavaScript para controlar la excepción.

Por ejemplo, para registrar el tiempo de espera agotado en el registro de Orchestrator, escriba esta función:

```
System.log("No response from user. Timed out.");
```

- 10 Vincule y enlace el elemento de tarea de scripts que controla las excepciones al elemento que lo sigue en el flujo de trabajo.

Por ejemplo, vincule y enlace el elemento de tarea de scripts a un elemento **Lanzar excepción** para finalizar el flujo de trabajo con un error.

Ha definido el comportamiento de excepciones si se agota el tiempo de espera de interacción del usuario.

### Pasos siguientes

Cree el cuadro de diálogo en el que los usuarios proporcionan parámetros de entrada. Consulte [Crear el cuadro de diálogo de parámetros de entrada para la interacción del usuario](#).

## Crear el cuadro de diálogo de parámetros de entrada para la interacción del usuario

Los usuarios proporcionan parámetros de entrada durante la ejecución de un flujo de trabajo en un cuadro de diálogo de parámetros de entrada, del mismo modo en el que proporcionan parámetros de entrada la primera vez que se inicia un flujo de trabajo.

El diseño del cuadro de diálogo se crea en la pestaña **Presentación** del elemento de interacción del usuario, no en la pestaña **Presentación** de todo el flujo de trabajo. La pestaña **Presentación** de todo el flujo de trabajo crea el diseño del cuadro de diálogo de parámetros de entrada que se muestra en pantalla al iniciar un flujo de trabajo. La pestaña **Presentación** del elemento de interacción del usuario crea el diseño del cuadro de diálogo de parámetros de entrada que se abre cuando un flujo de trabajo llega a un elemento de interacción del usuario durante su ejecución.

### Requisitos previos

- Añada un elemento de interacción del usuario al esquema del flujo de trabajo.
- Establezca los atributos `security.group` y `timer.date` para la interacción del usuario.
- Defina los parámetros de entrada externos para la interacción del usuario.
- Defina el comportamiento de excepciones.

### Procedimiento

- 1 Haga clic en el icono **Editar** (✎) del elemento **Interacción del usuario** en el esquema del flujo de trabajo.



- 2 Haga clic en la pestaña **Presentación** del elemento de interacción del usuario.

La pestaña **Presentación** muestra los parámetros de entrada externos que ha creado para la interacción del usuario.

- 3 (opcional) Haga clic con el botón derecho en el nodo **Presentación** en la pestaña **Presentación** y seleccione **Crear paso**.

Los pasos le permiten crear secciones en el cuadro de diálogo, con descripciones y encabezados con los que organizar los parámetros de entrada.

- 4 (opcional) Haga clic con el botón derecho en el nodo **Presentación** en la pestaña **Presentación** y seleccione **Crear grupo de visualización**.

Los grupos de visualización le permiten establecer el orden de aparición de los parámetros de entrada en los pasos; asimismo, le permiten añadir encabezados secundarios e instrucciones al cuadro de diálogo.

- 5 Haga clic en un parámetro de entrada en la lista; en la pestaña **General** de dicho parámetro, añada una descripción del parámetro de entrada.

El texto de la descripción que se escribe aparece como etiqueta en el cuadro de diálogo de parámetros de entrada. Su función es indicar al usuario la información que se debe proporcionar cuando responden a la interacción del usuario.

- 6 Defina las propiedades de los parámetros de entrada.

Las propiedades de los parámetros de entrada le permiten cualificar los valores de los parámetros de entrada que los usuarios pueden proporcionar. También sirven para determinar los valores de parámetros de manera dinámica mediante expresiones de OGNL.

- 7 Haga clic en **Guardar y cerrar** para cerrar el Editor de flujos de trabajo.

Ha creado el cuadro de diálogo de parámetros de entrada en el que los usuarios proporcionan parámetros de entrada para responder a una interacción del usuario durante la ejecución de un flujo de trabajo.

### Pasos siguientes

Para obtener información sobre cómo crear los pasos de presentación y los grupos, y sobre cómo establecer las propiedades de los parámetros de entrada, consulte [Crear el cuadro de diálogo Parámetros de entrada en la pestaña Presentación](#).

## Responder a una solicitud de interacción del usuario

Los flujos de trabajo que requieren interacciones del usuario durante la ejecución se ponen en pausa hasta que el usuario proporciona la información requerida o hasta que se agota el tiempo de espera del flujo.

Los flujos de trabajo que requieren interacciones del usuario definen qué usuarios pueden proporcionar la información requerida y dirigen la solicitud de interacción.

## Requisitos previos

Verifique que hay al menos un flujo de trabajo con el estado A la espera de interacción del usuario.

## Procedimiento


1 En el menú desplegable del cliente de Orchestrator, seleccione **Ejecutar**.

2 Haga clic en la vista **My Orchestrator** del cliente de Orchestrator.

3 Haga clic en la pestaña **A la espera de información**.

En la pestaña **A la espera de información** hay una lista de los flujos de trabajo para los que usted u otros miembros del grupo de usuarios con el permiso requerido deben introducir información.

4 Haga doble clic en un flujo de trabajo que esté a la espera de información.

El token de flujo de trabajo a la espera de información aparece en la lista jerárquica **Flujos de trabajo** con este símbolo: .

5 Haga clic con el botón derecho en el flujo de trabajo y seleccione **Responder**.

6 Siga las instrucciones del cuadro de diálogo de parámetro de entrada para proporcionar la información requerida por el flujo de trabajo.

Ha proporcionado información para un flujo de trabajo que estaba a la espera de información del usuario durante la ejecución.

## Llamar a flujos de trabajo desde flujos de trabajo

Los flujos de trabajo pueden llamar a otros flujos de trabajo durante su ejecución. Un flujo de trabajo puede iniciar otro flujo de trabajo porque requiere el resultado del otro flujo de trabajo como parámetro de entrada para su propia ejecución, o bien puede iniciar un flujo de trabajo y dejar que prosiga su proceso de ejecución de forma independiente. Los flujos de trabajo también pueden iniciar un flujo de trabajo en un momento dado del futuro o bien iniciar varios flujos de trabajo de manera simultánea.

### ■ Elementos de los flujos de trabajo que llaman a flujos de trabajo

Hay cuatro formas de llamar a otros flujos de trabajo desde un flujo de trabajo. Cada forma de llamar a un flujo de trabajo o flujos de trabajo se representa por un elemento distinto del esquema del flujo de trabajo.

### ■ Llamar a un flujo de trabajo de forma sincrónica

Si se llama a un flujo de trabajo de forma sincrónica, el flujo de trabajo llamado se ejecuta como parte del flujo de trabajo que llama. El flujo de trabajo que llama puede utilizar los parámetros de salida del flujo de trabajo llamado cuando ejecuta sus elementos de esquema subsiguientes.

### ■ Llamar a un flujo de trabajo de forma asincrónica

Si se llama a un flujo de trabajo de forma asincrónica, el flujo de trabajo llamado se ejecuta de forma independiente al flujo de trabajo que llama. El flujo de trabajo que llama se sigue ejecutando sin esperar a que concluya el flujo de trabajo llamado.

- **Programar un flujo de trabajo**

Puede llamar a un flujo de trabajo desde un flujo de trabajo y programarlo para que se inicie en una fecha y una hora posteriores.

- **Requisitos previos para llamar a un flujo de trabajo remoto desde otro flujo de trabajo**

Si el flujo de trabajo desarrollado por usted llama a otro flujo que reside en un servidor de Orchestrator remoto, deben cumplirse ciertos requisitos previos para que el flujo de trabajo remoto se ejecute correctamente.

- **Llamar a varios flujos de trabajo simultáneamente**

Si se llama a varios flujos de trabajo de forma simultánea, dichos flujos de trabajo llamados se ejecutan como parte del flujo de trabajo que llama. El flujo de trabajo que llama espera a que concluyan todos los flujos de trabajo llamados antes de continuar. El flujo de trabajo que llama puede utilizar los resultados de los flujos de trabajo llamados como parámetros de entrada cuando ejecuta sus elementos de esquema subsiguientes.

## Elementos de los flujos de trabajo que llaman a flujos de trabajo

Hay cuatro formas de llamar a otros flujos de trabajo desde un flujo de trabajo. Cada forma de llamar a un flujo de trabajo o flujos de trabajo se representa por un elemento distinto del esquema del flujo de trabajo.

### Flujos de trabajo sincrónicos

Un flujo de trabajo puede iniciar otro flujo de trabajo de forma sincrónica. El flujo de trabajo al que se ha llamado se ejecuta como parte integral de la ejecución del flujo de trabajo que llama, y se ejecuta en el mismo espacio de memoria que el flujo de trabajo que llama. El flujo de trabajo que llama inicia otro flujo de trabajo y, a continuación, espera hasta el final de la ejecución del flujo de trabajo llamado antes de empezar a ejecutar el siguiente elemento de su esquema. Normalmente, puede llamar a un flujo de trabajo de forma sincrónica porque el flujo de trabajo que llama necesita la salida del flujo de trabajo llamado como parámetro de entrada para un elemento de esquema posterior. Por ejemplo, un flujo de trabajo puede llamar al flujo de trabajo Iniciar máquina virtual y esperar para iniciar una máquina virtual y, a continuación, obtener la dirección IP de esta máquina virtual para pasársela a otro elemento o a otro usuario por correo electrónico.

### Flujos de trabajo asincrónicos

Un flujo de trabajo puede iniciar otro flujo de trabajo de forma asincrónica. El flujo de trabajo que llama inicia otro flujo de trabajo pero continúa ejecutando inmediatamente el siguiente elemento de su esquema sin esperar al resultado del flujo de trabajo llamado. El flujo de trabajo llamado se ejecuta con parámetros de entrada que define el flujo de trabajo que llama, pero el ciclo de vida del flujo de trabajo llamado es independiente del ciclo de vida del flujo de trabajo que llama. Los flujos de trabajo asincrónicos le permiten crear cadenas de flujos de trabajo que pasan los parámetros de entrada de un flujo de trabajo al siguiente. Por ejemplo, un

flujo de trabajo puede crear varios objetos durante su ejecución. A continuación, el flujo de trabajo puede iniciar flujos de trabajo asincrónicos que utilizan estos objetos como parámetros de entrada en sus propias ejecuciones. Cuando el flujo de trabajo original ha iniciado todos los flujos de trabajo necesarios y ejecutado sus elementos restantes, finaliza. Sin embargo, los flujos de trabajo asincrónicos que ha iniciado continúan con sus ejecuciones de manera independiente del flujo de trabajo que los inició.

Para conseguir que el flujo de trabajo que llama espere al resultado del flujo de trabajo llamado, puede utilizar un flujo de trabajo anidado o crear una tarea de scripts que recupere el estado del token del flujo de trabajo llamado y, a continuación, recupere el resultado del flujo de trabajo cuando se complete.

### Flujos de trabajo programados

Un flujo de trabajo puede llamar a otro flujo de trabajo pero retrasar el inicio de ese flujo de trabajo hasta una fecha y hora posteriores. A continuación, el flujo de trabajo que llama continúa su ejecución hasta que finaliza. Al llamar a un flujo de trabajo programado, se crea una tarea para iniciar ese flujo de trabajo en la fecha y la hora definidas. Cuando el flujo de trabajo que llama se ha ejecutado, puede ver el flujo de trabajo programado en las vistas **Programador** y **Mi Orchestrator** del cliente de Orchestrator.

Los flujos de trabajo programados solo se ejecutan una vez. Puede programar un flujo de trabajo para que se ejecute de manera periódica llamando al método `Workflow.scheduleRecurrently` en un elemento de tarea de scripts en un flujo de trabajo sincrónico.

### Flujos de trabajo anidados

Un flujo de trabajo puede iniciar varios flujos de trabajo de manera simultánea anidando varios flujos de trabajo en un solo elemento de esquema. Todos los flujos de trabajo que figuran en el elemento de flujo de trabajo anidado se inician de manera simultánea cuando el flujo de trabajo que llama llega al elemento de flujo de trabajo anidado en su esquema. Cada flujo de trabajo anidado se inicia en un espacio de memoria distinto del espacio de memoria del flujo de trabajo que llama. El flujo de trabajo que llama espera hasta que todos los flujos de trabajo anidados han completado sus ejecuciones para iniciar la ejecución del siguiente elemento de su esquema. El flujo de trabajo que llama puede así utilizar los resultados de los flujos de trabajo anidados como parámetros de entrada cuando ejecuta sus elementos restantes.

## Propagar cambios en el flujo de trabajo a otros flujos de trabajo

Si llama a un flujo de trabajo desde otro flujo de trabajo, Orchestrator importa los parámetros de entrada del flujo de trabajo secundario al flujo de trabajo principal en el momento en que el elemento de flujo de trabajo se añade al esquema.

Si modifica el flujo de trabajo secundario después de haberlo añadido a otro flujo de trabajo, el flujo de trabajo principal llama a la nueva versión del flujo de trabajo secundario, pero no importa ningún parámetro de entrada nuevo. Para evitar que los cambios en los flujos de trabajo afecten al comportamiento de otros flujos de trabajo que los llaman, Orchestrator no propaga los nuevos parámetros de entrada automáticamente a los flujos de trabajo que efectúan la llamada.

Para propagar los parámetros de un flujo de trabajo a otros que lo llaman, debe buscar los flujos de trabajo que llaman al flujo de trabajo y sincronizarlos manualmente.

### Requisitos previos

Compruebe que tenga un flujo de trabajo al que llaman otros flujos de trabajo.

### Procedimiento

- 1 Modifique y guarde un flujo de trabajo al que llamen otros flujos de trabajo.
- 2 Cierre el Editor de flujos de trabajo.
- 3 Vaya al flujo de trabajo que ha cambiado en la lista jerárquica de la vista **Flujos de trabajo** en el cliente de Orchestrator.
- 4 Haga clic con el botón secundario en el flujo de trabajo y seleccione **Referencias > Buscar elementos que utilizan este elemento**.  
Aparece una lista de los flujos de trabajo que llaman a este flujo de trabajo.
- 5 Haga doble clic en un flujo de trabajo en la lista para resaltarlo en la vista **Flujos de trabajo** en el cliente de Orchestrator.
- 6 Haga clic con el botón secundario en el flujo de trabajo y seleccione **Editar**.  
Se abre el Editor de flujos de trabajo.
- 7 Haga clic en la pestaña **Esquema** en el Editor de flujos de trabajo.
- 8 Haga clic con el botón secundario en el elemento de flujo de trabajo para el flujo de trabajo cambiado desde el esquema de flujo de trabajo; a continuación, seleccione **Sincronizar > Sincronizar parámetros**.
- 9 Seleccione **Continuar** en el cuadro de diálogo de confirmación.
- 10 Guarde y cierre el Editor de flujos de trabajo.
- 11 Repita del [Paso 5](#) al [Paso 10](#) para todos los flujos de trabajo que utilicen el flujo de trabajo modificado.

Ha propagado un flujo de trabajo cambiado a otros flujos de trabajo que lo llaman.

### Propagar los parámetros de entrada y la presentación de un flujo de trabajo secundario al flujo de trabajo principal

Si desarrolla un flujo de trabajo que llama a otros flujos, puede propagar los parámetros de entrada y la presentación de los flujos de trabajo secundarios al flujo principal.

**Procedimiento**

- 1 En el menú desplegable del cliente de Orchestrator, seleccione **Ejecutar**.
- 2 Haga clic con el botón derecho en el flujo de trabajo que desea modificar y seleccione **Editar**.  
El Editor de flujos de trabajo se abre.
- 3 Seleccione la pestaña **Esquema**.
- 4 Haga clic con el botón derecho en el elemento del flujo de trabajo secundario cuyos parámetros de entrada y presentación desea propagar al flujo de trabajo principal y seleccione **Sincronizar > Sincronizar presentación**.
- 5 En el cuadro de diálogo de confirmación, seleccione **Aceptar**.
- 6 (opcional) Repita el [Paso 4](#) y el [Paso 5](#) para todos los flujos de trabajo secundarios cuyos parámetros de entrada y presentación desea propagar al flujo de trabajo principal.

Los parámetros de entrada del flujo de trabajo secundario se añaden a los parámetros de entrada del flujo principal. La presentación del flujo de trabajo principal se amplía con las presentaciones de los flujos de trabajo secundarios.

**Llamar a un flujo de trabajo de forma sincrónica**


Si se llama a un flujo de trabajo de forma sincrónica, el flujo de trabajo llamado se ejecuta como parte del flujo de trabajo que llama. El flujo de trabajo que llama puede utilizar los parámetros de salida del flujo de trabajo llamado cuando ejecuta sus elementos de esquema subsiguientes.

Puede llamar a flujos de trabajo de forma sincrónica desde otro flujo de trabajo mediante el elemento **Flujo de trabajo**.

**Requisitos previos**

- Abra un flujo de trabajo para editar en el Editor de flujos de trabajo.
- Añada algunos elementos al esquema del flujo de trabajo.

**Procedimiento**

- 1 Arrastre un elemento **Flujo de trabajo** del menú **Genérico** a la posición pertinente del esquema del flujo de trabajo.  
Aparece el cuadro de diálogo de selección **Elegir flujo de trabajo**.
- 2 Busque y seleccione un flujo de trabajo; a continuación, haga clic en **Aceptar**.  
Si la búsqueda devuelve un resultado parcial, restrinja el criterio de búsqueda o incremente el número de resultados de búsqueda en el menú **Herramientas > Preferencias del usuario** en el cliente.
- 3 Haga clic en el elemento **Flujo de trabajo** para que muestre sus pestañas de propiedades en la mitad inferior de la pestaña **Esquema**.
- 4 Haga clic en el icono **Editar** () del elemento **Flujo de trabajo** en el esquema del flujo de trabajo.

- 5 Enlace los parámetros de entrada correspondientes al flujo de trabajo en la pestaña **IN** del elemento de esquema del flujo de trabajo.
- 6 Enlace los parámetros de salida correspondientes al flujo de trabajo en la pestaña **OUT** del elemento de esquema del flujo de trabajo.
- 7 Defina el comportamiento de excepciones del flujo de trabajo en la pestaña **Excepciones**.
- 8 Haga clic en **Cerrar**.
- 9 Haga clic en **Guardar** en la parte inferior del Editor de flujos de trabajo.

Ha llamado a un flujo de trabajo de forma sincrónica desde otro flujo de trabajo. Cuando el flujo de trabajo alcanza el flujo de trabajo sincrónico durante su ejecución, empieza el flujo de trabajo sincrónico y el flujo de trabajo inicial espera a que concluya antes de seguir ejecutándose.

### Pasos siguientes

Puede llamar a un flujo de trabajo de forma sincrónica desde un flujo de trabajo.

## Llamar a un flujo de trabajo de forma asincrónica

Si se llama a un flujo de trabajo de forma asincrónica, el flujo de trabajo llamado se ejecuta de forma independiente al flujo de trabajo que llama. El flujo de trabajo que llama se sigue ejecutando sin esperar a que concluya el flujo de trabajo llamado.

Puede llamar a flujos de trabajo de forma asincrónica desde otro flujo de trabajo mediante el elemento **Flujo de trabajo asincrónico**.

### Requisitos previos

- Abra un flujo de trabajo para editar en el Editor de flujos de trabajo.
- Añada algunos elementos al esquema del flujo de trabajo.

### Procedimiento

- 1 Arrastre un elemento **Flujo de trabajo asincrónico** del menú **Genérico** a la posición pertinente del esquema del flujo de trabajo.  
Aparece el cuadro de diálogo de selección **Elegir flujo de trabajo**.
- 2 Busque y seleccione un flujo de trabajo en la lista; a continuación, haga clic en **Aceptar**.
- 3 Haga clic en el icono **Editar** (✎) del elemento **Flujo de trabajo asincrónico** en el esquema del flujo de trabajo.
- 4 Enlace los parámetros de entrada correspondientes al flujo de trabajo en la pestaña **IN** del elemento de flujo de trabajo asincrónico.

- 5 Enlace el parámetro de salida correspondiente en la pestaña **OUT** del elemento de flujo de trabajo asincrónico.

Puede enlazar el parámetro de salida al flujo de trabajo llamado o bien al resultado de ese flujo de trabajo.

- Enlace el flujo de trabajo llamado para devolver dicho flujo de trabajo como parámetro de salida.
- Enlace al token de flujo de trabajo del flujo de trabajo llamado para devolver el resultado de ejecutar el flujo de trabajo llamado.

- 6 Defina el comportamiento de excepciones del elemento de flujo de trabajo asincrónico en la pestaña **Excepciones**.

- 7 Haga clic en **Cerrar**.

- 8 Haga clic en **Guardar** en la parte inferior del Editor de flujos de trabajo.

Ha llamado a un flujo de trabajo de forma asincrónica desde otro flujo de trabajo. Cuando el flujo de trabajo alcanza el flujo de trabajo asincrónico durante su ejecución, empieza el flujo de trabajo asincrónico y el flujo de trabajo inicial continúa ejecutándose sin esperar a que concluya el flujo de trabajo asincrónico.

### Pasos siguientes

Puede programar un flujo de trabajo para que se inicie en una fecha y hora posteriores.

## Programar un flujo de trabajo

Puede llamar a un flujo de trabajo desde un flujo de trabajo y programarlo para que se inicie en una fecha y una hora posteriores.

Para programar flujos de trabajo en otro flujo de trabajo, utilice el elemento **Programar flujo de trabajo**.

### Requisitos previos

- Abra un flujo de trabajo para editar en el Editor de flujos de trabajo.
- Añada algunos elementos al esquema del flujo de trabajo.

### Procedimiento

- 1 Arrastre un elemento **Programar flujo de trabajo** del menú **Genérico** a la posición adecuada en el esquema de flujo de trabajo.
- 2 Busque el flujo de trabajo al que llamar escribiendo parte de su nombre en el cuadro de texto.
- 3 Seleccione el flujo de trabajo en la lista y haga clic en **Aceptar**.
- 4 Haga clic en el icono **Editar** (✎) del elemento **Programar flujo de trabajo** en el esquema del flujo de trabajo.
- 5 Haga clic en la pestaña de propiedades **IN**.

Aparece un parámetro llamado `workFlowScheduleDate` en la lista de propiedades para definir, junto con los parámetros de entrada del flujo de trabajo que llama.



- 6 Haga clic en **No establecido** para que el parámetro workflowScheduleDate establezca el parámetro.
- 7 Haga clic en **Crear parámetro o atributo en flujo de trabajo** para crear el parámetro y establecer su valor.
- 8 Haga clic en **No establecido** para **Valor** con el fin de establecer el valor del parámetro.
- 9 Utilice el calendario que aparece para establecer la fecha y la hora para iniciar el flujo de trabajo programado; a continuación, haga clic en **Aceptar**.
- 10 Enlace los demás parámetros de entrada al flujo de trabajo programado en la pestaña **IN** del elemento de flujo de trabajo programado.
- 11 Enlace los parámetros de salida necesarios al objeto Task en la pestaña **OUT** del elemento de flujo de trabajo programado.
- 12 Defina el comportamiento de excepciones del elemento de flujo de trabajo programado en la pestaña **Excepciones**.
- 13 Haga clic en **Cerrar**.
- 14 Haga clic en **Guardar** en la parte inferior del Editor de flujos de trabajo.

Ha programado un flujo de trabajo para que se inicie en una fecha y una hora concretas desde otro flujo de trabajo.

### Pasos siguientes

Puede llamar a varios flujos de trabajo a la vez desde un flujo de trabajo.

## Requisitos previos para llamar a un flujo de trabajo remoto desde otro flujo de trabajo

Si el flujo de trabajo desarrollado por usted llama a otro flujo que reside en un servidor de Orchestrator remoto, deben cumplirse ciertos requisitos previos para que el flujo de trabajo remoto se ejecute correctamente.

- Todos los parámetros de entrada del flujo de trabajo remoto se deben poder resolver en el servidor de Orchestrator remoto.
- Todos los parámetros de salida del flujo de trabajo remoto se deben poder resolver en el servidor de Orchestrator local.

Para asegurar que los parámetros del flujo de trabajo remoto pueden resolverse, los objetos de inventario usados por el flujo de trabajo deben estar disponibles tanto en el servidor remoto como en el servidor local de Orchestrator. Si el flujo de trabajo remoto usa objetos de un complemento, dicho complemento debe estar disponible en ambos servidores de Orchestrator. Los inventarios de los complementos remoto y local deben ser idénticos. Si el flujo de trabajo remoto usa objetos de sistema en Orchestrator (como flujos de trabajo y acciones), esos objetos deben existir en los inventarios de los servidores remoto y local de Orchestrator.

Por ejemplo, supongamos que inserta el flujo de trabajo Cambiar nombre de máquina virtual en un elemento de flujo de trabajo anidado en el flujo de trabajo Prueba desarrollado por usted y quiere ejecutar el flujo Cambiar nombre de máquina virtual en un servidor de Orchestrator remoto. Al ejecutarse el flujo de trabajo Prueba, se llama al flujo Cambiar nombre de máquina virtual desde la ejecución del flujo Prueba. Usted especifica una máquina virtual cuyo nombre se debe cambiar desde el inventario del servidor de Orchestrator local. Dado que el flujo de trabajo Cambiar nombre de máquina virtual se ejecuta en el servidor de Orchestrator remoto, la misma máquina virtual debe estar disponible en el inventario de ese servidor. De lo contrario, el flujo Cambiar nombre de máquina virtual no podrá resolver su parámetro de entrada vm. Por lo tanto, el complemento de vCenter Server presente en los servidores de Orchestrator local y remoto se debe conectar a la misma instancia de vCenter Server.

## Llamar a varios flujos de trabajo simultáneamente

Si se llama a varios flujos de trabajo de forma simultánea, dichos flujos de trabajo llamados se ejecutan como parte del flujo de trabajo que llama. El flujo de trabajo que llama espera a que concluyan todos los flujos de trabajo llamados antes de continuar. El flujo de trabajo que llama puede utilizar los resultados de los flujos de trabajo llamados como parámetros de entrada cuando ejecuta sus elementos de esquema subsiguientes.

Puede llamar a varios flujos de trabajo de forma simultánea desde otro flujo de trabajo mediante el elemento **Flujos de trabajo anidados**. Estos flujos de trabajo anidados se pueden utilizar para ejecutar flujos de trabajo con credenciales de usuario que son diferentes de las credenciales de usuario del flujo de trabajo que llama.

### Requisitos previos

- Abra un flujo de trabajo para editar en el Editor de flujos de trabajo.
- Añada algunos elementos al esquema del flujo de trabajo.

### Procedimiento

- 1 Arrastre un elemento **Flujo de trabajo anudado** del menú **Acción y flujo de trabajo** a la posición correspondiente del esquema del flujo de trabajo.

Aparece el cuadro de diálogo de selección **Elegir flujo de trabajo**.

- 2 Busque y seleccione un flujo de trabajo para comenzar; a continuación, haga clic en **Aceptar**.
- 3 Haga clic en el icono **Editar** (✎) del elemento **Flujos de trabajo anidados** en el esquema del flujo de trabajo.
- 4 Haga clic en la pestaña **Flujos de trabajo**.

El flujo de trabajo seleccionado en el paso [Paso 2](#) aparece en la pestaña.

- 5 Establezca los enlaces de entrada y de salida de este flujo de trabajo en las pestañas respectivas **IN** y **OUT** del panel derecho en la pestaña Propiedades del elemento de esquema **Flujos de trabajo**.

- 6 Haga clic en la pestaña **Información de conexión** del panel derecho en la pestaña Propiedades del elemento de esquema **Flujos de trabajo**.

La pestaña **Información de conexión** permite acceder a flujos de trabajo almacenados en un servidor diferente del local mediante las credenciales apropiadas.

- 7 Para acceder a flujos de trabajo en un servidor remoto, seleccione **Remoto**; a continuación, haga clic en **No establecido** para proporcionar un nombre de host o una dirección IP al servidor remoto.

---

**Nota** Utilice el complemento Multinodo de vRealize Orchestrator para llamar a flujos de trabajo en un servidor remoto.

---

- 8 Defina las credenciales con las que acceder al servidor remoto.
  - Seleccione **Heredar** para utilizar las mismas credenciales que el usuario que ejecuta el flujo de trabajo que llama.
  - Seleccione **Dinámicas** y haga clic en **No establecido** para seleccionar un conjunto de credenciales dinámicas que el tipo `credentials` define en cualquier otro lugar del flujo de trabajo.
  - Seleccione **Estáticas** y haga clic en **No establecido** para proporcionar las credenciales directamente.
- 9 Haga clic en el botón **Añadir flujo de trabajo** en la pestaña **Flujos de trabajo** para seleccionar más flujos de trabajo que añadir al elemento de flujo de trabajo anidado.
- 10 Repita los pasos [Paso 2](#) al [Paso 8](#) para definir la configuración de cada flujo de trabajo que se añada.
- 11 Haga clic en el elemento de flujos de trabajo anidados en el esquema del flujo de trabajo.

La cantidad de flujos de trabajo anidados en el elemento aparece como numeral en el elemento de flujos de trabajo anidados.

Ha llamado a varios flujos de trabajo de forma simultánea desde un flujo de trabajo.

### Pasos siguientes

Puede definir flujos de trabajo de larga ejecución.

## Ejecutar un flujo de trabajo en una selección de objetos

Puede automatizar las tareas repetitivas ejecutando un flujo de trabajo en una selección de objetos. Por ejemplo, puede crear un flujo de trabajo que tome una instantánea de todas las máquinas virtuales en una carpeta de máquinas virtuales, o bien puede crear un flujo de trabajo que desactive todas las máquinas virtuales de un host concreto.

Puede utilizar uno de los métodos siguientes para ejecutar un flujo de trabajo en una selección de objetos.

- Ejecute el flujo de trabajo **Biblioteca > vCenter > Lote > Ejecutar un flujo de trabajo en una selección de objetos**.

- Cree un flujo de trabajo que llame a los flujos de trabajo **Biblioteca > Orchestrator > Iniciar flujos de trabajo en serie** o **Iniciar flujos de trabajo en paralelo**.
- Cree un flujo de trabajo que obtenga una matriz de objetos y ejecute un flujo de trabajo en cada objeto de la matriz en un bucle de elementos de flujo de trabajo.
- Ejecute un flujo de trabajo desde JavaScript llamando al método `Workflow.execute()` en un bucle For en un elemento con script de un flujo de trabajo.

El método que elija para ejecutar un flujo de trabajo en una selección de objetos depende del flujo de trabajo que se vaya a ejecutar; asimismo, puede afectar al rendimiento de dicho flujo de trabajo. Por ejemplo, el flujo de trabajo Ejecutar un flujo de trabajo en una selección de objetos es la forma más sencilla de ejecutar un flujo de trabajo en varios objetos y no requiere ningún desarrollo de flujo de trabajo; sin embargo, solo puede ejecutar flujos de trabajo que tomen un solo parámetro de entrada.

Si crea un flujo de trabajo que llame a los flujos de trabajo Iniciar flujos de trabajo en serie o Iniciar flujos de trabajo en paralelo, puede ejecutarlo en flujos de trabajo de varios objetos para que tomen más de un parámetro de entrada. El flujo de trabajo de llamada debe crear una matriz de propiedades para transferir los parámetros de entrada a los flujos de trabajo Iniciar flujos de trabajo en serie o Iniciar flujos de trabajo en paralelo. Estos flujos de trabajo solo se utilizan en otros flujos de trabajo. No los ejecute directamente.

La ejecución de un flujo de trabajo en un bucle For en un elemento con scripts es más rápida que la ejecución de un flujo de trabajo en un bucle de elementos de flujo de trabajo; sin embargo, es menos flexible y limita el potencial de reutilización. Y, lo que es más importante, al ejecutarse un flujo de trabajo en un bucle con script, se pierde la creación de puntos de comprobación que lleva a cabo Orchestrator cuando inicia cada elemento en la ejecución de un flujo de trabajo. Como consecuencia, si el servidor de Orchestrator se detiene mientras se ejecuta el bucle con scripts, cuando se reinicie el servidor, el flujo de trabajo se reanudará al inicio del elemento con script y se repetirá el bucle completo. Si el servidor de Orchestrator se detiene durante la ejecución de un flujo de trabajo con un bucle de elementos de flujo de trabajo, el flujo de trabajo se reanudará en el elemento concreto del bucle que se ejecutaba cuando se detuvo el servidor.

Para obtener más información sobre los flujos de trabajo por lotes, consulte *Uso de complementos de VMware vRealize Orchestrator*.

En la sección [Desarrollar un flujo de trabajo complejo](#), se indica cómo crear un flujo de trabajo que ejecute un flujo de trabajo en una matriz de objetos en un bucle de elementos de flujo de trabajo.

Para obtener información sobre cómo ejecutar un flujo de trabajo en un bucle For con script, consulte [Ejemplos de creación de scripts de flujos de trabajo](#).

## Implementar los flujos de trabajo Iniciar flujos de trabajo en serie e Iniciar flujos de trabajo en paralelo

Puede utilizar los flujos de trabajo Iniciar flujos de trabajo en serie e Iniciar flujos de trabajo en paralelo para ejecutar un flujo de trabajo en una selección de objetos.

No puede ejecutar directamente los flujos de trabajo Iniciar flujos de trabajo en serie e Iniciar flujos de trabajo en paralelo. Debe incluirlos en otro flujo de trabajo que haya creado. Para utilizar los flujos de trabajo Iniciar flujos de trabajo en serie e Iniciar flujos de trabajo en paralelo para ejecutar un flujo de trabajo en una selección de objetos, debe obtener los objetos en los que ejecutar el flujo de trabajo. Transferirá al flujo de trabajo estos objetos y cualquier otro parámetro de entrada que necesite el flujo de trabajo como una matriz de propiedades. Los flujos de trabajo Iniciar flujos de trabajo en serie e Iniciar flujos de trabajo en paralelo emiten los resultados de la ejecución del flujo de trabajo en la selección de objetos como una matriz de objetos `WorkflowToken`.

Puede implementar los flujos de trabajo Iniciar flujos de trabajo en serie e Iniciar flujos de trabajo en paralelo de la misma manera. El flujo de trabajo Iniciar flujos de trabajo en serie ejecuta el flujo de trabajo en cada objeto de manera secuencial. El flujo de trabajo Iniciar flujos de trabajo en paralelo ejecuta el flujo de trabajo en todos los objetos de manera simultánea.

### Requisitos previos

Abra un flujo de trabajo para editar en el Editor de flujos de trabajo.

### Procedimiento

- 1 En el esquema del flujo de trabajo, añada un elemento de tarea de scripts o una acción para obtener una lista de objetos en los que ejecutar el flujo de trabajo.

Por ejemplo, para ejecutar un flujo de trabajo en todas las máquinas virtuales de una carpeta de máquinas virtuales, puede añadir la acción `getAllVirtualMachinesByFolder` al flujo de trabajo.

- 2 Vincule el elemento o la acción con script y enlace la entrada y la salida del elemento o de la acción con script a las entradas o a los atributos del flujo de trabajo.

Por ejemplo, puede enlazar la entrada `vmFolder` de la acción `getAllVirtualMachinesByFolder` a un parámetro de entrada de un flujo de trabajo y la salida `actionResult` a un atributo del flujo de trabajo que llama.

- 3 Añada un elemento de tarea con script para lanzar la lista de objetos a una matriz de propiedades.

Por ejemplo, si los objetos en los que se ejecuta el flujo de trabajo son una matriz de máquinas virtuales, `allVMs`, devuelta por la salida `actionResult` de la acción `getAllVirtualMachinesByFolder`, puede escribir el siguiente script para lanzar los objetos a una matriz de propiedades.

```
propsArray = new Array();

for each (var vm in allVMs) {
    var prop = new Properties();
    prop.put("vm", vm);
    propsArray.push(prop);
}
```

- 4 Enlace las entradas y las salidas de un elemento de tarea con script a los atributos del flujo de trabajo.

En el ejemplo del elemento de tarea con script del [Paso 3](#), debe enlazar la entrada a la matriz `allVMs` de máquinas virtuales y crear el atributo de salida `propsArray` como una matriz de objetos `Properties`.

- 5 Añada un elemento de flujo de trabajo al esquema del flujo de trabajo.
- 6 Seleccione el flujo de trabajo Iniciar flujos de trabajo en serie o el flujo de trabajo Iniciar flujos de trabajo en paralelo y vincule el elemento de flujo de trabajo a otros elementos.
- 7 Enlace la entrada `wf` del flujo de trabajo Iniciar flujos de trabajo en serie o Iniciar flujos de trabajo en paralelo al flujo de trabajo para ejecutarlo en los objetos.

Por ejemplo, para eliminar todas las instantáneas de todas las máquinas virtuales devueltas por la acción `getAllVirtualMachinesByFolder`, debe seleccionar el flujo de trabajo Eliminar todas las instantáneas.

- 8 Enlace la entrada `parameters` del flujo de trabajo Iniciar flujos de trabajo en serie o Iniciar flujos de trabajo en paralelo a la matriz de objetos `Properties` que contiene los objetos en los que ejecutar el flujo de trabajo.

Por ejemplo, enlace la entrada `parameters` al atributo `propsArray` definido en el [Paso 4](#).

- 9 (opcional) Enlace la salida `workflowTokens` del flujo de trabajo Iniciar flujos de trabajo en serie o Iniciar flujos de trabajo en paralelo a un atributo del flujo de trabajo.
- 10 (opcional) Siga añadiendo más elementos que utilicen los resultados de la ejecución del flujo de trabajo Iniciar flujos de trabajo en serie o Iniciar flujos de trabajo en paralelo.

Ha creado un flujo de trabajo que utiliza el flujo de trabajo Iniciar flujos de trabajo en serie o Iniciar flujos de trabajo en paralelo para ejecutar un flujo de trabajo en una selección de objetos.

## Desarrollar flujos de trabajo de larga ejecución

Un flujo de trabajo en espera consume recursos del sistema porque de manera continuada sondea el objeto del que requiere una respuesta. Si sabe que un flujo de trabajo puede terminar esperando un intervalo de tiempo prolongado antes de recibir la respuesta que necesita, puede añadir elementos de flujos de trabajo de larga ejecución al flujo de trabajo.

Cada flujo de trabajo en ejecución consume un proceso del sistema. Cuando un flujo de trabajo alcanza un elemento de flujo de trabajo de larga ejecución, este establece el flujo de trabajo en estado pasivo. A continuación, el elemento de flujo de trabajo de larga ejecución pasa la información del flujo de trabajo a un solo proceso que sondea el sistema para detectar todos los elementos de flujo de trabajo de larga ejecución que se ejecutan en el servidor. En lugar de que cada elemento de flujo de trabajo de larga ejecución intente de manera continua recuperar información del sistema, los elementos de flujo de trabajo de larga ejecución se quedan en un estado pasivo durante un intervalo de tiempo establecido, mientras el proceso de flujo de trabajo de larga ejecución sondea el sistema de su parte.

La duración de la espera se establece de una de estas formas:

- Establezca un temporizador, encapsulado en un objeto `Date`, que suspende el flujo de trabajo hasta una fecha y una hora concretas. Implemente los elementos de flujo de trabajo de larga ejecución basados en un temporizador incluyendo un elemento **Temporizador de espera** en el esquema.
- Defina un evento de activador, encapsulado en un objeto `Trigger`, que reinicia el flujo de trabajo después de producirse el evento de activador. Implemente los elementos de flujo de trabajo de larga ejecución basados en un temporizador incluyendo un **Evento en espera** o un elemento **Interacción del usuario** en el esquema.

## Establecer una fecha y una hora relativas para flujos de trabajo basados en temporizador

Puede establecer el atributo `timer.date` de un elemento Temporizador de espera en una fecha y una hora relativas enlazándolo a un objeto `Date`. El objeto `Date` se define en una función con script.

Cuando llega la fecha de la hora en cuestión, el flujo de trabajo de larga ejecución basado en un temporizador se reactiva y continúa su ejecución. Por ejemplo, puede establecer la reactivación del flujo de trabajo a mediodía del 12 de febrero. También puede crear un elemento de flujo de trabajo que calcule y genere un objeto `Date` relativo de acuerdo con una función que se defina. Por ejemplo, puede crear un objeto `Date` relativo que añada 24 horas al tiempo actual.

### Requisitos previos

- Cree un flujo de trabajo.
- Abra el flujo de trabajo para editar en el Editor de flujos de trabajo.
- Añada algunos elementos al esquema del flujo de trabajo.

### Procedimiento

- 1 Arrastre un elemento **Tarea de scripts** del menú **Genérico** al esquema de un flujo de trabajo, antes del elemento que requiere el objeto relativo `Date` para su atributo `timeout.date`.
- 2 Haga clic en el icono **Editar** (✎) del elemento **Tarea de scripts** en el esquema del flujo de trabajo.
- 3 Proporcione un nombre y una descripción para el elemento de flujo de trabajo con script en la pestaña de propiedades **Información**.
- 4 Haga clic en la pestaña de propiedades **OUT**; a continuación, haga clic en el icono de **Enlazar a parámetro o atributo de flujo de trabajo** (🔗).
- 5 Haga clic en **Crear parámetro o atributo en flujo de trabajo** para crear un atributo de flujo de trabajo.
  - a Asigne un nombre al atributo `timerDate`.
  - b Seleccione `Date` en la lista de tipos de atributo.
  - c Seleccione **Crear atributo de flujo de trabajo con el mismo nombre**.

- d Deje el valor del atributo en **No establecido**, ya que una función con script proporcionará este valor.
- e Haga clic en **Aceptar**.

6 Haga clic en la pestaña **Creación de scripts** para el elemento de flujo de trabajo con script.

7 Defina una función para calcular y generar un objeto Date denominado timerDate en la ventana de creación de scripts de la pestaña **Creación de scripts**.

Por ejemplo, puede crear un objeto Date implementando la función de JavaScript siguiente, en la que el periodo de tiempo de espera es un retraso relativo en milisegundos.

```
timerDate = new Date();
System.log( "Current date : '" + timerDate + "'" );
timerDate.setTime( timerDate.getTime() + (86400 * 1000) );
System.log( "Timer will expire at '" + timerDate + "'" );
```

La función de JavaScript de ejemplo anterior define un objeto Date que obtiene la fecha y la hora actuales con el método getTime, y añade 86.400.000 milisegundos o 24 horas. El elemento **Tarea de scripts** genera este valor como su parámetro de salida.

8 Haga clic en **Cerrar**.

9 Haga clic en **Guardar**.

Ha creado una función que calcula y genera un objeto Date. Un elemento **Temporizador de espera** puede recibir este objeto Date como parámetro de entrada para suspender un flujo de trabajo de larga ejecución hasta la fecha encapsulada en este objeto. Cuando el flujo de trabajo llega al elemento **Temporizador de espera**, suspende su ejecución y espera 24 horas antes de continuar.

### Pasos siguientes

Debe añadir un elemento **Temporizador de espera** a un flujo de trabajo para implementar un flujo de trabajo de larga ejecución basado en un temporizador.

## Crear un flujo de trabajo de larga ejecución basado en temporizador

Si sabe que un flujo de trabajo debe esperar una respuesta de un origen externo durante un tiempo predecible, puede implementarlo como flujo de trabajo de larga ejecución basado en temporizador. Un flujo de trabajo de larga ejecución basado en temporizador espera hasta una fecha y una hora específicas antes de reanudarse.

Un flujo de trabajo se implementa como flujo de trabajo de larga ejecución basado en temporizador mediante el elemento **Temporizador de espera**.

### Requisitos previos


- Cree un flujo de trabajo.
- Abra el flujo de trabajo para editar en el Editor de flujos de trabajo.
- Añada algunos elementos al esquema del flujo de trabajo.



## Procedimiento

- 1 Arrastre un elemento **Temporizador de espera** del menú **Genérico** a la posición en el esquema de flujo de trabajo en el que se suspenderá la ejecución del flujo de trabajo.

Si implementa una tarea de scripts para calcular la fecha y la hora, este elemento debe preceder al elemento **Temporizador de espera**.

- 2 Haga clic en el icono **Editar** () del elemento **Temporizador de espera** en el esquema de flujo de trabajo.

- 3 Proporcione una descripción del motivo para implementar el temporizador en la pestaña de propiedades **Información**.

- 4 Haga clic en la pestaña de propiedades **Atributos**.

El parámetro `timer.date` aparece en la lista de atributos.

- 5 En el parámetro `timer.date`, haga clic en el botón **No establecido** para enlazar el parámetro con un objeto `Date` adecuado.

Se abre el cuadro de diálogo de selección **Temporizador de espera**, con una lista de los posibles enlaces.

- Seleccione un objeto `Date` predefinido en la lista que aparece, por ejemplo uno definido mediante un elemento **Tarea de scripts** en otro lugar del flujo de trabajo.
- También puede crear un objeto `Date` que establezca una fecha y una hora específicas para la espera del flujo de trabajo.

- 6 (opcional) Cree un objeto `Date` que establezca una fecha y una hora específicas para que espere el flujo de trabajo.

- a Haga clic en **Crear parámetro o atributo en flujo de trabajo** en el cuadro de diálogo de selección **Temporizador de espera**.

Se abre el cuadro de diálogo **Información de parámetro**.

- b Asigne un nombre apropiado al parámetro.
- c Deje el tipo establecido como `Date`.
- d Haga clic en **Crear atributo de flujo de trabajo con el mismo nombre**.
- e En la propiedad **Valor**, haga clic en el botón **No establecido** para establecer el valor del parámetro.  
Aparece un calendario.
- f Utilice el calendario para establecer una fecha y una hora en las que se reiniciará el flujo de trabajo.
- g Haga clic en **Aceptar**.

- 7 Haga clic en **Cerrar**.

- 8 Haga clic en **Guardar** en la parte inferior del Editor de flujos de trabajo.

Ha definido un temporizador que suspende un flujo de trabajo de larga ejecución basado en temporizador hasta una fecha y una hora establecidas.

### Pasos siguientes

Puede crear un flujo de trabajo de larga ejecución que espere a un evento activador antes de continuar.

## Crear un objeto activador

Los objetos activadores supervisan los activadores de eventos definidos por los complementos. Por ejemplo, el complemento vCenter Server define estos eventos como objetos `Task`. Cuando finaliza la tarea, el activador envía un mensaje a un elemento de flujo de trabajo de larga ejecución basado en temporizador que está en espera, con el fin de reiniciar el flujo de trabajo.

El evento que requiere mucho tiempo al que espera el flujo de trabajo de larga ejecución basado en activador debe devolver un objeto `VC:Task`. Por ejemplo, la acción `startVM` para iniciar una máquina virtual devuelve un objeto `VC:Task`, a fin de que los elementos subsiguientes de un flujo de trabajo puedan supervisar su progreso. Un evento activador de un flujo de trabajo de larga ejecución basado en activador requiere este objeto `VC:Task` como parámetro de entrada.

Cree un objeto `Trigger` en una función de JavaScript en un elemento **Tarea de scripts**. Este elemento **Tarea de scripts** puede formar parte del flujo de trabajo de larga ejecución basado en activador que espera el evento activador. También puede formar parte de otro flujo de trabajo que proporcione parámetros de entrada al flujo de trabajo de larga ejecución basado en activador. La función activadora debe implementar el método `createEndOfTaskTrigger()` desde la API de Orchestrator.

---

**Importante** Debe definir un periodo de tiempo de espera para todos los activadores; de lo contrario, el flujo de trabajo puede esperar indefinidamente.

---

### Requisitos previos

- Cree un flujo de trabajo.
- Abra el flujo de trabajo para editar en el Editor de flujos de trabajo.
- Añada algunos elementos al esquema del flujo de trabajo.
- En el flujo de trabajo, declare un objeto `VC:Task` como parámetro de entrada o atributo, por ejemplo un objeto `VC:Task` de un flujo de trabajo o un elemento de flujo de trabajo que inicie o clone una máquina virtual.

### Procedimiento

- 1 Arrastre un elemento **Tarea de scripts** del menú **Genérico** al esquema de un flujo de trabajo.  
Uno de los elementos que precede a la **Tarea de scripts** debe generar un objeto `VC:Task` como su parámetro de salida.
- 2 Haga clic en el icono **Editar** (✎) del elemento **Tarea de scripts** en el esquema del flujo de trabajo.
- 3 Proporcione un nombre y una descripción para el activador en la pestaña de propiedades **Información**.

4 Haga clic en la pestaña de propiedades **IN**.

5 Haga clic en el icono de **Enlazar a parámetro o atributo de flujo de trabajo** ()

Se abre el cuadro de diálogo de selección de parámetros de entrada.

6 Seleccione o cree un parámetro de entrada del tipo **VC:Tarea**.

Este objeto VC:Task representa el evento que requiere mucho tiempo iniciado por otro elemento o flujo de trabajo.

7 (opcional) Seleccione o cree un parámetro de entrada del tipo Número para definir un periodo de tiempo de espera en segundos.

8 Haga clic en la pestaña de propiedades **OUT**.

9 Haga clic en el icono de **Enlazar a parámetro o atributo de flujo de trabajo** ()

Se abre el cuadro de diálogo de selección de parámetros de salida.

10 Cree un parámetro de salida con las propiedades siguientes.

- a Cree la propiedad Nombre con el valor trigger.
- b Cree la propiedad Tipo con el valor Trigger.
- c Haga clic en **Crear atributo con el mismo nombre** para crear el atributo.
- d Deje el valor como **No establecido**.

11 Defina cualquier comportamiento de excepción en la pestaña de propiedades **Excepciones**.

12 Defina una función para generar un objeto Trigger en la pestaña **Creación de scripts**.

Por ejemplo, podría crear un objeto Trigger implementando la función siguiente de JavaScript.

```
trigger = task.createEndOfTaskTrigger(timeout);
```

El método createEndOfTaskTrigger() devuelve un objeto Trigger que supervisa un objeto VC:Task denominado task.

13 Haga clic en **Cerrar**.

14 Haga clic en **Guardar** en la parte inferior del Editor de flujos de trabajo.

Ha definido un elemento de flujo de trabajo que crea un evento activador para un flujo de trabajo de larga ejecución basado en activador. El elemento activador genera un objeto Trigger como su parámetro de salida, al que el elemento **Evento de espera** puede enlazar.

### Pasos siguientes

Debe enlazar este evento activador a un elemento **Evento de espera** en un flujo de trabajo de larga ejecución basado en activador.

## Crear un flujo de trabajo de larga ejecución basado en activador

Si sabe que un flujo de trabajo debe esperar una respuesta de un origen externo durante su ejecución, pero ignora cuánto durará la espera, puede implementarlo como flujo de trabajo de larga ejecución basado en activador. Un flujo de trabajo de larga ejecución basado en activador espera a que ocurra un evento activador definido antes de reanudarse.

Un flujo de trabajo se implementa como flujo de trabajo de larga ejecución basado en activador mediante el elemento **Evento de espera**. Cuando el flujo de trabajo de larga ejecución basado en activador alcanza el elemento **Evento de espera**, suspende su ejecución y espera en estado pasivo hasta recibir un mensaje del activador. Durante el periodo de espera, el flujo de trabajo pasivo no consume ningún subproceso, sino que el elemento del flujo de trabajo de larga ejecución transfiere la información del flujo de trabajo al subproceso único que supervisa todos los flujos de trabajo de larga ejecución en el servidor.

### Requisitos previos

- Cree un flujo de trabajo.
- Abra el flujo de trabajo para editar en el Editor de flujos de trabajo.
- Añada algunos elementos al esquema del flujo de trabajo.
- Defina un evento activador encapsulado en un objeto Trigger.

### Procedimiento

- 1 Arrastre un elemento **Evento de espera** del menú **Genérico** a la posición en el esquema de flujo de trabajo en el que se suspenderá la ejecución del flujo de trabajo.

La tarea de scripts que declara el activador debe ir justo antes del elemento **Evento de espera**.

- 2 Haga clic en el icono **Editar** (✎) del elemento **Evento de espera** en el esquema de flujo de trabajo.
- 3 Proporcione una descripción del motivo de la espera en la pestaña de propiedades **Información**.
- 4 Haga clic en la pestaña de propiedades **Atributos**.

El parámetro `trigger.ref` aparece en la lista de atributos.

- 5 En el parámetro `trigger.ref`, haga clic en el enlace **No establecido** para enlazar el parámetro con un objeto Trigger adecuado.

Se abre el cuadro de diálogo de selección **Evento de espera**, con una lista de los parámetros a los que poderse enlazar.

- 6 Seleccione un objeto Trigger predefinido en la lista que aparece.

Este objeto Trigger representa un objeto activador que define otro flujo de trabajo o elemento de flujo de trabajo.

- 7 Defina cualquier comportamiento de excepción en la pestaña de propiedades **Excepciones**.
- 8 Haga clic en **Cerrar**.
- 9 Haga clic en **Guardar** en la parte inferior del Editor de flujos de trabajo.

Ha definido un elemento de flujo de trabajo que suspende un flujo de trabajo de larga ejecución basado en activador, que espera un evento activador específico antes de reiniciar.

### Pasos siguientes

Puede ejecutar un flujo de trabajo.

## Elementos de configuración

Un elemento de configuración es una lista de atributos que puede utilizar para configurar constantes en toda una implementación del servidor de Orchestrator.

Todos los flujos de trabajo, las acciones y las políticas que se ejecutan en un servidor específico de Orchestrator pueden utilizar los atributos establecidos en un elemento de configuración. La configuración de atributos en elementos de configuración permite poner a disposición los mismos valores de atributos a todos los flujos de trabajo, las acciones y las políticas que se ejecutan en el servidor de Orchestrator.

Si crea un paquete que contiene un flujo de trabajo, una acción o una política que utiliza un atributo desde un elemento de configuración, Orchestrator incluye automáticamente el elemento de configuración en el paquete. Si importa un paquete que contiene un elemento de configuración a otro servidor de Orchestrator, también puede importar los valores de atributo de elementos de configuración. Por ejemplo, si crea un flujo de trabajo que requiere valores de atributo que dependen del servidor de Orchestrator en el que se ejecuta, al configurar dichos atributos en un elemento de configuración puede exportar dicho flujo de trabajo para que otro servidor de Orchestrator pueda utilizarlo. Por lo tanto, los elementos de configuración permiten intercambiar flujos de trabajo, acciones y políticas entre servidores con mayor facilidad.

---

**Nota** No es posible importar valores de un atributo de elemento de configuración de un elemento de configuración exportado desde Orchestrator 5.1 o una versión anterior.

---

## Crear un elemento de configuración

Los elementos de configuración permiten establecer atributos comunes en un servidor de Orchestrator. Todos los elementos que se ejecutan en el servidor pueden llamar a los atributos establecidos en un elemento de configuración. La creación de elementos de configuración permite definir atributos comunes desde el servidor, en lugar de hacerlo individualmente en cada elemento.

### Procedimiento

- 1 Seleccione **Diseño** en el menú desplegable del cliente de Orchestrator.
- 2 Haga clic en la vista **Configuraciones**.
- 3 Haga clic con el botón secundario en la lista jerárquica de carpetas y seleccione **Nueva carpeta** para crear una carpeta.
- 4 Proporcione el nombre de la carpeta y haga clic en **Aceptar**.
- 5 Haga clic con el botón secundario en la carpeta que ha creado y seleccione **Nuevo elemento**.


- 6 Proporcione un nombre para el elemento de configuración y haga clic en **Aceptar**.  
Se abre el editor de elementos de configuración.
- 7 Aumente el número de versión haciendo clic en los dígitos de versión de la pestaña **General** y proporcionando un comentario de la versión.
- 8 Escriba una descripción del elemento de configuración en el cuadro de texto **Descripción** en la pestaña **General**.

9 Haga clic en la pestaña **Atributos**.

10 Haga clic en el icono **Añadir atributo** () para crear un atributo nuevo.

11 Haga clic en los valores de atributo de **Nombre**, **Tipo**, **Valor** y **Descripción**, respectivamente, para establecer el nombre, el tipo, el valor y la descripción de los atributos.

12 Haga clic en la pestaña **Permisos**.

13 Haga clic en el icono **Añadir derechos de acceso** () para conceder permiso para acceder a este elemento de configuración a un grupo de usuarios.

14 Busque un grupo de usuarios en el cuadro de texto **Filtro** y seleccione el grupo de usuarios pertinente en la lista que aparece.

15 Marque las casillas de verificación correspondientes para establecer los derechos de acceso para el grupo de usuarios seleccionado.

Puede establecer los permisos siguientes en el elemento de configuración.

Permiso	Descripción
<b>Ver</b>	Los usuarios pueden ver el elemento de configuración, pero no los esquemas ni la creación de scripts.
<b>Inspeccionar</b>	Los usuarios pueden ver el elemento de configuración, incluidos los esquemas y la creación de scripts.
<b>Administrar</b>	Los usuarios pueden establecer permisos en los elementos del elemento de configuración y tener todos los demás permisos.
<b>Ejecutar</b>	Los usuarios pueden ejecutar los elementos del elemento de configuración.
<b>Editar</b>	Los usuarios pueden editar los elementos del elemento de configuración.

16 Haga clic en **Seleccionar**.

17 Haga clic en **Guardar y cerrar** para salir del editor de elementos de configuración.

Ha definido un elemento de configuración que establece los atributos comunes en un servidor de Orchestrator.

#### Pasos siguientes

Puede utilizar el elemento de configuración a fin de proporcionar atributos para flujos de trabajo o acciones.

## Permisos de usuario de un flujo de trabajo

Orchestrator define niveles de permisos que puede aplicar a grupos para permitirles o prohibirles el acceso a los flujos de trabajo.

<b>Ver</b>	El usuario puede ver los elementos del flujo de trabajo, pero no el esquema ni la creación de scripts.
<b>Inspeccionar</b>	El usuario puede ver los elementos del flujo de trabajo, incluidos el esquema y la creación de scripts.
<b>Ejecutar</b>	El usuario puede ejecutar el flujo de trabajo.
<b>Editar</b>	El usuario puede editar el flujo de trabajo.
<b>Administrar</b>	El usuario puede establecer permisos en el flujo de trabajo y tiene el resto de permisos.

El permiso para **Administrar** incluye los permisos para **Ver**, **Inspeccionar**, **Editar** y **Ejecutar**. Todos los permisos requieren el permiso para **Ver**.

Si no define ningún permiso en un flujo de trabajo, el flujo de trabajo hereda los permisos de la carpeta que lo contiene. Si define permisos en un flujo de trabajo, esos permisos reemplazan a los permisos de la carpeta que lo contiene, incluso aunque los permisos de la carpeta sean más restrictivos.

## Establecer permisos de usuario en un flujo de trabajo

Se establecen niveles de permiso en un flujo de trabajo para limitar el acceso que los grupos de usuarios puedan tener a dicho flujo de trabajo.

Puede seleccionar los usuarios y los grupos de usuarios para los que establecer permisos en el servidor LDAP de Orchestrator.

### Requisitos previos

- Cree un flujo de trabajo.
- Abra el flujo de trabajo para editar en el Editor de flujos de trabajo.
- Añada algunos elementos al esquema del flujo de trabajo.

### Procedimiento

- 1 Haga clic en la pestaña **Permisos**.
- 2 Haga clic en el icono **Añadir derechos de acceso** () para definir permisos para un grupo de usuarios nuevo.
- 3 Busque un grupo de usuarios.

Los resultados de búsqueda contienen todos los grupos de usuarios del servidor LDAP de Orchestrator que coinciden con la búsqueda.

- 4 Seleccione un grupo de usuarios y seleccione las casillas de verificación correspondientes para establecer el nivel de permisos de este grupo de usuarios.

Para permitir que un usuario de este grupo de usuarios vea el flujo de trabajo, inspeccione el esquema y la creación de scripts, ejecute y edite los elementos y cambie los permisos, debe marcar todas las casillas de verificación.

- 5 Haga clic en **Seleccionar**.

El grupo de usuarios aparece en la lista de permisos.

- 6 Haga clic en **Guardar y cerrar** para salir del editor.

## Validar flujos de trabajo

Orchestrator proporciona una herramienta para la validación de flujos de trabajo. Al validar un flujo de trabajo, se pueden identificar errores en el flujo de trabajo y comprobar que los datos fluyan correctamente de un elemento al siguiente.

Cuando valida un flujo de trabajo, la herramienta de validación crea una lista con todos los errores y advertencias. Si se hace clic en un error de la lista, el elemento del flujo de trabajo que contiene el error queda resaltado.

Si ejecuta la herramienta de validación en el editor de flujos de trabajo, la herramienta proporciona correcciones rápidas sugeridas para los errores detectados. Algunas correcciones rápidas requieren que proporcione información adicional o parámetros de entrada. Otras correcciones rápidas resuelven los errores por usted.

La validación de flujos de trabajo comprueba los enlaces y las conexiones entre los elementos. La validación de flujos de trabajo no comprueba el procesamiento de datos que realiza cada elemento en el flujo de trabajo. Por lo tanto, un flujo de trabajo válido se puede ejecutar de manera incorrecta y producir resultados erróneos si una función de un elemento del esquema es incorrecta.

De manera predeterminada Orchestrator siempre efectúa la validación de flujos de trabajo cuando ejecuta un flujo de trabajo. Puede cambiar el comportamiento predeterminado de la validación en el cliente de Orchestrator. Consulte [Probar flujos de trabajo durante el desarrollo](#). Por ejemplo, en ocasiones durante el desarrollo de flujos de trabajo, para realizar pruebas, podría ejecutar un flujo de trabajo que sabe que no es válido.

## Validar un flujo de trabajo y corregir los errores de validación

Antes de poder ejecutar un flujo de trabajo, debe validarlo. Puede validar flujos de trabajo en el cliente de Orchestrator o en el editor de flujos de trabajo. Sin embargo, solo puede corregir errores de validación si ha abierto el flujo de trabajo para editarlo en el editor de flujos de trabajo.

### Requisitos previos

Compruebe que disponga de un flujo de trabajo completo para validar, con elementos de esquema vinculados y enlaces definidos.



## Procedimiento

- 1 Haga clic en la vista **Flujos de trabajo**.
- 2 Vaya a un flujo de trabajo en la lista jerárquica **Flujos de trabajo**.
- 3 (opcional) Haga clic con el botón secundario en el flujo de trabajo y seleccione **Validar flujo de trabajo**.

Si el flujo de trabajo es válido, aparece un mensaje de confirmación. Si el flujo de trabajo no es válido, aparece una lista de errores.

- 4 (opcional) Cierre el cuadro de diálogo Validación de flujos de trabajo.
- 5 Para abrir el editor de flujos de trabajo, haga clic con el botón secundario en el flujo de trabajo y seleccione **Editar**.
- 6 Haga clic en la pestaña **Esquema**.
- 7 Haga clic en el botón **Validar** de la barra de herramientas de la pestaña **Esquema**.

Si el flujo de trabajo es válido, aparece un mensaje de confirmación. Si el flujo de trabajo no es válido, aparece una lista de errores.

- 8 En caso de que el flujo de trabajo no sea válido, haga clic en un mensaje de error.

La herramienta de validación resalta el elemento del esquema en el que se encuentra el error añadiendo un icono rojo. Siempre que sea posible, la herramienta de validación muestra una corrección rápida.

- Si está de acuerdo con la corrección rápida propuesta, haga clic en ella para llevar a cabo la acción.
- Si no está de acuerdo con la corrección rápida propuesta, cierre el cuadro de diálogo Validación de flujos de trabajo y corrija manualmente el elemento del esquema.

---

**Importante** Compruebe siempre si la corrección que propone Orchestrator es adecuada.

---

Por ejemplo, la acción propuesta puede consistir en eliminar un atributo no utilizado, cuando es posible que ese atributo no esté enlazado correctamente.

- 9 Repita los pasos anteriores hasta que haya eliminado todos los errores de validación.

Ya ha validado un flujo de trabajo y ha corregido los errores de validación.

## Pasos siguientes

Ahora puede ejecutar el flujo de trabajo.

## Depurar flujos de trabajo

Orchestrator proporciona una herramienta para la depuración de flujos de trabajo. Puede depurar un flujo de trabajo para inspeccionar los parámetros de entrada y de salida, así como los atributos al comienzo de cualquier actividad, reemplazar valores de parámetros o de atributos durante la ejecución de un flujo

de trabajo en modo de edición, y reanudar un flujo de trabajo a partir de la última actividad que no se pudo realizar.

Puede depurar flujos de trabajo en la biblioteca de flujos de trabajo estándar y flujos de trabajo personalizados. Puede depurar flujos de trabajo personalizados mientras los desarrolla en el Editor de flujos de trabajo.

## Depurar un flujo de trabajo

Puede depurar elementos de un flujo de trabajo añadiendo puntos de interrupción a los elementos en el esquema de flujo de trabajo.

Cuando se llega a un punto de interrupción, hay varias opciones para continuar el proceso de depuración. Al depurar un elemento del esquema de flujo de trabajo, puede ver información general sobre la ejecución del flujo de trabajo, modificar las variables de flujo de trabajo y ver los mensajes de registro.

### Requisitos previos

Inicie sesión en el cliente de Orchestrator como usuario que puede ejecutar flujos de trabajo.

### Procedimiento



- 1 Seleccione **Diseño** en el menú desplegable del cliente de Orchestrator.
- 2 Haga clic en la vista **Flujos de trabajo**.
- 3 Seleccione un flujo de trabajo en la biblioteca de flujos de trabajo y haga clic en la pestaña **Esquema**.
- 4 Para añadir puntos de interrupción al esquema que desea depurar, haga clic con el botón derecho en un elemento de flujo de trabajo y seleccione **Activar/desactivar punto de interrupción**.



Los puntos de interrupción se pueden habilitar o deshabilitar.

- 5 Haga clic en el icono de **Depurar flujo de trabajo** (.

Si el flujo de trabajo requiere parámetros de entrada, se deben facilitar.

- 6 Cuando la ejecución del flujo de trabajo esté en pausa al llegar a un punto de interrupción, seleccione una de las opciones disponibles.

Opción	Descripción
 <b>Reanudar</b>	Reanuda la ejecución del flujo de trabajo hasta que se llega a otro punto de interrupción.
 <b>Entrar</b>	Le permite entrar en un elemento de flujo de trabajo.
<b>Nota</b> No se puede entrar en un elemento de flujo de trabajo anidado si se depura un flujo de trabajo en el Editor de flujos de trabajo.	

Opción	Descripción
 <b>Ignorar</b>	Se ignora el elemento actual del esquema y la ejecución del flujo de trabajo se detiene en el próximo elemento.
 <b>Volver</b>	Sale del elemento de flujo de trabajo en el que se ha entrado.

7 (opcional) En la pestaña **Puntos de interrupción**, modifique los puntos de interrupción.

Puede habilitar, deshabilitar o eliminar puntos de interrupción.

8 (opcional) En la pestaña **Variables**, revise las variables.

Durante el proceso de depuración es posible modificar los valores de algunas variables.

## Depurar un flujo de trabajo de ejemplo



Puede depurar un flujo de trabajo en la biblioteca de flujos de trabajo estándar.

Por ejemplo, puede facilitar una dirección de destinatario incorrecta y puede corregir el valor cuando depure la interacción de ejemplo con el flujo de trabajo de correo electrónico.

### Requisitos previos

Inicie la sesión en el cliente de Orchestrator como usuario que puede ejecutar flujos de trabajo de Mail.

### Procedimiento

- 1 Seleccione **Diseño** en el menú desplegable del cliente de Orchestrator.
- 2 Haga clic en la vista **Flujos de trabajo**.
- 3 En la lista jerárquica de los flujos de trabajo, abra **Biblioteca > Correo electrónico**.
- 4 Seleccione la interacción de ejemplo con el flujo de trabajo de correo electrónico y haga clic en la pestaña **Esquema**.
- 5 Haga clic con el botón derecho en el elemento del flujo de trabajo **Enviar correo electrónico (interacción)** y seleccione **Activar/desactivar punto de interrupción**.
- 6 Haga clic en el icono de **Depurar flujo de trabajo** (.
- 7 Proporcione la información requerida.
  - a En el cuadro de texto **Dirección de destino**, escriba una dirección de destinatario incompleta.  
Por ejemplo, *nombre@empresa.c*.
  - b Seleccione un grupo de usuarios de LDAP que tenga autorización para responder a la consulta.
  - c Haga clic en **Enviar**.
- 8 Cuando se alcance el punto de interrupción, haga clic en el icono **Entrar** (.
- 9 En la pestaña **Variables**, compruebe los valores.

**10** En el cuadro de texto **aDirección**, escriba el valor correcto de la dirección del destinatario.

Por ejemplo, *nombre@empresa.com*.

**11** Haga clic en el icono **Reanudar** () para continuar ejecutando el flujo de trabajo.

El flujo de trabajo utiliza el valor que indicó durante el proceso de depuración y continúa ejecutando el flujo de trabajo.

## Flujos de trabajo en ejecución

Un flujo de trabajo de Orchestrator se ejecuta de acuerdo con un flujo lógico de eventos.

Cuando ejecuta un flujo de trabajo, cada elemento de esquema del flujo de trabajo se ejecuta de acuerdo con la secuencia siguiente.

- 1 El flujo de trabajo enlaza los atributos del token de flujo de trabajo y los parámetros de entrada con los parámetros de entrada del elemento de esquema.
- 2 Se ejecuta el elemento de esquema.
- 3 Los parámetros de salida del elemento de esquema se copian en los atributos del token de flujo de trabajo y los parámetros de salida de flujo de trabajo.
- 4 Los atributos del token de flujo de trabajo y los parámetros de salida se almacenan en la base de datos.
- 5 El siguiente elemento de esquema empieza a ejecutarse.

Esta secuencia se repite en cada elemento de esquema hasta que se llega al final del flujo de trabajo.

## Puntos de comprobación del token de flujo de trabajo

Cuando se ejecuta un flujo de trabajo, cada elemento de esquema es un punto de comprobación. Después de la ejecución de cada elemento de esquema, Orchestrator guarda los atributos de token de flujo de trabajo en la base de datos; a continuación, empieza a ejecutarse el siguiente elemento de esquema. Si el flujo de trabajo se detiene inesperadamente, la próxima vez que se reinicia el servidor de Orchestrator, el elemento de esquema activo se vuelve a ejecutar y el flujo de trabajo continúa desde el inicio del elemento de esquema que se ejecutaba cuando se produjo la interrupción. Ahora bien, Orchestrator no implementa la administración de transacciones ni la función de reversión.

## Fin del flujo de trabajo

El flujo de trabajo finaliza si el elemento de esquema activo es un elemento de fin. Cuando el flujo de trabajo alcanza un elemento de fin, otros flujos de trabajo o aplicaciones pueden utilizar los parámetros de salida del flujo de trabajo.

## Ejecutar un flujo de trabajo en el Editor de flujos de trabajo

Puede ejecutar un flujo de trabajo mientras lo desarrolla.

Al ejecutar un flujo de trabajo en el Editor de flujos de trabajo, puede verificar que se ejecute correctamente sin interrumpir el proceso de desarrollo. Puede ver los mensajes de registro que proporcionan información sobre la ejecución del flujo de trabajo. Si la ejecución del flujo de trabajo devuelve resultados inesperados, puede modificar el flujo de trabajo y volver a ejecutarlo sin cerrar el Editor de flujos de trabajo.

#### Requisitos previos

- Cree un flujo de trabajo.
- Abra el flujo de trabajo para editar en el Editor de flujos de trabajo.
- Valide el flujo de trabajo.

#### Procedimiento

- 1 Haga clic en la pestaña **Esquema**.
- 2 Haga clic en **Ejecutar**.
- 3 (opcional) Revise los mensajes de la pestaña **Registros**.

## Ejecutar un flujo de trabajo

Puede realizar operaciones automatizadas en vCenter Server ejecutando flujos de trabajo desde la biblioteca estándar o flujos de trabajo que cree.

Por ejemplo, puede crear una máquina virtual ejecutando el flujo de trabajo Crear máquina virtual simple.

#### Requisitos previos

Compruebe que haya configurado el complemento vCenter Server. Para obtener más información, consulte *Instalación y configuración de VMware vCenter Orchestrator*.

#### Procedimiento

- 1 En el menú desplegable del cliente de Orchestrator, seleccione **Ejecutar** o **Diseño**.
- 2 Haga clic en la vista **Flujos de trabajo**.
- 3 En la lista jerárquica de flujos de trabajo, abra **Biblioteca > vCenter > Administración de máquinas virtuales > Básica** para ir al flujo de trabajo Crear máquina virtual simple.
- 4 Haga clic con el botón secundario en el flujo de trabajo Crear máquina virtual simple y seleccione **Iniciar flujo de trabajo**.

- 5 Proporcione la información siguiente en el cuadro de diálogo de parámetros de entrada **Iniciar flujo de trabajo** para crear una máquina virtual en una instancia de vCenter Server conectada a Orchestrator.

Opción	Acción
Nombre de máquina virtual	Asigne a la máquina virtual el nombre <b>prueba-orchestrator</b> .
Carpeta de máquina virtual	<p>a Haga clic en <b>No establecido</b> para el valor <b>Carpeta de máquina virtual</b>.</p> <p>b Seleccione una carpeta de máquina virtual en el inventario.</p> <p>El botón <b>Seleccionar</b> está inactivo hasta que se selecciona un objeto del tipo correcto, en este caso, VC:VmFolder.</p>
Tamaño del nuevo disco en GB	Escriba un valor numérico adecuado.
Tamaño de memoria en MB	Escriba un valor numérico adecuado.
Número de CPU virtuales	Seleccione un número apropiado de CPU en el menú desplegable <b>Número de CPU virtuales</b> .
SO invitado de máquina virtual	Haga clic en el vínculo <b>No establecido</b> y seleccione un sistema operativo invitado en la lista.
Host donde crear la máquina virtual	Haga clic en <b>No establecido</b> para el valor <b>Host donde crear la máquina virtual</b> ; a continuación, busque una máquina host en la jerarquía de la infraestructura de vCenter Server.
Grupo de recursos	Haga clic en <b>No establecido</b> para el valor <b>Grupo de recursos</b> y busque un grupo de recursos en la jerarquía de la infraestructura de vCenter Server.
Red a la que conectar	<p>Haga clic en <b>No establecido</b> para el valor <b>Red a la que conectar</b> y seleccione una red.</p> <p>Pulse Entrar en el cuadro de texto <b>Filtro</b> para ver todas las redes disponibles.</p>
Almacén de datos donde guardar los archivos de máquina virtual	Haga clic en <b>No establecido</b> para el valor <b>Almacén de datos donde guardar los archivos de máquina virtual</b> y busque un almacén de datos en la jerarquía de la infraestructura de vCenter Server.

- 6 Haga clic en **Enviar** para ejecutar el flujo de trabajo.

Aparece un token de flujo de trabajo bajo el flujo de trabajo Crear máquina virtual simple, que muestra el icono de flujo de trabajo en ejecución.

- 7 Haga clic en el token de flujo de trabajo para ver el estado del flujo de trabajo mientras se ejecuta.

- 8 Haga clic en la pestaña **Eventos** en la vista de token de flujo de trabajo para seguir el progreso del token del flujo de trabajo hasta que se complete.

- 9 Haga clic en la vista **Inventario**.

- 10 Busque el grupo de recursos que ha definido en la jerarquía de la infraestructura de vCenter Server.

Si la máquina virtual no aparece en la lista, haga clic en el botón de actualizar para volver a cargar el inventario.

La máquina virtual orchestrator-test está presente en el grupo de recursos.

- 11 (opcional) Haga clic con el botón secundario en la máquina virtual orchestrator-test en la vista **Inventario** para ver una lista contextual de los flujos de trabajo que puede ejecutar en la máquina virtual orchestrator-test.

El flujo de trabajo Crear máquina virtual simple se ha ejecutado correctamente.

### Pasos siguientes

Puede iniciar sesión en vSphere Client y administrar la nueva máquina virtual.

## Reanudar la ejecución de un flujo de trabajo fallido

Si falla un flujo de trabajo, Orchestrator proporciona una opción para reanudar la ejecución del flujo de trabajo a partir de la última actividad fallida.

Es posible cambiar los parámetros del flujo de trabajo e intentar resumirlo, o conservar los parámetros y efectuar cambios en componentes externos que afectan a la ejecución del flujo de trabajo. Por ejemplo, si un flujo de trabajo falla debido a un problema de un sistema de terceros, puede efectuar cambios en el sistema y reanudar la ejecución del flujo de trabajo a partir de la última actividad fallida, sin modificar los parámetros del flujo de trabajo ni repetir actividades ya realizadas correctamente.

## Comportamiento para reanudar una ejecución de flujo de trabajo fallida

Puede configurar el comportamiento para reanudar una ejecución fallida de cada flujo de trabajo personalizado. Los flujos de trabajo predeterminados de la biblioteca utilizan la configuración de sistema predeterminada para reanudar una ejecución de flujo de trabajo fallida.

Puede cambiar el comportamiento predeterminado del sistema modificando un archivo de configuración. Consulte [Definición de propiedades personalizadas para reanudar las ejecuciones de flujos de trabajo que han fallado](#).

### Requisitos previos

Compruebe que tenga permisos para editar el flujo de trabajo.

### Procedimiento

- 1 Seleccione **Diseño** en el menú desplegable del cliente de Orchestrator.
- 2 Haga clic en la vista **Flujos de trabajo**.
- 3 Expanda la lista jerárquica de flujos de trabajo para acceder al flujo de trabajo cuyo comportamiento desea configurar.
- 4 Haga clic con el botón derecho en el flujo de trabajo y seleccione **Editar**.

El Editor de flujos de trabajo se abre.

- En la pestaña **General**, seleccione una opción del menú desplegable **Comportamiento de reanudar fallido**.

Opción	Descripción
<b>Predeterminado del sistema</b>	Sigue el comportamiento predeterminado.
<b>Habilitado</b>	Si falla una ejecución de flujo de trabajo, aparece una ventana emergente con una opción para reanudar dicha ejecución.
<b>Deshabilitado</b>	Si falla un flujo de trabajo, no se puede reanudar.

- Haga clic en **Guardar y cerrar**.

## Definición de propiedades personalizadas para reanudar las ejecuciones de flujos de trabajo que han fallado

De forma predeterminada, Orchestrator no está configurado para reanudar las ejecuciones de flujos de trabajo que han fallado. Puede permitir que Orchestrator reanude las ejecuciones de flujo de trabajo que han fallado y establecer un tiempo de espera transcurrido el cual los flujos de trabajo fallidos no se podrán reanudar.

### Procedimiento

- En el sistema del servidor de Orchestrator, vaya a `/etc/vco/app-server/`.
- Abra el archivo de configuración `vmo.properties` en un editor de texto.
- Configure Orchestrator para que reanude las ejecuciones de flujos de trabajo fallidas editando la línea siguiente en el archivo `vmo.properties`.

```
com.vmware.vco.engine.execute.resume-from-failed=true
```

- Establezca un tiempo de espera para reanudar las ejecuciones de flujos de trabajo fallidas editando la línea siguiente en el archivo `vmo.properties`.

```
com.vmware.vco.engine.execute.resume-from-failed.timeout-sec=<segundos>
```

El valor establecido reemplaza la configuración de tiempo de espera predeterminada de 86.400 segundos.

- Guarde el archivo `vmo.properties`.
- Reinicie el servidor de Orchestrator.

## Reanudar la ejecución de un flujo de trabajo fallido

Puede reanudar la ejecución de un flujo de trabajo desde la última actividad fallida, si está activada la función de reanudación de ejecuciones fallidas en el flujo de trabajo.



Cuando está activada la opción para reanudar una ejecución fallida, se pueden cambiar los parámetros del flujo de trabajo e intentar reanudarla usando las opciones en la ventana que aparece al fallar el flujo. También es posible conservar los parámetros y efectuar cambios en componentes externos que afectan a la ejecución del flujo de trabajo. Si no selecciona una opción, se agotará el tiempo de espera del flujo de trabajo y no se podrá reanudar. Para modificar el periodo de tiempo de espera, consulte [Definición de propiedades personalizadas para reanudar las ejecuciones de flujos de trabajo que han fallado](#).

#### Procedimiento

- 1 En el menú desplegable de la ventana emergente, seleccione **Reanudar** y haga clic en **Siguiente**.  
Si selecciona **Cancelar**, no se podrá reanudar el flujo de trabajo posteriormente.
- 2 (opcional) Modifique los parámetros del flujo de trabajo.
- 3 Haga clic en **Enviar**.

## Generar documentación de flujo de trabajo

Puede exportar documentación en formato PDF acerca de un flujo de trabajo o una carpeta de flujos de trabajo que seleccione en cualquier momento.

El documento exportado contiene información detallada acerca del flujo de trabajo seleccionado o de los flujos de trabajo de la carpeta. La información acerca de cada flujo de trabajo incluye el nombre, el historial de las versiones del flujo de trabajo, los atributos, la presentación de los parámetros, el esquema del flujo de trabajo y las acciones del flujo de trabajo. Además, la documentación también proporciona el código fuente de las acciones empleadas.

#### Procedimiento

- 1 En el menú desplegable del cliente de Orchestrator, seleccione **Ejecutar** o **Diseño**.
- 2 Haga clic en la vista **Flujos de trabajo**.
- 3 Vaya al flujo de trabajo o a la carpeta de flujos de trabajo para los que desee generar documentación y haga clic con el botón derecho sobre ellos.
- 4 Seleccione **Generar documentación**.
- 5 Busque la carpeta en la que desea guardar el archivo PDF, indique un nombre de archivo y haga clic en **Guardar**.

El archivo PDF que contiene la información acerca del flujo de trabajo seleccionado o de los flujos de trabajo de la carpeta se guarda en su sistema.

## Historial de versiones de flujos de trabajo

Puede utilizar el historial de versiones para revertir un flujo de trabajo a un estado guardado previamente. Puede revertir el estado del flujo de trabajo a una versión anterior o una posterior. También puede comparar las diferencias entre el estado actual del flujo de trabajo y una versión guardada de este.

Orchestrator crea un nuevo elemento de historial de versiones para cada flujo de trabajo cuando incrementa y guarda la versión del flujo de trabajo. Los cambios subsiguientes en el flujo de trabajo no modifican la versión actual guardada. Por ejemplo, cuando crea la versión de flujo de trabajo 1.0.0 y la guarda, el estado del flujo de trabajo se guarda en el historial de versiones. Si realiza cambios en el flujo de trabajo, puede guardar el estado del flujo de trabajo en el cliente de Orchestrator, pero no puede aplicar los cambios en la versión del flujo de trabajo 1.0.0. Para guardar los cambios en el historial de versiones, debe crear una versión de flujo de trabajo subsiguiente y guardarla. El historial de versiones se conserva en la base de datos junto con el propio flujo de trabajo.

Cuando elimina un flujo de trabajo, Orchestrator marca el elemento como eliminado en la base de datos sin eliminar el historial de versiones del elemento de la base de datos. De este modo, es posible restaurar los flujos de trabajo eliminados. Consulte [Restaurar flujos de trabajo eliminados](#).

### Requisitos previos

Abra un flujo de trabajo para editar en el Editor de flujos de trabajo.

### Procedimiento

- 1 Haga clic en la pestaña **General** en el Editor de flujos de trabajo; a continuación, haga clic en **Mostrar historial de versiones**.
- 2 Seleccione una versión de flujo de trabajo y haga clic en **Diferencias con actual** para comparar las diferencias.

Se abre una ventana que muestra las diferencias entre la versión de flujo de trabajo actual y la versión de flujo de trabajo seleccionada.

- 3 Seleccione una versión de flujo de trabajo y haga clic en **Revertir** para restaurar el estado del flujo de trabajo.

---

**Precaución** Si no ha guardado la versión de flujo de trabajo actual, se elimina del historial de versiones y no se puede revertir a la versión actual.

---

El estado del flujo de trabajo se revierte al estado de la versión seleccionada.

## Restaurar flujos de trabajo eliminados

Puede restaurar flujos de trabajo que se han eliminado de la biblioteca.

### Procedimiento

- 1 En el menú desplegable del cliente de Orchestrator, seleccione **Ejecutar** o **Diseño**.
- 2 Haga clic en la vista **Flujos de trabajo**.
- 3 Vaya a la carpeta de flujo de trabajo en la que desea restaurar flujos de trabajo eliminados.
- 4 Haga clic con el botón derecho en la carpeta y seleccione **Restaurar flujos de trabajo eliminados**.
- 5 Seleccione los flujos de trabajo que desee restaurar y haga clic en **Restaurar**.

Los flujos de trabajo restaurados aparecerán en la carpeta seleccionada.

## Desarrollar un flujo de trabajo simple de ejemplo

El desarrollo de un flujo de trabajo simple de ejemplo demuestra los pasos más habituales del proceso de desarrollo de flujos de trabajo.

El flujo de trabajo de ejemplo que va a crear inicia una máquina virtual en vCenter Server y envía un mensaje de correo electrónico al administrador para confirmar que la máquina virtual se haya iniciado.

El flujo de trabajo de ejemplo efectúa las tareas siguientes:

- 1 Solicita al usuario que seleccione una máquina virtual que iniciar.
- 2 Solicita al usuario que proporcione una dirección de correo electrónico a la que poder enviar notificaciones.
- 3 Comprueba si la máquina virtual seleccionada ya está encendida.
- 4 Envía una solicitud a la instancia de vCenter Server para iniciar la máquina virtual.
- 5 Espera a que vCenter Server inicie la máquina virtual, y devuelve un error si esta no se puede iniciar o si tarda demasiado en iniciarse.
- 6 Espera a que vCenter Server inicie VMware Tools en la máquina virtual, y devuelve un error si esta no se puede iniciar o si VMware Tools tarda demasiado en iniciarse.
- 7 Verifica que la máquina virtual esté en ejecución.
- 8 Envía una notificación a la dirección de correo electrónico proporcionada para informar de que la máquina se ha iniciado o que se ha producido un error.

El archivo ZIP de ejemplos de Orchestrator que puede descargar de la página de inicio de la documentación de Orchestrator contiene una versión completada del flujo de trabajo Iniciar VM y enviar correo electrónico.

El proceso para desarrollar el flujo de trabajo de ejemplo se compone de varias tareas.

### Requisitos previos

Antes de intentar desarrollar el flujo de trabajo simple, lea [Conceptos clave de los flujos de trabajo](#).

### Procedimiento

#### 1 Crear el ejemplo de flujo de trabajo simple

El proceso de desarrollo de un flujo de trabajo empieza con la creación del flujo de trabajo en el cliente de Orchestrator.

#### 2 Crear el esquema del ejemplo de flujo de trabajo simple

Puede crear un esquema de flujo de trabajo en el Editor de flujos de trabajo. El esquema de flujo de trabajo contiene los elementos que ejecuta el flujo de trabajo; asimismo, determina el proceso lógico del flujo de trabajo.

### 3 (opcional) Crear las zonas de ejemplo de flujo de trabajo simple

Puede resaltar diferentes zonas del flujo de trabajo añadiendo notas de flujo de trabajo de varios colores. La creación de diferentes zonas de flujo de trabajo facilita la lectura y la comprensión de esquemas complicados de flujo de trabajo.

### 4 Definir los parámetro del ejemplo de flujo de trabajo simple

En esta fase de desarrollo del flujo de trabajo, se definen los parámetros de entrada necesarios para que se ejecute el flujo de trabajo. En el flujo de trabajo de ejemplo, se necesita un parámetro de entrada para encender la máquina virtual, así como un parámetro para que la dirección de correo electrónico de la persona informe sobre el resultado de la operación. Cuando los usuarios ejecuten el flujo de trabajo, se les indicará que especifiquen la máquina virtual que se debe encender y una dirección de correo electrónico.

### 5 Definir los enlaces de decisiones del ejemplo de flujo de trabajo simple

Los elementos de un flujo de trabajo se pueden enlazar en la pestaña **Esquema** del Editor de flujos de trabajo. Los enlaces de decisiones definen el modo en el que los elementos de decisiones comparan los parámetros de entrada recibidos en la instrucción de decisión; asimismo, generan parámetros de salida conforme a si los parámetros de entrada coinciden con la instrucción de decisión.

### 6 Enlazar los elementos de acción del ejemplo de flujo de trabajo simple

Los elementos de un flujo de trabajo se pueden enlazar en el Editor de flujos de trabajo. Los enlaces definen el modo en el que los elementos de acción procesan los parámetros de entrada y generan parámetros de salida.

### 7 Enlazar los elementos de tarea de scripts de ejemplo de flujo de trabajo simple

Los elementos de un flujo de trabajo se pueden enlazar en la pestaña **Esquema** del Editor de flujos de trabajo. Los enlaces definen el modo en el que los elementos de tarea de scripts procesan los parámetros de entrada y generan parámetros de salida. Los elementos de tarea de scripts también se pueden enlazar a sus funciones de JavaScript.

### 8 Definir los enlaces de excepciones del ejemplo de flujo de trabajo simple

Los enlaces de excepciones se definen en la pestaña **Esquema** del Editor de flujos de trabajo. Los enlaces de excepciones definen el modo en el que los elementos procesan errores.

### 9 Establecer las propiedades de lectura-escritura para atributos del ejemplo de flujo de trabajo simple

Puede definir si los parámetros y los atributos son constantes o variables de solo lectura. También puede establecer limitaciones en los valores que los usuarios pueden proporcionar para parámetros de entrada.

### 10 Definir las propiedades de parámetro del ejemplo de flujo de trabajo simple

Las propiedades de parámetro se pueden establecer en el Editor de flujos de trabajo. Establecer las propiedades de parámetro afecta al comportamiento del parámetro y coloca restricciones en los valores posibles para ese parámetro.

## 11 Establecer el diseño del cuadro de diálogo de parámetros de entrada del ejemplo de flujo de trabajo simple

Cree el diseño o la presentación del cuadro de diálogo de parámetros de entrada en el Editor de flujos de trabajo. El cuadro de diálogo de parámetros de entrada se abre cuando los usuarios ejecutan un flujo de trabajo que necesita la ejecución de parámetros de entrada.

## 12 Validar y ejecutar el ejemplo de flujo de trabajo simple

Después de crear un flujo de trabajo, puede validarlo para detectar posibles errores. Si el flujo de trabajo no contiene errores, puede ejecutarlo.

# Crear el ejemplo de flujo de trabajo simple

El proceso de desarrollo de un flujo de trabajo empieza con la creación del flujo de trabajo en el cliente de Orchestrator.

### Requisitos previos

Verifique que los componentes siguientes estén instalados y configurados en el sistema.

- vCenter Server controlando varias máquinas virtuales y como mínimo una de ellas apagada
- Acceso a un servidor SMTP
- Dirección de correo electrónico válida

Para obtener información sobre cómo instalar y configurar vCenter Server, consulte la documentación *Guía de instalación y configuración de vSphere*. Para obtener información sobre cómo configurar Orchestrator a fin de utilizar un servidor SMTP, consulte *Instalación y configuración de VMware vRealize Orchestrator*.

Para escribir un flujo de trabajo, debe tener una cuenta de usuario de Orchestrator con al menos permisos para **Ver**, **Ejecutar**, **Inspeccionar**, **Editar** y, preferiblemente, **Administrar** en el servidor o en la carpeta de flujos de trabajo en la que se trabaja.

### Procedimiento

- 1 Seleccione **Diseño** en el menú desplegable del cliente de Orchestrator.
- 2 Haga clic en la vista **Flujos de trabajo**.
- 3 Haga clic con el botón secundario en la raíz de la lista de flujos de trabajo y seleccione **Añadir carpeta**.
- 4 Asigne el nombre **Ejemplos de flujo de trabajo** a la carpeta nueva y haga clic en **Aceptar**.
- 5 Haga clic con el botón secundario en la carpeta **Ejemplos de flujo de trabajo** y seleccione **Nuevo flujo de trabajo**.
- 6 Asigne el nombre **Iniciar VM y enviar correo electrónico** al flujo de trabajo nuevo y haga clic en **Aceptar**.

Se abre el Editor de flujos de trabajo.

- 7 En la pestaña **General**, haga clic en los dígitos del número de versión para incrementar el número de versión.

Como esta es la creación inicial del flujo de trabajo, establezca la versión en **0.0.1**.

- 8 Haga clic en el valor **Comportamiento de reinicio del servidor** en la pestaña **General** para establecer si el flujo de trabajo se reanuda tras el reinicio de un servidor.

- 9 Escriba una descripción de lo que hace el flujo de trabajo en el cuadro de texto **Descripción** en la pestaña **General**.

Por ejemplo, puede añadir la descripción siguiente.

**Este flujo de trabajo inicia una máquina virtual y envía un correo electrónico de confirmación al administrador de Orchestrator.**

- 10 Haga clic en **Guardar** en la parte inferior de la pestaña **General**.

Ha creado un flujo de trabajo denominado Iniciar VM y enviar correo electrónico, pero no ha definido sus funciones.

### Pasos siguientes

Cree el esquema de flujo de trabajo.

## Crear el esquema del ejemplo de flujo de trabajo simple

Puede crear un esquema de flujo de trabajo en el Editor de flujos de trabajo. El esquema de flujo de trabajo contiene los elementos que ejecuta el flujo de trabajo; asimismo, determina el proceso lógico del flujo de trabajo.

### Requisitos previos

Complete las siguientes tareas.

- [Crear el ejemplo de flujo de trabajo simple.](#)
- Abra el flujo de trabajo en el editor de flujos de trabajo para editarlo.

### Procedimiento

- 1 Haga clic en la pestaña **Esquema** en el Editor de flujos de trabajo.
- 2 En el menú **Genérico**, arrastre un elemento de decisión a la flecha que vincula el elemento Iniciar y el elemento End en el esquema.
- 3 Haga doble clic en el elemento de decisión; a continuación, cámbiele el nombre a **¿La VM está encendida?**.

El elemento de decisión corresponde a una función booleana que comprueba si la máquina virtual ya está encendida.

- 4 En el menú **Genérico**, arrastre un elemento de decisión a la flecha roja que vincula el elemento de decisión y un elemento End.

Aparece el cuadro de diálogo para seleccionar la acción.

5 Escriba **start** en el cuadro de texto **Filtro**; a continuación, seleccione la acción **startVM** en la lista filtrada de acciones y haga clic en **Seleccionar**.

6 Arrastre los elementos de acción siguientes, uno detrás de otro, a la flecha azul que vincula el elemento de acción **startVM** a un elemento **End**.

**vim3WaitTaskEnd** Suspende la ejecución del flujo de trabajo y efectúa un ping de una tarea en curso de vCenter Server a intervalos regulares, hasta la finalización de la tarea. La acción **startVM** inicia una máquina virtual y la acción **vim3WaitTaskEnd** hace que el flujo de trabajo espere mientras se inicia la máquina virtual. Tras iniciarse la máquina virtual, **vim3WaitTaskEnd** permite la reanudación del flujo de trabajo.

**vim3WaitToolsStarted** Suspende la ejecución del flujo de trabajo y espera hasta que VMware Tools se inicie en la máquina virtual de destino.

7 En el menú **Genérico**, arrastre un elemento de tarea de scripts a la flecha azul que vincula el elemento de acción **vim3WaitToolsStarted** a un elemento **End**.

8 Haga doble clic en el elemento de tarea de scripts y cámbiele el nombre a **OK**.

9 Arrastre otro elemento de tarea de scripts a la flecha verde que vincula el elemento de decisión **VM powered on?** a un elemento **End**; a continuación, asigne el nombre **Ya iniciado** a este elemento de tarea de scripts.

10 Modifique el vínculo del elemento de tarea de scripts **Already started**.

- Arrastre el elemento de tarea de scripts **Already started** a la izquierda del elemento de acción **startVM**.
- Elimine la flecha azul que conecta elemento de tarea de scripts **Already started** a un elemento **End**.
- Vincule el elemento de tarea de scripts **Already started** al elemento de acción **vim3WaitToolsStarted** con una flecha azul.

11 En el menú **Genérico**, arrastre los elementos de tarea de scripts siguientes al esquema.

- Arrastre un elemento de tarea de scripts al elemento de acción **startVM** y asígnele el nombre **Error inicio VM**.
- Arrastre un elemento de tarea de scripts al elemento de acción **vim3WaitTaskEnd** y asígnele el nombre **Tiempo espera agotado 1**.
- Arrastre un elemento de tarea de scripts al elemento de acción **vim3WaitToolsStarted** y asígnele el nombre **Tiempo espera agotado 2**.
- Arrastre un elemento de tarea de scripts a la flecha azul que vincula el elemento de tarea de scripts **OK** a un elemento **End**; a continuación, asigne el nombre **Enviar correo** al elemento de tarea de scripts nuevo y arrástrelo a la derecha del elemento de tarea de scripts **OK**.
- Vincule los elementos de tarea de scripts **Start VM Failed**, **Timeout 1** y **Timeout 2** al elemento de tarea de scripts **Send Email** con flechas azules.

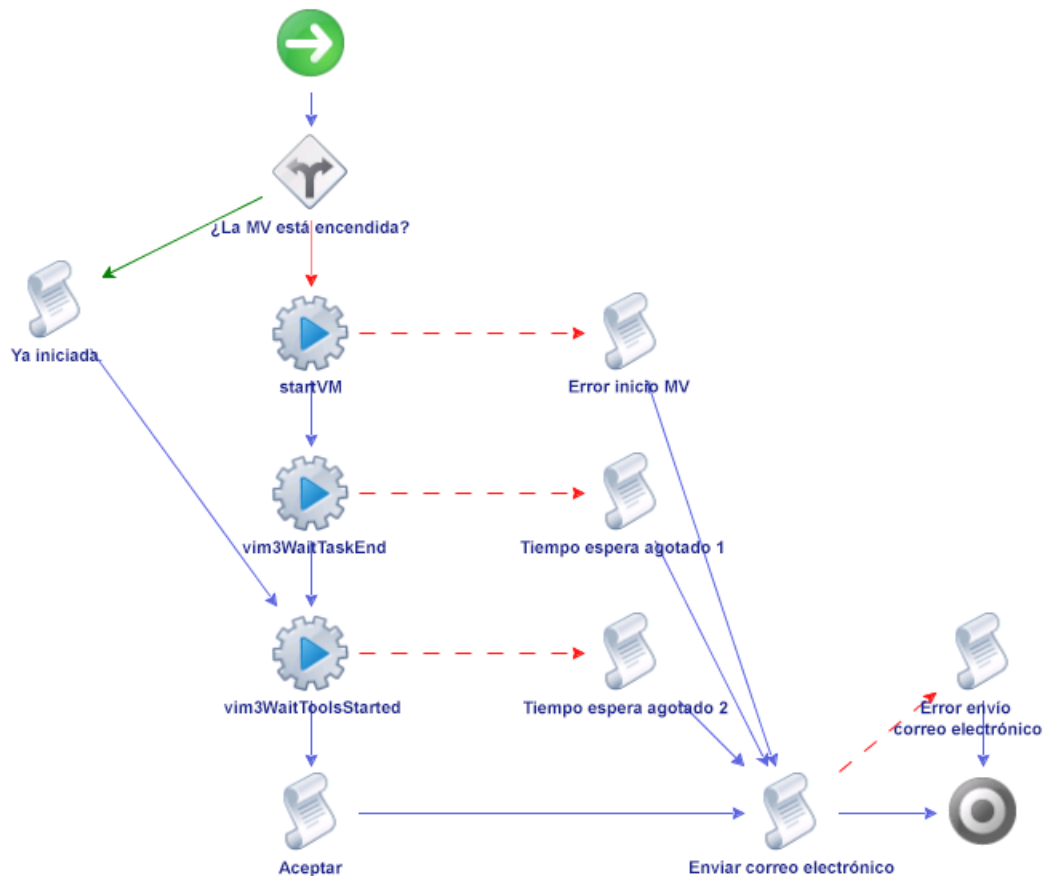
- Arrastre un elemento de tarea de scripts al elemento de tarea de scripts Send Email; a continuación, asigne el nombre **Error envío correo electrónico** al elemento de tarea de scripts nuevo, arrástrelo a la derecha del elemento de tarea de scripts Timeout 2 y vincúlelo al elemento End con una flecha azul.

12 Arrastre el elemento End a la derecha del elemento de tarea de scripts Send Email.

13 Haga clic en **Guardar** en la parte inferior de la pestaña **Esquema**.

La figura siguiente muestra la organización de los elementos de esquema de flujo de trabajo Iniciar VM y enviar correo electrónico.

**Figura 1-10. Vincular los elementos del flujo de trabajo de ejemplo Iniciar VM y enviar correo electrónico**



### Pasos siguientes

Puede resaltar diferentes zonas del flujo de trabajo.

## Crear las zonas de ejemplo de flujo de trabajo simple

Puede resaltar diferentes zonas del flujo de trabajo añadiendo notas de flujo de trabajo de varios colores. La creación de diferentes zonas de flujo de trabajo facilita la lectura y la comprensión de esquemas complicados de flujo de trabajo.



## Requisitos previos

Complete las siguientes tareas.

- [Crear el ejemplo de flujo de trabajo simple.](#)
- [Crear el esquema del ejemplo de flujo de trabajo simple.](#)
- Abra el flujo de trabajo en el editor de flujos de trabajo para editarlo.

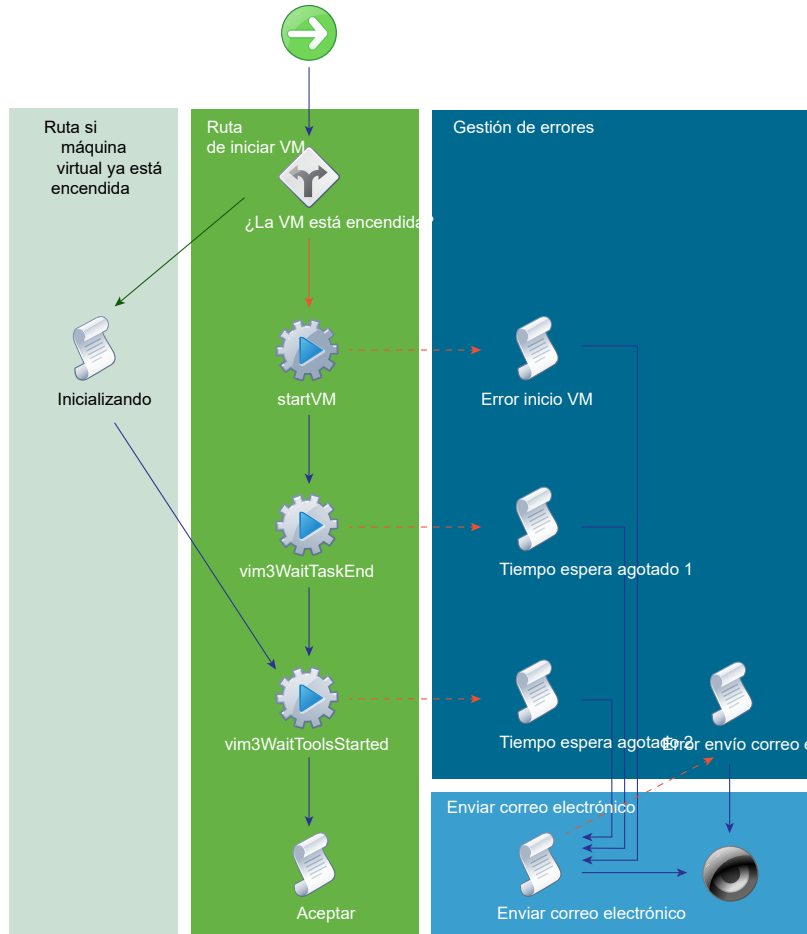
## Procedimiento

- 1 Arrastre un elemento de nota de flujo de trabajo del menú **Genérico** al Editor de flujos de trabajo.
- 2 Coloque la nota de flujo de trabajo sobre el elemento de tarea de scripts `Already started`.
- 3 Arrastre los bordes de la nota de flujo de trabajo para cambiarle el tamaño y que envuelva el elemento de tarea de scripts `Already started`.
- 4 Haga doble clic en el texto y escriba una descripción.

Por ejemplo, **Ruta si la máquina virtual ya está encendida.**

- 5 Pulse Ctrl+E para seleccionar el color de fondo.
- 6 Repita los pasos anteriores para resaltar otras zonas del flujo de trabajo.
  - Coloque una nota sobre la secuencia vertical de elementos del elemento de decisión `VM powered on?` al elemento `OK`. Añada la descripción **Ruta de iniciar VM.**
  - Coloque una nota sobre los elementos de tarea de scripts `startVM failed`, los dos elementos de tarea de scripts `Timeout` y el elemento de tarea de scripts `Send Email Failed`. Añada la descripción **Gestión de errores.**
  - Coloque una nota sobre el elemento de tarea de scripts `Send Email`. Añada la descripción **Enviar correo electrónico.**

La figura siguiente muestra el aspecto aproximado que tendrán las zonas de flujo de trabajo.

**Figura 1-11. Zonas del flujo de trabajo de ejemplo Iniciar VM y enviar correo electrónico****Pasos siguientes**

Debe definir los atributos y los parámetros de entrada y de salida del flujo de trabajo.

**Definir los parámetro del ejemplo de flujo de trabajo simple**

En esta fase de desarrollo del flujo de trabajo, se definen los parámetros de entrada necesarios para que se ejecute el flujo de trabajo. En el flujo de trabajo de ejemplo, se necesita un parámetro de entrada para encender la máquina virtual, así como un parámetro para que la dirección de correo electrónico de la persona informe sobre el resultado de la operación. Cuando los usuarios ejecuten el flujo de trabajo, se les indicará que especifiquen la máquina virtual que se debe encender y una dirección de correo electrónico.

**Requisitos previos**

Complete las siguientes tareas.

- [Crear el ejemplo de flujo de trabajo simple.](#)
- Abra el flujo de trabajo en el editor de flujos de trabajo para editarlo.

**Procedimiento**

- 1 Haga clic en la pestaña **Entradas** en el Editor de flujos de trabajo.
- 2 Haga clic con el botón secundario en la pestaña **Entradas** y seleccione **Añadir parámetro**.  
Un parámetro denominado `arg_in_0` aparece en la pestaña **Entradas**.
- 3 Haga clic en `arg_in_0`.
- 4 Escriba el nombre `vm` en el cuadro de diálogo Elegir nombre de atributo y haga clic en **Aceptar**.
- 5 Haga clic en el cuadro de texto **Tipo**; a continuación, escriba `vc:virtualm` en el cuadro de texto de búsqueda del cuadro de diálogo Tipo de parámetro.
- 6 Seleccione **VC:VirtualMachine** en la lista propuesta de tipos de parámetros y haga clic en **Aceptar**.
- 7 En el cuadro de texto **Descripción**, escriba una descripción del parámetro.  
Por ejemplo, escriba `La máquina virtual que se encenderá`.
- 8 Repita el [Paso 2](#) al [Paso 7](#) para crear un segundo parámetro de entrada con los valores siguientes.
  - Nombre: `toAddress`
  - Tipo: cadena
  - Descripción:  
`La dirección de correo electrónico a la que enviar el resultado de este flujo de trabajo`
- 9 Haga clic en **Guardar** en la parte inferior de la pestaña **Entradas**.

Ha definido los parámetros de entrada del flujo de trabajo.

**Pasos siguientes**

Defina los enlaces entre los parámetros del elemento.

## Definir los enlaces de decisiones del ejemplo de flujo de trabajo simple

Los elementos de un flujo de trabajo se pueden enlazar en la pestaña **Esquema** del Editor de flujos de trabajo. Los enlaces de decisiones definen el modo en el que los elementos de decisiones comparan los parámetros de entrada recibidos en la instrucción de decisión; asimismo, generan parámetros de salida conforme a si los parámetros de entrada coinciden con la instrucción de decisión.


**Requisitos previos**

Complete las siguientes tareas.

- [Crear el ejemplo de flujo de trabajo simple](#).
- [Crear el esquema del ejemplo de flujo de trabajo simple](#).
- [Definir los parámetro del ejemplo de flujo de trabajo simple](#).

- Abra el flujo de trabajo en el editor de flujos de trabajo para editarlo.

### Procedimiento

- 1 En la pestaña **Esquema**, haga clic en el icono **Editar** () del elemento de decisión **VM Powered On?**.
- 2 En la pestaña **Decisión**, haga clic en el botón **No establecido (nulo)** y seleccione **vm** como parámetro de entrada del elemento de decisión en la lista de parámetros propuestos.
- 3 Seleccione la instrucción **El estado de encendido es igual a** en la lista de instrucciones de decisión propuestas en el menú desplegable.

Aparece el botón **No establecido** en el cuadro de texto del valor, que ofrece una serie limitada de selección de valores.

- 4 Seleccione **poweredOn**.
- 5 Haga clic en **Guardar** en la parte inferior de la pestaña **Esquema** del Editor de flujos de trabajo.

Ha definido la instrucción true o false respecto a la cual el elemento de decisión comparará el valor del parámetro de entrada que recibe.

### Pasos siguientes

Debe definir los enlaces de los otros elementos del flujo de trabajo.

## Enlazar los elementos de acción del ejemplo de flujo de trabajo simple

Los elementos de un flujo de trabajo se pueden enlazar en el Editor de flujos de trabajo. Los enlaces definen el modo en el que los elementos de acción procesan los parámetros de entrada y generan parámetros de salida.

### Requisitos previos

Complete las siguientes tareas.

- [Crear el ejemplo de flujo de trabajo simple.](#)
- [Crear el esquema del ejemplo de flujo de trabajo simple.](#)
- [Definir los parámetro del ejemplo de flujo de trabajo simple.](#)
- [Definir los enlaces de decisiones del ejemplo de flujo de trabajo simple.](#)
- Abra el flujo de trabajo en el editor de flujos de trabajo para editarlo.

### Procedimiento

- 1 En la pestaña **Esquema**, haga clic en el icono **Editar** () del elemento de acción **startVM**.

## 2 Establezca la información general siguiente en la pestaña **Información**.

Opción	Acción
Interacción	Seleccione <b>Sin interacción externa</b> .
Estado empresarial	Seleccione la casilla de verificación y añada el texto <b>Enviando inicio VM</b> .
Descripción	Deje el texto Iniciar/Reanudar una VM. Vuelva a la tarea de inicio.

## 3 Haga clic en la pestaña **IN**.

La pestaña **IN** muestra los dos parámetros posibles de entrada para la acción startVM y host.

Orchestrator enlaza automáticamente el parámetro vm a vm[in-parameter] porque la acción startVM solo puede tomar un VC:VirtualMachine como parámetro de entrada. Orchestrator detecta el parámetro vm que ha definido al establecer los parámetros de entrada del flujo de trabajo y así lo vincula a la acción automáticamente.

## 4 Establezca host en **NULO**.

Es un parámetro opcional, por lo que se puede establecer en nulo. Ahora bien, si lo deja en el estado **No establecido**, el flujo de trabajo no puede validar.

## 5 Haga clic en la pestaña **OUT**.

Aparece actionResult, el parámetro de salida que generan todas las acciones.

## 6 En el parámetro actionResult, haga clic en **No establecido**.

## 7 Haga clic en **Crear parámetro o atributo en flujo de trabajo**.

El cuadro de diálogo Información de parámetro muestra los valores que puede establecer para este parámetro de salida. El tipo de parámetro de salida para la acción startVM es un objeto VC:Task.

## 8 Asigne el nombre **powerOnTask** al parámetro y proporcione una descripción.

Por ejemplo, **Contiene el resultado de encender una VM**.

## 9 Haga clic en **Crear atributo de flujo de trabajo con el mismo nombre**; a continuación, haga clic en **Aceptar** para salir del cuadro de diálogo Información de parámetro.

## 10 Repita los pasos anteriores para enlazar los parámetros de entrada y de salida a los elementos de acción vim3WaitTaskEnd y vim3WaitToolsStarted.

[Enlaces de elemento de acción del ejemplo de flujo de trabajo simple](#) muestra los enlaces de los elementos de acción vim3WaitTaskEnd y vim3WaitToolsStarted.

## 11 Haga clic en **Guardar** en la parte inferior de la pestaña **Esquema** del Editor de flujos de trabajo.

Los parámetros de entrada y de salida de los elementos de acción se enlazan a los correspondientes valores y tipos de parámetros.

### Pasos siguientes

Enlace los elementos de tarea de scripts y defina sus funciones.

## Enlaces de elemento de acción del ejemplo de flujo de trabajo simple

Los enlaces definen el modo en el que los elementos de acción del ejemplo de flujo de trabajo simple procesan parámetros de entrada y de salida.

Al definir enlaces, Orchestrator presenta parámetros que ya ha definido en el flujo de trabajo como candidatos para enlazar. Si todavía no ha definido el parámetro pertinente en el flujo de trabajo, NULL es la única opción posible de parámetro. Haga clic en **Crear parámetro o atributo en flujo de trabajo** para crear un parámetro.

### Acción vim3WaitTaskEnd

El elemento de acción vim3WaitTaskEnd declara constantes para efectuar el seguimiento del progreso de una tarea y de una tasa de sondeo. En la tabla siguiente, figuran los enlaces de parámetros de entrada y de salida requeridos por la acción vim3WaitTaskEnd.

**Tabla 1-53. Valores de enlace de la acción vim3WaitTaskEnd**

Nombre del parámetro	Tipo de enlace	¿Enlazar con parámetro existente o crear parámetro?	Valores de enlace
task	IN	Enlazar	<ul style="list-style-type: none"> <li>■ Parámetro local: powerOnTask</li> <li>■ Parámetro de origen: task[attribute]</li> <li>■ Tipo: VC:Task</li> <li>■ Descripción: <b>Contiene el resultado de encender una VM.</b></li> </ul>
progress	IN	Crear	<ul style="list-style-type: none"> <li>■ Parámetro local: progress</li> <li>■ Parámetro de origen: progress[attribute]</li> <li>■ Tipo: booleano</li> <li>■ Valor: no (false)</li> <li>■ Descripción: <b>Progreso de registro mientras se espera a que finalice la tarea de vCenter Server.</b></li> </ul>

Tabla 1-53. Valores de enlace de la acción vim3WaitTaskEnd (continuación)

Nombre del parámetro	Tipo de enlace	¿Enlazar con parámetro existente o crear parámetro?	Valores de enlace
pollRate	IN	Crear	<ul style="list-style-type: none"> <li>■ Parámetro local: pollRate</li> <li>■ Parámetro de origen: pollRate[attribute]</li> <li>■ Tipo: número</li> <li>■ Valor: 2</li> <li>■ Descripción: <b>Tasa de sondeo en segundos a la que vim3WaitTaskEnd comprueba el avance de la tarea de vCenter Server.</b></li> </ul>
actionResult	OUT	Crear	<ul style="list-style-type: none"> <li>■ Parámetro local: actionResult[attribute]</li> <li>■ Parámetro de origen: returnedManagedObject[attribute]</li> <li>■ Tipo: cualquiera</li> <li>■ Descripción: <b>El objeto administrado devuelto por la acción waitTaskEnd.</b></li> </ul>

### Acción vim3WaitToolsStarted

El elemento de acción vim3WaitToolsStarted espera hasta que VMware Tools se haya instalado en una máquina virtual, y define una tasa de sondeo y un periodo de tiempo de espera. En la tabla siguiente, figuran los enlaces de parámetros de entrada requeridos por la acción vim3WaitToolsStarted.

El elemento de acción vim3WaitToolsStarted no tiene salida, de modo que no necesita enlace de salida.

Tabla 1-54. Valores de enlace de la acción vim3WaitToolsStarted

Nombre del parámetro	Tipo de enlace	¿Enlazar con parámetro existente o crear parámetro?	Valores de enlace
vm	IN	Enlace automático	<ul style="list-style-type: none"> <li>■ Parámetro local: vm</li> <li>■ Parámetro de origen: vm[in-parameter]</li> <li>■ Tipo: VC:VirtualMachine</li> <li>■ Valor: no editable, la variable no es un atributo de flujo de trabajo.</li> <li>■ Descripción: <b>La máquina virtual que se iniciará.</b></li> </ul>
pollingRate	IN	Enlazar	<ul style="list-style-type: none"> <li>■ Parámetro local: pollRate</li> <li>■ Parámetro de origen: pollRate[attribute]</li> <li>■ Tipo: número</li> <li>■ Descripción: <b>La tasa de sondeo en segundos a la que vim3WaitTaskEnd comprueba el avance de la tarea de vCenter Server.</b></li> </ul>
timeout	IN	Crear	<ul style="list-style-type: none"> <li>■ Parámetro local: timeout</li> <li>■ Parámetro de origen: timeout[attribute]</li> <li>■ Tipo: número</li> <li>■ Valor: <b>10</b></li> <li>■ Descripción: <b>El límite de tiempo que vim3WaitToolsStarted espera antes de generar una excepción.</b></li> </ul>

## Enlazar los elementos de tarea de scripts de ejemplo de flujo de trabajo simple

Los elementos de un flujo de trabajo se pueden enlazar en la pestaña **Esquema** del Editor de flujos de trabajo. Los enlaces definen el modo en el que los elementos de tarea de scripts procesan los parámetros de entrada y generan parámetros de salida. Los elementos de tarea de scripts también se pueden enlazar a sus funciones de JavaScript.

### Requisitos previos


Complete las siguientes tareas.

- [Crear el ejemplo de flujo de trabajo simple.](#)
- [Crear el esquema del ejemplo de flujo de trabajo simple.](#)




- Definir los parámetro del ejemplo de flujo de trabajo simple.
- Definir los enlaces de decisiones del ejemplo de flujo de trabajo simple.
- Abra el flujo de trabajo en el editor de flujos de trabajo para editarlo.

### Procedimiento

- 1 En la pestaña **Esquema**, haga clic en el icono **Editar** () del elemento de tarea de scripts Already Started.
- 2 Establezca la información general siguiente en la pestaña **Información**.

Opción	Acción
Interacción	Seleccione <b>Sin interacción externa</b> .
Estado empresarial	Seleccione la casilla de verificación y añada el texto <b>VM ya encendida</b> .
Descripción	Deje el texto VM ya encendida omitiendo startVM y waitTaskEnd, y comprobando que las herramientas de VM estén en funcionamiento.

- 3 Haga clic en la pestaña **IN**.  
Como se trata de un elemento de tarea de scripts personalizable, no hay propiedades predefinidas.
- 4 Haga clic en el icono de **Enlazar a parámetro o atributo de flujo de trabajo** ()
- 5 Seleccione vm en la lista propuesta de parámetros.
- 6 Deje en blanco las pestañas **OUT** y **Excepción**.  
Este elemento no genera un parámetro de salida ni una excepción.
- 7 Haga clic en la pestaña **Creación de scripts**.
- 8 Añada la función de JavaScript siguiente.

```
//Writes the following event in the Orchestrator database
Server.log("VM '"+ vm.name +"' already started");
```

- 9 Repita los pasos anteriores para enlazar los demás parámetros de entrada y de salida a los elementos de tarea de scripts.

[Enlaces de elemento de tarea de scripts del ejemplo de flujo de trabajo simple](#) enumera los enlaces de Start VM failed, Timeout o ErrorSend Email Failed, y los elementos de tarea de scripts OK.

- 10 Haga clic en **Guardar** en la parte inferior de la pestaña **Esquema** del Editor de flujos de trabajo.

Ha enlazado los elementos de tarea de scripts a sus parámetros de entrada y de salida, y proporcionado la creación de scripts que define su función.

### Pasos siguientes

Debe definir el control de excepciones.

## Enlaces de elemento de tarea de scripts del ejemplo de flujo de trabajo simple

Los enlaces definen el modo en el que los elementos de tarea de scripts del ejemplo de flujo de trabajo simple procesan parámetros de entrada. Los elementos de tarea de scripts también se pueden enlazar a sus funciones de JavaScript.

Al definir enlaces, Orchestrator presenta parámetros que ya ha definido en el flujo de trabajo como candidatos para enlazar. Si todavía no ha definido el parámetro pertinente en el flujo de trabajo, NULL es la única opción posible de parámetro. Haga clic en **Crear parámetro o atributo en flujo de trabajo** para crear un parámetro.

### Tarea de scripts de Error inicio VM

El elemento de tarea de scripts Error inicio VM administra todas las excepciones lanzadas por la acción startVM estableciendo el contenido de una notificación por correo electrónico sobre el error al iniciar la máquina virtual y escribiendo el evento en el registro de Orchestrator.

En la tabla siguiente, figuran los enlaces de parámetros de entrada y de salida requeridos por el elemento de tarea de scripts Error inicio VM.

**Tabla 1-55. Enlaces del elemento de tarea de scripts Error inicio VM**

Nombre del parámetro	Tipo de enlace	¿Enlazar con parámetro existente o crear parámetro?	Valores de enlace
vm	IN	Enlazar	<ul style="list-style-type: none"> <li>■ Parámetro local: vm</li> <li>■ Parámetro de origen: vm[in-parameter]</li> <li>■ Tipo: VC:VirtualMachine</li> <li>■ Descripción: <b>La máquina virtual que se encenderá.</b></li> </ul>
errorCode	IN	Crear	<ul style="list-style-type: none"> <li>■ Parámetro local: errorCode</li> <li>■ Parámetro de origen: errorCode[attribute]</li> <li>■ Tipo: cadena</li> <li>■ Descripción: <b>Capturar excepciones durante el encendido de una VM.</b></li> </ul>
body	OUT	Crear	<ul style="list-style-type: none"> <li>■ Parámetro local: body</li> <li>■ Parámetro de origen: body[attribute]</li> <li>■ Tipo: cadena</li> <li>■ Descripción: <b>El cuerpo del correo electrónico</b></li> </ul>

El elemento de tarea de scripts Error inicio VM efectúa la función con scripts siguiente.

```
body = "Unable to execute powerOnVM_Task() on VM '"+vm.name+"', exception found: "+errorCode;
//Writes the following event in the Orchestrator database
Server.error("Unable to execute powerOnVM_Task() on VM '"+vm.name+"', exception found: "+errorCode);
```

### Elemento de tarea de scripts Tiempo de espera 1

El elemento de tarea de scripts Tiempo de espera 1 administra todas las excepciones lanzadas por la acción vim3WaitTaskEnd estableciendo el contenido de una notificación por correo electrónico sobre el error de la tarea y escribiendo el evento en el registro de Orchestrator.

En la tabla siguiente, figuran los enlaces de parámetros de entrada y de salida requeridos por el elemento de tarea de scripts Tiempo de espera 1.

**Tabla 1-56. Enlaces del elemento de tarea de scripts Tiempo de espera 1**

Nombre del parámetro	Tipo de enlace	¿Enlazar con parámetro existente o crear parámetro?	Valores de enlace
vm	IN	Enlazar	<ul style="list-style-type: none"> <li>■ Parámetro local: vm</li> <li>■ Parámetro de origen: vm[in-parameter]</li> <li>■ Tipo: VC:VirtualMachine</li> <li>■ Descripción: <b>La máquina virtual que se iniciará.</b></li> </ul>
errorCode	IN	Enlazar	<ul style="list-style-type: none"> <li>■ Parámetro local: errorCode</li> <li>■ Parámetro de origen: errorCode[attribute]</li> <li>■ Tipo: cadena</li> <li>■ Descripción: <b>Capturar excepciones durante el encendido de una VM.</b></li> </ul>
body	OUT	Enlazar	<ul style="list-style-type: none"> <li>■ Parámetro local: body</li> <li>■ Parámetro de origen: body[attribute]</li> <li>■ Tipo: cadena</li> <li>■ Descripción: <b>El cuerpo del correo electrónico</b></li> </ul>

El elemento de tarea de scripts Tiempo de espera 1 requiere la función con scripts siguiente.

```
body = "Error while waiting for poweredOnVM_Task() to complete on VM '"+vm.name+"', exception found: "+errorCode;
//Writes the following event in the Orchestrator database
Server.error("Error while waiting for poweredOnVM_Task() to complete on VM '"+vm.name+"', exception found: "+errorCode);
```

## Elemento de tarea de scripts Tiempo de espera 2

El elemento de tarea de scripts Tiempo de espera 2 administra todas las excepciones lanzadas por la acción `vim3WaitToolsStarted` estableciendo el contenido de una notificación por correo electrónico sobre el error de la tarea y escribiendo el evento en el registro de Orchestrator.

En la tabla siguiente, figuran los enlaces de parámetros de entrada y de salida requeridos por el elemento de tarea de scripts Tiempo de espera 2.

**Tabla 1-57. Enlaces del elemento de tarea de scripts Tiempo de espera 2**

Nombre del parámetro	Tipo de enlace	¿Enlazar con parámetro existente o crear parámetro?	Valores de enlace
vm	IN	Enlazar	<ul style="list-style-type: none"> <li>■ Parámetro local: vm</li> <li>■ Parámetro de origen: vm[in-parameter]</li> <li>■ Tipo: VC:VirtualMachine</li> <li>■ Descripción: <b>La máquina virtual que se encenderá.</b></li> </ul>
errorCode	IN	Enlazar	<ul style="list-style-type: none"> <li>■ Parámetro local: errorCode</li> <li>■ Parámetro de origen: errorCode[attribute]</li> <li>■ Tipo: cadena</li> <li>■ Descripción: <b>Capturar excepciones durante el encendido de una VM.</b></li> </ul>
body	OUT	Enlazar	<ul style="list-style-type: none"> <li>■ Parámetro local: body</li> <li>■ Parámetro de origen: body[attribute]</li> <li>■ Tipo: cadena</li> <li>■ Descripción: <b>El cuerpo del correo electrónico</b></li> </ul>

El elemento de tarea de scripts Tiempo de espera 2 requiere la función con scripts siguiente.

```
body = "Error while waiting for VMware tools to be up on VM '"+vm.name+"', exception found:
"+errorCode;
//Writes the following event in the Orchestrator database
Server.error("Error while waiting for VMware tools to be up on VM '"+vm.name+"', exception found:
"+errorCode);
```

## Elemento de tarea de scripts Correcto

El elemento de tarea de scripts Correcto recibe el aviso de que la máquina virtual se ha iniciado correctamente, establece el contenido de una notificación por correo electrónico sobre el inicio correcto de la máquina virtual y escribe el evento en el registro de Orchestrator.

En la tabla siguiente, figuran los enlaces de parámetros de entrada y de salida requeridos por el elemento de tarea de scripts Correcto.

**Tabla 1-58. Enlaces del elemento de tarea de scripts Correcto**

Nombre del parámetro	Tipo de enlace	¿Enlazar con parámetro existente o crear parámetro?	Valores de enlace
vm	IN	Enlazar	<ul style="list-style-type: none"> <li>■ Parámetro local: vm</li> <li>■ Parámetro de origen: vm[in-parameter]</li> <li>■ Tipo: VC:VirtualMachine</li> <li>■ Descripción: <b>La máquina virtual que se encenderá.</b></li> </ul>
body	OUT	Enlazar	<ul style="list-style-type: none"> <li>■ Parámetro local: body</li> <li>■ Parámetro de origen: body[attribute]</li> <li>■ Tipo: cadena</li> <li>■ Descripción: <b>El cuerpo del correo electrónico</b></li> </ul>

El elemento de tarea de scripts Correcto requiere la función con scripts siguiente.

```
body = "The VM '"+vm.name+"' has started successfully and is ready for use";
//Writes the following event in the Orchestrator database
Server.log(body);
```

### Elemento de tarea de scripts Error envío correo electrónico

El elemento de tarea de scripts Error envío correo electrónico recibe el aviso del error de envío de correo electrónico y escribe el evento en el registro de Orchestrator.

En la tabla siguiente, figuran los enlaces de parámetros de entrada y de salida requeridos por el elemento de tarea de scripts Error envío correo electrónico.

Tabla 1-59. Enlaces del elemento de tarea de scripts Error envío correo electrónico

Nombre del parámetro	Tipo de enlace	¿Enlazar con parámetro existente o crear parámetro?	Valores de enlace
vm	IN	Enlazar	<ul style="list-style-type: none"> <li>■ Parámetro local: vm</li> <li>■ Parámetro de origen: vm[in-parameter]</li> <li>■ Tipo: VC:VirtualMachine</li> <li>■ Descripción: <b>La máquina virtual que se encenderá.</b></li> </ul>
toAddress	IN	Enlazar	<ul style="list-style-type: none"> <li>■ Parámetro local: toAddress</li> <li>■ Parámetro de origen: toAddress[in-parameter]</li> <li>■ Tipo: cadena</li> <li>■ Descripción: <b>La dirección de correo electrónico de la persona a la que se informará del resultado de este flujo de trabajo</b></li> </ul>
emailErrorCode	IN	Crear	<ul style="list-style-type: none"> <li>■ Parámetro local: emailErrorCode</li> <li>■ Parámetro de origen: emailErrorCode[attribute]</li> <li>■ Tipo: cadena</li> <li>■ Descripción: <b>Capturar excepciones durante el envío de un correo electrónico</b></li> </ul>

El elemento de tarea de scripts Error envío correo electrónico requiere la función con scripts siguiente.

```
//Writes the following event in the Orchestrator database
Server.error("Couldn't send result email to '"+toAddress+"' for VM '"+vm.name+"', exception found: "+emailErrorCode);
```

### Elemento de tarea de scripts Enviar correo electrónico

La finalidad del flujo de trabajo Iniciar VM y enviar correo electrónico es informar a un administrador cuando se inicia una máquina virtual. Para ello, se debe definir la tarea de scripts que envía un correo electrónico. Para enviar el correo electrónico, el elemento de tarea de scripts Enviar correo electrónico necesita un servidor SMTP, las direcciones del remitente y del destinatario del correo electrónico, el asunto y el contenido del correo electrónico.

En la tabla siguiente, figuran los enlaces de parámetros de entrada y de salida requeridos por el elemento de tarea de scripts Enviar correo electrónico.

Tabla 1-60. Enlaces del elemento de tarea de scripts Enviar correo electrónico

Nombre del parámetro	Tipo de enlace	¿Enlazar con parámetro existente o crear parámetro?	Valores de enlace
vm	IN	Enlazar	<ul style="list-style-type: none"> <li>■ Parámetro local: vm</li> <li>■ Parámetro de origen: vm[in-parameter]</li> <li>■ Tipo: VC:VirtualMachine</li> <li>■ Descripción: <b>La máquina virtual que se encenderá.</b></li> </ul>
toAddress	IN	Enlazar	<ul style="list-style-type: none"> <li>■ Parámetro local: toAddress</li> <li>■ Parámetro de origen: toAddress[in-parameter]</li> <li>■ Tipo: cadena</li> <li>■ Descripción: <b>La dirección de correo electrónico de la persona a la que se informará del resultado de este flujo de trabajo</b></li> </ul>
body	IN	Enlazar	<ul style="list-style-type: none"> <li>■ Parámetro local: body</li> <li>■ Parámetro de origen: body[attribute]</li> <li>■ Tipo: cadena</li> <li>■ Descripción: <b>El cuerpo del correo electrónico</b></li> </ul>
smtpHost	IN	Crear	<ul style="list-style-type: none"> <li>■ Parámetro local: smtpHost</li> <li>■ Parámetro de origen: smtpHost[attribute]</li> <li>■ Tipo: cadena</li> <li>■ Descripción: <b>El servidor SMTP del correo electrónico</b></li> </ul>

Tabla 1-60. Enlaces del elemento de tarea de scripts Enviar correo electrónico (continuación)

Nombre del parámetro	Tipo de enlace	¿Enlazar con parámetro existente o crear parámetro?	Valores de enlace
fromAddress	IN	Crear	<ul style="list-style-type: none"> <li>■ Parámetro local: fromAddress</li> <li>■ Parámetro de origen: fromAddress[attribute]</li> <li>■ Tipo: cadena</li> <li>■ Descripción: <b>La dirección de correo electrónico del remitente</b></li> </ul>
subject	IN	Crear	<ul style="list-style-type: none"> <li>■ Parámetro local: subject</li> <li>■ Parámetro de origen: subject[attribute]</li> <li>■ Tipo: cadena</li> <li>■ Descripción: <b>El asunto del correo electrónico</b></li> </ul>

El elemento de tarea de scripts Enviar correo electrónico requiere la función con scripts siguiente.

```
//Create an instance of EmailMessage
var myEmailMessage = new EmailMessage() ;

//Apply methods on this instance that populate the email message
myEmailMessage.smtpHost = smtpHost;
myEmailMessage.fromAddress = fromAddress;
myEmailMessage.toAddress = toAddress;
myEmailMessage.subject = subject;
myEmailMessage.addMimePart(body , "text/html");

//Apply the method that sends the email message
myEmailMessage.sendMessage();
System.log("Sent email to '"+toAddress+"'");
```

## Definir los enlaces de excepciones del ejemplo de flujo de trabajo simple

Los enlaces de excepciones se definen en la pestaña **Esquema** del Editor de flujos de trabajo. Los enlaces de excepciones definen el modo en el que los elementos procesan errores.

Los elementos siguientes del flujo de trabajo devuelven excepciones: startVM, vim3WaitTaskEnd, Send Email y vim3WaitToolsStarted.

### Requisitos previos



Complete las siguientes tareas.

- [Crear el ejemplo de flujo de trabajo simple.](#)
- [Crear el esquema del ejemplo de flujo de trabajo simple.](#)



- Definir los parámetro del ejemplo de flujo de trabajo simple.
- Definir los enlaces de decisiones del ejemplo de flujo de trabajo simple.
- Enlazar los elementos de acción del ejemplo de flujo de trabajo simple.
- Enlazar los elementos de tarea de scripts de ejemplo de flujo de trabajo simple.
- Abra el flujo de trabajo en el editor de flujos de trabajo para editarlo.

#### Procedimiento

- 1 En la pestaña **Esquema**, haga clic en el icono **Editar** () del elemento de acción **startVM**.
- 2 Haga clic en la pestaña **Excepción**.
- 3 Haga clic en el botón **No establecido**.
- 4 Seleccione **errorCode** en la lista propuesta.
- 5 Repita los pasos anteriores para establecer el enlace de excepción en **errorCode** para **vim3WaitTaskEnd** y **vim3WaitToolsStarted**.
- 6 Haga clic en el icono **Editar** () del elemento de tarea de scripts **Send Email**.
- 7 Haga clic en la pestaña **Excepción**.
- 8 Haga clic en el botón **No establecido**.
- 9 Seleccione **emailErrorCode** en la lista propuesta.
- 10 Haga clic en **Guardar** en la parte inferior de la pestaña **Esquema** del Editor de flujos de trabajo.

Ha definido el enlace de excepción de los elementos que devuelven excepciones.

#### Pasos siguientes

Debe establecer las propiedades de lectura y de escritura en los atributos y los parámetros.

## Establecer las propiedades de lectura-escritura para atributos del ejemplo de flujo de trabajo simple

Puede definir si los parámetros y los atributos son constantes o variables de solo lectura. También puede establecer limitaciones en los valores que los usuarios pueden proporcionar para parámetros de entrada.

Establecer determinados parámetros en solo lectura permite que otros desarrolladores adapten el flujo de trabajo o lo modifique sin alterar la función principal del flujo de trabajo.

#### Requisitos previos

Complete las siguientes tareas.

- [Crear el ejemplo de flujo de trabajo simple.](#)
- [Crear el esquema del ejemplo de flujo de trabajo simple.](#)
- [Definir los parámetro del ejemplo de flujo de trabajo simple.](#)

- Definir los enlaces de decisiones del ejemplo de flujo de trabajo simple.
- Enlazar los elementos de acción del ejemplo de flujo de trabajo simple.
- Enlazar los elementos de tarea de scripts de ejemplo de flujo de trabajo simple.
- Definir los enlaces de excepciones del ejemplo de flujo de trabajo simple.
- Abra el flujo de trabajo en el editor de flujos de trabajo para editarlo.

### Procedimiento

- 1 Haga clic en la pestaña **General** en la parte superior del Editor de flujos de trabajo.  
Bajo **Atributos**, hay una lista de todos los atributos definidos, todos ellos provistos de casillas de verificación. Al seleccionar estas casillas de verificación, los atributos se establecen en solo lectura.
- 2 Seleccione las casillas de verificación para que los atributos siguientes sean constantes de solo lectura:
  - progress
  - pollRate
  - timeout
  - smtpHost
  - fromAddress
  - subject

Ha definido los atributos del flujo de trabajo que son constantes y los que son variables.

### Pasos siguientes

Defina las propiedades de parámetro y coloque restricciones en los valores posibles para ese parámetro.

## Definir las propiedades de parámetro del ejemplo de flujo de trabajo simple

Las propiedades de parámetro se pueden establecer en el Editor de flujos de trabajo. Establecer las propiedades de parámetro afecta al comportamiento del parámetro y coloca restricciones en los valores posibles para ese parámetro.

### Requisitos previos

Complete las siguientes tareas.

- [Crear el ejemplo de flujo de trabajo simple.](#)
- [Crear el esquema del ejemplo de flujo de trabajo simple.](#)
- [Definir los parámetro del ejemplo de flujo de trabajo simple.](#)
- [Definir los enlaces de decisiones del ejemplo de flujo de trabajo simple.](#)
- [Enlazar los elementos de acción del ejemplo de flujo de trabajo simple.](#)

- [Enlazar los elementos de tarea de scripts de ejemplo de flujo de trabajo simple.](#)
- [Definir los enlaces de excepciones del ejemplo de flujo de trabajo simple.](#)
- Abra el flujo de trabajo en el editor de flujos de trabajo para editarlo.

### Procedimiento

- 1 Haga clic en la pestaña **Presentación** en el Editor de flujos de trabajo.  
Aparecen los dos parámetros de entrada que ha definido para este flujo de trabajo.
- 2 Haga clic en el parámetro **(VC:VirtualMachine)vm**.
- 3 Añada una descripción en la pestaña **General** en la mitad inferior de la pantalla.  
Por ejemplo, escriba **La máquina virtual que se iniciará**.
- 4 Haga clic en la pestaña **Propiedades** en la mitad inferior de la pantalla.  
En esta pestaña, puede establecer las propiedades del parámetro **(VC:VirtualMachine)vm**.
- 5 Haga clic en el icono **Añadir propiedad** (➤+).
- 6 En la lista de propiedades propuestas, seleccione la propiedad **Entrada obligatoria**; a continuación, haga clic en **Aceptar** y establezca su valor en **Sí**.  
  
Si se habilita esta propiedad, los usuarios no pueden ejecutar el flujo de trabajo Iniciar VM y enviar correo electrónico sin proporcionar una máquina virtual que iniciar.
- 7 Haga clic en el icono **Añadir propiedad** (➤+).
- 8 En la lista de propiedades propuestas, seleccione **Seleccionar valor como**; a continuación, haga clic en **Aceptar** y seleccione **lista** en la lista de valores posibles.  
  
Cuando establece esta propiedad, define el modo en el que el usuario selecciona el valor del parámetro de entrada **(VC:VirtualMachine)vm**.
- 9 Haga clic en el parámetro **(string)toAddress** en la mitad superior de la pestaña **Presentación**.
- 10 Añada una descripción en la pestaña **Descripción** en la mitad inferior de la pantalla.  
Por ejemplo, escriba  
**La dirección de correo electrónico de la persona a la que se enviará la notificación.**
- 11 Haga clic en la pestaña **Propiedades** para **(string)toAddress**; a continuación, haga clic en el icono **Añadir propiedad** (➤+).
- 12 En la lista de propiedades propuestas, seleccione la propiedad **Entrada obligatoria**; a continuación, haga clic en **Aceptar** y establezca su valor en **Sí**.
- 13 Haga clic en el icono **Añadir propiedad** (➤+).

- 14 En la lista de propiedades propuestas, seleccione **Expresión regular coincidente** y haga clic en **Aceptar**.

Esta propiedad permite establecer restricciones en el contenido que los usuarios pueden proporcionar como entrada.

- 15 Haga clic en el cuadro de texto **Valor** para **Expresión regular coincidente** y establezca las restricciones en `[a-zA-Z0-9_%+.]+@[a-zA-Z0-9-]+\.[a-zA-Z]{2,4}`.

Al establecer estas restricciones, la entrada del usuario se limita a los caracteres que son adecuados para direcciones de correo electrónico. Si el usuario intenta utilizar cualquier otro carácter en la dirección de correo electrónico del destinatario cuando inicie el flujo de trabajo, el flujo de trabajo no se inicia.

Ha establecido los dos parámetros como obligatorios, ha definido el modo en el que el usuario puede seleccionar la máquina virtual que iniciar y ha limitado los caracteres que se pueden introducir en la dirección de correo electrónico del destinatario.

### Pasos siguientes

Debe crear el diseño, o la presentación, del cuadro de diálogo de parámetros de entrada en el que los usuarios especifican valores de parámetros de entrada del flujo de trabajo cuando lo ejecutan.

## Establecer el diseño del cuadro de diálogo de parámetros de entrada del ejemplo de flujo de trabajo simple

Cree el diseño o la presentación del cuadro de diálogo de parámetros de entrada en el Editor de flujos de trabajo. El cuadro de diálogo de parámetros de entrada se abre cuando los usuarios ejecutan un flujo de trabajo que necesita la ejecución de parámetros de entrada.

### Requisitos previos

Complete las siguientes tareas.

- [Crear el ejemplo de flujo de trabajo simple.](#)
- [Crear el esquema del ejemplo de flujo de trabajo simple.](#)
- [Definir los parámetro del ejemplo de flujo de trabajo simple.](#)
- [Definir los enlaces de decisiones del ejemplo de flujo de trabajo simple.](#)
- [Enlazar los elementos de acción del ejemplo de flujo de trabajo simple.](#)
- [Enlazar los elementos de tarea de scripts de ejemplo de flujo de trabajo simple.](#)
- [Definir los enlaces de excepciones del ejemplo de flujo de trabajo simple.](#)
- [Establecer las propiedades de lectura-escritura para atributos del ejemplo de flujo de trabajo simple.](#)
- [Definir las propiedades de parámetro del ejemplo de flujo de trabajo simple.](#)
- Abra el flujo de trabajo en el editor de flujos de trabajo para editarlo.

## Procedimiento

- 1 Haga clic en la pestaña **Presentación** en el Editor de flujos de trabajo.
- 2 Haga clic con el botón secundario en el nodo **Presentación** en la lista jerárquica de presentaciones y seleccione **Crear grupo de visualización**.

Aparecen el nodo **Nuevo paso** y un subnodo **Nuevo grupo** bajo el nodo **Presentación**.

- 3 Haga clic con el botón secundario en **Nuevo paso** y seleccione **Eliminar**.

Como este flujo de trabajo tiene solo dos parámetros, no se necesitan varias capas de secciones de visualización en el cuadro de diálogo de parámetros de entrada.

- 4 Haga doble clic en **Nuevo grupo** para editar el nombre del grupo y pulse Entrar.

Por ejemplo, asigne el nombre **Máquina virtual** al grupo de visualización.

El texto que se especifica aquí aparece como encabezado en el cuadro de diálogo de parámetros de entrada cuando se inicia el flujo de trabajo.

- 5 En el cuadro de texto **Descripción** de la pestaña **General** en la parte inferior de la pestaña **Presentación**, escriba una descripción del grupo de visualización nuevo.

Por ejemplo, escriba **Seleccione la máquina virtual que se iniciará**.

El texto que se especifica aquí aparece como solicitud en el cuadro de diálogo de parámetros de entrada cuando se inicia el flujo de trabajo.

- 6 Arrastre el parámetro **(VC:VirtualMachine)vm** al grupo de visualización **Máquina virtual**.

En el cuadro de diálogo de parámetros de entrada, bajo el encabezado Máquina virtual, aparecerá un cuadro de texto en el que el usuario escribe el nombre de la máquina virtual.

- 7 Repita los pasos anteriores para crear un grupo de visualización para el parámetro **toAddress** estableciendo las propiedades siguientes:

- a Cree un grupo de visualización y asígnele el nombre **Dirección de correo electrónico del destinatario**.
- b Añada una descripción para el grupo de visualización, por ejemplo, **Escriba la dirección de correo electrónico de la persona a la que se enviará una notificación cuando se encienda esta máquina virtual**.
- c Arrastre el parámetro **toAddress** al grupo de visualización **Dirección de correo electrónico del destinatario**.

Ha configurado el diseño del cuadro de diálogo de parámetros de entrada que aparece cuando los usuarios ejecutan el flujo de trabajo.

## Pasos siguientes

Ha completado el desarrollo del ejemplo de flujo de trabajo simple. Ahora puede validar y ejecutar el flujo de trabajo.

## Validar y ejecutar el ejemplo de flujo de trabajo simple

Después de crear un flujo de trabajo, puede validarlo para detectar posibles errores. Si el flujo de trabajo no contiene errores, puede ejecutarlo.

### Requisitos previos

Complete las siguientes tareas.

- [Crear el ejemplo de flujo de trabajo simple.](#)
- [Crear el esquema del ejemplo de flujo de trabajo simple.](#)
- [Definir los parámetro del ejemplo de flujo de trabajo simple.](#)
- [Definir los enlaces de decisiones del ejemplo de flujo de trabajo simple.](#)
- [Enlazar los elementos de acción del ejemplo de flujo de trabajo simple.](#)
- [Enlazar los elementos de tarea de scripts de ejemplo de flujo de trabajo simple.](#)
- [Definir los enlaces de excepciones del ejemplo de flujo de trabajo simple.](#)
- [Establecer las propiedades de lectura-escritura para atributos del ejemplo de flujo de trabajo simple.](#)
- [Definir las propiedades de parámetro del ejemplo de flujo de trabajo simple.](#)
- [Establecer el diseño del cuadro de diálogo de parámetros de entrada del ejemplo de flujo de trabajo simple.](#)
- Abra el flujo de trabajo en el editor de flujos de trabajo para editarlo.

### Procedimiento

- 1 Haga clic en **Validar**, en la pestaña **Esquema** del editor de flujos de trabajo.

La herramienta de validación localiza cualquier error en la definición del flujo de trabajo.

- 2 Tras haber eliminado todos los errores, haga clic en **Guardar y cerrar** en la parte inferior del editor de flujos de trabajo.

Vuelva al cliente de Orchestrator.

- 3 Haga clic en la vista **Flujos de trabajo**.

- 4 Seleccione **Ejemplos de flujos de trabajo > Iniciar VM y enviar correo electrónico** en la lista jerárquica de flujos de trabajo.

- 5 Haga clic con el botón secundario en el flujo de trabajo **Iniciar VM y enviar correo electrónico** y seleccione **Iniciar flujo de trabajo**.

El cuadro de diálogo de parámetros de entrada se abre y le pide que especifique una máquina virtual para iniciar y una dirección de correo electrónico a la que enviar notificaciones.

- 6 Seleccione una máquina virtual para iniciar desde el inventario de vCenter Server.
- 7 Indique una dirección de correo electrónico a la que enviar las notificaciones.

- 8 Haga clic en **Enviar** para iniciar el flujo de trabajo.

Aparece un token de flujo de trabajo en el flujo de trabajo Iniciar VM y enviar correo electrónico.

- 9 Haga clic en el token del flujo de trabajo para comprobar el progreso del flujo de trabajo mientras se ejecuta.

Si el flujo de trabajo se ejecuta correctamente, la máquina virtual que seleccionó se encuentra encendida y el destinatario de correo electrónico que definió recibe un correo electrónico de confirmación.

#### Pasos siguientes

Puede generar un documento en el que revisar la información sobre el flujo de trabajo. Consulte [Generar documentación de flujo de trabajo](#).

## Desarrollar un flujo de trabajo complejo

El desarrollo de un flujo de trabajo complejo de ejemplo demuestra los pasos más habituales del proceso de desarrollo de flujos de trabajo, así como opciones más avanzadas, como la creación de decisiones personalizadas y de bucles.

En el ejercicio del flujo de trabajo complejo, se desarrolla un flujo de trabajo que toma una instantánea de todas las máquinas virtuales que se encuentran dentro de un determinado grupo de recursos. El flujo de trabajo que se cree efectuará las tareas siguientes:

- 1 Solicita al usuario un grupo de recursos que contiene máquinas virtuales de las que tomar instantáneas.
- 2 Determina si el grupo de recursos contiene máquinas virtuales en ejecución.
- 3 Determina la cantidad de máquinas virtuales en ejecución que contiene el recurso.
- 4 Verifica si una máquina virtual concreta que se ejecuta en el grupo cumple determinados criterios para tomar una instantánea.
- 5 Toma la instantánea de la máquina virtual.
- 6 Determina si en el grupo de recursos hay más máquinas virtuales de las que tomar instantáneas.
- 7 Repite el proceso de verificación y de instantánea hasta que el flujo de trabajo haya tomado instantáneas de todas las máquinas virtuales válidas del grupo de recursos.

El archivo ZIP de ejemplos de Orchestrator que puede descargar de la página de inicio de la documentación de Orchestrator contiene una versión completada del flujo de trabajo Tomar una instantánea de todas las máquinas virtuales en un grupo de recursos.

## Requisitos previos

Antes de intentar desarrollar este flujo de trabajo complejo, siga los ejercicios de [Desarrollar un flujo de trabajo simple de ejemplo](#). Los procedimientos para desarrollar un flujo de trabajo complejo proporcionan los pasos generales del proceso de desarrollo, pero no son tan detallados como los ejercicios del flujo de trabajo simple.

## Procedimiento

### 1 [Crear el ejemplo de flujo de trabajo complejo](#)

El proceso de desarrollo de un flujo de trabajo empieza con la creación del flujo de trabajo en el cliente de Orchestrator.

### 2 [Crear una acción personalizada para el ejemplo de flujo de trabajo completo](#)

El elemento de tarea de scripts Check VM llama a una acción que no existe en la API de Orchestrator. Debe crear la acción getVMDiskModes.

### 3 [Crear el esquema del ejemplo de flujo de trabajo complejo](#)

Puede crear un esquema de flujo de trabajo en el Editor de flujos de trabajo. El esquema de flujo de trabajo contiene los elementos que ejecuta el flujo de trabajo; asimismo, determina el proceso lógico del flujo de trabajo.

### 4 [\(opcional\) Crear zonas de ejemplo de flujo de trabajo complejo](#)

Opcionalmente, puede resaltar diferentes zonas del flujo de trabajo añadiendo notas de flujo de trabajo. La creación de diferentes zonas de flujo de trabajo permite facilitar la lectura y la comprensión de esquemas de flujo de trabajo complicados.

### 5 [Definir los parámetros del ejemplo de flujo de trabajo complejo](#)

Defina los parámetros de flujo de trabajo en el Editor de flujos de trabajo. Los parámetros de entrada proporcionan datos para que los procese el flujo de trabajo. Los parámetros de salida son los datos devueltos por el flujo de trabajo al término de su ejecución.

### 6 [Definir los enlaces para el ejemplo de flujo de trabajo complejo](#)

Los elementos de un flujo de trabajo se pueden enlazar en el Editor de flujos de trabajo. Los enlaces definen el flujo de datos del flujo de trabajo. Los elementos de tarea de scripts también se pueden enlazar a sus funciones de JavaScript.

### 7 [Establecer las propiedades de los atributos del ejemplo de flujo de trabajo complejo](#)

Las propiedades de los atributos se establecen en la pestaña **General** en el Editor de flujos de trabajo.

### 8 [Crear el diseño de los parámetros de entrada del ejemplo de flujo de trabajo complejo](#)

Cree el diseño o la presentación del cuadro de diálogo de parámetros de entrada en la pestaña **Presentación** del Editor de flujos de trabajo. El cuadro de diálogo de parámetros de entrada se abre cuando los usuarios ejecutan un flujo de trabajo, con lo cual los usuarios pueden indicar los parámetros de entrada con los que se ejecuta el flujo de trabajo.



## 9 Validar y ejecutar el ejemplo de flujo de trabajo complejo

Después de crear un flujo de trabajo, puede validarlo para detectar posibles errores. Si el flujo de trabajo no contiene errores, puede ejecutarlo.

## Crear el ejemplo de flujo de trabajo complejo

El proceso de desarrollo de un flujo de trabajo empieza con la creación del flujo de trabajo en el cliente de Orchestrator.

Para obtener información sobre cómo instalar y configurar vCenter Server, consulte la documentación *Guía de instalación y configuración de vSphere*. Para obtener información sobre cómo configurar Orchestrator, consulte *Instalación y configuración de VMware vRealize Orchestrator*.

### Requisitos previos

Verifique que los componentes siguientes estén instalados y configurados en el sistema.

- vCenter Server, que controla un grupo de recursos que contiene algunas máquinas virtuales.
- La carpeta Workflow Examples en la lista jerárquica de flujos de trabajo, que creó en [Crear el ejemplo de flujo de trabajo simple](#).

### Procedimiento

- 1 Seleccione **Diseño** en el menú desplegable del cliente de Orchestrator.
- 2 Seleccione **Flujos de trabajo > Ejemplos de flujo de trabajo**.
- 3 Haga clic con el botón secundario en la carpeta **Ejemplos de flujo de trabajo** y seleccione **Nuevo flujo de trabajo**.
- 4 Asigne el nombre **Tomar una instantánea de todas las máquinas virtuales en un grupo de recursos** al flujo de trabajo y haga clic en **Aceptar**.  
Se abre el Editor de flujos de trabajo.
- 5 En la pestaña **General** del Editor de flujos de trabajo, haga clic en los dígitos del número de versión para incrementar el número de versión.  
Para la creación inicial del flujo de trabajo, establezca la versión en **0.0.1**.
- 6 Haga clic en el valor **Comportamiento de reinicio del servidor** para establecer si el flujo de trabajo se reanuda tras el reinicio del servidor.
- 7 En el cuadro de texto **Descripción**, describa lo que hace el flujo de trabajo.
- 8 Haga clic en **Guardar** en la parte inferior de la pestaña **General**.

Ha creado el flujo de trabajo Tomar una instantánea de todas las máquinas virtuales en un grupo de recursos.

### Pasos siguientes

Debe crear una acción personalizada.

## Crear una acción personalizada para el ejemplo de flujo de trabajo completo

El elemento de tarea de scripts `Check VM` llama a una acción que no existe en la API de Orchestrator. Debe crear la acción `getVMDiskModes`.

Para obtener más información sobre la creación de acciones, consulte [Capítulo 3 Desarrollar acciones](#).

### Requisitos previos

Cree el flujo de trabajo `Tomar una instantánea de todas las máquinas virtuales en un grupo de recursos`. Consulte [Crear el ejemplo de flujo de trabajo complejo](#).

### Procedimiento

- 1 Cierre el Editor de flujos de trabajo haciendo clic en **Guardar y cerrar**.
- 2 Haga clic en la vista **Acciones** en el cliente de Orchestrator.
- 3 Haga clic con el botón secundario en la raíz de la lista jerárquica de acciones y seleccione **Nuevo módulo**.
- 4 Asigne al nuevo módulo el nombre `com.vmware.ejemplo`.
- 5 Haga clic con el botón secundario en el módulo `com.vmware.ejemplo` y seleccione **Añadir acción**.
- 6 Cree una acción denominada `getVMDiskModes`.
- 7 Aumente el número de versión en la pestaña **General** en el editor de acciones haciendo clic en los dígitos de la versión.
- 8 Añada la descripción siguiente de la acción en la pestaña **General**.

```
This action returns an array containing the disk modes of all disks on a VM.
The elements in the array each have one of the following string values:
- persistent
- independent-persistent
- nonpersistent
- independent-nonpersistent
Legacy values:
- undoable
- append
```

- 9 Haga clic en la pestaña **Creación de scripts**.
- 10 Haga clic con el botón secundario en el panel superior de la pestaña **Creación de scripts**; a continuación, seleccione **Añadir parámetro** para crear el parámetro de entrada siguiente.
  - Nombre: `vm`
  - Tipo: `VC:VirtualMachine`
  - Descripción: **La máquina virtual para la cual se devuelven los modos de disco**

## 11 Añada el siguiente script en la parte inferior de la pestaña **Creación de scripts**.

El código siguiente devuelve una matriz de modos de disco para los discos de la máquina virtual.

```
var devicesArray = vm.config.hardware.device;
var retArray = new Array();
if (devicesArray!=null && devicesArray.length!=0) {
    for (i in devicesArray) {
        if (devicesArray[i] instanceof VcVirtualDisk) {
            retArray.push(devicesArray[i].backing.diskMode);
        }
    }
}
return retArray;
```

## 12 Haga clic en **Guardar y cerrar** para salir de la paleta **Acciones**.

Ha definido la acción personalizada que requiere el flujo de trabajo Tomar una instantánea de todas las máquinas virtuales en un grupo de recursos.

### Pasos siguientes

Cree el esquema del flujo de trabajo.

## Crear el esquema del ejemplo de flujo de trabajo complejo

Puede crear un esquema de flujo de trabajo en el Editor de flujos de trabajo. El esquema de flujo de trabajo contiene los elementos que ejecuta el flujo de trabajo; asimismo, determina el proceso lógico del flujo de trabajo.

### Requisitos previos

Complete las siguientes tareas.

- [Crear el ejemplo de flujo de trabajo complejo.](#)
- [Crear una acción personalizada para el ejemplo de flujo de trabajo completo.](#)
- Abra el flujo de trabajo en el editor de flujos de trabajo para editarlo.

### Procedimiento

- 1 Haga clic en la pestaña **Esquema** en el Editor de flujos de trabajo.
- 2 Añada los elementos de esquema siguientes al esquema de flujo de trabajo.

Tipo de elemento	Nombre de elemento	Posición en esquema
Tarea de scripts	<b>Initializing</b>	Bajo el elemento Start
Decisión	<b>VMs to Process?</b>	Bajo el elemento de tarea de scripts Initializing
Tarea de scripts	<b>Pool Has No VMs</b>	Bajo el elemento de decisión personalizado VMs to Process?, vinculado con una flecha roja
Decisión personalizada	<b>Remaining VMs?</b>	A la derecha del elemento de decisión personalizado VMs to Process?, vinculado con una flecha verde

Tipo de elemento	Nombre de elemento	Posición en esquema
Acción	getVMDiskModes	A la derecha del elemento de decisión personalizado Remaining VMs?, vinculado con una flecha verde
Decisión personalizada	<b>Create Snapshot?</b>	A la derecha del elemento de acción getVMDiskModes, vinculado con una flecha azul
Flujo de trabajo	Create a snapshot	Encima del elemento de decisión personalizado Create Snapshot?, vinculado con una flecha verde
Tarea de scripts	<b>VM Snapshots</b>	A la izquierda del flujo de trabajo Create a snapshot, vinculado con una flecha azul
Tarea de scripts	<b>Increment</b>	A la izquierda del elemento de tarea de scripts VM Snapshots, vinculado con una flecha azul
Tarea de scripts	<b>Set Output</b>	A la derecha del elemento de tarea de scripts Pool Has No VMs, vinculado con una flecha azul

### 3 Añada un elemento de tarea de scripts Log Exception.

- Cree un vínculo de control de excepciones con el flujo de trabajo Crear una instantánea a un elemento End.
- Arrastre un elemento de tarea de scripts a la flecha discontinua roja que vincula el flujo de trabajo Crear una instantánea con un elemento End.
- Haga doble clic en el elemento de tarea de scripts y cámbiele el nombre a **Excepción de registro**.
- Mueva el elemento de tarea de scripts Log Exception encima del elemento de tarea de scripts VM Snapshots.

### 4 Desvincule todos los elementos End, excepto el elemento End que está a la derecha del elemento de tarea de scripts Set Output.

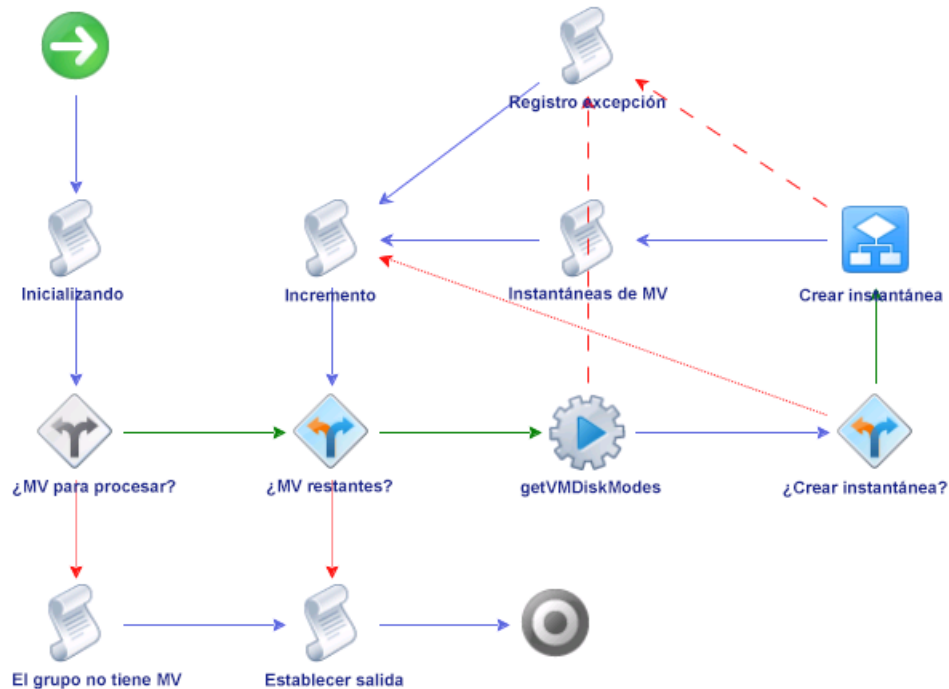
### 5 Vincule los demás elementos como se describe en la tabla siguiente.

Elemento	Vincular a	Tipo de flecha	Descripción
Elemento de acción getVMDiskModes	Elemento de tarea de scripts Log Exception	Discontinua roja	Control de excepciones
Elemento de decisión personalizado Create Snapshot?	Elemento de tarea de scripts Increment	Roja	Resultado False
Elemento de tarea de scripts Log Exception	Elemento de tarea de scripts Increment	Azul	Progreso de flujo de trabajo normal
Elemento de tarea de scripts Increment	Elemento de decisión personalizado Remaining VMs?	Azul	Progreso de flujo de trabajo normal
Elemento de decisión personalizado Remaining VMs?	Elemento de tarea de scripts Set Output	Roja	Resultado False

### 6 Haga clic en **Guardar** en la parte inferior de la pestaña **Esquema**.

En la figura siguiente, se muestra cómo deben ser los elementos vinculados del flujo de trabajo Tomar una instantánea de todas las máquinas virtuales en un grupo de recursos.

**Figura 1-12. Vincular el flujo de trabajo de ejemplo Tomar una instantánea de todas las máquinas virtuales en un grupo de recursos**



### Pasos siguientes

También puede definir zonas de flujo de trabajo mediante notas de flujo de trabajo.

## Crear zonas de ejemplo de flujo de trabajo complejo

Opcionalmente, puede resaltar diferentes zonas del flujo de trabajo añadiendo notas de flujo de trabajo. La creación de diferentes zonas de flujo de trabajo permite facilitar la lectura y la comprensión de esquemas de flujo de trabajo complicados.

### Requisitos previos

Complete las siguientes tareas.

- [Crear el ejemplo de flujo de trabajo complejo.](#)
- [Crear el esquema del ejemplo de flujo de trabajo complejo.](#)
- Abra el flujo de trabajo en el editor de flujos de trabajo para editarlo.

## Procedimiento

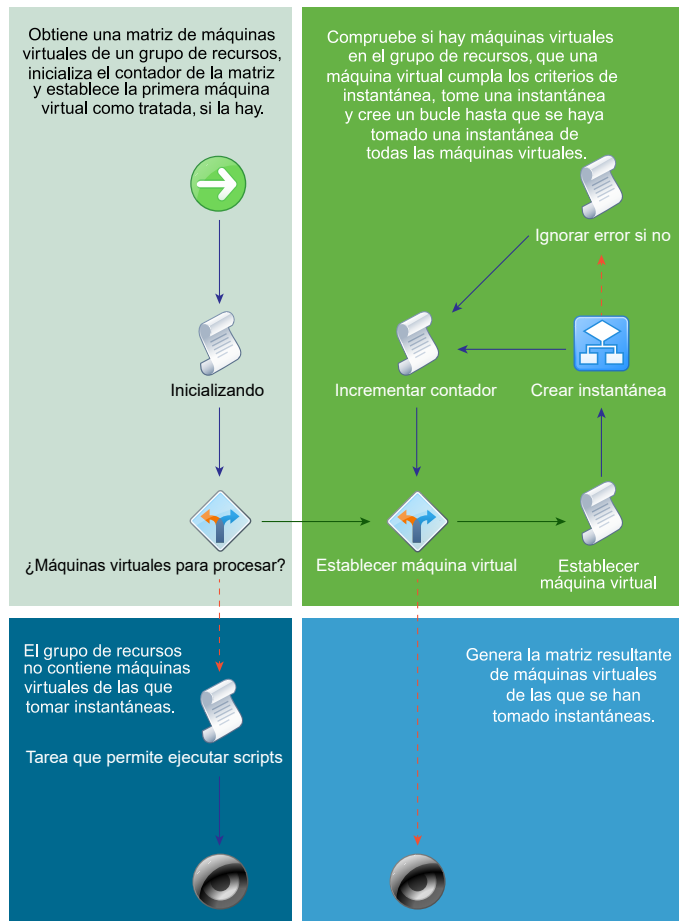
- 1 Cree las zonas de flujo de trabajo siguientes mediante notas de flujo de trabajo.

Elementos en zona	Descripción
Elemento Iniciar; Inicializar tarea de scripts; elemento de decisión ¿VM para procesar?	Obtiene una matriz de máquinas virtuales de un grupo de recursos, inicializa el contador de la matriz y establece la primera máquina virtual como tratada, si la hay.
El grupo no tiene tareas de scripts de VM.	El grupo de recursos no contiene máquinas virtuales de las que tomar instantáneas.
Elemento de decisión ¿VM restantes?; acción getVMDisksModes, decisión ¿Crear instantánea?; flujo de trabajo Crear una instantánea; tarea de scripts Instantáneas de VM; Tarea de scripts de incremento; Tarea de scripts de excepción de registro	Compruebe si hay máquinas virtuales en el grupo de recursos, que una máquina virtual cumpla los criterios de instantánea, tome una instantánea y cree un bucle hasta que se haya tomado una instantánea de todas las máquinas virtuales.
Establecer tarea de scripts de salida; elemento Fin	Genera la matriz resultante de máquinas virtuales de las que se han tomado instantáneas.

- 2 Seleccione una nota de flujo de trabajo y pulse Ctrl+E para seleccionar el color de fondo.
- 3 Haga clic en **Guardar** en la parte inferior de la pestaña **Esquema** del Editor de flujos de trabajo.

Las zonas de flujo de trabajo tienen el aspecto que se muestra en el diagrama siguiente.

**Figura 1-13. Diagrama de esquema para el flujo de trabajo de ejemplo Tomar una instantánea de todas las máquinas virtuales en un grupo de recursos**



### Pasos siguientes

Debe definir los parámetros de entrada y de salida del flujo de trabajo.

## Definir los parámetros del ejemplo de flujo de trabajo complejo

Defina los parámetros de flujo de trabajo en el Editor de flujos de trabajo. Los parámetros de entrada proporcionan datos para que los procese el flujo de trabajo. Los parámetros de salida son los datos devueltos por el flujo de trabajo al término de su ejecución.

### Requisitos previos

Complete las siguientes tareas.

- [Crear el ejemplo de flujo de trabajo complejo.](#)
- [Crear el esquema del ejemplo de flujo de trabajo complejo.](#)
- Abra el flujo de trabajo en el editor de flujos de trabajo para editarlo.

**Procedimiento**

- 1 Haga clic en la pestaña **Entradas** en el Editor de flujos de trabajo.
- 2 Defina el parámetro de entrada siguiente .
  - Nombre: resourcePool
  - Tipo: VC:ResourcePool
  - Descripción:  
El grupo de recursos que contiene las máquinas virtuales de las que tomar instantáneas..
- 3 Haga clic en la pestaña **Salidas** en el Editor de flujos de trabajo.
- 4 Defina el parámetro de salida siguiente.
  - Nombre: snapshotVmArrayOut
  - Tipo: Array/VC:VirtualMachine
  - Descripción:  
La matriz de máquinas virtuales de la que se han tomado las instantáneas.

Ha definido los parámetros de entrada y salida del flujo de trabajo.

**Pasos siguientes**

Debe definir los enlaces entre los parámetros del elemento.

**Definir los enlaces para el ejemplo de flujo de trabajo complejo**

Los elementos de un flujo de trabajo se pueden enlazar en el Editor de flujos de trabajo. Los enlaces definen el flujo de datos del flujo de trabajo. Los elementos de tarea de scripts también se pueden enlazar a sus funciones de JavaScript.

**Requisitos previos**

Complete las siguientes tareas.

- [Crear el ejemplo de flujo de trabajo complejo.](#)
- [Crear el esquema del ejemplo de flujo de trabajo complejo](#)
- [Definir los parámetros del ejemplo de flujo de trabajo complejo](#)
- Revise los enlaces que debe definir. Consulte [Enlaces de ejemplo de flujos de trabajo complejos.](#)
- Abra el flujo de trabajo en el editor de flujos de trabajo para editarlo.

**Procedimiento**

- 1 Haga clic en la pestaña **Esquema** en el Editor de flujos de trabajo.
- 2 Defina los enlaces.
- 3 Haga clic en **Guardar** en la parte inferior de la pestaña **Esquema**.



Todos los parámetros de entrada y de salida de los elementos se enlazan a los correspondientes valores y tipos de parámetros.

### Pasos siguientes

Establezca las propiedades de los atributos.

## Enlaces de ejemplo de flujos de trabajo complejos

Los enlaces definen el modo en el que los elementos de acción del ejemplo de flujo de trabajo simple procesan parámetros de entrada y de salida.

El flujo de trabajo Tomar una instantánea de todas las máquinas virtuales en un flujo de trabajo de grupo de recursos requiere los parámetros siguientes de entrada y de salida. También define las funciones de JavaScript para los elementos de tarea de scripts.

En los casos en que enlace a parámetros existentes, el enlace hereda los valores de tipo y de descripción del parámetro original.

### Inicializar tarea de scripts

El elemento Inicializar tarea de scripts inicializa los atributos del flujo de trabajo. En la tabla siguiente, figuran los enlaces de parámetros de entrada y de salida requeridos por el elemento Inicializar tarea de scripts.

**Tabla 1-61. Enlaces del elemento Inicializar tarea de scripts**

Nombre del parámetro	Tipo de enlace	¿Enlazar con parámetro existente o crear parámetro?	Valores de enlace
resourcePool	IN	Enlazar	<ul style="list-style-type: none"> <li>■ Parámetro local: resourcePool</li> <li>■ Parámetro de origen: resourcePool[in-parameter]</li> <li>■ Tipo: VC:ResourcePool</li> <li>■ Descripción: <b>El grupo de recursos que contiene las máquinas virtuales de las que tomar instantáneas</b></li> </ul>
allVMs	OUT	Crear	<ul style="list-style-type: none"> <li>■ Parámetro local: allVMs</li> <li>■ Parámetro de origen: allVMs[attribute]</li> <li>■ Tipo: Array/VC:VirtualMachine</li> <li>■ Descripción: <b>Las máquinas virtuales en el grupo de recursos.</b></li> </ul>

Tabla 1-61. Enlaces del elemento Inicializar tarea de scripts (continuación)

Nombre del parámetro	Tipo de enlace	¿Enlazar con parámetro existente o crear parámetro?	Valores de enlace
numberOfVms	OUT	Crear	<ul style="list-style-type: none"> <li>■ Parámetro local: numberOfVms</li> <li>■ Parámetro de origen: numberOfVms[attribute]</li> <li>■ Tipo: número</li> <li>■ Descripción: <b>El número de máquinas virtuales encontradas en el grupo de recursos</b></li> </ul>
vmCounter	OUT	Crear	<ul style="list-style-type: none"> <li>■ Parámetro local: vmCounter</li> <li>■ Parámetro de origen: vmCounter[attribute]</li> <li>■ Tipo: número</li> <li>■ Descripción: <b>El recuento de máquinas virtuales dentro de la matriz</b></li> </ul>
vm	OUT	Crear	<ul style="list-style-type: none"> <li>■ Parámetro local: vm</li> <li>■ Parámetro de origen: vm[attribute]</li> <li>■ Tipo: VC:VirtualMachine</li> <li>■ Descripción: <b>La máquina virtual actual de la que se toma una instantánea</b></li> </ul>
snapshotVmArray	OUT	Crear	<ul style="list-style-type: none"> <li>■ Parámetro local: snapshotVmArray</li> <li>■ Parámetro de origen: snapshotVmArray[attribute]</li> <li>■ Tipo: Array/VC:VirtualMachine</li> <li>■ Descripción: <b>La matriz de máquinas virtuales de la que se han tomado las instantáneas</b></li> </ul>

El elemento Inicializar tarea de scripts efectúa la función con scripts siguiente.

```
//Retrieve an array of virtual machines contained in the specified Resource Pool
allVms = resourcePool.vm;
//Initialize the size of the Array and the first VM to snapshot
if (allVms!=null && allVms.length!=0) {
    numberOfVms = allVms.length;
    vm = allVms[0];
} else {
    numberOfVms = 0;
}
//Initialize the VM counter
vmCounter = 0;
```

```
//Initializing the array of VM snapshots
snapshotVmArray = new Array();
```

### Elemento de decisión ¿VM para procesar?

El elemento de decisión ¿VM para procesar? determina si en el grupo de recursos hay máquinas virtuales de las que tomar instantáneas. La tabla siguiente contiene los enlaces requeridos por el elemento de decisión ¿VM para procesar?

**Tabla 1-62. Elemento de decisión ¿VM para procesar?**

Nombre del parámetro	Tipo de enlace	¿Enlazar con parámetro existente o crear parámetro?	Valores de enlace
numberOfVms	Decisión	Enlazar	<ul style="list-style-type: none"> <li>■ Parámetro de origen: numberOfVms[attribute]</li> <li>■ Instrucción de decisión: mayor que</li> <li>■ Valor: 0.0</li> <li>■ Descripción: <b>El número de máquinas virtuales encontradas en el grupo de recursos</b></li> </ul>

### El grupo no tiene elemento de tarea de scripts de VM

El grupo no tiene elemento de tarea de scripts de VM registra el hecho de que el grupo de recursos no contiene máquinas virtuales válidas en la base de datos de Orchestrator. La tabla siguiente contiene los enlaces requeridos por El grupo no tiene elemento de tarea de scripts de VM.

**Tabla 1-63. Enlaces de El grupo no tiene elemento de tarea de scripts de VM**

Nombre del parámetro	Tipo de enlace	¿Enlazar con parámetro existente o crear parámetro?	Valores de enlace
resourcePool	IN	Enlazar	<ul style="list-style-type: none"> <li>■ Parámetro local: resourcePool</li> <li>■ Parámetro de origen: resourcePool[in-parameter]</li> <li>■ Tipo: VC:ResourcePool</li> <li>■ Descripción: <b>El grupo de recursos que contiene las máquinas virtuales de las que tomar instantáneas..</b></li> </ul>

El grupo no tiene elemento de tarea de scripts de VM efectúa la función de script siguiente.

```
//Writes the following event in the Orchestrator database
Server.warn("The specified ResourcePool "+resourcePool.name+" does not contain any VMs.");
```

## Elemento de decisión personalizada ¿VM restantes?

El elemento de decisión personalizada ¿VM restantes? determina si cualquier máquina virtual de la que tomar instantáneas se mantiene en el grupo de recursos. La tabla siguiente contiene los enlaces requeridos por el elemento de decisión personalizada ¿VM restantes?

**Tabla 1-64. Elemento de decisión personalizada ¿VM restantes?**

Nombre del parámetro	Tipo de enlace	¿Enlazar con parámetro existente o crear parámetro?	Valores de enlace
numberOfVMs	IN	Enlazar	<ul style="list-style-type: none"> <li>■ Parámetro de origen: numberOfVMs[attribute]</li> <li>■ Instrucción de decisión: mayor que</li> <li>■ Valor: 0.0</li> <li>■ Descripción: <b>El número de máquinas virtuales encontradas en el grupo de recursos</b></li> </ul>
vmCounter	IN	Enlazar	<ul style="list-style-type: none"> <li>■ Parámetro local: vmCounter</li> <li>■ Parámetro de origen: vmCounter[attribute]</li> <li>■ Tipo: número</li> <li>■ Descripción: <b>El recuento de máquinas virtuales dentro de la matriz</b></li> </ul>

El elemento de decisión personalizada ¿VM restantes? efectúa la función con scripts siguiente.

```
//Checks if the workflow has reached the end of the array of VMs
if (vmCounter < numberOfVMs) {
    return true;
} else {
    return false;
}
```

## Elemento de acción getVMDisksModes

El elemento de acción getVMDisksModes obtiene los modos de los discos que se ejecutan en una máquina virtual. La tabla siguiente contiene los enlaces requeridos por el elemento de acción getVMDisksModes.

Tabla 1-65. Enlaces del elemento de acción getVMDisksModes

Nombre del parámetro	Tipo de enlace	¿Enlazar con parámetro existente o crear parámetro?	Valores de enlace
vm	IN	Enlazar	<ul style="list-style-type: none"> <li>■ Parámetro local: vm</li> <li>■ Parámetro de origen: vm[attribute]</li> <li>■ Tipo: VC:VirtualMachine</li> <li>■ Descripción: <b>La máquina virtual actual de la que se toma una instantánea</b></li> </ul>
actionResult	OUT	Crear	<ul style="list-style-type: none"> <li>■ Parámetro local: actionResult</li> <li>■ Parámetro de origen: vmDisksModes[attribute]</li> <li>■ Tipo: matriz/cadena</li> <li>■ Descripción: <b>Los modos de disco actuales de la máquina virtual</b></li> </ul>
errorCode	Excepción	Crear	Parámetro local: errorCode

### Elemento de decisión personalizada ¿Crear instantánea?

El elemento de decisión personalizada ¿Crear instantánea? determina si se toman instantáneas de máquinas virtuales, en función de los modos de disco de las máquinas virtuales. La tabla siguiente contiene los enlaces requeridos por el elemento de decisión personalizada ¿Crear instantánea?

Tabla 1-66. Enlaces elemento de decisión personalizada ¿Crear instantánea?

Nombre del parámetro	Tipo de enlace	¿Enlazar con parámetro existente o crear parámetro?	Valores de enlace
vmDisksMode	IN	Enlazar	<ul style="list-style-type: none"> <li>■ Parámetro local: vmDisksMode</li> <li>■ Parámetro de origen: vmDisksMode[attribute]</li> <li>■ Tipo: matriz/cadena</li> <li>■ Descripción: <b>Los modos de disco actuales de la máquina virtual</b></li> </ul>
vm	IN	Enlazar	<ul style="list-style-type: none"> <li>■ Parámetro local: vm</li> <li>■ Parámetro de origen: vm[attribute]</li> <li>■ Tipo: VC:VirtualMachine</li> <li>■ Descripción: <b>La máquina virtual actual de la que se toma una instantánea</b></li> </ul>

El elemento de decisión personalizada ¿Crear instantánea? efectúa la función con scripts siguiente.

```
//A snapshot cannot be taken if one of its disks is in independent mode
// (independent-persistent or independent-nonpersistent)
var containsIndependentDisks = false;
if (vmDisksModes!=null && vmDisksModes.length>0) {
    for (i in vmDisksModes) {
        if (vmDisksModes[i].charAt(0)=="i") {
            containsIndependentDisks = true;
        }
    }
} else {
    //if no disk found no need to try to snapshot the VM
    System.warn("Won't snapshot '"+vm.name+"', no disks found");
    return false;
}
if (containsIndependentDisks) {
    System.warn("Won't snapshot '"+vm.name+"', independent disk(s) found");
    return false;
} else {
    System.log("Snapshotting '"+vm.name+"'");
    return true;
}
```

### Elemento de flujo de trabajo Crear una instantánea

El elemento de flujo de trabajo Crear una instantánea toma instantáneas de máquinas virtuales. La tabla siguiente contiene los enlaces requeridos por el elemento de flujo de trabajo Crear una instantánea.

**Tabla 1-67. Enlaces del elemento de flujo de trabajo Crear una instantánea**

Nombre del parámetro	Tipo de enlace	¿Enlazar con parámetro existente o crear parámetro?	Valores de enlace
vm	IN	Enlazar	<ul style="list-style-type: none"> <li>■ Parámetro local: vm</li> <li>■ Parámetro de origen: vm[attribute]</li> <li>■ Tipo: VC:VirtualMachine</li> <li>■ Descripción: <b>Una máquina virtual activa de la que se toma una instantánea.</b></li> </ul>
name	IN	Crear	<ul style="list-style-type: none"> <li>■ Parámetro local: name</li> <li>■ Parámetro de origen: snapshotName[attribute]</li> <li>■ Tipo: cadena</li> <li>■ Descripción: <b>El nombre de esta instantánea. No es necesario que el nombre sea exclusivo para esta máquina virtual.</b></li> </ul>

Tabla 1-67. Enlaces del elemento de flujo de trabajo Crear una instantánea (continuación)

Nombre del parámetro	Tipo de enlace	¿Enlazar con parámetro existente o crear parámetro?	Valores de enlace
description	IN	Crear	<ul style="list-style-type: none"> <li>■ Parámetro local: description</li> <li>■ Parámetro de origen: snapshotDescription[attribute]</li> <li>■ Tipo: cadena</li> <li>■ Descripción: <b>Una descripción para esta instantánea.</b></li> </ul>
memory	IN	Crear	<ul style="list-style-type: none"> <li>■ Parámetro local: memory</li> <li>■ Parámetro de origen: snapshotMemory[attribute]</li> <li>■ Tipo: booleano</li> <li>■ Valor: no</li> <li>■ Descripción: <b>Si es TRUE, se incluye en la instantánea un volcado del estado interno de la máquina virtual (un volcado de memoria).</b></li> </ul>
quiesce	IN	Crear	<ul style="list-style-type: none"> <li>■ Parámetro local: quiesce</li> <li>■ Parámetro de origen: snapshotQuiesce[attribute]</li> <li>■ Tipo: booleano</li> <li>■ Valor: sí</li> <li>■ Descripción: <b>Si es TRUE y la máquina virtual está encendida cuando se toma la instantánea, VMware Tools se utiliza para poner en modo inactivo el sistema de archivos en la máquina virtual.</b></li> </ul>
snapshot	OUT	Crear	<ul style="list-style-type: none"> <li>■ Parámetro local: snapshot</li> <li>■ Parámetro de origen: nulo</li> <li>■ Tipo: VC:VirtualMachineSnapshot</li> <li>■ Descripción: <b>La instantánea que se toma.</b></li> </ul>
errorCode	Excepción	Crear	Parámetro local: errorCode

### Elemento Tareas de scripts de VM

El elemento Tareas de scripts de VM añade las instantáneas a una matriz. La tabla siguiente contiene los enlaces requeridos por el elemento Tareas de scripts de VM.

Tabla 1-68. Enlaces del elemento Tareas de scripts de VM

Nombre del parámetro	Tipo de enlace	¿Enlazar con parámetro existente o crear parámetro?	Valores de enlace
vm	IN	Enlazar	<ul style="list-style-type: none"> <li>■ Parámetro local: vm</li> <li>■ Parámetro de origen: vm[attribute]</li> <li>■ Tipo: VC:VirtualMachine</li> <li>■ Descripción: <b>Una máquina virtual activa de la que se toma una instantánea.</b></li> </ul>
snapshotVmArray	IN	Enlazar	<ul style="list-style-type: none"> <li>■ Parámetro local: snapshotVmArray</li> <li>■ Parámetro de origen: snapshotVmArray[attribute]</li> <li>■ Tipo: Array/VC:VirtualMachine</li> <li>■ Descripción: <b>La matriz de máquinas virtuales de la que se han tomado las instantáneas</b></li> </ul>
snapshotVmArray	OUT	Enlazar	<ul style="list-style-type: none"> <li>■ Parámetro local: snapshotVmArray</li> <li>■ Parámetro de origen: snapshotVmArray[attribute]</li> <li>■ Tipo: Array/VC:VirtualMachine</li> <li>■ Descripción: <b>La matriz de máquinas virtuales de la que se han tomado las instantáneas</b></li> </ul>

El elemento Tareas de scripts de VM efectúa la función con scripts siguiente.

```
//Writes the following event in the Orchestrator database
Server.log("Successfully took snapshot of the VM '"+vm.name);
//Inserts the VM snapshot in an array
snapshotVmArray.push(vm);
```

### Elemento Tarea de scripts de incremento

El elemento Tarea de scripts de incremento incrementa el contador que recuenta el número de máquinas virtuales de la matriz. La tabla siguiente contiene los enlaces requeridos por el elemento Tarea de scripts de incremento.



Tabla 1-69. Enlaces del elemento Tarea de scripts de incremento

Nombre del parámetro	Tipo de enlace	¿Enlazar con parámetro existente o crear parámetro?	Valores de enlace
vmCounter	IN	Enlazar	<ul style="list-style-type: none"> <li>■ Parámetro local: vmCounter</li> <li>■ Parámetro de origen: vmCounter[attribute]</li> <li>■ Tipo: número</li> <li>■ Descripción: <b>El recuento de máquinas virtuales dentro de la matriz</b></li> </ul>
allVMs	IN	Enlazar	<ul style="list-style-type: none"> <li>■ Parámetro local: allVMs</li> <li>■ Parámetro de origen: allVMs[attribute]</li> <li>■ Tipo: Array/VC:VirtualMachine</li> <li>■ Descripción: <b>Las máquinas virtuales en el grupo de recursos.</b></li> </ul>
vmCounter	OUT	Enlazar	<ul style="list-style-type: none"> <li>■ Parámetro local: vmCounter</li> <li>■ Parámetro de origen: vmCounter[attribute]</li> <li>■ Tipo: número</li> <li>■ Descripción: <b>El recuento de máquinas virtuales dentro de la matriz</b></li> </ul>
vm	OUT	Enlazar	<ul style="list-style-type: none"> <li>■ Parámetro local: vm</li> <li>■ Parámetro de origen: vm[attribute]</li> <li>■ Tipo: VC:VirtualMachine</li> <li>■ Descripción: <b>La máquina virtual actual de la que se toma una instantánea</b></li> </ul>

El elemento Tarea de scripts de incremento efectúa la función con scripts siguiente.

```
//Increases the array VM counter
vmCounter++;
//Sets the next VM to be snapshot in the attribute vm
vm = allVMs[vmCounter];
```

### Elemento Tarea de scripts de excepción de registro

El elemento Tarea de scripts de excepción de registro controla las excepciones en los elementos de flujo de trabajo y de acción. La tabla siguiente contiene los enlaces requeridos por el elemento Tarea de scripts de excepción de registro.

Tabla 1-70. Enlaces del elemento Tarea de scripts de excepción de registro

Nombre del parámetro	Tipo de enlace	¿Enlazar con parámetro existente o crear parámetro?	Valores de enlace
vm	IN	Enlazar	<ul style="list-style-type: none"> <li>■ Parámetro local: vm</li> <li>■ Parámetro de origen: vm[attribute]</li> <li>■ Tipo: VC:VirtualMachine</li> <li>■ Descripción: <b>La máquina virtual actual de la que se toma una instantánea</b></li> </ul>
errorCode	IN	Enlazar	<ul style="list-style-type: none"> <li>■ Parámetro local: errorCode</li> <li>■ Parámetro de origen: errorCode[attribute]</li> <li>■ Tipo: cadena</li> <li>■ Descripción: <b>Una excepción detectada al tomar una instantánea de una máquina virtual</b></li> </ul>

El elemento Tarea de scripts de excepción de registro efectúa la función con scripts siguiente.

```
//Writes the following event in the Orchestrator database
Server.error("Coudln't snapshot the VM '"+vm.name+"', exception: "+errorCode);
```

### Elemento Establecer tarea de scripts de salida

Establecer tarea de scripts de salida genera el parámetro de salida del flujo de trabajo, que contiene la matriz de máquinas virtuales de las que se han tomado instantáneas. La tabla siguiente contiene los enlaces requeridos por el elemento Establecer tarea de scripts de salida.

Tabla 1-71. Enlaces del elemento Establecer tarea de scripts de salida

Nombre del parámetro	Tipo de enlace	¿Enlazar con parámetro existente o crear parámetro?	Valores de enlace
snapshotVmArray	IN	Enlazar	<ul style="list-style-type: none"> <li>■ Parámetro local: snapshotVmArray</li> <li>■ Parámetro de origen: snapshotVmArray[attribute]</li> <li>■ Tipo: Array/VC:VirtualMachine</li> <li>■ Descripción: <b>La matriz de máquinas virtuales de la que se han tomado las instantáneas</b></li> </ul>
snapshotVmArrayOut	OUT	Enlazar	<ul style="list-style-type: none"> <li>■ Parámetro local: snapshotVmArrayOut</li> <li>■ Parámetro de origen: snapshotVmArrayOut[out-parameter]</li> <li>■ Tipo: Array/VC:VirtualMachine</li> <li>■ Descripción: <b>La matriz de máquinas virtuales de la que se han tomado instantáneas.</b></li> </ul>

El elemento Establecer tarea de scripts de salida efectúa la función con scripts siguiente.

```
//Passes the value of the internal attribute to a workflow output parameter
snapshotVmArrayOut = snapshotVmArray;
```

## Establecer las propiedades de los atributos del ejemplo de flujo de trabajo complejo

Las propiedades de los atributos se establecen en la pestaña **General** en el Editor de flujos de trabajo.

### Requisitos previos

Complete las siguientes tareas.

- [Crear el ejemplo de flujo de trabajo complejo.](#)
- [Crear el esquema del ejemplo de flujo de trabajo complejo.](#)
- [Definir los enlaces para el ejemplo de flujo de trabajo complejo.](#)
- Abra el flujo de trabajo en el editor de flujos de trabajo para editarlo.

### Procedimiento

- 1 Haga clic en la pestaña **General**.
- 2 Seleccione la casilla de verificación de solo lectura de los atributos siguientes para convertirlos en constantes de solo lectura:
  - snapshotName

- snapshotDescription
- snapshotMemory
- snapshotQuiesce

Ha definido los atributos del flujo de trabajo que son constantes y los que son variables.

### Pasos siguientes

Debe crear la presentación del flujo de trabajo, que crea el diseño del cuadro de diálogo de parámetros de entrada en el que los usuarios especifican valores de parámetros de entrada de un flujo de trabajo cuando lo ejecutan.

## Crear el diseño de los parámetros de entrada del ejemplo de flujo de trabajo complejo

Cree el diseño o la presentación del cuadro de diálogo de parámetros de entrada en la pestaña **Presentación** del Editor de flujos de trabajo. El cuadro de diálogo de parámetros de entrada se abre cuando los usuarios ejecutan un flujo de trabajo, con lo cual los usuarios pueden indicar los parámetros de entrada con los que se ejecuta el flujo de trabajo.

### Requisitos previos

Complete las siguientes tareas.

- [Crear el ejemplo de flujo de trabajo complejo.](#)
- [Crear el esquema del ejemplo de flujo de trabajo complejo.](#)
- [Definir los parámetros del ejemplo de flujo de trabajo complejo.](#)
- [Definir los enlaces para el ejemplo de flujo de trabajo complejo.](#)
- [Establecer las propiedades de los atributos del ejemplo de flujo de trabajo complejo.](#)
- Abra el flujo de trabajo en el editor de flujos de trabajo para editarlo.

### Procedimiento

- 1 Haga clic en la pestaña **Presentación** en el Editor de flujos de trabajo.  
El flujo de trabajo Tomar una instantánea de todas las máquinas virtuales en un grupo de recursos tiene un solo parámetro de entrada; por lo tanto, la creación de la presentación es sencilla.
- 2 Haga clic con el botón secundario en el nodo **Presentación** en la lista jerárquica de presentaciones y seleccione **Crear grupo de visualización**.
- 3 Elimine el elemento **Nuevo paso** que aparece sobre el elemento **Nuevo grupo**.
- 4 Haga doble clic en el elemento **Nuevo grupo** y cambie el nombre del grupo a **Grupo de recursos**.

- 5 Escriba una descripción del grupo de visualización **Grupo de recursos** en el cuadro de texto **Descripción** en la pestaña **General** en la parte inferior de la pestaña **Presentación**.

Por ejemplo,

**Indicar el nombre del grupo de recursos que contiene máquinas virtuales de las que tomar una instantánea.**

- 6 Haga clic en el parámetro (VC:ResourcePool) resourcePool.
- 7 Haga clic en la pestaña **Propiedades** de (VC:ResourcePool) resourcePool.
- 8 Haga clic con el botón secundario en la pestaña **Propiedades** y seleccione **Añadir propiedad > Entrada obligatoria**.
- 9 Haga clic con el botón secundario en la pestaña **Propiedades** y seleccione **Añadir propiedad > Seleccionar valor como**.  
  
Cuando establece esta propiedad, establece el modo en el que el usuario selecciona el valor del parámetro de entrada (VC:ResourcePool) resourcePool.
- 10 Arrastre el parámetro (VC:ResourcePool) resourcePool bajo el grupo de visualización **Grupo de recursos**.

Ha creado el diseño del cuadro de diálogo que aparece cuando los usuarios ejecutan el flujo de trabajo.

#### Pasos siguientes

Ha completado el desarrollo del ejemplo de flujo de trabajo complejo. Ahora puede validar y ejecutar el flujo de trabajo.

## Validar y ejecutar el ejemplo de flujo de trabajo complejo

Después de crear un flujo de trabajo, puede validarlo para detectar posibles errores. Si el flujo de trabajo no contiene errores, puede ejecutarlo.

#### Requisitos previos

Cree un flujo de trabajo, diseñe su esquema, defina sus vínculos y enlaces, defina sus propiedades de parámetros y cree la presentación del cuadro de diálogo de parámetros de entrada.

Complete las siguientes tareas.

- [Crear el ejemplo de flujo de trabajo complejo.](#)
- [Crear una acción personalizada para el ejemplo de flujo de trabajo completo.](#)
- [Crear el esquema del ejemplo de flujo de trabajo complejo.](#)
- [Definir los parámetros del ejemplo de flujo de trabajo complejo.](#)
- [Definir los enlaces para el ejemplo de flujo de trabajo complejo.](#)
- [Establecer las propiedades de los atributos del ejemplo de flujo de trabajo complejo.](#)
- [Crear el diseño de los parámetros de entrada del ejemplo de flujo de trabajo complejo.](#)
- Abra el flujo de trabajo en el editor de flujos de trabajo para editarlo.

## Procedimiento

- 1 Haga clic en **Validación**, en la pestaña **Esquema** del editor de flujos de trabajo.  
La herramienta de validación detecta cualquier error en la definición del flujo de trabajo.
- 2 Tras haber eliminado todos los errores, haga clic en **Guardar y cerrar** en la parte inferior del editor de flujos de trabajo.  
Vuelva al cliente de Orchestrator.
- 3 Haga clic en la vista **Flujos de trabajo**.
- 4 En la lista jerárquica de flujos de trabajo, seleccione **Ejemplos de flujo de trabajo > Tomar una instantánea de todas las máquinas virtuales en un grupo de recursos**.
- 5 Haga clic con el botón secundario en el flujo de trabajo **Tomar una instantánea de todas las máquinas virtuales en un grupo de recursos** y seleccione **Iniciar flujo de trabajo**.  
El cuadro de diálogo de parámetros de entrada se abre y le pide que especifique un grupo de recursos que contenga máquinas virtuales de las que tomar instantáneas.
- 6 Haga clic en **Enviar** para ejecutar el flujo de trabajo.  
Aparece un token de flujo de trabajo en el flujo de trabajo Tomar una instantánea de todas las máquinas virtuales en un grupo de recursos.
- 7 Haga clic en el token del flujo de trabajo para comprobar el progreso del flujo de trabajo mientras se ejecuta.

Si el flujo de trabajo se ejecuta correctamente, toma una instantánea de todas las máquinas virtuales del grupo de recursos seleccionado.

## Pasos siguientes

Puede generar un documento en el que revisar la información sobre el flujo de trabajo. Consulte [Generar documentación de flujo de trabajo](#).

## Crear scripts

Orchestrator utiliza JavaScript para generar bloques de creación a partir de los cuales se crean acciones, elementos de flujo de trabajo y políticas que acceden a las API de las tecnologías que se conectan a Orchestrator.

Orchestrator utiliza el motor de JavaScript de Rhino de Mozilla 1.7R4 como motor de creación de scripts. El motor de creación de scripts proporciona comprobación de tipos variables, administración de espacio de nombres y control de excepciones.

El motor de creación de Orchestrator permite utilizar funciones básicas del lenguaje JavaScript, por ejemplo condiciones, bucles, matrices y cadenas. Puede utilizar objetos en la creación de scripts proporcionados por la API de Orchestrator API, o bien objetos de cualquier otra API que se importan a Orchestrator mediante un complemento y que se asignan a objetos de JavaScript. Para obtener información sobre Rhino, consulte el sitio web de Mozilla Rhino.

Este capítulo incluye los siguientes temas:

- [Elementos de Orchestrator que requieren creación de scripts](#)
- [Limitaciones de la implementación de Rhino de Mozilla en Orchestrator](#)
- [Uso de la API de creación de scripts de Orchestrator](#)
- [Usar expresiones XPath con el complemento vCenter Server](#)
- [Directrices para el control de excepciones](#)
- [Ejemplos de JavaScript de Orchestrator](#)

### Elementos de Orchestrator que requieren creación de scripts

No todos los elementos de Orchestrator requieren la creación de scripts. Para conferir flexibilidad máxima a las aplicaciones, puede personalizar algunos elementos añadiendo funciones de JavaScript.

Puede añadir scripts en los elementos siguientes de Orchestrator.

#### Acciones

Las acciones son funciones con scripts. Puede limitar la creación de scripts que se escriben para una acción a una sola operación a fin de maximizar el potencial de reutilización de la acción por parte de otros elementos, por ejemplo otros flujos de trabajo. Asimismo, una acción puede contener muchas operaciones para limitar la complejidad de los flujos de trabajo; sin

embargo, de este modo la capacidad de volver a utilizar la acción se reduce.

## Políticas

Las políticas se establecen mediante scripts que controlan los eventos de activador. Cuando se producen eventos de activador, las políticas inician operaciones de orquestación que se definen en scripts.

## Flujos de trabajo

El elemento de flujo de trabajo Tarea de scripts permite escribir una operación de scripts personalizadas o una secuencia de operaciones que se pueden utilizar en los flujos de trabajo. También se define la instrucción de decisión booleana para elementos de decisión personalizada en scripts que devuelven `true` o `false`.

# Limitaciones de la implementación de Rhino de Mozilla en Orchestrator

Orchestrator utiliza el motor de JavaScript de Rhino de Mozilla 1.7R4. Sin embargo, la implementación de Rhino en Orchestrator presenta algunas limitaciones.

Al escribir scripts para flujos de trabajo, deben tenerse en cuenta las siguientes limitaciones de la implementación de Rhino de Mozilla en Orchestrator.

- Cuando se ejecuta un flujo de trabajo, los objetos que pasan de un elemento del flujo de trabajo a otro no son objetos JavaScript. Lo que se pasa de un elemento al siguiente es la serialización de un objeto Java que tiene una imagen JavaScript. Debido a esto no se puede utilizar todo el lenguaje JavaScript, solo las clases que están presentes en el Explorador de API. No es posible pasar objetos de función de un elemento de flujo de trabajo a otro.
- Orchestrator ejecuta el código en elementos de tarea de scripts en un contexto que no es el contexto raíz de Rhino. Orchestrator encapsula de forma transparente las acciones y los elementos de tarea de scripts en funciones de JavaScript, que ejecuta a continuación. Un elemento de tarea de scripts que contenga `System.log(this)`; no mostrará el objeto global `this` de la misma forma que lo hace la implementación estándar de Rhino.
- Solo se puede llamar a acciones que devuelven objetos no serializables de scripts, y no de flujos de trabajo. Para llamar a una acción que devuelva un objeto no serializable, se debe escribir un elemento de tarea con scripts que llame a la acción mediante el uso del método `System.getModuleModuleName.action()`.
- La validación del flujo de trabajo no comprueba si un tipo de atributo de flujo de trabajo es diferente de un tipo de entrada de una acción o flujo de trabajo secundario. Si se cambia el tipo de un parámetro de entrada de flujo de trabajo, por ejemplo de `VIM3:VirtualMachine` a `VC:VirtualMachine`, pero no se actualiza ninguna acción o tarea de scripts que use el tipo de entrada original, el flujo de trabajo se valida pero no se ejecuta.



# Uso de la API de creación de scripts de Orchestrator

La API de Orchestrator expone todos los objetos y las funciones de las tecnologías, a las que accede Orchestrator a través de sus complementos, como métodos y objetos de JavaScript.

Por ejemplo, puede acceder a implementaciones de JavaScript de la API de vCenter Server mediante la API de Orchestrator para incluir las operaciones de vCenter en elementos con scripts que usted mismo haya creado. También puede acceder a implementaciones de objetos de JavaScript desde el resto de complementos que tenga instalados en el servidor de Orchestrator. Si crea un complemento personalizado para una aplicación externa, puede asignar los objetos de su API a los objetos de JavaScript que luego expone la API de Orchestrator.

## Procedimiento

### 1 [Acceder al motor de creación de scripts desde el Editor de flujos de trabajo](#)

El motor de creación de scripts de Orchestrator utiliza el motor JavaScript Rhino 1.7R4 de Mozilla a fin de ayudarle a escribir scripts para elementos con scripts en flujos de trabajo. Puede acceder al motor de creación de scripts para elementos de flujos de trabajo con scripts desde la pestaña **Creación de scripts** en el Editor de flujos de trabajo.

### 2 [Acceder al motor de creación de scripts desde el editor de acciones o de políticas](#)

El motor de creación de scripts de Orchestrator utiliza el motor JavaScript Rhino de Mozilla a fin de ayudar a escribir scripts para acciones o políticas. Puede acceder al motor de creación de scripts para acciones o de políticas desde las respectivas pestañas **Creación de scripts** en los editores de acciones y políticas.

### 3 [Acceder al Explorador de API de Orchestrator](#)

Orchestrator proporciona un Explorador de API para buscar en la API de Orchestrator y acceder a la documentación de objetos de JavaScript que se pueden utilizar en elementos con scripts.

### 4 [Utilizar el Explorador de API de Orchestrator](#)

La API de Orchestrator expone la API de todas las tecnologías conectadas, incluida toda la API de vCenter Server. El explorador de API de Orchestrator le ayuda a buscar los objetos que necesita añadir a los scripts.

### 5 [Escribir scripts](#)

El motor de creación de scripts de Orchestrator permite escribir scripts. La inserción automática de las funciones y la finalización automática de las líneas de creación de scripts aceleran el proceso de creación de scripts y minimizan el potencial de generación de errores de escritura en scripts.

### 6 [Añadir parámetros a scripts](#)

El motor de creación de scripts de Orchestrator permite importar parámetros disponibles a scripts.

### 7 [Acceder al sistema de archivos del servidor de Orchestrator desde JavaScript y flujos de trabajo](#)

Orchestrator limita el acceso al sistema de archivos del servidor de Orchestrator desde JavaScript y los flujos de trabajo a determinados directorios.

## 8 Acceder a las clases de Java desde JavaScript

De forma predeterminada, Orchestrator restringe el acceso de JavaScript a un conjunto limitado de clases de Java. Si necesita que JavaScript acceda a una mayor cantidad de clases de Java, debe establecer una propiedad del sistema Orchestrator para permitir este acceso.

## 9 Acceder a los comandos del sistema operativo desde JavaScript

La API de Orchestrator ofrece una clase de script, `Command`, que ejecuta comandos en el sistema operativo que aloja el servidor de Orchestrator. Para impedir el acceso no autorizado al host del servidor de Orchestrator, de forma predeterminada, las aplicaciones de Orchestrator no tienen permiso para ejecutar la clase `Command`.

## Acceder al motor de creación de scripts desde el Editor de flujos de trabajo

El motor de creación de scripts de Orchestrator utiliza el motor JavaScript Rhino 1.7R4 de Mozilla a fin de ayudarlo a escribir scripts para elementos con scripts en flujos de trabajo. Puede acceder al motor de creación de scripts para elementos de flujos de trabajo con scripts desde la pestaña **Creación de scripts** en el Editor de flujos de trabajo.

### Procedimiento

- 1 Seleccione **Diseño** en el menú desplegable del cliente de Orchestrator.
- 2 Haga clic con el botón secundario en la vista **Flujos de trabajo** del cliente de Orchestrator y seleccione **Editar**.
- 3 Haga clic en la pestaña **Esquema** en el Editor de flujos de trabajo.
- 4 Añada un elemento de tarea de scripts o de decisión personalizada al esquema de flujo de trabajo.
- 5 En la pestaña **Creación de scripts**, haga clic en el elemento de tarea de scripts.

Ha accedido al motor de creación de scripts para definir las funciones de creación de scripts de elementos de flujo de trabajo. La pestaña **Creación de scripts** permite navegar por la API, consultar documentación sobre objetos, buscar objetos y escribir JavaScript.

### Pasos siguientes

Busque la API de Orchestrator mediante el Explorador de API.

## Acceder al motor de creación de scripts desde el editor de acciones o de políticas

El motor de creación de scripts de Orchestrator utiliza el motor JavaScript Rhino de Mozilla a fin de ayudar a escribir scripts para acciones o políticas. Puede acceder al motor de creación de scripts para acciones o de políticas desde las respectivas pestañas **Creación de scripts** en los editores de acciones y políticas.

## Procedimiento

- 1 En el menú desplegable del cliente de Orchestrator, seleccione una opción dependiendo del tipo de elemento cuya creación de scripts desee editar.

Opción	Descripción
Diseño	Seleccione esta opción para editar la creación de scripts de un elemento de acción.
Ejecutar	Seleccione esta opción para editar la creación de scripts de una política.

- 2 Haga clic con el botón secundario en una acción o una política en las vistas **Acciones** o **Políticas**; a continuación, seleccione **Editar**.
- 3 Haga clic en la pestaña **Creación de scripts** del editor de acciones o de políticas.

Ha accedido al motor de creación de scripts para definir las funciones de creación de scripts de elementos de acción o de política. La pestaña **Creación de scripts** permite navegar por la API, consultar documentación sobre objetos, buscar objetos y escribir JavaScript.

### Pasos siguientes

Busque la API de Orchestrator mediante el Explorador de API.

## Acceder al Explorador de API de Orchestrator

Orchestrator proporciona un Explorador de API para buscar en la API de Orchestrator y acceder a la documentación de objetos de JavaScript que se pueden utilizar en elementos con scripts.

Puede consultar una versión en línea de la API de creación de scripts para el complemento vCenter Server en la página de inicio de la documentación de Orchestrator.

## Procedimiento

- 1 Inicie sesión en el cliente de Orchestrator.
- 2 Seleccione **Herramientas > Explorador de API**.

Aparece el Explorador de API. Utilícelo para buscar todos los objetos y todas las funciones de la API de Orchestrator.

### Pasos siguientes

Utilice el Explorador de API para escribir scripts para elementos de tarea de scripts.

## Utilizar el Explorador de API de Orchestrator

La API de Orchestrator expone la API de todas las tecnologías conectadas, incluida toda la API de vCenter Server. El explorador de API de Orchestrator le ayuda a buscar los objetos que necesita añadir a los scripts.

### Requisitos previos

Abra el Explorador de API.

## Procedimiento

- 1 Escriba el nombre o una parte del nombre de un objeto en el cuadro de texto **Buscar** del Explorador de API y haga clic en **Buscar**.

Para limitar su búsqueda a un tipo de objeto concreto, marque o desmarque las casillas de verificación **Clase de script**, **Atributos y métodos** y **Tipos y enumeraciones**.

- 2 Haga doble clic en el elemento de la lista propuesta.

El objeto queda resaltado en la lista jerárquica de la izquierda. Un panel de documentación bajo la lista jerárquica presenta la información sobre el objeto.

## Pasos siguientes

Utilice los objetos que encuentre en los scripts.

## Objetos de JavaScript en el Explorador de API

El Explorador de API de Orchestrator identifica y agrupa los diferentes tipos de objetos de JavaScript en el árbol jerárquico que hay a la izquierda de la pestaña **Creación de scripts** o el cuadro de diálogo Explorador de API. El Explorador de API utiliza iconos para ayudarle a identificar los distintos tipos de objetos.

En la tabla siguiente se describen los objetos de la API de Orchestrator y se muestran sus iconos.

**Tabla 2-1. Objetos de JavaScript en la API de Orchestrator**

Objeto	Icono en la lista jerárquica	Descripción
Tipo		Tipos
Conjunto de funciones		Tipo interno que contiene un conjunto de métodos estáticos
Primitivo		Tipos primitivos
Objeto		Objetos de creación de scripts de Orchestrator estándar
Atributo		Atributos de JavaScript
Método		Métodos de JavaScript
Constructor		Constructores de JavaScript
Enumeración		Enumeraciones de JavaScript
Conjunto de cadenas		Conjunto de cadenas, valores predeterminados
Módulo		Conjunto de acciones
Complemento	Imagen que define el complemento	Las API que exponen los complementos a Orchestrator

## Escribir scripts

El motor de creación de scripts de Orchestrator permite escribir scripts. La inserción automática de las funciones y la finalización automática de las líneas de creación de scripts aceleran el proceso de creación de scripts y minimizan el potencial de generación de errores de escritura en scripts.

### Requisitos previos

Abra un elemento con scripts para editar y haga clic en su pestaña **Creación de scripts**.

### Procedimiento

- 1 Navegue por la lista jerárquica de objetos en la parte izquierda de la pestaña **Creación de scripts**, o bien utilice la función de búsqueda Explorador de API, para seleccionar un tipo, una clase o un método que añadir al script.

- 2 Haga clic con el botón secundario en el tipo, la clase o el método; a continuación, seleccione **Copiar**.

Si el motor de creación de scripts no permite copiar el elemento seleccionado, significa que este objeto no es posible en el contexto del script.

- 3 Haga clic con el botón secundario en la ventana de creación de scripts; a continuación, pegue el elemento que ha copiado en el lugar pertinente del script.

El motor de creación de scripts inserta el elemento en el script, y lo completa con su constructor y un nombre de instancia.

Por ejemplo, si ha copiado el objeto `Date`, el motor de creación de scripts pega el código siguiente en el script.

```
var myDate = new Date();
```

- 4 Copie y pegue un método para añadir al script.

El motor de creación de scripts completa la llamada del método añadiendo los atributos correspondientes.

Por ejemplo, si ha copiado el objeto `cloneVM()` del módulo `com.vmware.library.vc.vm`, el motor de creación de scripts pega el código siguiente en el script.

```
System.getModule("com.vmware.library.vc.vm").cloneVM(vm, folder, name, spec)
```

El motor de creación de scripts resalta esos parámetros que ya ha definido en el elemento. Los parámetros que no estén definidos permanecen sin resaltar.

- 5 Coloque el cursor al final de un elemento que haya pegado en el script; a continuación, pulse Ctrl + barra espaciadora para seleccionar en una lista contextual de métodos y atributos posibles que el objeto pueda llamar.

---

**Nota** La función de finalización automática está, por ahora, en fase de pruebas.

---

Ha añadido objetos y funciones al script.

## Pasos siguientes

Añada parámetros al script.

## Codificación de color de las palabras clave de creación de scripts

Cuando añades scripts en la pestaña **Creación de scripts** de un elemento de flujo de trabajo con script, aparecen ciertos tipos de palabras clave en diferentes colores para mejorar la legibilidad del código.

Todos los scripts aparecen con una fuente negra estándar, a menos que se indique lo contrario.

**Tabla 2-2. Codificación de color de las palabras clave de creación de scripts**

Tipo de palabra clave	Color de texto en la pestaña Creación de scripts
Palabras de JavaScript estándar, como if, else, for y new	Negrita
Declaraciones de variables, concretamente var	Verde
Modificadores en bucles, por ejemplo in	Rojo
Valores de variables nulas	Púrpura
Valores de variables no nulas	Verde
Comentarios de código	Cursiva gris
Tipos de objetos de complemento de Orchestrator, por ejemplo VC:VirtualMachine o VC:Host	Verde
Texto de salida	Verde
Atributos de flujo de trabajo	Rosa
Entradas de flujo de trabajo	Rosa
Salidas de flujo de trabajo	Rosa

## Añadir parámetros a scripts

El motor de creación de scripts de Orchestrator permite importar parámetros disponibles a scripts.

Si ya ha definido parámetros para el elemento que edita, aparecen como vínculos en la barra de herramientas de la pestaña **Creación de scripts**.

### Requisitos previos

Un elemento con scripts se abre para editar y su pestaña **Creación de scripts** está abierta.

### Procedimiento

- 1 Coloque el cursor en la posición adecuada en un script en la ventana de creación de scripts de la pestaña **Creación de scripts**.
- 2 En la barra de herramientas de la pestaña **Creación de scripts**, haga clic en el vínculo del parámetro.

Orchestrator inserta el parámetro en la posición del cursor.

### 3 Inserte un parámetro con un valor nulo en el script.

Si transfiere valores nulos a tipos primitivos como enteros, booleanos y cadenas, la API de creación de scripts de Orchestrator establece automáticamente el valor predeterminado para este argumento.

Ha añadido parámetros al script.

#### Pasos siguientes

Añada acceso a clases de Java en scripts.

## Acceder al sistema de archivos del servidor de Orchestrator desde JavaScript y flujos de trabajo

Orchestrator limita el acceso al sistema de archivos del servidor de Orchestrator desde JavaScript y los flujos de trabajo a determinados directorios.

Las funciones y los flujos de trabajo de JavaScript solo tienen permisos de lectura, escritura y ejecución en el directorio permanente `c:\orchestrator`.

El administrador de Orchestrator puede modificar las carpetas a las cuales las funciones y los flujos de trabajo de JavaScript tienen acceso de lectura, escritura y ejecución estableciendo una propiedad del sistema. Consulte *Instalación y configuración de VMware vRealize Orchestrator* para obtener información sobre cómo establecer las propiedades del sistema.

Las funciones y los flujos de trabajo de JavaScript también tienen permisos de lectura, escritura y ejecución en la carpeta de E/S temporal predeterminada del sistema del servidor. Escribir en la carpeta de E/S temporal predeterminada es la única forma garantizada, independiente de la configuración y con posibilidad de cambiarse de acceder al sistema de archivos con los permisos completos. Ahora bien, los archivos que se guardan en la carpeta de E/S temporal se pierden al reiniciar el servidor.

Para obtener la carpeta de E/S temporal predeterminada, es necesario utilizar el método `System.getTempDirectory` en las funciones de JavaScript.

### Acceder al sistema de archivos de servidor utilizando el método `System.getTempDirectory`

Como alternativa a escribir en las carpetas en el sistema de servidor de Orchestrator en que el administrador ha establecido los permisos correspondientes, puede escribir en la carpeta de E/S temporal predeterminada.

De forma predeterminada, Orchestrator tiene permisos completos de lectura, escritura y ejecución en la carpeta de E/S temporal predeterminada. La carpeta de E/S temporal predeterminada se obtiene mediante el método `System.getTempDirectory` en las funciones de JavaScript.

#### Procedimiento

- ◆ Incluya la línea de código siguiente en las funciones de JavaScript para acceder a la carpeta `java.io.temp-dir`.

```
var tempDir = System.getTempDirectory()
```

## Acceder a las clases de Java desde JavaScript

De forma predeterminada, Orchestrator restringe el acceso de JavaScript a un conjunto limitado de clases de Java. Si necesita que JavaScript acceda a una mayor cantidad de clases de Java, debe establecer una propiedad del sistema Orchestrator para permitir este acceso.

De forma predeterminada, el motor de JavaScript de Orchestrator solo puede acceder a las clases del paquete `java.util.*`.

El administrador de Orchestrator puede permitir el acceso a otras clases de Java desde funciones de JavaScript estableciendo una propiedad del sistema. Consulte *Instalación y configuración de VMware vRealize Orchestrator* para obtener información sobre cómo establecer las propiedades del sistema.

## Acceder a los comandos del sistema operativo desde JavaScript

La API de Orchestrator ofrece una clase de script, `Command`, que ejecuta comandos en el sistema operativo que aloja el servidor de Orchestrator. Para impedir el acceso no autorizado al host del servidor de Orchestrator, de forma predeterminada, las aplicaciones de Orchestrator no tienen permiso para ejecutar la clase `Command`.

El administrador de Orchestrator puede permitir el acceso a la clase de script `Command` estableciendo la propiedad del sistema `com.vmware.js.allow-local-process=true`.

Para obtener información sobre cómo establecer las propiedades del sistema, consulte *Instalación y configuración de VMware vCenter Orchestrator*.

Para obtener información sobre cómo establecer las propiedades del sistema, consulte *Instalación y configuración de VMware vCenter Orchestrator*.

## Usar expresiones XPath con el complemento vCenter Server

Puede utilizar los métodos de buscador en el complemento vCenter Server para consultar los objetos de inventario de vCenter Server. Las expresiones XPath permiten definir parámetros de búsqueda.

El complemento vCenter Server incluye un conjunto de métodos de buscador de objetos, como `getAllDatastores()`, `getAllResourcePools()` o `findAllForType()`. Puede utilizar estos métodos para acceder a los inventarios de las instancias de vCenter Server que están conectadas al servidor de Orchestrator y buscar objetos por ID, nombre u otras propiedades.

Por motivos de rendimiento, los métodos de buscador no devuelven las propiedades de los objetos consultados a menos que especifique un conjunto de propiedades en la consulta de búsqueda.

Puede consultar una versión en línea de la API de creación de scripts para el complemento vCenter Server en la página de inicio de la documentación de Orchestrator.

---

**Importante** Las consultas basadas en expresiones XPath podrían afectar al rendimiento de Orchestrator porque el método de buscador devuelve todos los objetos de un tipo específico en el lado de vCenter Server y los filtros de consulta se aplican en el lado del complemento vCenter Server.

---



## Usar expresiones XPath con el complemento vCenter Server

Cuando invoca un método de buscador, puede utilizar expresiones basadas en el lenguaje de consulta XPath. La búsqueda devuelve todos los objetos de inventario que coinciden con las expresiones XPath. Si desea consultar alguna de las propiedades, puede incluirlas en el script de búsqueda en forma de matriz de cadena.

El ejemplo de JavaScript siguiente utiliza el objeto de creación de scripts `VcPlugin` y una expresión XPath para devolver los nombres de todos los objetos de base de datos que forman parte de los objetos administrados de vCenter Server y que contienen la cadena **ds** en el nombre.

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

La misma expresión XPath se puede invocar mediante el objeto de creación de scripts `Server` y el método de buscador `findAllForType`.

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

El ejemplo de script siguiente devuelve los nombres de todos los objetos de sistema host cuyo ID empieza por el dígito **1**.

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

El script siguiente devuelve los nombres y los ID de todos los objetos de centro de datos que contienen la cadena **DC** en el nombre, ya sea en mayúsculas o en minúsculas. El script también recupera la propiedad **etiqueta**.

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

## Directrices para el control de excepciones

La implementación de Orchestrator del motor de JavaScript de Rhino de Mozilla admite el control de excepciones para permitir el procesamiento de errores. Siga estas directrices al escribir controladores de excepciones en scripts.

- Utilice los tipos siguientes de error de la ECMA (European Computer Manufacturers Association). Utilice `Error` como excepción genérica que devuelven las funciones del complemento y los tipos siguientes de errores específicos.
  - `TypeError`
  - `RangeError`
  - `EvalError`
  - `ReferenceError`
  - `URIError`
  - `SyntaxError`

El ejemplo siguiente muestra una definición de `URIError`.

```
try {
    ...
    throw new URIError("VirtualMachine with ID 'vm-0056'
                        not found on 'vcenter-test-1'");
    ...
} catch ( e if e instanceof URIError ) {
}
}
```

- Todas las excepciones no capturadas por los scripts deben ser objetos de cadena simple con el formato `<type>:SPACE<human readable message>`, tal como se muestra en el ejemplo siguiente.

```
throw "ValidationError: The input parameter 'myParam' of type 'string' is too short."
```

- Escriba mensajes legibles que sean lo más claros posible.
- La comprobación de tipos de excepción de cadenas simples debe utilizar el patrón siguiente.

```
try {
    throw "VMwareNoSpaceLeftOnDatastore: Datastore 'myDatastore' has no space left" ;
} catch ( e if (typeof(e)=="string" && e.indexOf("VMwareNoSpaceLeftOnDatastore:") == 0) ) {
    System.log("No space left on device") ;
    // Do something useful here
}
```

- La comprobación de tipos de excepción de cadenas simples debe seguir este patrón en los elementos de scripts en los flujos de trabajo.

```
if (typeof(errorCode)=="string"
    && errorCode.indexOf("VMwareNoSpaceLeftOnDatastore:")
    == 0) {
    // Do something useful here
}
```

## Ejemplos de JavaScript de Orchestrator

Puede cortar, pegar y adaptar los ejemplos de JavaScript de Orchestrator a fin de escribir JavaScripts para tareas de orquestación comunes.

- [Ejemplos básicos de creación de scripts](#)

Las acciones, las políticas y los elementos con scripts de flujos de trabajo requieren la creación de scripts básicos de tareas comunes. Estos ejemplos se pueden cortar, pegar y adaptar en los elementos con scripts.

- [Ejemplos de creación de scripts de correo electrónico](#)

Los elementos de scripts de flujo de trabajo pueden incluir la creación de scripts de tareas comunes relacionadas con el correo electrónico. Estos ejemplos se pueden cortar, pegar y adaptar en los elementos con scripts.

- [Ejemplos de creación de scripts del sistema de archivos](#)

Las políticas, las acciones y los elementos con scripts de flujos de trabajo requieren la creación de scripts básicos de tareas comunes. Estos ejemplos se pueden cortar, pegar y adaptar en los elementos con scripts.

- [Ejemplos de creación de scripts de LDAP](#)

Las políticas, las acciones y los elementos con scripts de los flujos de trabajo requieren la creación de scripts de las tareas de LDAP comunes. Estos ejemplos se pueden cortar, pegar y adaptar en los elementos con scripts.

- [Ejemplos de creación de scripts de registro](#)

Las políticas, las acciones y los elementos con scripts de los flujos de trabajo requieren la creación de scripts de las tareas de registro comunes. Estos ejemplos se pueden cortar, pegar y adaptar en los elementos con scripts.

- [Ejemplos de creación de scripts de redes](#)

Las políticas, las acciones y los elementos con scripts de los flujos de trabajo requieren la creación de scripts de tareas de redes comunes. Estos ejemplos se pueden cortar, pegar y adaptar en los elementos con scripts.

## ■ Ejemplos de creación de scripts de flujos de trabajo

Las políticas, las acciones y los elementos con scripts de los flujos de trabajo requieren ejemplos de creación de scripts de las tareas comunes del flujo de trabajo. Estos ejemplos se pueden cortar, pegar y adaptar en los elementos con scripts.

## Ejemplos básicos de creación de scripts

Las acciones, las políticas y los elementos con scripts de flujos de trabajo requieren la creación de scripts básicos de tareas comunes. Estos ejemplos se pueden cortar, pegar y adaptar en los elementos con scripts.

### Acceder a documentos XML

El ejemplo siguiente de JavaScript permite acceder a documentos XML desde JavaScript mediante la implementación de ECMAScript para XML (E4X) en la API de JavaScript de Orchestrator.

**Nota** Además de implementar E4X en la API de JavaScript, Orchestrator proporciona una implementación XML de modelo de objetos de documento (DOM) en el complemento XML. Para obtener información sobre el complemento XML y sus flujos de trabajo de ejemplo, consulte *Uso de los complementos de vRealize Orchestrator*.

```
var people = <people>
    <person id="1">
        <name>Moe</name>
    </person>
    <person id="2">
        <name>Larry</name>
    </person>
</people>;

System.log("'people' = " + people);

// built-in XML type
System.log("'people' is of type : " + typeof(people));

// list-like interface System.log("which contains a list of " +
people.person.length() + " persons");
System.log("whose first element is : " + people.person[0]);

// attribute 'id' is mapped to field '@id'
people.person[0].@id='47';
// change Moe's id to 47
// also supports search by constraints
System.log("Moe's id is now : " + people.person.(name=='Moe').@id);

// suppress Moe from the list
delete people.person[0];
System.log("Moe is now removed.");

// new (sub-)document can be built from a string
people.person[1] = new XML("<person id='3'><name>James</name></person>");
System.log("Added James to the list, which is now :");
```

```
for each(var person in people..person)

for each(var person in people..person){
    System.log("- " + person.name + " (id=" + person.@id + ")");
}
```

## Establecer propiedades y obtenerlas de una tabla hash

El ejemplo de JavaScript siguiente establece propiedades en una tabla hash y obtiene las propiedades de dicha tabla. En el ejemplo siguiente, la clave siempre es una cadena y el valor es un objeto, un número, un booleano o una cadena.

```
var table = new Properties() ;
table.put("myKey",new Date()) ;
// get the object back
var myDate= table.get("myKey") ;
System.log("Date is : "+myDate) ;
```

## Reemplazar el contenido de una cadena

El ejemplo de JavaScript siguiente reemplaza el contenido de una cadena por contenido nuevo.

```
var str1 = "'hello'" ;
var reg = new RegExp("'", "g");
var str2 = str1.replace(reg,"\\'") ;
System.log(""+str2) ; // result : \'hello\'
```

## Comparar tipos

El ejemplo de JavaScript siguiente comprueba si un objeto concuerda con un determinado tipo de objeto.

```
var path = 'myurl/test';
if(typeof(path, string)){
    throw("string");
} else {
    throw("other");
}
```

## Ejecutar un comando en el servidor de Orchestrator

El ejemplo de JavaScript siguiente permite ejecutar una línea de comandos en el servidor de Orchestrator. Utilice las mismas credenciales que las empleadas al iniciar el servidor.

---

**Nota** El acceso al sistema de archivos está limitado de forma predeterminada.

---

```
var cmd = new Command("ls -al") ;
cmd.execute(true) ;
System.log(cmd.output) ;
```

## Ejemplos de creación de scripts de correo electrónico

Los elementos de scripts de flujo de trabajo pueden incluir la creación de scripts de tareas comunes relacionadas con el correo electrónico. Estos ejemplos se pueden cortar, pegar y adaptar en los elementos con scripts.

Cuando se ejecuta un flujo de trabajo de correo, utiliza la configuración predeterminada de servidor de correo que se establece en el flujo de trabajo Configurar correo. Los valores predeterminados se pueden modificar mediante parámetros de entrada o definiendo valores personalizados en los elementos de script de flujo de trabajo.

### Obtener una dirección de correo electrónico

El ejemplo siguiente de JavaScript obtiene la dirección de correo electrónico del propietario actual de un script en ejecución.

```
var emailAddress = Server.getRunningUser().emailAddress ;
```

### Enviar un correo electrónico

El siguiente ejemplo de JavaScript envía un correo electrónico al destinatario definido, a través de un servidor SMTP, con el contenido indicado.

```
var message = new EmailMessage() ;
message.smtpHost = "smtpHost" ;
message.subject= "my subject" ;
message.toAddress = "receiver@vmware.com" ;
message.fromAddress = "sender@vmware.com" ;
message.addMimePart("This is a simple message","text/html") ;
message.sendMessage() ;
```

### Recuperar mensajes de correo electrónico

El ejemplo siguiente de JavaScript recupera, sin borrarlos, los mensajes de una cuenta de correo electrónico mediante la API de scripts proporcionada por la clase MailClient.

```
var myMailClient = new MailClient();

myMailClient.setProtocol(mailProtocol);
if(useSSL){
    myMailClient.enableSSL();
}

myMailClient.connect( mailServer, mailPort, mailUsername, mailPassword);
System.log("Successfully login!");

try {
    myMailClient.openFolder("Inbox");

    var messages = myMailClient.getMessages();
    System.log("Reading messages...!");
    if ( messages != null && messages.length > 0 ) {
```

```

    System.log( "You have " + messages.length + " email(s) in your inbox" );
    for (i = 0; i < messages.length; i++) {
        System.log("");
        System.log("-----MSG-----");
        System.log("Headers: ");
        var headerProp = messages[i].getHeaders();
        for each(key in headerProp.keys){
            System.log(key+": "+headerProp.get(key));
        }
        System.log("");

        System.log( "Message["+ i +"] with from: " + messages[i].from + " to: " + messages[i].to);
        System.log( "Message["+ i +"] with subject: " + messages[i].subject);
        var content = messages[i].getContent();
        System.log("Msg content as string: " + content);
    }
} else {
    System.warn( "No messages found" );
}
} finally {
    myMailClient.closeFolder();
    myMailClient.close();
}

```

## Ejemplos de creación de scripts del sistema de archivos

Las políticas, las acciones y los elementos con scripts de flujos de trabajo requieren la creación de scripts básicos de tareas comunes. Estos ejemplos se pueden cortar, pegar y adaptar en los elementos con scripts.

### Añadir contenido a un archivo de texto sencillo

El ejemplo siguiente de JavaScript añade contenido a un archivo de texto.

```

var tempDir = System.getTempDirectory() ;
var fileWriter = new FileWriter(tempDir + "/readme.txt") ;
fileWriter.open() ;
fileWriter.writeLine("File written at : "+new Date()) ;
fileWriter.writeLine("Another line") ;
fileWriter.close() ;

```

### Obtener el contenido de un archivo

El ejemplo siguiente de JavaScript obtiene el contenido de un archivo de la máquina host del servidor de Orchestrator.

```

var tempDir = System.getTempDirectory() ;
var fileReader = new FileReader(tempDir + "/readme.txt") ;
fileReader.open() ;
var fileContentAsString = fileReader.readAll();
fileReader.close() ;

```

## Ejemplos de creación de scripts de LDAP

Las políticas, las acciones y los elementos con scripts de los flujos de trabajo requieren la creación de scripts de las tareas de LDAP comunes. Estos ejemplos se pueden cortar, pegar y adaptar en los elementos con scripts.

### Convertir objetos LDAP en objetos de Active Directory

El ejemplo de JavaScript siguiente convierte elementos de grupo de LDAP en objetos de grupo de usuarios de Active Directory y viceversa.

```
var ldapGroup ;
// convert from ldap element to Microsoft:UserGroup object
var adGroup = ActiveDirectory.search("UserGroup",ldapGroup.commonName) ;
// convert back to LdapGroup element
var ldapElement = Server.getLdapElement(adGroup.distinguishedName) ;
```

## Ejemplos de creación de scripts de registro

Las políticas, las acciones y los elementos con scripts de los flujos de trabajo requieren la creación de scripts de las tareas de registro comunes. Estos ejemplos se pueden cortar, pegar y adaptar en los elementos con scripts.

### Registro persistente

El siguiente ejemplo de JavaScript crea entradas de registro persistentes.

```
Server.log("This is a persistant message", "enter a long description here");
Server.warn("This is a persistant warning", "enter a long description here");
Server.error("This is a persistant error", "enter a long description here");
```

### Registro no persistente

El ejemplo siguiente de JavaScript crea entradas de registro no persistentes.

```
System.log("This is a non-persistant log message");
System.warn("This is a non-persistant log warning");
System.error("This is a non-persistant log error");
```

## Ejemplos de creación de scripts de redes

Las políticas, las acciones y los elementos con scripts de los flujos de trabajo requieren la creación de scripts de tareas de redes comunes. Estos ejemplos se pueden cortar, pegar y adaptar en los elementos con scripts.

### Obtener texto de una URL

El ejemplo de JavaScript siguiente accede a una URL, obtiene texto y lo convierte en una cadena.

```
var url = new URL("http://www.vmware.com") ;
var htmlContentAsString = url.getContent() ;
```



## Ejemplos de creación de scripts de flujos de trabajo

Las políticas, las acciones y los elementos con scripts de los flujos de trabajo requieren ejemplos de creación de scripts de las tareas comunes del flujo de trabajo. Estos ejemplos se pueden cortar, pegar y adaptar en los elementos con scripts.

### Devolver todos los flujos de trabajo ejecutados por el usuario actual

El siguiente ejemplo de JavaScript obtiene todas las ejecuciones de los flujos de trabajo del servidor y comprueba si pertenecen al usuario actual.

```
var allTokens = Server.findAllForType('WorkflowToken');
var currentUser = Server.getCredential().username;
var res = [];
for(var i = 0; i<res.length; i++){
    if(allTokens[i].runningUserName == currentUser){
        res.push(allTokens[i]);
    }
}
return res;
```

### Acceder al token del flujo de trabajo actual

Puede acceder al token del flujo de trabajo actual utilizando la variable `workflow`. Es un objeto de tipo simple `WorkflowToken` que proporciona acceso a la ejecución del flujo de trabajo actual. El siguiente ejemplo de JavaScript obtiene el ID del token del flujo de trabajo y su fecha de inicio.

```
System.log("Current workflow run ID: " + workflow.id);
System.log("Current workflow run start date: "+workflow.startDate);
```

### Programar un flujo de trabajo

El siguiente ejemplo de JavaScript inicia un flujo de trabajo con un conjunto de propiedades determinadas y, a continuación, lo programa para que se inicie una hora más tarde.

```
var workflowToLaunch = myWorkflow ;
// create parameters
var workflowParameters = new Properties() ;
workflowParameters.put("name","John Doe") ;
// change the task name
workflowParameters.put("__taskName","Workflow for John Doe") ;

// create scheduling date one hour in the future
var workflowScheduleDate = new Date() ;
var time = workflowScheduleDate.getTime() + (60*60*1000) ;
workflowScheduleDate.setTime(time) ; var scheduledTask =
workflowToLaunch.schedule(workflowParameters,workflowScheduleDate);
```

## Ejecutar un flujo de trabajo sobre una selección de objetos en bucle

El siguiente ejemplo de JavaScript toma la matriz de máquinas virtuales y ejecuta un flujo de trabajo sobre cada una de ellas en un bucle For. VMs y workflowToRun son entradas del flujo de trabajo.

```
var len=VMs.length;
for (var i=0; i < len; i++ )
{
    var VM = VMs[i];
    //var workflowToLaunch = Server.getWorkflowWithId("workflowId");
    var workflowToLaunch = workflowToRun;
    if (workflowToLaunch == null) {
        throw "Workflow not found";
    }
    var workflowParameters = new Properties();
    workflowParameters.put("vm",VM);
    var wfToken = workflowToLaunch.execute(workflowParameters);
    System.log ("Ran workflow on " +VM.name);
}
```

## Desarrollar acciones

Orchestrator proporciona bibliotecas de acciones predefinidas. Las acciones representan funciones individuales que se utilizan como bloques de creación en flujos de trabajo y en scripts.

Las acciones son funciones de JavaScript. Admiten múltiples parámetros de entrada y tienen un solo valor de devolución. Pueden llamar a cualquier objeto en la API de Orchestrator o a objetos en cualquier API que se pueda importar a Orchestrator mediante un complemento.

Cuando se ejecuta un flujo de trabajo, una acción toma parámetros de entrada de los atributos del flujo de trabajo. Estos atributos pueden ser parámetros de entrada iniciales del flujo de trabajo o bien atributos establecidos por otros elementos del flujo de trabajo cuando se ejecutan.

Este capítulo incluye los siguientes temas:

- [Reutilizar acciones](#)
- [Acceder a la vista Acciones](#)
- [Componentes de la vista Acciones](#)
- [Crear acciones](#)
- [Historial de versiones de acciones](#)
- [Restaurar acciones eliminadas](#)

### Reutilizar acciones

Cuando defina una función individual como una acción, en lugar de codificarla directamente en un elemento de flujo de trabajo de tarea de scripts, expóngala en la biblioteca. Cuando una acción aparece en la biblioteca, otros flujos de trabajo pueden utilizarla.

Cuando define acciones independientemente de los flujos de trabajo que las llaman, puede actualizarlas u optimizarlas con mayor facilidad. La definición de acciones individuales también permite a otros flujos de trabajo reutilizar las acciones. Cuando se ejecuta un flujo de trabajo, Orchestrator guarda en caché cada acción solo la primera vez que la ejecuta el flujo de trabajo. A continuación, Orchestrator puede reutilizar la acción guardada en caché. Guardar las acciones en caché es útil para las llamadas recursivas en un flujo de trabajo o para bucles rápidos.

Puede duplicar acciones, exportarlas a otros flujos de trabajo o paquetes, o moverlas a otro módulo en la lista jerárquica de acciones.

## Acceder a la vista Acciones

La interfaz del cliente de Orchestrator presenta una vista **Acciones** que proporciona acceso a las bibliotecas de acciones del servidor de Orchestrator.

La vista **Acciones** de la interfaz del cliente de Orchestrator presenta una lista jerárquica de todas las acciones disponibles en el servidor de Orchestrator.

### Procedimiento

- 1 Seleccione **Diseño** en el menú desplegable del cliente de Orchestrator.
- 2 Haga clic en la vista **Acciones**.
- 3 Examine las bibliotecas de acciones expandiendo los nodos de la lista jerárquica de acciones.

Puede utilizar la vista **Acciones** para ver información sobre las acciones de las bibliotecas, así como crear y editar acciones.

## Componentes de la vista Acciones

Cuando hace clic en una acción en la lista jerárquica de acciones, aparece información sobre dicha acción en el panel derecho del cliente de Orchestrator.

La vista **Acciones** cuenta con cuatro pestañas.

<b>General</b>	Muestra información general sobre la acción, como el nombre, el número de versión, los permisos y una descripción.
<b>Creación de scripts</b>	Muestra los tipos de valor devuelto de la acción, así como el código de JavaScript que define la función de la acción.
<b>Eventos</b>	Muestra todos los eventos que ha encontrado o activado esta acción.
<b>Permisos</b>	Muestra los usuarios y los grupos de usuarios que tienen permiso para acceder a esta acción.

## Crear acciones

Puede definir funciones individuales como acciones que otros elementos, por ejemplo los flujos de trabajo, pueden utilizar. Las acciones son funciones de JavaScript con permisos y parámetros de entrada y de salida definidos.

### ■ Crear una acción

Cuando defina una función individual como acción, en lugar de codificarla directamente en un elemento de flujo de trabajo de tarea de scripts, puede exponerla en la biblioteca para que la utilicen otros flujos de trabajo.

## ■ Buscar elementos que implementan una acción

Si edita una acción y cambia su comportamiento, puede que interrumpa accidentalmente un flujo de trabajo o una aplicación que implementa dicha acción. Orchestrator proporciona una función para buscar todas las acciones, los flujos de trabajo o los paquetes que implementan un elemento específico. Puede comprobar si la modificación del elemento afecta al funcionamiento de otros elementos.

## ■ Directrices de codificación de acciones

Para optimizar el rendimiento de los flujos de trabajo y maximizar el potencial de volver a utilizar acciones, es aconsejable seguir una serie de directivas básicas de codificación al crear acciones.

## Crear una acción

Cuando defina una función individual como acción, en lugar de codificarla directamente en un elemento de flujo de trabajo de tarea de scripts, puede exponerla en la biblioteca para que la utilicen otros flujos de trabajo.

### Procedimiento

- 1 Seleccione **Diseño** en el menú desplegable del cliente de Orchestrator.
- 2 Haga clic en la vista **Acciones**.
- 3 Expanda la raíz de la lista jerárquica de acciones y acceda al módulo en el que desea crear la acción.
- 4 Haga clic con el botón secundario en el módulo y seleccione **Añadir acción**.
- 5 Escriba un nombre para la acción en el cuadro de texto y haga clic en **Aceptar**.  
La acción personalizada se añade a la biblioteca de acciones.
- 6 Haga clic con el botón secundario en la acción y seleccione **Editar**.
- 7 Haga clic en la pestaña **Creación de scripts**.
- 8 Para cambiar el tipo de valor devuelto predeterminado, haga clic en el vínculo **vacío**.
- 9 Añada los parámetros de entrada de acción haciendo clic en el icono de flecha.
- 10 Escriba el script de acción.
- 11 Defina los permisos de acción.
- 12 Haga clic en **Guardar y cerrar**.

Ha creado una acción personalizada y ha añadido los parámetros de entrada de acción.

### Pasos siguientes

Puede utilizar la nueva acción personalizada en un flujo de trabajo.

## Buscar elementos que implementan una acción

Si edita una acción y cambia su comportamiento, puede que interrumpa accidentalmente un flujo de trabajo o una aplicación que implementa dicha acción. Orchestrator proporciona una función para buscar todas las acciones, los flujos de trabajo o los paquetes que implementan un elemento específico. Puede comprobar si la modificación del elemento afecta al funcionamiento de otros elementos.

---

**Importante** La función **Buscar elementos que utilizan este elemento** comprueba todos los paquetes, los flujos de trabajo y las políticas, pero no comprueba los scripts. En consecuencia, modificar una acción podría afectar a un elemento que llama a esta acción en un script que la función **Buscar elementos que utilizan este elemento** no ha identificado.

---

### Procedimiento

- 1 Seleccione **Diseño** en el menú desplegable del cliente de Orchestrator.
- 2 Haga clic en la vista **Acciones**.
- 3 Expanda los nodos de la lista jerárquica de acciones para acceder a una acción específica.
- 4 Haga clic con el botón secundario en la acción y seleccione **Buscar elementos que utilizan este elemento**.

Un cuadro de diálogo muestra todos los elementos, por ejemplo flujos de trabajo o paquetes, que implementan esta acción.

- 5 Haga doble clic en un elemento en la lista de resultados para mostrar dicho elemento en el cliente de Orchestrator.

Ha localizado todos los elementos que implementan una acción.

### Pasos siguientes

Puede comprobar si la modificación de este elemento afecta a otros elementos.

## Directrices de codificación de acciones

Para optimizar el rendimiento de los flujos de trabajo y maximizar el potencial de volver a utilizar acciones, es aconsejable seguir una serie de directivas básicas de codificación al crear acciones.

### Directrices básicas de acciones

Al crear una acción, debe utilizar directrices básicas.

- Cada acción debe incluir una descripción de su rol y su función.
- Escriba acciones breves y elementales, y combínelas en un flujo de trabajo.
- Procure no escribir acciones que efectúen varias funciones, ya que eso limita el potencial de volver a utilizar la acción.
- Evite acciones que se ejecuten durante intervalos de tiempo prolongados. Cree más bien un bucle en el flujo de trabajo, e incluya un evento de espera o un elemento Temporizador de espera tras el elemento de acción.

- No escriba puntos de comprobación en acciones. Los flujos de trabajo establecen un punto de comprobación al inicio y al final de cada ejecución del elemento.
- Procure no escribir bucles en una acción. Lo aconsejable es crear bucles en el flujo de trabajo. Si el servidor se reinicia, un flujo de trabajo en ejecución se reanuda en su último punto de comprobación, al inicio de un elemento. Si escribe un bucle dentro de una acción y el servidor se reinicia mientras el flujo de trabajo ejecuta dicha acción, el flujo de trabajo se reanuda en el punto de comprobación el inicio de dicha acción y el bucle se inicia de nuevo desde el comienzo.

## Directrices de denominación de acciones

Utilice directrices básicas cuando asigne nombres a acciones.

- Escriba los nombres de las acciones en inglés.
- Los nombres de las acciones deben empezar en minúscula. Utilice un carácter en mayúscula al comienzo de cada palabra compuesta del nombre. Por ejemplo, `myAction`.
- Los nombres de las acciones deben ser lo más explícitos posible para que la función de la acción esté clara. Por ejemplo, `backupAllVMsInPool`.
- Los nombres de los módulos deben ser lo más explícitos posible.
- Dichos nombres deben ser exclusivos.
- En el caso de los nombres de los módulos, utilice el formato de dirección de Internet invertida. Por ejemplo, `com.vmware.myactions.myAction`.

## Directrices de parámetros de acciones

Utilice directrices básicas cuando escriba definiciones de parámetros de acciones.

- Escriba los nombres de los parámetros en inglés.
- Los nombres de los parámetros deben empezar en minúscula.
- Los nombres de los parámetros deben ser lo más explícitos posible.
- Es preferible que los nombres de los parámetros sean de una sola palabra. Si el nombre debe tener más de una palabra, utilice un carácter en mayúscula al comienzo de cada palabra compuesta del nombre. Por ejemplo, `myParameter`.
- Utilice la forma del plural para parámetros que representen una matriz de objetos.
- Los nombres de las variables no deben ser ambiguos, por ejemplo `displayName`.
- Incluya una descripción de cada parámetro para explicar su finalidad.
- No utilice un número excesivo de parámetros en una sola acción.

## Historial de versiones de acciones

Puede utilizar el historial de versiones para revertir una acción a una versión anterior. Puede revertir el estado de la acción a una versión anterior o una posterior. También puede comparar las diferencias entre el estado actual de la acción y una versión guardada de esta.

Orchestrator crea un nuevo elemento de historial de versiones para cada acción cuando incrementa y guarda la versión de la acción. Los cambios subsiguientes en la acción no modifican el elemento de versión actual. Por ejemplo, cuando crea la versión de acción 1.0.0 y la guarda, el estado de la acción se guarda en la base de datos. Si realiza cambios en la acción, puede guardar el estado de la acción en el cliente de Orchestrator, pero no puede aplicar los cambios en la versión de la acción 1.0.0. Para guardar los cambios en la base de datos, debe crear una versión de acción subsiguiente y guardarla. El historial de versiones se conserva en la base de datos junto con la propia acción.

Cuando elimina una acción, Orchestrator marca el elemento como eliminado en la base de datos sin eliminar el historial de versiones del elemento de la base de datos. De este modo, es posible restaurar las acciones eliminadas. Consulte [Restaurar acciones eliminadas](#).

### Requisitos previos

Abra una acción para editarla.

### Procedimiento

- 1 Haga clic en la pestaña **General** en el editor de acciones.
- 2 Haga clic en **Mostrar historial de versiones**.  
Se abre una ventana con el historial de versiones.
- 3 Seleccione una versión de acción y haga clic en **Diferencia con actual** para comparar las diferencias.  
Se abre una ventana que muestra las diferencias entre la versión de acción actual y la versión de acción seleccionada.
- 4 Seleccione una versión de acción y haga clic en **Revertir** para restaurar el estado de la acción.

---

**Precaución** Si no ha guardado la versión de acción actual, se elimina del historial de versiones y no se puede revertir a la versión actual.

---

El estado de la acción se revierte al estado de la versión seleccionada.

## Restaurar acciones eliminadas

Puede restaurar acciones que se han eliminado de la biblioteca.

### Procedimiento

- 1 Seleccione **Diseño** en el menú desplegable del cliente de Orchestrator.
- 2 Haga clic en la vista **Acciones**.
- 3 Vaya a la carpeta en la que desea restaurar una o más acciones eliminadas.
- 4 Haga clic con el botón derecho en la carpeta y seleccione **Restaurar acciones eliminadas**.
- 5 Seleccione las acciones que desee restaurar y haga clic en **Restaurar**.

Las acciones aparecerán en la carpeta seleccionada.



# Creación de elementos de recursos

# 4

Los flujos de trabajo pueden llegar a requerir que los objetos que cree independientemente de Orchestrator se usen como atributos. Para usar objetos externos como atributos en flujos de trabajo, impórtelos al servidor de Orchestrator como elementos de recursos.

Los objetos que los flujos de trabajo pueden usar como elementos de recursos son, entre otros, archivos de imagen, scripts, plantillas XML o archivos HTML. Los flujos de trabajo que se ejecutan en el servidor de Orchestrator pueden utilizar cualquier elemento de recursos que se importe a Orchestrator.

Si se importa un objeto a Orchestrator como elemento de recursos, es posible efectuar cambios en el objeto en una sola ubicación y propagarlos automáticamente a todos los flujos de trabajo que usan ese elemento de recursos.

Los elementos de recursos se pueden organizar en carpetas. Un elemento de recursos tiene un tamaño máximo de 16 MB.

Este capítulo incluye los siguientes temas:

- [Visualizar un elemento de recursos](#)
- [Importar un objeto externo para utilizar como elemento de recursos](#)
- [Editar la información de los elementos de los recursos y los derechos de acceso](#)
- [Guardar un elemento de recursos en un archivo](#)
- [Actualizar un elemento de recursos](#)
- [Añadir un elemento de recursos a un flujo de trabajo](#)

## Visualizar un elemento de recursos

Puede ver los elementos de recursos en el cliente de Orchestrator, para examinar su contenido y descubrir los flujos de trabajo que utilizan ese elemento de recursos.

### Procedimiento

- 1 Seleccione **Diseño** en el menú desplegable del cliente de Orchestrator.
- 2 Haga clic en la vista **Recursos**.
- 3 Expanda el visor del árbol jerárquico para ir al elemento de recursos.

- 4 Haga clic en un elemento de recursos para ver información sobre él en el panel derecho.
- 5 Haga clic en la pestaña **Visor** para ver el contenido del elemento de recursos.
- 6 Haga clic con el botón derecho en el elemento de recursos y seleccione **Buscar elementos que utilizan este elemento**.

Orchestrator enumera todos los flujos de trabajo que utilizan este elemento de recursos.

#### Pasos siguientes

Importe y edite un elemento de recursos.

## Importar un objeto externo para utilizar como elemento de recursos

Los flujos de trabajo pueden requerir que los objetos que cree independientemente de Orchestrator se utilicen como atributos. Para utilizar objetos externos como atributos en flujos de trabajo, debe importarlos al servidor de Orchestrator como elementos de recursos.

#### Requisitos previos

Compruebe que tiene un archivo de imagen, script, plantilla XML, archivo HTML u otro tipo de objeto que importar.

#### Procedimiento

- 1 Seleccione **Diseño** en el menú desplegable del cliente de Orchestrator.
- 2 Haga clic en la vista **Recursos**.
- 3 Haga clic con el botón derecho en una carpeta de recursos en la lista jerárquica o en la raíz y seleccione **Nueva carpeta** para crear una carpeta en la que almacenar el elemento de recurso.
- 4 Haga clic con el botón derecho en la carpeta de recursos en la que importar el elemento de recursos y seleccione **Importar recursos**.
- 5 Seleccione el recurso que importar y haga clic en **Abrir**.

Orchestrator añade el elemento de recursos a la carpeta seleccionada.

Importó un elemento de recursos en el servidor de Orchestrator.

#### Pasos siguientes

Edite la información general del elemento de recursos y establezca los permisos de acceso de usuarios.


## Editar la información de los elementos de los recursos y los derechos de acceso

Después de importar un objeto en el servidor de Orchestrator como un elemento de recurso, puede editar los detalles y los permisos del elemento de recurso.

**Requisitos previos**

Compruebe que haya importado una imagen, script, archivo XML o HTML, o cualquier otro tipo de objeto importado a Orchestrator como elemento de recurso.

**Procedimiento**

- 1 Seleccione **Diseño** en el menú desplegable del cliente de Orchestrator.
- 2 Haga clic en la vista **Recursos**.
- 3 Haga clic con el botón derecho en el elemento de recurso y seleccione **Editar**.
- 4 Haga clic en la pestaña **General** y defina el nombre, la versión y la descripción del elemento de recurso.
- 5 Haga clic en la pestaña **Permisos** y en el icono **Añadir derechos de acceso** () para definir los permisos de un grupo de usuarios.
- 6 Indique el nombre del grupo de usuarios en el campo de texto **Filtro**.
- 7 Seleccione un grupo de usuarios y haga clic en **Aceptar**.
- 8 Haga clic con el botón derecho en un grupo de usuarios y seleccione **Añadir derechos de acceso**.
- 9 Compruebe las casillas de verificación correspondientes para definir el nivel de permisos para este grupo de usuarios y haga clic en **Aceptar**.

Los permisos no son acumulativos. Para permitir que un usuario vea el elemento de recurso, lo utilice en sus flujos de trabajo y modifique los permisos, debe marcar todas las casillas de verificación.

- 10 Haga clic en **Guardar y cerrar** para salir del editor.

Ya editó la información general del elemento de recurso y estableció los derechos de acceso de los usuarios.

**Pasos siguientes**

Guarde el elemento de recurso en un archivo para actualizarlo o añada el elemento de recurso a un flujo de trabajo.

## Guardar un elemento de recursos en un archivo

Puede guardar un elemento de recursos en un archivo del sistema local. Al hacerlo, puede editarlo.

No es posible editar un elemento de recursos en el cliente de Orchestrator. Por ejemplo, si el elemento de recursos es un archivo de configuración XML o un script, debe guardarlo localmente para poder modificarlo.

**Requisitos previos**

Compruebe que el servidor de Orchestrator contenga un elemento de recursos que pueda guardar en un archivo.

### Procedimiento

- 1 Seleccione **Diseño** en el menú desplegable del cliente de Orchestrator.
- 2 Haga clic en la vista **Recursos**.
- 3 Haga clic con el botón derecho en el elemento de recursos y seleccione **Guardar en archivo**.
- 4 Lleve a cabo las modificaciones necesarias en el archivo.

Ha guardado un elemento de recursos en un archivo.

### Pasos siguientes

Cargue el elemento de recursos en el servidor de Orchestrator.

## Actualizar un elemento de recursos

Si desea actualizar un elemento de recursos, debe exportarlo al sistema de archivos, editar el archivo exportado con una herramienta adecuada y actualizar el elemento de recursos importando el archivo editado.

### Requisitos previos

Compruebe que haya importado una imagen, script, archivo XML o HTML, o cualquier otro tipo de objeto importado a Orchestrator como elemento de recurso.

### Procedimiento

- 1 Modifique el archivo de origen del elemento de recursos en el sistema local.
- 2 Seleccione **Diseño** en el menú desplegable del cliente de Orchestrator.
- 3 Haga clic en la vista **Recursos**.
- 4 Localice en la lista jerárquica el elemento de recursos que ha actualizado.
- 5 Haga clic con el botón derecho en el elemento de recursos y seleccione **Actualizar recurso**.
- 6 (opcional) Haga clic en la pestaña **Visor** para verificar que Orchestrator haya actualizado el elemento de recursos.

Ha actualizado un elemento de recursos que contiene el servidor de Orchestrator.

## Añadir un elemento de recursos a un flujo de trabajo

Los elementos de recursos son objetos externos que puede importar al servidor de Orchestrator para que los flujos de trabajo los utilicen como atributos durante su ejecución. Por ejemplo, un flujo de trabajo puede utilizar un archivo XML importado que defina una asignación para convertir un tipo de datos en otro o un script que defina una función, cuando se ejecute.

## Requisitos previos

Compruebe que disponga de los siguientes objetos en el servidor de Orchestrator:

- Una imagen, script, archivo XML o HTML, o cualquier otro tipo de objeto importado a Orchestrator como elemento de recurso.
- Un flujo de trabajo que requiera el elemento de recurso como atributo.

## Procedimiento

- 1 Seleccione **Diseño** en el menú desplegable del cliente de Orchestrator.
- 2 Haga clic en la vista **Flujos de trabajo**.
- 3 Expanda el visor del árbol jerárquico para acceder al flujo de trabajo que requiera el elemento de recurso como atributo.
- 4 Haga clic con el botón derecho en el flujo de trabajo y seleccione **Editar**.
- 5 En la pestaña **General**, en el panel Atributos, haga clic en el icono **Añadir atributo** (A+).
- 6 Haga clic en el nombre de atributo y escriba un nuevo nombre de atributo.
- 7 Haga clic en **Tipo** para indicar el tipo de atributo.
- 8 En el cuadro de diálogo **Seleccionar un tipo**, escriba **recurso** en el cuadro de texto **Filtro** para buscar un tipo de objeto.

Opción	Acción
Defina un elemento de un solo recurso como atributo	Seleccione ResourceElement de la lista.
Defina una carpeta que contenga varios elementos de recursos como atributo	Seleccione ResourceElementCategory de la lista.

- 9 Haga clic en **Valor** y escriba el nombre del elemento de recurso o la categoría de los elementos de recurso en el cuadro de texto **Filtro**.
- 10 En la lista propuesta, seleccione el elemento de recursos o una carpeta que contenga elementos de recursos; a continuación, haga clic en **Seleccionar**.
- 11 Haga clic en **Guardar y cerrar** para salir del editor.

Se ha añadido un elemento de recursos o una carpeta de elementos de recursos como atributo en un flujo de trabajo.

## Crear paquetes

Los paquetes se utilizan para distribuir contenido de un servidor de Orchestrator a otro. Los paquetes pueden contener flujos de trabajo, acciones, plantillas de políticas, configuraciones o recursos.

Al añadir un elemento a un paquete, Orchestrator comprueba las dependencias y añade cualquier elemento dependiente al paquete. Por ejemplo, si añade un flujo de trabajo que utiliza otros flujos de trabajo o acciones, Orchestrator añade esos flujos de trabajo y esas acciones al paquete.

Al importar un paquete, el servidor compara las versiones de los diferentes elementos de su contenido con los elementos locales correspondientes. La comparación muestra las diferencias de versiones entre los elementos locales y los importados. El administrador puede decidir si se importa el paquete; asimismo, puede seleccionar la importación de determinados elementos.

Los paquetes utilizan la administración de derechos digitales para controlar el modo en el que el servidor destinatario puede utilizar el contenido del paquete. Orchestrator firma paquetes y los cifra para la protección de datos. Los paquetes pueden hacer el seguimiento de los usuarios que exportan y redistribuyen elementos mediante el uso de certificados X509.

Para obtener más información sobre la utilización de paquetes, consulte *Uso del cliente de VMware vRealize Orchestrator*.

- **Crear un paquete**

Puede exportar flujos de trabajo, plantillas de políticas, acciones, referencias de complementos, recursos y elementos de configuración en paquetes. Todos los elementos que implementa un elemento en un paquete se añaden al paquete automáticamente, con el fin de garantizar la compatibilidad entre versiones. Si no desea añadir los elementos referenciados, puede eliminarlos en el editor de paquetes.

- **Establecer permisos de usuario en un paquete**

Se establecen diferentes niveles de permiso en un paquete para limitar el acceso que distintos usuarios o grupos de usuarios puedan tener al contenido de dicho paquete.

## Crear un paquete

Puede exportar flujos de trabajo, plantillas de políticas, acciones, referencias de complementos, recursos y elementos de configuración en paquetes. Todos los elementos que implementa un elemento en un paquete se añaden al paquete automáticamente, con el fin de garantizar la compatibilidad entre versiones. Si no desea añadir los elementos referenciados, puede eliminarlos en el editor de paquetes.

## Requisitos previos

Compruebe que el servidor de Orchestrator contenga elementos como flujos de trabajo, acciones y plantillas de políticas que pueda añadir a un paquete.

## Procedimiento

1 En el menú desplegable del cliente de Orchestrator, seleccione **Administrar**.

2 Haga clic en la vista **Paquetes**.

3 Haga clic con el botón secundario en el panel izquierdo y seleccione **Añadir paquete**.

4 Escriba el nombre del nuevo paquete y haga clic en **Aceptar**.

La sintaxis de los nombres de paquetes es *domain.your\_company.folder.package\_name*.

Por ejemplo, `com.vmware.myfolder.mypackage`.

5 Haga clic con el botón secundario en el paquete y seleccione **Editar**.

Se abre el editor de paquetes.

6 En la pestaña **General**, añada una descripción para el paquete.

7 En la pestaña **Flujos de trabajo**, añada flujos de trabajo al paquete.

- Haga clic en **Insertar flujos de trabajo (búsqueda de lista)** para buscar y seleccionar flujos de trabajo en un cuadro de diálogo de selección.
- Haga clic en **Insertar flujos de trabajo (examinar árbol)** para buscar y seleccionar carpetas de flujos de trabajo en la lista jerárquica.

8 En las pestañas **Plantillas de políticas**, **Acciones**, **Configuraciones**, **Recursos** y **Complementos utilizados**, añada plantillas de políticas, acciones, elementos de configuración, elementos de recursos y complementos al paquete.

9 Haga clic en **Guardar y cerrar** para salir del editor.

Ha creado un paquete y le ha añadido elementos.

## Pasos siguientes

Defina los permisos de usuario para este paquete.

# Establecer permisos de usuario en un paquete

Se establecen diferentes niveles de permiso en un paquete para limitar el acceso que distintos usuarios o grupos de usuarios puedan tener al contenido de dicho paquete.


Puede seleccionar los usuarios y los grupos de usuarios para los que establecer permisos en los usuarios y los grupos de usuarios del servidor LDAP de Orchestrator o del servidor vCenter Single Sign-On. Orchestrator define niveles de permisos que puede aplicar a usuarios o grupos de usuarios.

<b>Ver</b>	El usuario puede ver los elementos del paquete, pero no los esquemas ni la creación de scripts.
<b>Inspeccionar</b>	El usuario puede ver los elementos del paquete, incluidos los esquemas y la creación de scripts.
<b>Editar</b>	El usuario puede editar los elementos del paquete.
<b>Administrar</b>	El usuario puede establecer permisos en los elementos del paquete.

#### Requisitos previos

Cree un paquete, ábralo para editarlo en el editor de paquetes y añada los elementos necesarios al paquete.

#### Procedimiento

- 1 Haga clic en la pestaña **Permisos** en el editor de paquetes.
- 2 Haga clic en el icono **Añadir derechos de acceso** () para definir permisos para un usuario o un grupo de usuarios nuevos.
- 3 Busque un usuario o un grupo de usuarios.  
Los resultados de búsqueda muestran todos los usuarios y los grupos de usuarios que coinciden con la búsqueda.
- 4 Seleccione un usuario o un grupo de usuarios.
- 5 Seleccione las casillas de verificación correspondientes para definir el nivel de permisos de este usuario y haga clic en **Seleccionar**.  
Para permitir que un usuario vea los elementos, inspeccione el esquema y la creación de scripts, ejecute y edite los elementos y cambie los permisos, debe marcar todas las casillas de verificación.
- 6 Haga clic en **Guardar y cerrar** para salir del editor.

Ha creado un paquete y ha establecido los correspondientes permisos de usuario.



# Desarrollar complementos

Orchestrator permite la integración con soluciones de administración mediante su arquitectura de complementos abierta. Puede utilizar el cliente de Orchestrator para ejecutar y crear flujos de trabajo de complementos y acceder a la API de complementos.

Este capítulo incluye los siguientes temas:

- [Descripción general de los complementos](#)
- [Contenido y estructura de un complemento](#)
- [Referencia de API del complemento de Orchestrator](#)
- [Elementos del archivo de definición del complemento vso.xml](#)
- [Recomendaciones para el desarrollo de complementos de Orchestrator](#)

## Descripción general de los complementos

Los complementos de Orchestrator deben incluir un conjunto estándar de componentes y seguir una arquitectura estándar. Estas prácticas facilitan la creación de complementos para la mayor variedad posible de tecnologías externas.

- [Estructura de un complemento de Orchestrator](#)

Los complementos de Orchestrator tienen una estructura común compuesta de varios tipos de capa que implementan funciones específicas.

- [Exponer una API externa a Orchestrator](#)

Puede exponer una API de un producto externo a la plataforma Orchestrator creando un complemento de Orchestrator. Puede crear un complemento para cualquier tecnología que exponga una API que pueda asignar a objetos Javascript que use Orchestrator.

- [Componentes de un complemento](#)

Los complementos se componen de un conjunto estándar de componentes que exponen los objetos en la tecnología conectada a la plataforma de Orchestrator.

- [Función del archivo vso.xml](#)

El archivo `vso.xml` se utiliza para asignar los objetos, las clases, los métodos y los atributos de la tecnología conectada a los objetos de inventario y los atributos, así como a los tipos, las clases y los métodos de creación de scripts de Orchestrator. El archivo `vso.xml` también define la configuración y el comportamiento de inicio del complemento.

- **Funciones del adaptador de complemento**

El adaptador de complemento es el punto de entrada del complemento al servidor de Orchestrator. El adaptador de complemento sirve como almacén de datos para la tecnología conectada en el servidor de Orchestrator, crea la fábrica del complemento y administra los eventos que tienen lugar en la tecnología conectada.

- **Funciones de la fábrica del complemento**

La fábrica del complemento define el modo en el que Orchestrator busca objetos en la tecnología conectada y lleva a cabo operaciones en los objetos.

- **Función de los objetos de buscador**

Los objetos de buscador identifican y localizan determinadas instancias de los tipos de objetos administrados en la tecnología conectada. Orchestrator puede modificar e interactuar con los objetos que encuentre en la tecnología conectada ejecutando flujos de trabajo en los objetos de buscador.

- **Función de los objetos de creación de scripts**

Los objetos de creación de scripts son representaciones de JavaScript de objetos de la tecnología conectada. Los objetos de creación de scripts de los complementos aparecen en la API de JavaScript de Orchestrator, y se pueden usar en los elementos con script en los flujos de trabajo y las acciones.

- **Función de los gestores de eventos**

Los eventos son cambios en los estados o los atributos de los objetos que Orchestrator detecta en la tecnología conectada. Orchestrator supervisa los eventos implementando gestores de eventos.

## Estructura de un complemento de Orchestrator

Los complementos de Orchestrator tienen una estructura común compuesta de varios tipos de capa que implementan funciones específicas.

Las tres capas inferiores de un complemento de Orchestrator, que son clases de infraestructura, clases de ajuste y objetos de creación de scripts, implementan la conexión entre la tecnología conectada y Orchestrator.

Las partes de un complemento de Orchestrator que son visibles para el usuario son las tres capas superiores: acciones, bloques de creación y flujos de trabajo de alto nivel.

**Figura 6-1. Estructura de un complemento de Orchestrator****Clases de infraestructura**

Un conjunto de clases que proporciona la conexión entre la tecnología conectada y Orchestrator. Entre las clases de infraestructura están las utilizadas para implementaciones según la definición del complemento, por ejemplo fábrica de complementos, adaptador de complementos, etc. Las clases de infraestructura también incluyen las que proporcionan funciones para objetos y tareas comunes, por ejemplo para ayuda, almacenamiento en caché o inventario.

**Clases de ajuste**

Un conjunto de clases que adapta el modelo de objetos de la tecnología conectada al modelo de objetos que quiere exponer dentro de Orchestrator.

**Objetos de creación de scripts**

Son tipos de objeto de JavaScript que proporcionan acceso a las clases de ajuste, los métodos y los atributos en la tecnología conectada. En el archivo `vso.xml`, debe definir qué clases de ajuste, atributos y métodos de tecnología conectada se expondrán a Orchestrator.

**Acciones**

Un conjunto de funciones de JavaScript que puede usar directamente en flujos de trabajo y tareas de creación de scripts. Las acciones admiten múltiples parámetros de entrada y tienen un solo valor de devolución.

**Flujos de trabajo de bloques de creación**

Un conjunto de flujos de trabajo que cubren todas las funciones genéricas que quiera proporcionar con el complemento. Un flujo de trabajo de bloques de creación suele representar una operación en la interfaz de usuario de la tecnología orquestada. Los flujos de trabajo de bloques de creación se pueden usar directamente o se pueden incluir en flujos de trabajo de alto nivel.

**Flujos de trabajo de alto nivel**

Un conjunto de flujos de trabajo que cubren funciones específicas del complemento. Puede proporcionar flujos de trabajo de alto nivel para

cumplir requisitos concretos o mostrar ejemplos complejos del uso de complementos.

## Exponer una API externa a Orchestrator

Puede exponer una API de un producto externo a la plataforma Orchestrator creando un complemento de Orchestrator. Puede crear un complemento para cualquier tecnología que exponga una API que pueda asignar a objetos Javascript que use Orchestrator.

Los complementos asignan objetos y métodos Java que añaden a la API de creación de scripts de Orchestrator. Si una tecnología externa expone una API de Java, puede asignar la API directamente a JavaScript para que Orchestrator la use en los flujos de trabajo y en las acciones.

Puede crear complementos para aplicaciones que exponen una API en un lenguaje distinto a Java utilizando WSDL (Web service definition language), REST (Representational state transfer) o un servicio de mensajería para integrar la API expuesta con objetos Java. A continuación, puede asignar los objetos Java integrados a JavaScript para que Orchestrator pueda usarlos.

La tecnología conectada es independiente de Orchestrator. Puede crear complementos de Orchestrator para productos externos aunque solo tenga acceso al código binario, por ejemplo en archivos Java (archivos JAR), en lugar de código fuente.

## Componentes de un complemento

Los complementos se componen de un conjunto estándar de componentes que exponen los objetos en la tecnología conectada a la plataforma de Orchestrator.

Los principales componentes de un complemento son el adaptador de complemento, la fábrica y las implementaciones de eventos. Los objetos y las operaciones que se han definido en el adaptador, la fábrica y las implementaciones de eventos se asignan a los objetos de Orchestrator en un archivo de definición XML denominado `vso.xml`. El archivo `vso.xml` asigna los objetos y las funciones de la tecnología conectada a los objetos de creación de scripts de JavaScript que aparecen en la API de JavaScript de Orchestrator. El archivo `vso.xml` también asigna tipos de objetos de la tecnología conectada a los buscadores, que aparecen en la pestaña **Inventario** de Orchestrator.

Los componentes incluyen los componentes que se indican a continuación.

### Módulo de complemento

El complemento propiamente dicho, como se define mediante un conjunto de clases de Java, un archivo `vso.xml` y paquetes de los flujos de trabajo y las acciones que interactúan con los objetos a los que se accede con el complemento. El módulo de complemento es obligatorio.

### Adaptador de complemento

Define la interfaz entre la tecnología conectada y el servidor de Orchestrator. El adaptador es el punto de entrada del complemento a la plataforma de Orchestrator. El adaptador crea la fábrica del complemento, administra la carga y descarga del complemento, y gestiona los eventos que tienen lugar en los objetos en la tecnología conectada. El adaptador de complemento es obligatorio.

**Fábrica de complemento**

Define el modo en el que Orchestrator busca los objetos en la tecnología conectada y lleva a cabo operaciones con ellos. El adaptador crea una fábrica para la sesión de cliente que se abre entre Orchestrator y una tecnología conectada. La fábrica permite compartir una sesión entre todas las conexiones de cliente o abrir una sesión por conexión de cliente. La fábrica de complemento es obligatoria.

**Configuración**

Orchestrator no define un método estándar para que el complemento almacene su configuración. Puede guardar la información de configuración utilizando registros de Windows, archivos de configuración estáticos, bases de datos o archivos XML. Los complementos de Orchestrator se pueden configurar ejecutando flujos de trabajo de configuración en el cliente de Orchestrator.

**Buscadores**

Reglas de interacción que definen el modo en el que Orchestrator localiza y representa los objetos en la tecnología conectada. Los buscadores recuperan objetos del conjunto de objetos que expone la tecnología conectada a Orchestrator. En el archivo `vso.xml`, se definen las relaciones entre los objetos para permitir la navegación por la red de objetos. Orchestrator representa el modelo de objetos de la tecnología conectada en la pestaña **Inventario**. Los buscadores son obligatorios si se desea exponer objetos en la tecnología conectada a Orchestrator.

**Objetos de creación de scripts**

Tipos de objetos de JavaScript que proporcionan acceso a los objetos, las operaciones y los atributos en la tecnología conectada. Los objetos de creación de scripts definen la forma en la que Orchestrator accede al modelo de objeto de la tecnología conectada mediante JavaScript. Las clases y los métodos de la tecnología conectada se asignan a los objetos de JavaScript en el archivo `vso.xml`. Puede acceder a los objetos de JavaScript en la API de creación de scripts de Orchestrator e integrarlos en los flujos de trabajo, las acciones y las tareas de scripts de Orchestrator. Los objetos de creación de scripts son obligatorios si se desea añadir métodos, clases y tipos de creación de scripts a la API de JavaScript de Orchestrator.

**Inventario**

Las instancias de los objetos en la tecnología conectada que Orchestrator localiza utilizando buscadores aparecen en la vista **Inventario** en el cliente de Orchestrator. Puede realizar operaciones en los objetos del inventario ejecutando flujos de trabajo en ellos. El inventario es opcional. Puede crear un complemento que solo añada clases y tipos de creación de scripts a la API de JavaScript de Orchestrator y no exponga ninguna instancia de objetos en el inventario.

**Eventos**

Cambios en el estado de un objeto en la tecnología conectada. Orchestrator puede escuchar de forma pasiva los eventos que ocurren en

la tecnología conectada. Orchestrator también puede activar eventos en la tecnología conectada. Los eventos son opcionales.

## Función del archivo vso.xml

El archivo `vso.xml` se utiliza para asignar los objetos, las clases, los métodos y los atributos de la tecnología conectada a los objetos de inventario y los atributos, así como a los tipos, las clases y los métodos de creación de scripts de Orchestrator. El archivo `vso.xml` también define la configuración y el comportamiento de inicio del complemento.

El archivo `vso.xml` realiza las funciones principales siguientes.

<b>Comportamiento de inicio y de configuración</b>	Define el modo en que se inicia el complemento y localiza las implementaciones de configuración que define el complemento. Carga el adaptador del complemento.
<b>Objetos de inventario</b>	Define los tipos de objetos a los que accede el complemento en la tecnología conectada. Los métodos de buscador de la implementación de fábrica del complemento localizan las instancias de estos objetos y las muestran en el inventario de Orchestrator.
<b>Tipos de creación de scripts</b>	Añade tipos de creación de scripts a la API de JavaScript de Orchestrator para representar los distintos tipos de objetos en el inventario. Puede utilizar estos tipos de creación de scripts como parámetros en los flujos de trabajo.
<b>Clases de creación de scripts</b>	Añade clases a la API de JavaScript de Orchestrator que puede utilizar en los elementos con script de los flujos de trabajo, las acciones, las políticas, etc.
<b>Métodos de creación de scripts</b>	Añade métodos a la API de JavaScript de Orchestrator que puede utilizar en elementos con scripts en los flujos de trabajo, las acciones, las políticas, etc.
<b>Atributos de creación de scripts</b>	Añade los atributos de los objetos en la tecnología conectada a la API de JavaScript de Orchestrator que puede utilizar en los elementos con scripts en los flujos de trabajo, las acciones, las políticas, etc.

## Funciones del adaptador de complemento

El adaptador de complemento es el punto de entrada del complemento al servidor de Orchestrator. El adaptador de complemento sirve como almacén de datos para la tecnología conectada en el servidor de Orchestrator, crea la fábrica del complemento y administra los eventos que tienen lugar en la tecnología conectada.

Para crear un adaptador de complemento, cree una clase de Java que implemente la interfaz de `IPluginAdaptor`.

La clase del adaptador de complemento que cree administra la fábrica, los eventos y los activadores del complemento en la tecnología conectada. La interfaz `IPluginAdaptor` proporciona métodos que se utilizan para llevar a cabo estas tareas.

El adaptador de complemento desempeña las funciones principales siguientes.

<b>Crea una fábrica</b>	La función más importante del adaptador de complemento es cargar y descargar una instancia de fábrica de complemento para cada conexión de Orchestrator con la tecnología conectada. La clase del adaptador de complemento llama al método <code>IPluginAdaptor.createPluginFactory()</code> para crear una instancia de una clase que implementa la interfaz de <code>IPluginFactory</code> .
<b>Administra eventos</b>	El adaptador de complemento es la interfaz entre el servidor de Orchestrator y la tecnología conectada. El adaptador de complemento administra los eventos que realiza o supervisa Orchestrator en los objetos de la tecnología conectada. El adaptador administra eventos a través de los publicadores de eventos. Los publicadores de eventos son instancias de la interfaz de <code>IPluginEventPublisher</code> que crea el adaptador llamando al método <code>IPluginAdaptor.registerEventPublisher()</code> . Los publicadores de eventos establecen activadores y medidores en los objetos de la tecnología conectada, para permitir a Orchestrator lanzar acciones definidas si se producen determinados eventos en el objeto o si los valores del objeto superan ciertos umbrales. Asimismo, puede definir las instancias de <code>PluginTrigger</code> y de <code>PluginWatcher</code> que definen eventos que esperan los elementos Evento de espera en flujos de trabajo de larga ejecución.
<b>Establece el nombre del complemento</b>	El nombre del complemento se indica en el archivo <code>vso.xml</code> . El adaptador de complemento obtiene este nombre del archivo <code>vso.xml</code> y lo publica en la vista <b>Inventario</b> en el cliente de Orchestrator.
<b>Instala licencias</b>	Puede llamar a métodos para instalar los archivos de licencia que requiera la tecnología conectada en la implementación del adaptador.

Para obtener más información sobre la interfaz de `IPluginAdaptor`, todos sus métodos y todas las demás clases de la API del complemento, consulte [Referencia de API del complemento de Orchestrator](#).

## Funciones de la fábrica del complemento

La fábrica del complemento define el modo en el que Orchestrator busca objetos en la tecnología conectada y lleva a cabo operaciones en los objetos.

Para crear la fábrica del complemento, debe implementar y extender la interfaz de `IPluginFactory` desde la API del complemento de Orchestrator. La clase de fábrica del complemento que cree define las funciones de buscador que Orchestrator utiliza para acceder a objetos en la tecnología conectada. La fábrica permite al servidor de Orchestrator buscar objetos por su ID, su relación con otros objetos o una cadena de consulta.

La fábrica de complemento desempeña las funciones principales siguientes.

<b>Busca objetos</b>	Puede crear funciones que busquen objetos según su nombre y su tipo. Para buscar objetos por nombre y tipo, utilice el método <code>IPluginFactory.find()</code> .
<b>Buscar objetos en relación con otros objetos</b>	Puede crear funciones para buscar objetos que se relacionen con un objeto determinado mediante un determinado tipo de relación. Las relaciones se definen en el archivo <code>vso.xml</code> . También puede crear buscadores para buscar objetos secundarios dependientes que se relacionan con todos los objetos principales mediante un determinado tipo de relación. El método <code>IPluginFactory.findRelation()</code> permite buscar objetos que se relacionen con un determinado objeto principal mediante un tipo concreto de relación. El método <code>IPluginFactory.hasChildrenInRelation()</code> permite descubrir si existe al menos un objeto secundario en una instancia principal.
<b>Definir consultas para buscar objetos según sus propios criterios</b>	Puede crear buscadores de objetos que implementen reglas de consulta que se definan. El método <code>IPluginFactory.findAll()</code> permite buscar todos los objetos que cumplan las reglas de consulta que se definan cuando la fábrica llame a este método. Los resultados del método <code>findAll()</code> se obtienen en un objeto <code>QueryResult</code> que contiene una lista de todos los objetos encontrados que coincidan con las reglas de consulta definidas.

Para obtener más información sobre la interfaz de `IPluginFactory`, todos sus métodos y todas las demás clases de la API del complemento, consulte [Referencia de API del complemento de Orchestrator](#).

## Función de los objetos de buscador

Los objetos de buscador identifican y localizan determinadas instancias de los tipos de objetos administrados en la tecnología conectada. Orchestrator puede modificar e interactuar con los objetos que encuentre en la tecnología conectada ejecutando flujos de trabajo en los objetos de buscador.

Cualquier instancia de un determinado tipo de objeto administrado en la tecnología conectada debe tener un identificador único para que los objetos de buscador de Orchestrator la puedan encontrar. La tecnología conectada proporciona los identificadores únicos para las instancias de objetos como cadenas. Cuando se ejecuta un flujo de trabajo, Orchestrator establece los identificadores únicos de los objetos que encuentra como valores de atributo de flujo de trabajo. Los flujos de trabajo que requieren un objeto de un tipo concreto como parámetro de entrada se ejecutan en una instancia específica de ese tipo de objeto.

Los objetos de buscador que añaden los complementos a la API de JavaScript de Orchestrator tienen el nombre de complemento como prefijo. Por ejemplo, el tipo de objeto administrado `VirtualMachine` de la API de vCenter Server aparece en Orchestrator como tipo `VC:VirtualMachine` de JavaScript.



Por ejemplo, Orchestrator accede a una instancia `VC:VirtualMachine` específica a través del complemento de vCenter Server implementando un objeto de buscador que utiliza el atributo `id` de la máquina virtual como identificador único. Puede transferir esta instancia de objeto a los elementos de flujo de trabajo como valores de atributo.

Un complemento de Orchestrator asigna los objetos de la tecnología conectada a objetos de buscador equivalentes de Orchestrator en los elementos `<finder>` del archivo `vso.xml`. Los elementos `<finder>` identifican el método o la función de la tecnología conectada que obtiene el identificador único para una instancia determinada de un objeto. Los elementos `<finder>` también definen relaciones entre los objetos, para buscar los objetos por la forma en que se relacionan entre ellos.

Los objetos de buscador aparecen en la pestaña **Inventario** de Orchestrator, bajo el complemento que los contiene.

## Función de los objetos de creación de scripts

Los objetos de creación de scripts son representaciones de JavaScript de objetos de la tecnología conectada. Los objetos de creación de scripts de los complementos aparecen en la API de JavaScript de Orchestrator, y se pueden usar en los elementos con script en los flujos de trabajo y las acciones.

Los objetos de creación de scripts de los complementos aparecen en la API de JavaScript de Orchestrator como módulos, tipos y clases de JavaScript. La mayoría de los objetos de buscador tienen una representación de objeto de creación de scripts. Las clases de JavaScript pueden añadir métodos y atributos a la API de JavaScript de Orchestrator que representan los métodos y los atributos de los objetos de la API de la tecnología conectada. La tecnología conectada proporciona las implementaciones de los objetos, los tipos, las clases, los atributos y los métodos, independientemente de Orchestrator. Por ejemplo, el complemento de vCenter Server representa todos los objetos de la API de vCenter Server como objetos JavaScript en la API de JavaScript de Orchestrator, con representaciones JavaScript de todas las clases, los métodos y los atributos que define la API de vCenter Server. Puede utilizar las clases de creación de scripts de vCenter Server, así como los métodos y los atributos que definen en las funciones con scripts de Orchestrator.

Por ejemplo, el tipo de objeto administrado `VirtualMachine` de la API de vCenter Server se detecta mediante el buscador `VC:VirtualMachine` y aparece en la API de JavaScript de Orchestrator como clase `VcVirtualMachine` de JavaScript. La clase `VcVirtualMachine` de JavaScript en la API de JavaScript de Orchestrator define los mismos métodos y atributos que el objeto administrado `VirtualMachine` de la API de vCenter Server.

Un complemento de Orchestrator asigna los objetos, los tipos, las clases, los atributos y los métodos de la tecnología conectada a los objetos, los tipos, las clases, los atributos y los métodos JavaScript de Orchestrator en el elemento `<scripting-objects>` en el archivo `vso.xml`.

## Función de los gestores de eventos

Los eventos son cambios en los estados o los atributos de los objetos que Orchestrator detecta en la tecnología conectada. Orchestrator supervisa los eventos implementando gestores de eventos.

Los complementos de Orchestrator permiten supervisar eventos en una tecnología conectada de distintas formas. La API del complemento de Orchestrator permite crear los siguientes tipos de gestores de eventos para supervisar los eventos en una tecnología conectada.

### **Escuchas**

Supervise de forma pasiva los objetos de la tecnología conectada para detectar cambios en su estado. La implementación del complemento o DE la tecnología conectada define los eventos supervisados por las escuchas. Las escuchas no inician eventos, pero notifican a Orchestrator cuando se producen los eventos. Las escuchas detectan los eventos sondeando la tecnología conectada o recibiendo notificaciones de ella. Cuando se producen los eventos, las políticas o los flujos de trabajo de Orchestrator que esperan el evento pueden reaccionar iniciando operaciones en el servidor de Orchestrator. Los componentes de escucha son opcionales.

### **Políticas**

Supervise determinados elementos en la tecnología conectada e inicie operaciones en el servidor de Orchestrator si se producen los eventos. Las políticas pueden supervisar los activadores y los medidores de políticas. Los activadores de políticas definen un evento en la tecnología conectada que, cuando se produce, provoca que una política en ejecución inicie una operación en el servidor de Orchestrator, por ejemplo la ejecución de un flujo de trabajo. Los medidores de políticas definen los rangos de valores para los atributos de un objeto en la tecnología conectada; cuando se superan, hacen que Orchestrator inicie una operación. Las políticas son opcionales.

### **Activadores de flujo de trabajo**

Si un flujo de trabajo en ejecución contiene un elemento Evento de espera, cuando alcanza dicho elemento, suspende su ejecución y espera a que se produzca un evento en la tecnología conectada. Los activadores de flujo de trabajo definen los eventos en la tecnología conectada a los que esperan los elementos Evento de espera en los flujos de trabajo. Los activadores de flujo de trabajo se registran con monitores. Además, son opcionales.

### **Monitores**

Supervise los activadores de flujo de trabajo para un determinado evento en la tecnología conectada, en nombre de un elemento Evento de espera en un flujo de trabajo. Cuando tiene lugar el evento, los monitores lo notifican a los flujos de trabajo que esperan dicho evento. Los monitores son opcionales.

## **Contenido y estructura de un complemento**

Los complementos de Orchestrator deben contener un conjunto estándar de componentes y seguir una estructura de archivos estándar. Para que un complemento siga la estructura de archivos estándar, debe incluir carpetas y archivos específicos.

Para crear un complemento de Orchestrator, defina el modo en que Orchestrator accede e interactúa con los objetos en la tecnología conectada. Asigne también todos los objetos y las funciones de la tecnología conectada a los objetos y las funciones correspondientes de Orchestrator en el archivo `vso.xml`.

El archivo `vso.xml` debe incluir una referencia a todos los tipos de objeto u operaciones que se exponen a Orchestrator. Se debe proporcionar un identificador único para cada objeto que encuentre el complemento en la tecnología conectada. Defina los nombres de objeto en los elementos `finder` y en los elementos de objeto en el archivo `vso.xml`.

Un complemento se puede entregar como archivo de Java estándar (JAR) o como archivo ZIP; en cualquier caso, debe cambiarse su extensión a `.dar`.

---

**Nota** Puede utilizar el centro de control de Orchestrator para importar un archivo DAR al servidor de Orchestrator.

---

- **Definir la asignación de aplicaciones en el archivo `vso.xml`**

Los objetos que incluye en el archivo `vso.xml` aparecen como objetos de creación de scripts en la API de creación de scripts de Orchestrator, o como objetos de buscador en la pestaña **Inventario** de Orchestrator.

- **Formato del archivo de definición del complemento `vso.xml`**

El archivo `vso.xml` define la manera que el servidor de Orchestrator tiene de interactuar con la tecnología conectada. Debe incluir una referencia a cada tipo de objeto o de operación para exponer a Orchestrator en el archivo `vso.xml`.

- **Designar objetos de complemento**

Se debe proporcionar un identificador exclusivo para cada objeto que encuentre el complemento en la tecnología conectada. Los nombres de objetos se definen en los elementos `<finder>` y `<object>` del archivo `vso.xml`.

- **Convenciones de nomenclatura de objetos de complemento**

Al asignar nombres a objetos de complementos, debe seguir las convenciones de las clases de Java.

- **Estructura de archivos del complemento**

Un complemento debe seguir una estructura de archivos estándar, así como incluir determinados archivos y carpetas. Un complemento se entrega como archivo Java (JAR) o ZIP estándar, cuyo nombre se debe cambiar con la extensión `.dar`.

## Definir la asignación de aplicaciones en el archivo `vso.xml`

Los objetos que incluye en el archivo `vso.xml` aparecen como objetos de creación de scripts en la API de creación de scripts de Orchestrator, o como objetos de buscador en la pestaña **Inventario** de Orchestrator.

El archivo `vso.xml` proporciona la información siguiente al servidor de Orchestrator:

- Un nombre de versión y la descripción del complemento

- Referencias a las clases de la tecnología conectada y al adaptador de complementos asociado
- Inicializa el complemento cuando se inicia el servidor de Orchestrator
- Tipos de creación de scripts para representar los tipos de objetos en la tecnología conectada
- Relaciones entre tipos de objetos para definir el modo en el que los objetos se visualizan en el inventario de Orchestrator
- Clases de creación de scripts que asignan los objetos y las operaciones de la tecnología conectada a funciones y tipos de objetos en la API de JavaScript de Orchestrator
- Enumeraciones para definir una lista de valores constantes que se aplican a todos los objetos de un tipo concreto
- Eventos que Orchestrator supervisa en la tecnología conectada

El archivo `vso.xml` debe seguir la definición del esquema XML de los complementos de Orchestrator. Se puede acceder a la definición del esquema en el sitio de soporte de VMware.

```
http://www.vmware.com/support/orchestrator/plugin-4-1.xsd
```

Para obtener descripciones de todos los elementos del archivo `vso.xml`, consulte [Elementos del archivo de definición del complemento vso.xml](#).

## Formato del archivo de definición del complemento vso.xml

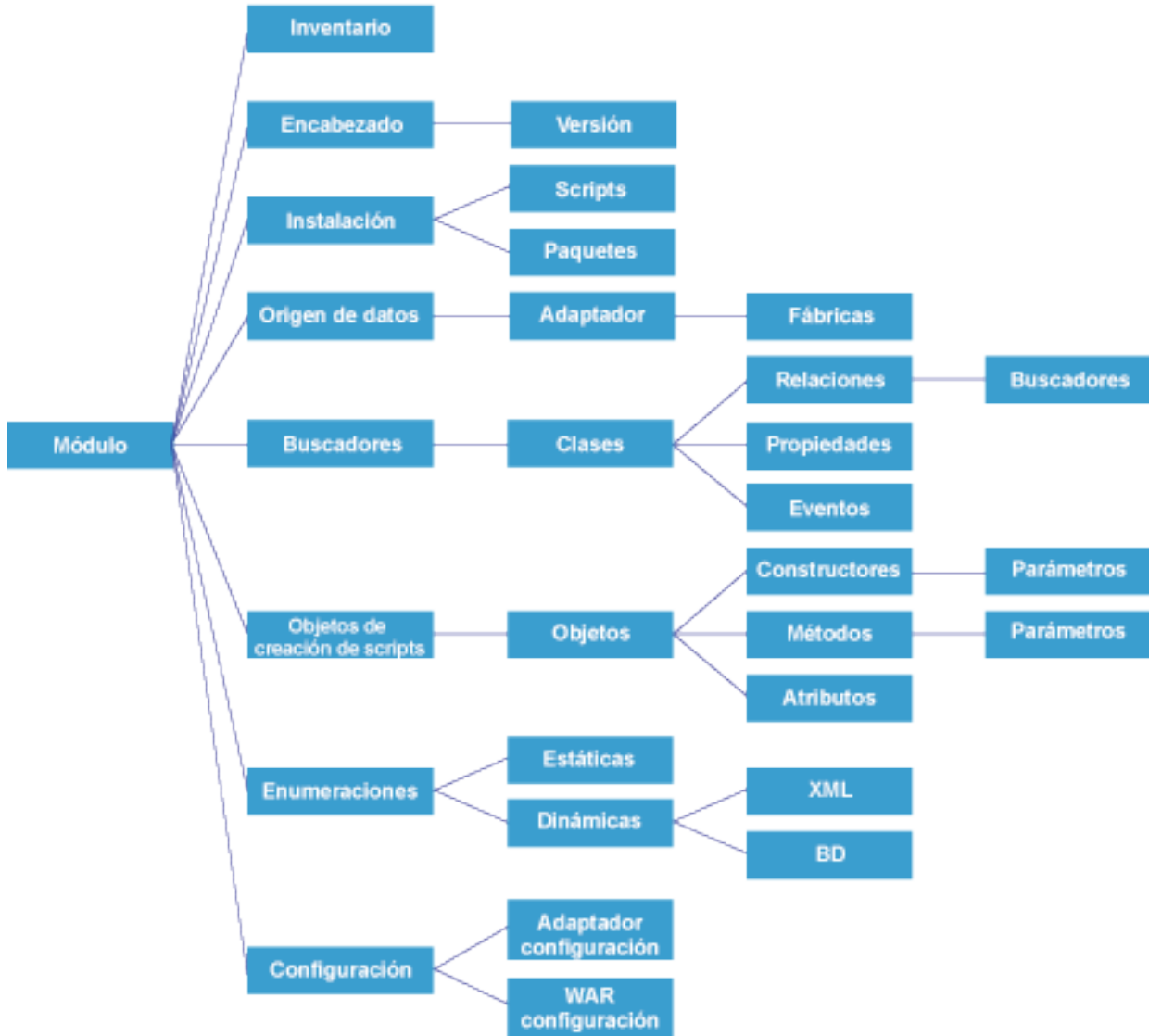
El archivo `vso.xml` define la manera que el servidor de Orchestrator tiene de interactuar con la tecnología conectada. Debe incluir una referencia a cada tipo de objeto o de operación para exponer a Orchestrator en el archivo `vso.xml`.

Los objetos que incluye en el archivo `vso.xml` aparecen como objetos de creación de scripts en la API de creación de scripts de Orchestrator o como objetos de buscador en la pestaña **Inventario** de Orchestrator.

Como parte de la arquitectura abierta y la implementación estandarizada de complementos, el archivo `vso.xml` debe tener un formato estándar.

El diagrama siguiente muestra el formato del archivo de definición del complemento `vso.xml` y cómo se anidan los elementos.

Figura 6-2. Formato del archivo de definición del complemento vso.xml



## Designar objetos de complemento

Se debe proporcionar un identificador exclusivo para cada objeto que encuentre el complemento en la tecnología conectada. Los nombres de objetos se definen en los elementos `<finder>` y `<object>` del archivo `vso.xml`.

Las operaciones del buscador que se definen en la implementación de fábrica buscan objetos en la tecnología conectada. Cuando el complemento encuentra objetos, puede utilizarlos en los flujos de trabajo de Orchestrator y transferirlos de un elemento de flujo de trabajo a otro. Los identificadores únicos que proporciona para los objetos permiten transferirlos entre los elementos de un flujo de trabajo.

El servidor de Orchestrator guarda solo el tipo y el identificador de cada objeto que procesa; no almacena ningún dato sobre el lugar o el modo en el que Orchestrator ha obtenido el objeto. Debe designar los objetos de forma coherente en la implementación del complemento para poder realizar un seguimiento de los objetos que se obtienen de los complementos.

Si el servidor de Orchestrator se detiene mientras los flujos de trabajo se ejecutan, cuando reinicia el servidor, los flujos de trabajo se reanudan en el elemento del flujo de trabajo que se estaba ejecutando cuando se detuvo el servidor. El flujo de trabajo utiliza identificadores para recuperar los objetos que el elemento estaba procesando cuando se detuvo el servidor.

## Convenciones de nomenclatura de objetos de complemento

Al asignar nombres a objetos de complementos, debe seguir las convenciones de las clases de Java.

**Importante** Debido al modo en que el motor de flujos de trabajo realiza la serialización de datos, no use las secuencias citadas a continuación en nombres de objeto. Si se usan esas secuencias de caracteres en los identificadores de objeto, el motor de flujos de trabajo analizará los flujos incorrectamente y la ejecución de flujos podría tener resultados imprevistos.

- #;#
- #,#
- #=#

Use estas directrices al asignar nombres a objetos de complementos:

- Use una inicial mayúscula para cada palabra del nombre.
- No use espacios para separar palabras.
- Como letras, use únicamente los caracteres estándar A - Z y a - z.
- No use caracteres especiales (por ejemplo, acentos).
- No use un número como primer carácter de un nombre.
- Si es posible, use menos de 10 caracteres.

La [Tabla 6-1. Reglas de nomenclatura de objetos de complemento](#) muestra reglas aplicables a tipos de objeto individuales.

**Tabla 6-1. Reglas de nomenclatura de objetos de complemento**

Tipo de objeto	Reglas de nomenclatura
Complemento	<ul style="list-style-type: none"> <li>■ Se define en el elemento <code>&lt;module&gt;</code> del archivo <code>vso.xml</code>.</li> <li>■ Debe ser conforme a las convenciones de nomenclatura de las clases de Java.</li> <li>■ Debe ser un nombre exclusivo. No se pueden ejecutar dos complementos con el mismo nombre en un servidor de Orchestrator.</li> </ul>
Objeto de buscador	<ul style="list-style-type: none"> <li>■ Se define en los elementos <code>&lt;finder&gt;</code> del archivo <code>vso.xml</code>.</li> <li>■ Debe ser conforme a las convenciones de nomenclatura de las clases de Java.</li> <li>■ Debe ser un nombre exclusivo en el complemento.</li> </ul> <p>Orchestrator añade el nombre del complemento y el signo de dos puntos a los nombres de objeto de buscador en los tipos de objeto de buscador de la API de creación de scripts de Orchestrator. Por ejemplo, el tipo de objeto <code>VirtualMachine</code> del complemento de vCenter Server aparece en la API de creación de scripts de Orchestrator como <code>VC:VirtualMachine</code>.</p>
Objeto de creación de scripts	<ul style="list-style-type: none"> <li>■ Se define en los elementos <code>&lt;scripting-object&gt;</code> del archivo <code>vso.xml</code>.</li> <li>■ Debe ser conforme a las convenciones de nomenclatura de las clases de Java.</li> <li>■ Debe ser un nombre exclusivo en el servidor de Orchestrator.</li> <li>■ Para evitar confundir objetos de script con objetos de buscador que tengan el mismo nombre, o con objetos de script de otros complementos, añada siempre como prefijo el nombre del complemento al nombre del objeto, pero sin un signo de dos puntos. Por ejemplo, la clase <code>VirtualMachine</code> del complemento de vCenter Server aparece en la API de creación de scripts de Orchestrator como la clase <code>VcVirtualMachine</code>.</li> </ul>

## Estructura de archivos del complemento

Un complemento debe seguir una estructura de archivos estándar, así como incluir determinados archivos y carpetas. Un complemento se entrega como archivo Java (JAR) o ZIP estándar, cuyo nombre se debe cambiar con la extensión `.dar`.

El contenido del archivo DAR debe seguir las convenciones siguientes de nomenclatura y de estructura de archivos.

Tabla 6-2. Estructura del archivo DAR

Carpetas	Descripción
<code>plug-in_name\VS0-INF\</code>	<p>Contiene el archivo <code>vso.xml</code> que define la asignación de los objetos en la tecnología conectada a objetos de Orchestrator. La carpeta <code>VS0-INF</code> y el archivo <code>vso.xml</code> son obligatorios.</p>
<code>plug-in_name\lib\</code>	<p>Contiene los archivos JAR con los archivos binarios de la tecnología conectada. También contiene archivos JAR con las implementaciones del adaptador, fábrica, controladores de notificaciones y otras interfaces del complemento. La carpeta <code>lib</code> y los archivos JAR son obligatorios.</p>
<code>plug-in_name\resources\</code>	<p>Contiene los archivos de recursos necesarios para el complemento. La carpeta <code>resources</code> puede incluir los tipos de elemento siguientes:</p> <ul style="list-style-type: none"> <li>■ Archivos de imagen, para representar los objetos del complemento en la pestaña <b>Inventario</b> de Orchestrator.</li> <li>■ Scripts, para definir el comportamiento de inicialización cuando se inicia el complemento.</li> <li>■ Paquetes de Orchestrator, que pueden contener acciones, flujos de trabajo personalizados y otros recursos que interactúan con los objetos a los que se accede mediante el complemento.</li> </ul> <p>Los recursos se pueden organizar en subcarpetas. Por ejemplo, <code>resources\images\</code>, <code>resources\scripts\</code> o <code>resources\packages\</code>.</p> <p>La carpeta <code>resources</code> es opcional.</p>

Utilice el centro de control de Orchestrator para importar un archivo DAR al servidor de Orchestrator.

## Referencia de API del complemento de Orchestrator

La API del complemento de Orchestrator define las clases y las interfaces de Java para implementar y extender cuando se desarrollan las implementaciones de `IPluginAdaptor` y `IPluginFactory` para crear un complemento.

Todas las clases están incluidas en el paquete `ch.dunes.vso.sdk.api`, a menos que se indique lo contrario.

### Interfaz IAop

La interfaz `IAop` proporciona métodos para obtener y definir propiedades sobre objetos en la tecnología conectada.

```
public interface IAop
```

La interfaz `IAop` define los siguientes métodos:



Método	Devuelve	Descripción
<code>get(java.lang.String propertyName, java.lang.Object object, java.lang.Object sdkObject)</code>	<code>java.lang.Object</code>	Obtiene una propiedad de un determinado objeto del complemento.
<code>set(java.lang.String propertyName, java.lang.String propertyValue, java.lang.Object object)</code>	Vacío	Define una propiedad de un determinado objeto del complemento.

## Interfaz IDynamicFinder

La interfaz IDynamicFinder devuelve el ID de las propiedades de un buscador mediante programación, en lugar de definir el ID y las propiedades en el archivo `vso.xml`.

La interfaz IDynamicFinder define los métodos siguientes:

Método	Devuelve	Descripción
<code>getIdAccessor(java.lang.String type)</code>	<code>java.lang.String</code>	Proporciona una expresión OGNL para obtener un ID de objeto mediante programación.
<code>getProperties(java.lang.String type)</code>	<code>java.util.List&lt;SDKFinderProperty&gt;</code>	Proporciona una lista de propiedades de objeto mediante programación.

## Interfaz IPluginAdaptor

La interfaz IPluginAdaptor se implementa para administrar fábricas, eventos y monitores del complemento. La interfaz IPluginAdaptor define un adaptador entre el complemento y el servidor de Orchestrator.

Las instancias de IPluginAdaptor se encargan de administrar las sesiones. La interfaz IPluginAdaptor define los métodos siguientes:

Método	Devuelve	Descripción
<code>addWatcher(PluginWatcher watcher)</code>	Vacío	Añade un monitor para supervisar un evento específico.
<code>createPluginFactory(java.lang.String sessionId, java.lang.String username, java.lang.String password, IPluginNotificationHandler notificationHandler)</code>	<code>IPluginFactory</code>	<p>Crea una instancia de <code>IPluginFactory</code>. El servidor de Orchestrator utiliza la fábrica para obtener objetos de la tecnología conectada por su ID, su relación con otros objetos, etcétera.</p> <p>El ID de sesión permite identificar una sesión que se está ejecutando. Por ejemplo, un usuario puede iniciar sesión en diferentes clientes de Orchestrator y ejecutar dos sesiones a la vez.</p> <p>De forma similar, el inicio de un flujo de trabajo crea una sesión independiente del cliente en el que se ha iniciado el flujo de trabajo. Un flujo de trabajo sigue ejecutándose aunque se cierre el cliente de Orchestrator.</p>
<code>installLicenses(PluginLicense[] licenses)</code>	Vacío	Instala la información de licencia de los complementos estándar que proporciona VMware.
<code>registerEventPublisher(java.lang.String type, java.lang.String id, IPluginEventPublisher publisher)</code>	Vacío	Establece los activadores y los medidores en un elemento del inventario.
<code>removeWatcher(java.lang.String watcherId)</code>	Vacío	Elimina un monitor.
<code>setPluginName(java.lang.String pluginName)</code>	Vacío	Obtiene el nombre del complemento del archivo <code>vso.xml</code> .
<code>setPluginPublisher(IPluginPublisher pluginPublisher)</code>	Vacío	Establece el publicador del complemento.
<code>uninstallPluginFactory(IPluginFactory plugin)</code>	Vacío	Desinstala una fábrica de complemento.
<code>unregisterEventPublisher(java.lang.String type, java.lang.String id, IPluginEventPublisher publisher)</code>	Vacío	Elimina los activadores y los medidores de un elemento del inventario.

## Interfaz `IPluginEventPublisher`

La interfaz `IPluginEventPublisher` publica medidores y activadores en un bus de notificación de eventos para la supervisión mediante políticas de Orchestrator.

Puede crear instancias de `IPluginEventPublisher` directamente en la implementación del adaptador de complementos o bien crearlas en clases de generadores de eventos independientes.

Puede implementar la interfaz `IPluginEventPublisher` para publicar eventos de la tecnología conectada en el motor de políticas de Orchestrator. Debe crear métodos para configurar medidores y activadores de políticas en objetos de la tecnología conectada y escuchas de eventos relacionados con esos objetos.

Las políticas pueden implementar medidores o activadores para supervisar objetos en la tecnología conectada. Los medidores de políticas supervisan los atributos de objetos e insertan un evento en el servidor de Orchestrator si los valores del objeto superan ciertos límites. Los activadores de políticas supervisan e insertan un evento en el servidor de Orchestrator si se produce un evento definido en el objeto. Debe registrar activadores y medidores de políticas con instancias `IPluginEventPublisher` para que las políticas de Orchestrator los supervisen.

La interfaz `IPluginEventPublisher` define los métodos siguientes:

Tipo	Devuelve	Descripción
<code>pushGauge(java.lang.String type, java.lang.String id, java.lang.String gaugeName, java.lang.String deviceName, java.lang.Double gaugeValue)</code>	Vacío	<p>Publica un medidor para la supervisión mediante políticas. Obtiene los parámetros siguientes:</p> <ul style="list-style-type: none"> <li>■ <code>type</code>: tipo de objeto que se supervisa.</li> <li>■ <code>id</code>: identificador del objeto que se supervisa.</li> <li>■ <code>gaugeName</code>: nombre de este medidor.</li> <li>■ <code>deviceName</code>: nombre del tipo de atributo que supervisa el medidor.</li> <li>■ <code>gaugeValue</code>: valor para el que el medidor supervisa el objeto.</li> </ul>
<code>pushTrigger(java.lang.String type, java.lang.String id, java.lang.String triggerName, java.util.Properties additionalProperties)</code>	Vacío	<p>Publica un activador para la supervisión mediante políticas. Obtiene los parámetros siguientes:</p> <ul style="list-style-type: none"> <li>■ <code>type</code>: tipo de objeto que se supervisa.</li> <li>■ <code>id</code>: identificador del objeto que se supervisa.</li> <li>■ <code>triggerName</code>: nombre de este activador.</li> <li>■ <code>additionalProperties</code>: cualquier propiedad adicional para la supervisión mediante el activador.</li> </ul>

## Interfaz `IPluginFactory`

`IPluginAdaptor` devuelve instancias de `IPluginFactory`. Las instancias de `IPluginFactory` ejecutan comandos en la aplicación conectada y busca objetos con los que realizar operaciones de Orchestrator.

La interfaz `IPluginFactory` define el campo siguiente:

```
static final java.lang.String RELATION_CHILDREN
```

La interfaz `IPluginFactory` define los métodos siguientes:

Método	Devuelve	Descripción
<code>executePluginCommand(java.lang.String cmd)</code>	Vacío	Utilice el complemento para ejecutar un comando. VMware recomienda no utilizar este método.
<code>find(java.lang.String type, java.lang.String id)</code>	<code>java.lang.Object</code>	Utilice un complemento para buscar un objeto. Identifique el objeto por su ID y su tipo.
<code>findAll(java.lang.String type, java.lang.String query)</code>	<code>QueryResult</code>	Utilice el complemento para buscar objetos de un tipo que coincidan con una cadena de consulta. La sintaxis de la consulta se define en la implementación <code>IPluginFactory</code> del complemento. Si no define la sintaxis de la consulta, <code>findAll()</code> devuelve todos los objetos del tipo especificado.
<code>findRelation(java.lang.String parentType, java.lang.String parentId, java.lang.String relationName)</code>	<code>java.util.List</code>	Determina si un objeto tiene elementos secundarios.
<code>hasChildrenInRelation(java.lang.String parentType, java.lang.String parentId, java.lang.String relationName)</code>	<code>HasChildrenResult</code>	Busca todos los elementos secundarios relativos a un determinado elemento principal mediante una relación concreta.
<code>invalidate(java.lang.String type, java.lang.String id)</code>	Vacío	Invalida los objetos por tipo y por ID.
<code>void invalidateAll()</code>	Vacío	Invalida todos los objetos de la caché.

## Interfaz IPluginNotificationHandler

`IPluginNotificationHandler` define métodos para notificar a Orchestrator los distintos tipos de evento que se producen en los objetos a los que accede Orchestrator a través del complemento.

La interfaz `IPluginNotificationHandler` define los métodos siguientes:

Método	Devuelve	Descripción
<code>getSessionID()</code>	<code>java.lang.String</code>	Devuelve el ID de sesión actual.
<code>notifyElementDeleted(java.lang.String type, java.lang.String id)</code>	Vacío	Notifica al sistema que se ha eliminado un objeto con el tipo y el ID especificados.
<code>notifyElementInvalidate(java.lang.String type, java.lang.String id)</code>	Vacío	Notifica al sistema que han cambiado las relaciones de un objeto. Puede utilizar el método <code>notifyElementInvalidate()</code> para notificar a Orchestrator todos los cambios en las relaciones entre objetos, no solo los cambios de relaciones que invalidan un objeto. Por ejemplo, cuando se añade un objeto secundario a uno principal, se produce un cambio en la relación entre los dos objetos.

Método	Devuelve	Descripción
<code>notifyElementUpdated(java.lang.String type, java.lang.String id)</code>	Vacío	Notifica al sistema que se han modificado los atributos de un objeto.
<code>notifyMessage(ch.dunes.vso.sdk.api.ErrorLevel severity, java.lang.String type, java.lang.String id, java.lang.String message)</code>	Vacío	Publica un mensaje de error en relación con el módulo actual.

## Interfaz IPluginPublisher

La interfaz `IPluginPublisher` publica un evento de monitor en un bus de notificación de eventos para la supervisión de los elementos Evento de espera de un flujo de trabajo de larga ejecución.

Cuando un activador de flujo de trabajo inicia un evento en la tecnología conectada, un monitor de complemento que supervisa el activador y está registrado con una instancia de `IPluginPublisher` notifica a cualquier flujo de trabajo en espera que el evento ha ocurrido.

La interfaz `IPluginPublisher` define el método siguiente:

Tipo	Valor	Descripción
<code>pushWatcherEvent(java.lang.String id, java.util.Properties properties)</code>	Vacío	Publica un evento de monitor en un bus de notificación de eventos.

## Interfaz WebConfigurationAdaptor

La interfaz `WebConfigurationAdaptor` implementa `IConfigurationAdaptor`, y define métodos para localizar e instalar una aplicación web en la pestaña de configuración de un complemento.

**Nota** La interfaz `WebConfigurationAdaptor` se ha dejado de usar desde Orchestrator 4.1. Para añadir una aplicación web a la configuración, implemente `IConfigurationAdaptor` y utilice el atributo `configuration-war` en el archivo `vso.xml` para identificar la aplicación web.

La interfaz `WebConfigurationAdaptor` define los siguientes métodos:

Método	Devuelve	Descripción
<code>getWebAppContext()</code>	String	Localiza el archivo WAR de la aplicación web para la pestaña de configuración. Proporciona el nombre y la ruta al archivo WAR desde el directorio <code>/webapps</code> del archivo DAR como cadena.
<code>setWebConfiguration(boolean webConfiguration)</code>	Booleano	Determina si el contenido de la pestaña de configuración se define a través de una aplicación web.

## Clase PluginTrigger

La clase `PluginTrigger` crea un módulo activador que obtiene información sobre los objetos y los eventos para supervisar en la tecnología conectada, en nombre de un elemento Evento de espera en un flujo de trabajo.

La clase `PluginTrigger` define métodos para obtener o configurar el tipo y el nombre del objeto que supervisar, la naturaleza del evento y un periodo de tiempo de espera.

Debe crear implementaciones de la clase `PluginTrigger` exclusivamente para los elementos Evento de espera de flujos de trabajo. Debe definir activadores de políticas para políticas de Orchestrator en clases que definen eventos e implementar el método `IPluginEventPublisher.pushTrigger()`.

```
public class PluginTrigger
    extends java.lang.Object
    implements java.io.Serializable
```

La clase `PluginTrigger` define los métodos siguientes:

Método	Devuelve	Descripción
<code>getModuleName()</code>	<code>java.lang.String</code>	Obtiene el nombre del módulo activador.
<code>getProperties()</code>	<code>java.util.Properties</code>	Obtiene una lista de las propiedades del activador.
<code>getSdkId()</code>	<code>java.lang.String</code>	Obtiene el ID del objeto para supervisar en la tecnología conectada.
<code>getSdkType()</code>	<code>java.lang.String</code>	Obtiene el tipo de objeto para supervisar en la tecnología conectada.
<code>getTimeout()</code>	Completo	Obtiene el periodo de tiempo de espera del activador.
<code>setModuleName(java.lang.String moduleName)</code>	Vacío	Establece el nombre del módulo activador.
<code>setProperties(java.util.Properties properties)</code>	Vacío	Establece una lista de las propiedades del activador.
<code>setSdkId(java.lang.String sdkId)</code>	Vacío	Establece el ID del objeto para supervisar en la tecnología conectada.
<code>setSdkType(java.lang.String sdkType)</code>	Vacío	Establece el tipo de objeto para supervisar en la tecnología conectada.
<code>setTimeout(long timeout)</code>	Vacío	Establece un periodo de tiempo de espera en segundos. Un valor negativo desactiva el tiempo de espera.

## Constructores

- `PluginTrigger()`
- `PluginTrigger(java.lang.String moduleName, long timeout, java.lang.String sdkType, java.lang.String sdkId)`

## Clase PluginWatcher

La clase `PluginWatcher` supervisa un módulo activador para un evento definido en la tecnología conectada en nombre de un elemento Evento de espera de un flujo de trabajo de larga ejecución.

La clase `PluginWatcher` define un constructor que usted puede usar para crear instancias de monitor del complemento. La clase `PluginWatcher` define métodos para obtener o configurar el nombre del activador de flujo de trabajo que supervisar y un periodo de tiempo de espera.

```
public class PluginWatcher
extends java.lang.Object
implements java.io.Serializable
```

La clase `PluginWatcher` define los métodos siguientes:

Método	Devuelve	Descripción
<code>getId()</code>	<code>java.lang.String</code>	Obtiene el ID del activador.
<code>getModuleName()</code>	<code>java.lang.String</code>	Obtiene el nombre del módulo activador.
<code>getTimeoutDate()</code>	Completo	Obtiene la fecha de tiempo de espera del activador.
<code>getTrigger()</code>	Vacío	Obtiene un activador.
<code>setId(java.lang.String id)</code>	Vacío	Establece el ID del activador.
<code>setTimeoutDate()</code>	Vacío	Establece la fecha de tiempo de espera del activador.

## Constructor

`PluginWatcher(PluginTrigger trigger)`

## Clase QueryResult

La clase `QueryResult` contiene los resultados de una consulta `find` realizada en los objetos a los que accede Orchestrator a través del complemento.

```
public class QueryResult
extends java.lang.Object
implements java.io.Serializable
```

El valor `totalCount` puede ser mayor que el número de elementos que devuelve `QueryResult`, si el número total de resultados encontrados es mayor que el número de resultados que devuelve la consulta. El número de resultados que devuelve la consulta se define en la sintaxis de consulta en el archivo `vso.xml`.

La clase `QueryResult` define los métodos siguientes:

Método	Devuelve	Descripción
<code>addElement(java.lang.Object element)</code>	Vacío	Añade un elemento a <code>QueryResult</code> .
<code>addElements(java.util.List elements)</code>	Vacío	Añade una lista de elementos a <code>QueryResult</code> .
<code>getElements()</code>	<code>java.util.List</code>	Obtiene elementos de la aplicación conectada.
<code>getTotalCount()</code>	Completo	Obtiene un recuento de todos los elementos disponibles en la tecnología conectada.
<code>isPartialResult()</code>	Booleano	Determina si el resultado obtenido está completo.
<code>removeElement(java.lang.Object element)</code>	Vacío	Elimina un elemento de la tecnología conectada.
<code>setElements(java.util.List elements)</code>	Vacío	Establece los elementos en la tecnología conectada.
<code>setTotalCount(long totalCount)</code>	Vacío	Establece el número total de elementos disponibles en la tecnología conectada.

## Constructores

- `QueryResult()`
- `QueryResult(java.util.List ret)`
- `QueryResult(java.util.List elements, long totalCount)`

## Clase SDKFinderProperty

La clase `SDKFinderProperty` define métodos para obtener y establecer propiedades en los objetos encontrados por los objetos de buscador de Orchestrator en la tecnología conectada. El método `IDynamicFinder.getProperties` devuelve objetos `SDKFinderProperty`.

```
public class SDKFinderProperty
extends java.lang.Object
```

La clase `SDKFinderProperty` define los siguientes métodos:

Método	Devuelve	Descripción
<code>getAttributeName()</code>	<code>java.lang.String</code>	Obtiene un nombre de atributo de objeto
<code>getBeanProperty()</code>	<code>java.lang.String</code>	Obtiene propiedades de un bean de Java
<code>getDescription()</code>	<code>java.lang.String</code>	Obtiene una descripción de objeto
<code>getDisplayName()</code>	<code>java.lang.String</code>	Obtiene un nombre para mostrar de objeto
<code>getPossibleResultType()</code>	<code>java.lang.String</code>	Obtiene los tipos posibles de resultado que devuelve el buscador



Método	Devuelve	Descripción
<code>getPropertyAccessor()</code>	<code>java.lang.String</code>	Obtiene un descriptor de propiedades de objeto
<code>getPropertyAccessorTree()</code>	<code>java.lang.Object</code>	Obtiene un árbol de descriptor de propiedades de objeto
<code>isHidden()</code>	Booleano	Muestra u oculta el objeto
<code>isShowInColumn()</code>	Booleano	Muestra u oculta el objeto en la columna de base de datos
<code>isShowInDescription()</code>	Booleano	Muestra u oculta la descripción del objeto
<code>setAttributeName(java.lang.String attributeName)</code>	Vacío	Establece un nombre de atributo de objeto
<code>setBeanProperty(java.lang.String beanProperty)</code>	Vacío	Establece propiedades en un bean de Java
<code>setDescription(java.lang.String description)</code>	Vacío	Establece una descripción de objeto
<code>setDisplayName(java.lang.String displayName)</code>	Vacío	Establece un nombre para mostrar de objeto
<code>setHidden(boolean hidden)</code>	Vacío	Muestra u oculta el objeto
<code>setPossibleResultType(java.lang.String possibleResultType)</code>	Vacío	Establece los tipos posibles de resultado que devuelve el buscador
<code>setPropertyAccessor(java.lang.String propertyAccessor)</code>	Vacío	Establece un descriptor de propiedades de objeto
<code>setPropertyAccessorTree(java.lang.Object propertyAccessorTree)</code>	Vacío	Establece un árbol de descriptor de propiedades de objeto
<code>setShowInColumn(boolean showInTable)</code>	Vacío	Muestra u oculta el objeto en la columna de base de datos
<code>setShowInDescription(boolean showInDescription)</code>	Vacío	Muestra u oculta la descripción del objeto

## Constructor

`SDKFinderProperty(java.lang.String attributeName, java.lang.String displayName, java.lang.String beanProperty, java.lang.String propertyAccessor)`

## Clase PluginExecutionException

La clase `PluginExecutionException` devuelve un mensaje de error si el complemento detecta una excepción cuando ejecuta una operación.

```
public class PluginExecutionException
extends java.lang.Exception
implements java.io.Serializable
```

La clase `PluginExecutionException` hereda los métodos siguientes de class `java.lang.Throwable`:

fillInStackTrace, getCause, getLocalizedMessage, getMessage, getStackTrace, initCause, printStackTrace, printStackTrace, printStackTrace, setStackTrace, toStringfillInStackTrace, getCause, getLocalizedMessage, getMessage, getStackTrace, initCause, printStackTrace

## Constructor

PluginExecutionException(java.lang.String message)

## Clase PluginOperationException

La clase PluginOperationException controla los errores detectados durante el funcionamiento de un complemento.

```
public class PluginOperationException
extends java.lang.RuntimeException
implements java.io.Serializable
```

La clase PluginOperationException hereda los métodos siguientes de class java.lang.Throwable:

fillInStackTrace, getCause, getLocalizedMessage, getMessage, getStackTrace, initCause, printStackTrace, printStackTrace, printStackTrace, setStackTrace, toString

## Constructor

PluginOperationException(java.lang.String message)

## Enumeración HasChildrenResult

La enumeración HasChildrenResult declara si un determinado elemento principal tiene elementos secundarios. El método IPluginFactory.hasChildrenInRelation devuelve objetos HasChildrenResult.

```
public enum HasChildrenResult
extends java.lang.Enum<HasChildrenResult>
implements java.io.Serializable
```

La enumeración HasChildrenResult define las constantes siguientes:

- public static final HasChildrenResult Yes
- public static final HasChildrenResult No
- public static final HasChildrenResult Unknown

La enumeración HasChildrenResult define los métodos siguientes:

Método	Devuelve	Descripción
getValue()	int	Devuelve uno de los valores siguientes: <div> <div>1</div> <div>El elemento principal tiene elementos secundarios</div> </div> <div> <div>-1</div> <div>El elemento principal no tiene elementos secundarios</div> </div> <div> <div>0</div> <div>Parámetro desconocido o no válido</div> </div>
valueOf(java.lang.String name)	static HasChildrenResult	Devuelve una constante de enumeración de este tipo con el nombre especificado. La cadena debe coincidir exactamente con el identificador utilizado para declarar una constante de enumeración de este tipo. No utilice caracteres de espacio en blanco en el nombre de la enumeración.
values()	static HasChildrenResult[]	Devuelve una matriz que incluye las constantes de este tipo de enumeración, en el orden en que se declaran. Este método se puede iterar en constantes, como se indica a continuación: <div> <pre>for (HasChildrenResult c : HasChildrenResult.values()) System.out.println(c);</pre> </div>

La enumeración `HasChildrenResult` hereda los métodos siguientes de class `java.lang.Enum`:

`clone`, `compareTo`, `equals`, `finalize`, `getDeclaringClass`, `hashCode`, `name`, `ordinal`, `toString`, `valueOf`

## Tipo de anotación `ScriptingAttribute`

El tipo de anotación `ScriptingAttribute` anota un atributo de un objeto en la tecnología conectada para utilizar como propiedad en la creación de scripts.

```
@Retention(value=RUNTIME)
@Target(value={METHOD, FIELD})
public @interface ScriptingAttribute
```

El tipo de anotación `ScriptingAttribute` tiene el valor siguiente:

```
public abstract java.lang.String value
```

## Tipo de anotación ScriptingFunction

El tipo de anotación `ScriptingFunction` anota un método para utilizar como propiedad en la creación de scripts.

```
@Retention(value=RUNTIME)
@Target(value={METHOD, CONSTRUCTOR})
public @interface ScriptingFunction
```

El tipo de anotación `ScriptingFunction` tiene el valor siguiente:

```
public abstract java.lang.String value
```

## Tipo de anotación ScriptingParameter

El tipo de anotación `ScriptingParameter` anota un parámetro para utilizar como propiedad en la creación de scripts.

```
@Retention(value=RUNTIME)
@Target(value=PARAMETER)
public @interface ScriptingParameter
```

El tipo de anotación `ScriptingParameter` tiene el valor siguiente:

```
public abstract java.lang.String value
```

## Elementos del archivo de definición del complemento vso.xml

El archivo `vso.xml` contiene un conjunto de flujos de trabajo estándar. Algunos de los elementos son obligatorios y otros son opcionales. Cada elemento tiene atributos que definen valores para los objetos y las operaciones que el usuario asigna a objetos y operaciones de Orchestrator.

Además, los elementos pueden tener varios elementos secundarios o ninguno. Un elemento secundario define el elemento principal con más detalle. El mismo elemento secundario puede aparecer en diferentes elementos principales. Por ejemplo, el elemento `description` carece de elementos secundarios, pero aparece como elemento secundario en muchos elementos principales: `module`, `example`, `trigger`, `gauge`, `finder`, `constructor`, `method`, `object` y `enumeration`.

Cada definición de elemento que sigue enumera sus atributos, elementos superiores y secundarios.

### Elemento module

Un módulo describe un conjunto de objetos de complemento que se ponen a disposición para Orchestrator.

El módulo contiene información sobre el modo en el que los datos de la tecnología conectada se asignan a las clases de Java, las versiones, el modo en el que se implementa el módulo y el modo en el que aparece el complemento en el inventario de Orchestrator.

El elemento `<module>` es opcional. El elemento `<module>` cuenta con los atributos siguientes:

Atributos	Valor	Descripción
name	String	Define el tipo de todos los elementos <code>&lt;finder&gt;</code> en el complemento. Atributo obligatorio.
version	Número	Número de versión del complemento; se utiliza cuando se vuelven a cargar los paquetes en una nueva versión del complemento. Atributo obligatorio.
build-number	Número	Número de compilación del complemento; se utiliza cuando se vuelven a cargar los paquetes en una nueva versión del complemento. Atributo obligatorio.
image	Archivo de imagen	Icono que se mostrará en el inventario de Orchestrator. Atributo obligatorio.
display-name	String	Nombre que aparece en el inventario de Orchestrator. Atributo opcional.
interface-mapping-allowed	true o false	VMware desaconseja la asignación de interfaces. Atributo opcional.

**Tabla 6-3. Jerarquía de elementos**

Elemento principal	Elementos secundarios
Ninguno	<ul style="list-style-type: none"> <li>■ <code>&lt;description&gt;</code></li> <li>■ <code>&lt;installation&gt;</code></li> <li>■ <code>&lt;configuration&gt;</code></li> <li>■ <code>&lt;finder-datasources&gt;</code></li> <li>■ <code>&lt;inventory&gt;</code></li> <li>■ <code>&lt;finders&gt;</code></li> <li>■ <code>&lt;scripting-objects&gt;</code></li> <li>■ <code>&lt;enumerations&gt;</code></li> </ul>

## Elemento description

El elemento `<description>` proporciona descripciones de los elementos del complemento que aparece en la documentación del Explorador de API.

Debe añadir el texto que aparece en la documentación del Explorador de API entre las etiquetas `<description>` y `</description>`.

El elemento `<description>` es opcional. El elemento `<description>` no tiene atributos.

Tabla 6-4. Jerarquía de elementos

Elementos principales	Elementos secundarios
<ul style="list-style-type: none"> <li>■ &lt;module&gt;</li> <li>■ &lt;example&gt;</li> <li>■ &lt;trigger&gt;</li> <li>■ &lt;gauge&gt;</li> <li>■ &lt;finder&gt;</li> <li>■ &lt;constructor&gt;</li> <li>■ &lt;method&gt;</li> <li>■ &lt;object&gt;</li> <li>■ &lt;enumeration&gt;</li> </ul>	Ninguno

## Elemento deprecated

El elemento `<deprecated>` marca objetos y métodos que han quedado obsoletos en la documentación del Explorador de API.

Debe añadir el texto que aparece en la documentación del Explorador de API entre las etiquetas `<deprecated>` y `</deprecated>`.

El elemento `<deprecated>` es opcional. El elemento `<deprecated>` no tiene atributos.

Tabla 6-5. Jerarquía de elementos

Elementos principales	Elementos secundarios
<ul style="list-style-type: none"> <li>■ &lt;method&gt;</li> <li>■ &lt;object&gt;</li> </ul>	Ninguno

## Elemento url

El elemento `<url>` proporciona una URL que apunta a documentación externa sobre un objeto o una enumeración.

La URL se proporciona entre las etiquetas `<url>` y `</url>`.

El elemento `<url>` es opcional. El elemento `<url>` no tiene atributos.

Tabla 6-6. Jerarquía de elementos

Elementos principales	Elementos secundarios
<ul style="list-style-type: none"> <li>■ &lt;enumeration&gt;</li> <li>■ &lt;object&gt;</li> </ul>	Ninguno

## Elemento installation

El elemento `<installation>` permite instalar un paquete o ejecutar un script cuando se inicia el servidor.

El elemento `<installation>` es opcional. El elemento `<installation>` tiene los atributos siguientes:

Atributos	Valor	Descripción
mode	always, never o version	La configuración del valor mode produce el comportamiento siguiente cuando se inicia el servidor de Orchestrator: <ul style="list-style-type: none"> <li>■ Se ejecuta la acción always</li> <li>■ Se ejecuta la acción never</li> <li>■ La acción se ejecuta cuando el servidor detecta una versión más reciente del complemento</li> </ul> Atributo obligatorio.

Tabla 6-7. Jerarquía de elementos

Elemento principal	Elemento secundario
<module>	<action>

## Elemento action

El elemento <action> especifica la acción que se ejecuta al iniciar el servidor de Orchestrator.

Los atributos del elemento <action> proporcionan la ruta del paquete de Orchestrator que define el comportamiento del complemento cuando se inicia.

El elemento <action> es opcional. Un complemento puede tener un número ilimitado de elementos <action>. El elemento <action> tiene los siguientes atributos.

Atributos	Valor	Descripción
resource	String	La ruta del script o del paquete de Java desde la raíz del archivo dar. Atributo obligatorio.
type	install-package o execute-script	Instala el paquete de Orchestrator indicado en el servidor de Orchestrator o bien ejecuta el script especificado. Atributo obligatorio.

Tabla 6-8. Jerarquía de elementos

Elemento principal	Elementos secundarios
<installation>	Ninguno

## Elemento finder-datasources

El elemento <finder-datasources> es el contenedor de los elementos <finder-datasource>.

El elemento <finder-datasources> es opcional. El elemento <finder-datasources> no tiene atributos.

Tabla 6-9. Jerarquía de elementos

Elemento principal	Elementos secundarios
<module>	<finder-datasource>

## Elemento finder-datasource

El elemento `<finder-datasource>` apunta al archivo de clase de Java de la implementación `IPluginAdaptor` que creó para el complemento.

Puede definir cómo Orchestrator accede a los objetos de la tecnología conectada en el elemento `<finder-datasource>`. El elemento `<finder-datasource>` identifica la clase de Java del adaptador de complementos que ha creado. La clase del adaptador de complementos crea instancias de la fábrica de complementos que ha creado. La fábrica de complementos define métodos que buscan objetos en la tecnología conectada. Puede definir tiempos de espera en el elemento `<finder-datasource>` para las llamadas de método de buscador que realiza la fábrica. En la interfaz `IPluginFactory` se aplican distintos tiempos de espera para los distintos métodos de buscador.

El elemento `<finder-datasource>` es opcional. Un complemento puede tener un número ilimitado de elementos `<finder-datasources>`. El elemento `<finder-datasource>` tiene los siguientes atributos.

Atributos	Valor	Descripción
<code>name</code>	String	Identifica la fuente de los datos del elemento <code>&lt;finder&gt;</code> , atributos <code>datasource</code> . Equivalente a un id de XML. Atributo obligatorio.
<code>adaptor-class</code>	Clase de Java	Apunta a la implementación <code>IPluginAdaptor</code> que usted define para crear el adaptador de complementos, por ejemplo <code>com.vmware.plugins.sample.Adaptor</code> . Atributo obligatorio.
<code>concurrent-call</code>	<code>true</code> (predeterminado) o <code>false</code>	Permite a varios usuarios acceder al adaptador al mismo tiempo. Debe configurar <code>concurrent-call</code> como <code>false</code> si el complemento no admite llamadas simultáneas. Atributo opcional.
<code>invoker-mode</code>	<code>direct</code> (predeterminado) o <code>timeout</code>	Establece un tiempo de espera en la función buscador. Si se configura como <code>direct</code> , el tiempo de espera de las llamadas a las funciones de buscador nunca se agota. Si se configura como <code>timeout</code> , el servidor de Orchestrator aplica el periodo de tiempo de espera que corresponde al método de buscador. Atributo opcional.
<code>anonymous-login-mode</code>	<code>never</code> (predeterminado) o <code>always</code>	Pasa (o no) el nombre y la contraseña del usuario al complemento. Atributo opcional.
<code>timeout-fetch-relation</code>	Número; de manera predeterminada, 30 segundos	Se aplica a las llamadas de <code>findRelation()</code> . Atributo opcional.
<code>timeout-find-all</code>	Número; de manera predeterminada, 60 segundos	Se aplica a las llamadas de <code>findAll()</code> . Atributo opcional.



Atributos	Valor	Descripción
timeout-find	Número; de manera predeterminada, 60 segundos	Se aplica a las llamadas de find(). Atributo opcional.
timeout-has-children-in-relation	Número; de manera predeterminada, 2 segundos	Se aplica a las llamadas de findChildrenInRelation(). Atributo opcional.
timeout-execute-plugin-command	Número; de manera predeterminada, 30 segundos	Se aplica a las llamadas de executePluginCommand(). Atributo opcional.

Tabla 6-10. Jerarquía de elementos

Elemento principal	Elementos secundarios
<finder-datasources>	Ninguno

## Elemento inventory

El elemento <inventory> define la raíz de la lista jerárquica del complemento que aparece en los cuadros de diálogo de selección de objetos y en la vista **Inventario** del cliente de Orchestrator.

El elemento <inventory> no representa un objeto en la aplicación conectada, sino el propio complemento como objeto de la API de creación de scripts de Orchestrator.

El elemento <inventory> es opcional. El elemento <inventory> tiene el atributo siguiente.

Atributos	Valor	Descripción
type	Un tipo de objeto de Orchestrator	El tipo de elemento <finder> que representa la raíz de la jerarquía de objetos. Atributo obligatorio.

Tabla 6-11. Jerarquía de elementos

Elemento principal	Elementos secundarios
<module>	Ninguno

## Elemento finders

El elemento <finders> es el contenedor de todos los elementos <finder>.

El elemento <finders> es opcional. El elemento <finders> no tiene atributos.

Tabla 6-12. Jerarquía de elementos

Elemento principal	Elemento secundario
<module>	<finder>

## Elemento finder

El elemento `<finder>` representa en el cliente de Orchestrator un tipo de objeto encontrado a través del complemento.

El elemento `<finder>` identifica la clase de Java que define el objeto que representa el buscador de objetos. El elemento `<finder>` define el modo en el que el objeto aparece en la interfaz del cliente de Orchestrator. También identifica el objeto de creación de scripts que define la API de creación de scripts de Orchestrator para representar este objeto.

Los buscadores actúan como interfaz entre formatos de objeto utilizados por diferentes tipos de tecnologías conectadas.

El elemento `<finder>` es opcional. Un complemento puede tener un número ilimitado de elementos `<finder>`. El elemento `<finder>` define los atributos siguientes:

Atributos	Valor	Descripción
type	Un tipo de objeto de Orchestrator	Tipo de objeto representado por el buscador. Atributo obligatorio.
datasource	Atributo <code>&lt;finder-datasource name&gt;</code>	Identifica la clase de Java que define el objeto utilizando el origen de datos refid. Atributo obligatorio.
dynamic-finder	Método de Java	Define el método de buscador personalizado que se implementa en una instancia de <code>IDynamicFinder</code> , para devolver el ID y las propiedades de un buscador mediante programación, en lugar de definirlo en el archivo <code>vso.xml</code> . Atributo opcional.
hidden	true o false (predeterminado)	Si es true, oculta el buscador en el cliente de Orchestrator. Atributo opcional.
image	Ruta a un archivo gráfico	Un icono de 16x16 para representar el buscador en listas jerárquicas en el cliente de Orchestrator. Atributo opcional.
java-class	Nombre de una clase de Java	La clase de Java que define el objeto que el buscador busca y asigna a un objeto de creación de scripts. Atributo opcional.
script-object	Atributo <code>&lt;scripting-object type&gt;</code>	El tipo <code>&lt;scripting-object&gt;</code> , si lo hay, al que se asigna este buscador. Atributo opcional.

Tabla 6-13. Jerarquía de elementos

Elemento principal	Elementos secundarios
<finders>	<ul style="list-style-type: none"> <li>■ &lt;id&gt;</li> <li>■ &lt;description&gt;</li> <li>■ &lt;properties&gt;</li> <li>■ &lt;default-sorting&gt;</li> <li>■ &lt;inventory-children&gt;</li> <li>■ &lt;relations&gt;</li> <li>■ &lt;inventory-tabs&gt;</li> <li>■ &lt;events&gt;</li> </ul>

## Elemento properties

El elemento <properties> es el contenedor de los elementos <finder><property>.

El elemento <properties> es opcional. El elemento <properties> no tiene atributos.

Tabla 6-14. Jerarquía de elementos

Elemento principal	Elemento secundario
<finder>	<property>

## Elemento property

El elemento <property> asigna las propiedades del objeto encontrado a las llamadas de método o las propiedades de Java.

Puede llamar a los métodos de la clase SDKFinderProperty cuando implementa la fábrica de complementos para obtener propiedades para el procesamiento de la implementación de la fábrica de complementos.

En las vistas del cliente de Orchestrator, puede mostrar u ocultar las propiedades de los objetos. Asimismo, puede utilizar enumeraciones para definir propiedades de objetos.

El elemento <property> es opcional. Un complemento puede tener un número ilimitado de elementos <property>. El elemento <property> tiene los atributos siguientes.

Atributos	Valor	Descripción
name	Nombre de buscador	El nombre que utiliza FinderResult para almacenar el elemento. Atributo obligatorio.
display-name	Nombre de buscador	El nombre de propiedad que se muestra. Atributo opcional.

Atributos	Valor	Descripción
bean-property	Nombre de propiedad	El atributo bean-property permite identificar una propiedad que se obtiene mediante las operaciones get y set. Si identifica una propiedad llamada MyProperty, el complemento define las operaciones getMyProperty y setMyProperty.  Puede establecer bean-property o property-accessor, pero no las dos a la vez. Atributo opcional.
property-accessor	El método que obtiene un valor de propiedad de un objeto	El atributo property-accessor permite definir una expresión OGNL para validar las propiedades de un objeto.  Puede establecer bean-property o property-accessor, pero no las dos a la vez. Atributo opcional.
show-in-column	true (predeterminado) o false	Si es true, esta propiedad aparece en la tabla de resultados del cliente de Orchestrator. Atributo opcional.
show-in-description	true (predeterminado) o false	Si es true, esta propiedad aparece en la descripción del objeto. Atributo opcional.
hidden	true o false (predeterminado)	Si es true, esta propiedad se oculta en todos los casos. Atributo opcional.
linked-enumeration	Nombre de enumeración	Vincula una propiedad de buscador con una enumeración. Atributo opcional.

Tabla 6-15. Jerarquía de elementos

Elemento principal	Elementos secundarios
<properties>	Elementos secundarios

## Elemento relations

El elemento <relations> es el contenedor de los elementos <finder><relation>.

El elemento <relations> es opcional. El elemento <relations> no tiene atributos.

Tabla 6-16. Jerarquía de elementos

Elemento principal	Elemento secundario
<finder>	<relation>

## Elemento relation

El elemento <relation> define el modo en el que los objetos se relacionan con otros objetos.

En el elemento <relation> se define el nombre de la relación.

El elemento `<relation>` es opcional. Un complemento puede tener un número ilimitado de elementos `<relation>`. El elemento `<relation>` tiene los atributos siguientes.

Atributos	Valor	Descripción
name	Nombre de relación	Nombre de esta relación. Atributo obligatorio.
type	Tipo de objeto de Orchestrator	Tipo de objeto que se relaciona con otro objeto mediante esta relación. Atributo obligatorio.
cardinality	to-one o to-many	Define la relación entre los objetos como de uno a uno o de uno a varios. Atributo opcional.

**Tabla 6-17. Jerarquía de elementos**

Elemento principal	Elementos secundarios
<code>&lt;relations&gt;</code>	Ninguno

## Elemento id

El elemento `<id>` define un método para obtener el ID exclusivo del objeto identificado por el buscador.

El elemento `<id>` es opcional. El elemento `<id>` tiene los atributos siguientes.

Atributos	Valor	Descripción
accessor	Nombre de método	El atributo <code>accessor</code> permite definir una expresión OGNL para validar las propiedades de un objeto. Atributo obligatorio.

**Tabla 6-18. Jerarquía de elementos**

Elemento principal	Elementos secundarios
<code>&lt;finder&gt;</code>	Ninguno

## Elemento inventory-children

El elemento `<inventory-children>` define la jerarquía de las listas que muestran los objetos en los cuadros de selección de objetos y en la vista **Inventario** del cliente de Orchestrator.

El elemento `<inventory-children>` es opcional. El elemento `<inventory-children>` no tiene atributos.

**Tabla 6-19. Jerarquía de elementos**

Elemento principal	Elemento secundario
<code>&lt;finder&gt;</code>	<code>&lt;relation-link&gt;</code>

## Elemento relation-link

El elemento `<relation-link>` define las jerarquías entre los objetos principales y secundarios en la pestaña **Inventario**.

El elemento `<relation-link>` es opcional. Un complemento puede tener un número ilimitado de elementos `<relation-link>`. El elemento `<relation-link>` tiene el atributo siguiente.

Tipo	Valor	Descripción
name	Nombre de relación	refid para un nombre de relación. Atributo obligatorio.

**Tabla 6-20. Jerarquía de elementos**

Elemento principal	Elementos secundarios
<code>&lt;inventory-children&gt;</code>	Ninguno

## Elemento events

El elemento `<events>` es el contenedor de los elementos `<trigger>` y `<gauge>`.

El elemento `<events>` puede contener un número ilimitado de activadores o de medidores.

El elemento `<events>` es opcional. El elemento `<events>` no tiene atributos.

**Tabla 6-21. Jerarquía de elementos**

Elemento principal	Elementos secundarios
<code>&lt;finder&gt;</code>	<ul style="list-style-type: none"> <li>■ <code>&lt;trigger&gt;</code></li> <li>■ <code>&lt;gauge&gt;</code></li> </ul>

## Elemento trigger

El elemento `<trigger>` declara los activadores que se pueden utilizar para este buscador. Debe implementar los métodos `registerEventPublisher()` y `unregisterEventPublisher()` de `IPluginAdaptor` para establecer activadores.

El elemento `<trigger>` es opcional. El elemento `<trigger>` tiene el atributo siguiente.

Tipo	Valor	Descripción
name	Nombre del activador	Nombre para este activador. Atributo obligatorio.

**Tabla 6-22. Jerarquía de elementos**

Elemento principal	Elementos secundarios
<code>&lt;events&gt;</code>	<ul style="list-style-type: none"> <li>■ <code>&lt;description&gt;</code></li> <li>■ <code>&lt;trigger-properties&gt;</code></li> </ul>

## Elemento trigger-properties

El elemento `<trigger-properties>` es el contenedor de los elementos `<trigger-property>`.

El elemento `<trigger-properties>` es opcional. El elemento `<trigger-properties>` no tiene atributos.

**Tabla 6-23. Jerarquía de elementos**

Elemento principal	Elemento secundario
<code>&lt;trigger&gt;</code>	<code>&lt;trigger-property&gt;</code>

## Elemento trigger-property

El elemento `<trigger-property>` define las propiedades que identifican un objeto activador.

El elemento `<trigger-property>` es opcional. Un complemento puede tener un número ilimitado de elementos `<trigger-property>`. El elemento `<trigger-property>` tiene los atributos siguientes.

Tipo	Valor	Descripción
<code>name</code>	Nombre del activador	Nombre del activador. Atributo opcional.
<code>display-name</code>	Nombre del activador	Nombre que se muestra en el cliente de Orchestrator. Atributo opcional.
<code>type</code>	Tipo de activador	Tipo de objeto que define el activador. Atributo obligatorio.

**Tabla 6-24. Jerarquía de elementos**

Elemento principal	Elementos secundarios
<code>&lt;trigger-properties&gt;</code>	Ninguno

## Elemento gauge

El elemento `<gauge>` define los medidores que puede utilizar para este buscador. Debe implementar los métodos `registerEventPublisher()` y `unregisterEventPublisher()` de `IPluginAdaptor` para establecer medidores.

El elemento `<gauge>` es opcional. Un complemento puede tener un número ilimitado de elementos `<gauge>`. El elemento `<gauge>` tiene los atributos siguientes.

Tipo	Valor	Descripción
<code>name</code>	Nombre de medidor	Nombre del medidor. Atributo obligatorio.
<code>min-value</code>	Número	Límite mínimo. Atributo opcional.
<code>max-value</code>	Número	Límite máximo. Atributo opcional.
<code>unit</code>	Tipo de objeto	Tipo de objeto que define el medidor. Atributo obligatorio.
<code>format</code>	Cadena	Formato del valor supervisado. Atributo opcional.

Tabla 6-25. Jerarquía de elementos

Elemento principal	Elemento secundario
<events>	<description>

## Elemento scripting-objects

El elemento <scripting-objects> es el contenedor de los elementos <object>.

El elemento <scripting-objects> es opcional. El elemento <scripting-objects> no tiene atributos.

Tabla 6-26. Jerarquía de elementos

Elemento principal	Elemento secundario
<module>	<object>

## Elemento object

El elemento <object> asigna los constructores, los atributos y los métodos de la tecnología conectada a tipos de objetos de JavaScript que expone la API de creación de scripts de Orchestrator.

Consulte [Designar objetos de complemento](#) para obtener información sobre las convenciones de nomenclatura de los objetos.

El elemento <object> es opcional. Un complemento puede tener un número ilimitado de elementos <object>. El elemento <object> tiene los atributos siguientes.

Tipo	Valor	Descripción
script-name	Nombre de JavaScript	Nombre de creación de scripts de la clase. Debe ser único globalmente. Atributo obligatorio.
java-class	Clase de Java	Clase de Java incluida por esta clase de JavaScript. Atributo obligatorio.
create	true (predeterminado) o false	Si es true, puede crear una nueva instancia de esta clase. Atributo opcional.
strict	true o false (predeterminado)	Si es true, solo puede llamar a métodos que anote o declare en el archivo vso.xml. Atributo opcional.
is-deprecated	true o false (predeterminado)	Si es true, el objeto asigna una clase de Java obsoleta. Atributo opcional.
since-version	Cadena	Versión a partir de la cual la clase de Java queda obsoleta. Atributo opcional.



Tabla 6-27. Jerarquía de elementos

Elemento principal	Elementos secundarios
<scripting-objects>	<ul style="list-style-type: none"> <li>■ &lt;description&gt;</li> <li>■ &lt;deprecated&gt;</li> <li>■ &lt;url&gt;</li> <li>■ &lt;constructors&gt;</li> <li>■ &lt;attributes&gt;</li> <li>■ &lt;methods&gt;</li> <li>■ &lt;singleton&gt;</li> </ul>

## Elemento constructors

El elemento <constructors> es el contenedor de los elementos <object><constructor>.

El elemento <constructors> es opcional. El elemento <constructors> no tiene atributos.

Tabla 6-28. Jerarquía de elementos

Elemento principal	Elemento secundario
<object>	<constructor>

## Elemento constructor

El elemento <constructor> define un método de constructor. El método de <constructor> produce documentación en el Explorador de API.

El elemento <constructor> es opcional. Un complemento puede tener un número ilimitado de elementos <constructor>. El elemento <constructor> no tiene atributos.

Tabla 6-29. Jerarquía de elementos

Elemento principal	Elementos secundarios
<constructors>	<ul style="list-style-type: none"> <li>■ &lt;description&gt;</li> <li>■ &lt;parameters&gt;</li> </ul>

## Elemento parameters del constructor

El elemento <parameters> es el contenedor de los elementos <constructor><parameter>.

El elemento <parameters> es opcional. El elemento <parameters> no tiene atributos.

Tabla 6-30. Jerarquía de elementos

Elemento principal	Elemento secundario
<constructor>	<parameter>

## Elemento parameter del constructor

El elemento <parameter> define los parámetros del constructor.

El elemento `<parameter>` es opcional. Un complemento puede tener un número ilimitado de elementos `<parameter>`. El elemento `<parameter>` tiene los atributos siguientes.

Tipo	Valor	Descripción
name	Cadena	Nombre de parámetro que se utiliza en la documentación de API. Atributo obligatorio.
type	Tipo de parámetro de Orchestrator	Tipo de parámetro que se utiliza en la documentación de API. Atributo obligatorio.
is-optional	true o false	Si está establecido como true, el valor puede ser nulo. Atributo opcional.
since-version	Cadena	Versión de método. Atributo opcional.

**Tabla 6-31. Jerarquía de elementos**

Elemento principal	Elementos secundarios
<code>&lt;parameters&gt;</code>	Ninguno

## Elemento attributes

El elemento `<attributes>` es el contenedor de los elementos `<object>``<attribute>`.

El elemento `<attributes>` es opcional. El elemento `<attributes>` no tiene atributos.

**Tabla 6-32. Jerarquía de elementos**

Elemento principal	Elemento secundario
<code>&lt;object&gt;</code>	<code>&lt;attribute&gt;</code>

## Elemento attribute

El elemento `<attribute>` asigna los atributos de una clase de Java de la tecnología conectada a atributos disponibles mediante el motor de JavaScript de Orchestrator.

El elemento `<attribute>` es opcional. Un complemento puede tener un número ilimitado de elementos `<attribute>`. El elemento `<attribute>` tiene los siguientes atributos.

Tipo	Valor	Descripción
java-name	Atributo de Java	Nombre del atributo de Java. Atributo obligatorio.
script-name	Objeto de JavaScript	Nombre del objeto de JavaScript correspondiente. Atributo obligatorio.

Tipo	Valor	Descripción
return-type	Cadena	Tipo de objeto devuelto por este atributo. Aparece en la documentación del Explorador de API. Atributo opcional.  <b>Nota</b> Si el tipo de valor devuelto de JavaScript es Properties, las implementaciones de Java subyacentes admitidas son java.util.HashMap y java.util.Hashtable.
read-only	true o false	Si es true, no puede modificar este atributo. Atributo opcional.
is-optional	true o false	Si es true, este campo puede ser nulo. Atributo opcional.
show-in-api	true o false	Si es false, este atributo no aparece en la documentación de la API. Atributo opcional.
is-deprecated	true o false	Si es true, el objeto asigna un atributo obsoleto. Atributo opcional.
since-version	Número	Versión en la que el atributo quedó obsoleto. Atributo opcional.

Tabla 6-33. Jerarquía de elementos

Elemento principal	Elementos secundarios
<attributes>	Ninguno

## Elemento methods

El elemento <methods> es el contenedor de los elementos <object><method>.

El elemento <methods> es opcional. El elemento <methods> no tiene atributos.

Tabla 6-34. Jerarquía de elementos

Elemento principal	Elemento secundario
<object>	<method>

## Elemento method

El elemento <method> asigna un método de Java de la tecnología conectada a un método de JavaScript que exponga el motor de JavaScript de Orchestrator.

El elemento <method> es opcional. Un complemento puede tener un número ilimitado de elementos <method>. El elemento <method> tiene los atributos siguientes.

Tipo	Valor	Descripción
java-name	Método de Java	Nombre de la firma del método de Java con tipos de argumento entre paréntesis, por ejemplo <code>getVms(DataStore)</code> . Atributo obligatorio.
script-name	Método de JavaScript	Nombre del método de JavaScript correspondiente. Atributo obligatorio.
return-type	Tipo de objeto de Java	El tipo que obtiene este método. Atributo opcional.  <b>Nota</b> Si el tipo de valor devuelto de JavaScript es <code>Properties</code> , las implementaciones de Java subyacentes admitidas son <code>java.util.HashMap</code> y <code>java.util.Hashtable</code> .
static	true o false	Si es true, este método es estático. Atributo opcional.
show-in-api	true o false	Si es false, este método no aparece en la documentación de la API. Atributo opcional.
is-deprecated	true o false	Si es true, el objeto asigna un método en desuso. Atributo opcional.
since-version	Número	La versión en la que el método queda en desuso. Atributo opcional.

Tabla 6-35. Jerarquía de elementos

Elemento principal	Elementos secundarios
<methods>	<ul style="list-style-type: none"> <li>■ &lt;deprecated&gt;</li> <li>■ &lt;description&gt;</li> <li>■ &lt;example&gt;</li> <li>■ &lt;parameters&gt;</li> </ul>

## Elemento de ejemplo

El elemento <example> le permite añadir ejemplos de código a los métodos de Javascript que aparecen en la documentación del Explorador de API.

El elemento <example> es opcional. El elemento <example> no tiene atributos.

Tabla 6-36. Jerarquía de elementos

Elemento principal	Elementos secundarios
<method>	<ul style="list-style-type: none"> <li>■ &lt;code&gt;</li> <li>■ &lt;description&gt;</li> </ul>

## Elemento code

El elemento `<code>` proporciona código de ejemplo que aparece en la documentación del Explorador de API.

El ejemplo de código se proporciona entre las etiquetas `<code>` y `</code>`. El elemento `<code>` es opcional. El elemento `<code>` no tiene atributos.

**Tabla 6-37. Jerarquía de elementos**

Elemento principal	Elementos secundarios
<code>&lt;example&gt;</code>	Ninguno

## Elemento method parameters

El elemento `<parameters>` es el contenedor de los elementos `<method>``<parameter>`.

El elemento `<parameters>` es opcional. El elemento `<parameters>` no tiene atributos.

**Tabla 6-38.**

Elemento principal	Elemento secundario
<code>&lt;method&gt;</code>	<code>&lt;parameter&gt;</code>

## Elemento method parameter

El elemento `<parameter>` define los parámetros de entrada del método.

El elemento `<parameter>` es opcional. Un complemento puede tener un número ilimitado de elementos `<parameter>`. El elemento `<parameter>` tiene los atributos siguientes.

Tipo	Valor	Descripción
name	Cadena	Nombre del parámetro. Atributo obligatorio.
type	Tipo de parámetro de Orchestrator	Tipo de parámetro. Atributo obligatorio.
is-optional	true o false	Si está establecido como true, el valor puede ser nulo. Atributo opcional.
since-version	Cadena	Versión de método. Atributo opcional.

**Tabla 6-39. Jerarquía de elementos**

Elemento principal	Elemento secundario
<code>&lt;parameters&gt;</code>	Ninguno

## Elemento singleton

El elemento `<singleton>` crea un objeto de creación de scripts de JavaScript como instancia de singleton.

Un objeto singleton se comporta del mismo modo que una clase estática de Java. Los objetos singleton definen objetos genéricos para su uso por parte del complemento, en lugar de definir instancias específicas de objetos a las que accede Orchestrator en la tecnología conectada. Por ejemplo, puede utilizar un objeto singleton para establecer la conexión con la tecnología conectada.

El elemento `<singleton>` es opcional. El elemento `<singleton>` tiene los atributos siguientes.

Tipo	Valor	Descripción
<code>script-name</code>	Objeto de JavaScript	Nombre del objeto de JavaScript correspondiente. Atributo obligatorio.
<code>datasource</code>	Objeto de Java	El objeto de origen de Java para este objeto de JavaScript. Atributo obligatorio.

**Tabla 6-40. Jerarquía de elementos**

Elemento principal	Elemento secundario
<code>&lt;object&gt;</code>	Ninguno

## Elemento enumerations

El elemento `<enumerations>` es el contenedor de los elementos `<enumeration>`.

El elemento `<enumerations>` es opcional. El elemento `<enumerations>` no tiene atributos.

**Tabla 6-41. Jerarquía de elementos**

Elemento principal	Elemento secundario
<code>&lt;module&gt;</code>	<code>&lt;enumeration&gt;</code>

## Elemento enumeration

El elemento `<enumeration>` define valores comunes que se aplican a todos los objetos de un tipo concreto.

Si todos los objetos de un tipo determinado requieren un cierto atributo, y si el rango de valores de dicho atributo es limitado, puede definir diferentes valores como entradas de enumeración. Por ejemplo, si un tipo de objeto requiere un atributo `color`, y si los únicos colores disponibles son el rojo, el azul y el verde, puede definir tres entradas de enumeración para establecer esos tres valores de color. Las entradas se definen como elementos secundarios del elemento de enumeración.

El elemento `<enumeration>` es opcional. Un complemento puede tener un número ilimitado de elementos `<enumeration>`. El elemento `<enumeration>` tiene el atributo siguiente.

Tipo	Valor	Descripción
<code>type</code>	Tipo de objeto de Orchestrator	Tipo de enumeración. Atributo obligatorio.

Tabla 6-42. Jerarquía de elementos

Elemento principal	Elementos secundarios
<enumerations>	<ul style="list-style-type: none"> <li>■ &lt;url&gt;</li> <li>■ &lt;description&gt;</li> <li>■ &lt;entries&gt;</li> </ul>

## Elemento entries

El elemento <entries> es el contenedor de los elementos <enumeration><entry>.

El elemento <entries> es opcional. El elemento <entries> no tiene atributos.

Tabla 6-43. Jerarquía de elementos

Elemento principal	Elemento secundario
<enumeration>	<entry>

## Elemento entry

El elemento <entry> proporciona un valor para un atributo de enumeración.

El elemento <entry> es opcional. Un complemento puede tener un número ilimitado de elementos <entry>. El elemento <entry> tiene los atributos siguientes.

Tipo	Valor	Descripción
id	Texto	El identificador que utilizan los objetos para establecer la entrada de enumeración como atributo. Atributo obligatorio.
name	Texto	El nombre de la entrada. Atributo obligatorio.

Tabla 6-44. Jerarquía de elementos

Elemento principal	Elementos secundarios
<entries>	Ninguno

# Recomendaciones para el desarrollo de complementos de Orchestrator

Podrá mejorar determinados aspectos de los complementos de Orchestrator que desarrolle si comprende la estructura y el contenido de los complementos, así como la forma de evitar determinados problemas.

- **Enfoques para crear complementos de Orchestrator**

Para crear complementos de Orchestrator se pueden utilizar diferentes enfoques. Se puede empezar por crear un complemento capa a capa, o se puede empezar por crear todas las capas del complemento al mismo tiempo.

- **Tipos de complementos de Orchestrator**

Mediante el uso de complementos, es posible integrar bibliotecas o utilidades generales como XML o SSH, así como sistemas completos como vCloud Director, con Orchestrator. Según la tecnología que se integre con Orchestrator, los módulos se pueden categorizar como complementos para los servicios, o bien como complementos generales o complementos para sistemas.

- **Implementación de complementos**

Puede usar técnicas y recomendaciones útiles para estructurar sus complementos, implementar las clases de Java y los objetos JavaScript requeridos, desarrollar los flujos de trabajo y las acciones del complemento y proporcionar la presentación del flujo de trabajo.

- **Recomendaciones para el desarrollo de complementos de Orchestrator**

El cumplimiento de ciertas recomendaciones durante el desarrollo de diversos componentes de los complementos de Orchestrator contribuye a una mejor calidad de los complementos.

- **Documentación de API y cadenas de interfaz de usuario para complementos**

Cuando escriba cadenas de la interfaz de usuario (IU) para complementos de Orchestrator y la documentación de las API relacionadas, siga las reglas de estilo y formato aceptadas.

## Enfoques para crear complementos de Orchestrator

Para crear complementos de Orchestrator se pueden utilizar diferentes enfoques. Se puede empezar por crear un complemento capa a capa, o se puede empezar por crear todas las capas del complemento al mismo tiempo.

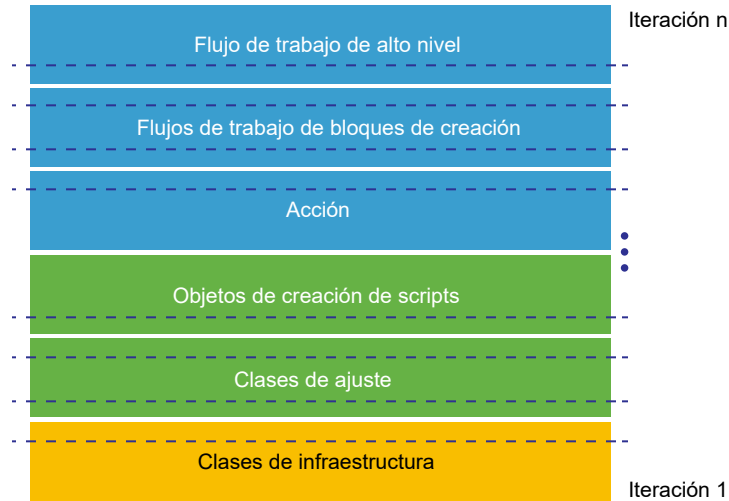
Para obtener información sobre las capas de los complementos, consulte [Estructura de un complemento de Orchestrator](#).

## Desarrollo de complementos de abajo arriba

Un complemento se puede crear capa por capa mediante el enfoque de desarrollo de abajo arriba.

El enfoque de desarrollo de abajo arriba crea el complemento capa por capa desde las capas de nivel inferior hasta las de nivel superior. Cuando este enfoque se combina con un enfoque de desarrollo interactivo e iterativo, se crea toda la capa o parte de ella para cada iteración. Tras un número N de iteraciones, el complemento se termina.



**Figura 6-3. Desarrollo de complementos de abajo arriba**

Una ventaja del enfoque de desarrollo de abajo arriba es que el desarrollo se efectúa capa por capa.

Sin embargo, también es necesario tener en cuenta las desventajas siguientes del enfoque de desarrollo de complementos de abajo arriba.

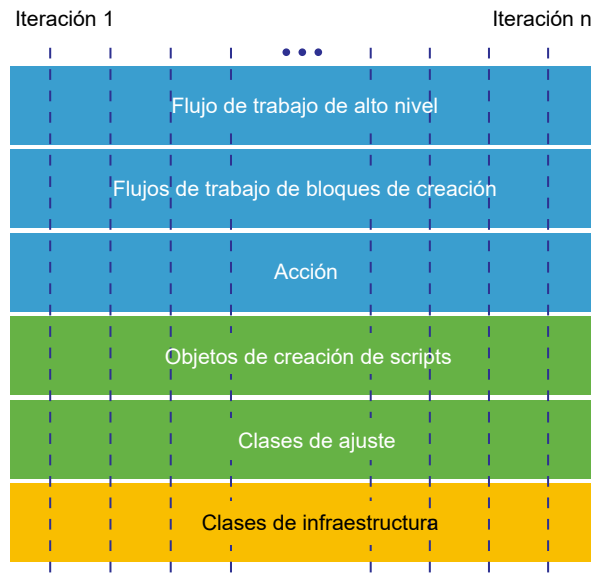
- El progreso del desarrollo de complementos es difícil de ver hasta que no se completan ciertas inserciones.
- No encaja muy bien en las prácticas de desarrollo de Agile.

El proceso de desarrollo de abajo arriba se considera apto para complementos pequeños, que tengan pocos objetos de creación de scripts, clases de ajuste, acciones o flujos de trabajo, o bien que no los tengan.

## Desarrollo de complementos de arriba abajo

Un complemento se puede crear dividiéndolo en una funcionalidad de arriba abajo mediante el enfoque de desarrollo de arriba abajo.

Cuando el enfoque de arriba abajo se combina con un proceso de desarrollo de Agile, se ofrece nueva funcionalidad para cada iteración. Como resultado, al final de la iteración N, el complemento se implementa del todo.

**Figura 6-4. Desarrollo de complementos de arriba abajo**

El enfoque de desarrollo de complementos de arriba abajo presenta las ventajas siguiente.

- El progreso del desarrollo de complementos es fácil de mostrar desde la primera iteración, ya que la nueva funcionalidad se completa para cada iteración, y el complemento se puede lanzar y utilizar después de cada iteración.
- Completar una división vertical de la funcionalidad permite establecer criterios de éxito bien definidos y determinar lo que se ha efectuado. Además, posibilita una mejor comunicación entre desarrolladores, ingenieros de administración de productos e ingenieros de control de calidad (CC).
- Permite a los ingenieros de CC empezar las pruebas y la automatización desde el inicio del proceso de desarrollo. Gracias a este enfoque, se obtienen comentarios valiosos y se reduce el tiempo de entrega global del proyecto.

Una desventaja del enfoque de desarrollo de complementos de arriba abajo es que el desarrollo está activo en diferentes capas al mismo tiempo.

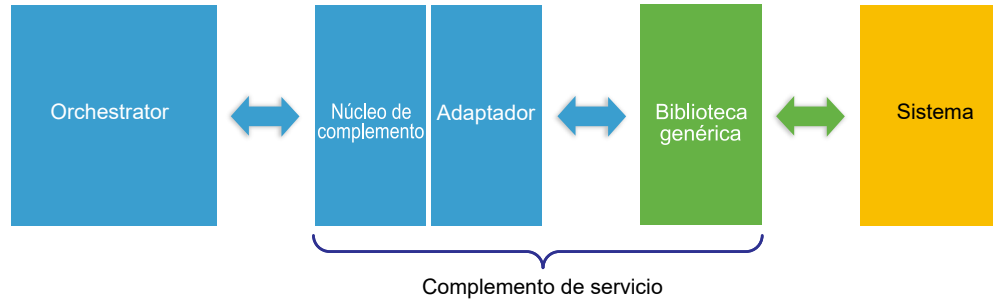
Es necesario aplicar el proceso de desarrollo de complementos de arriba abajo para la mayoría de los complementos. Resulta adecuado para los complementos con requisitos dinámicos.

## Tipos de complementos de Orchestrator

Mediante el uso de complementos, es posible integrar bibliotecas o utilidades generales como XML o SSH, así como sistemas completos como vCloud Director, con Orchestrator. Según la tecnología que se integre con Orchestrator, los módulos se pueden categorizar como complementos para los servicios, o bien como complementos generales o complementos para sistemas.

### Complementos para servicios

Los complementos para servicios o complementos de propósito general ofrecen funciones que pueden considerarse como un servicio dentro de Orchestrator.

**Figura 6-5. Arquitectura de complementos para servicios**

Los complementos para servicios exponen a Orchestrator utilidades o bibliotecas genéricas tales como XML, SSH o SOAP. Por ejemplo, estos complementos disponibles en Orchestrator son complementos para servicios:

<b>Complemento de JDBC</b>	Le permite usar cualquier base de datos dentro de un flujo de trabajo.
<b>Complemento de correo</b>	Le permite enviar correo electrónico dentro de un flujo de trabajo.
<b>Complemento de SSH</b>	Le permite abrir conexiones SSH y ejecutar comandos dentro de un flujo de trabajo.
<b>Complemento de XML</b>	Le permite administrar documentos XML dentro de un flujo de trabajo.

Los complementos para servicios tienen estas características:

<b>Complejidad</b>	Los complementos para servicios tienen niveles de complejidad bajos o intermedios. Los complementos para servicios exponen una biblioteca específica, o parte de una, dentro de Orchestrator para proporcionar funciones concretas. Por ejemplo, el complemento de XML añade una implementación de analizador de modelo de objetos de documento (DOM) XML a la API de JavaScript de Orchestrator.
<b>Tamaño</b>	Los complementos para servicios suelen ser relativamente pequeños. Requieren el mismo conjunto de clases básico que los demás complementos, y también necesitan otras clases que ofrezcan nuevos objetos de creación de scripts para añadir funciones nuevas.
<b>Inventario</b>	Los complementos para servicios requieren un inventario de objetos pequeño o puede que no lo necesiten en absoluto. Los complementos para servicios tienen un modelo de objetos pequeño y genérico, que no necesitan mostrar dentro del inventario de Orchestrator.

## Complementos para sistemas

Los complementos para sistemas conectan el motor de flujos de trabajo de Orchestrator a un sistema externo, para permitirle orquestar este último.

Los siguientes son ejemplos de complementos para sistemas:

<b>Complemento de vCenter Server</b>	Le permite administrar instancias de vCenter Server mediante flujos de trabajo.
<b>Complemento de vCloud Director</b>	Le permite interactuar con una instalación de vCloud Director dentro de un flujo de trabajo.
<b>Complemento de Cisco UCSM</b>	Le permite interactuar con entidades Cisco dentro de un flujo de trabajo.

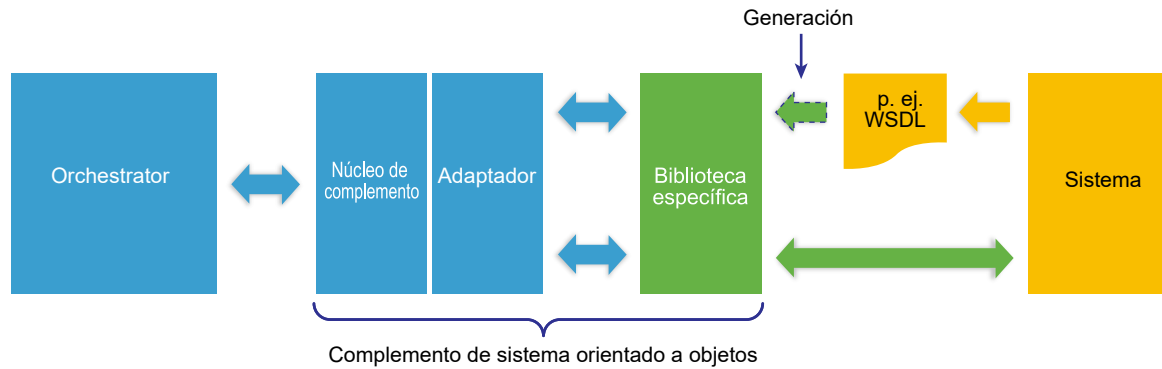
Estas son las principales características de los complementos para sistemas.

<b>Complejidad</b>	Los complementos para sistemas tienen un nivel de complejidad mayor que el de los complementos con propósito general, porque las tecnologías que exponen son relativamente complejas. Los complementos para sistemas deben representar todos los elementos del sistema externo dentro de Orchestrator para interactuar con el sistema externo y ofrecer sus funciones en Orchestrator. Si el sistema externo proporciona un mecanismo de integración, puede usarlo para exponer las funciones del sistema en Orchestrator más fácilmente. No obstante, además de representar los elementos del sistema externo en Orchestrator, los complementos para sistemas también podrían tener que ofrecer amplia escalabilidad, proporcionar un mecanismo de almacenamiento en caché, gestionar eventos y notificaciones, etc.
<b>Tamaño</b>	Los complementos para sistemas son de tamaño de mediano a grande. Los complementos para sistemas requieren muchas clases aparte de los conjuntos de clases básicos, ya que normalmente ofrecen un gran número de objetos de creación de scripts. Los complementos para sistemas podrían requerir otras clases auxiliares y de ayuda con las que interactuar.
<b>Inventario</b>	Por lo general, los complementos para sistemas tienen un gran número de objetos, que usted debe exponer debidamente en el inventario para poder localizarlos y trabajar fácilmente con ellos en Orchestrator. Dado el elevado número de objetos que deben exponer los complementos para sistemas, le convendría crear una herramienta auxiliar o un proceso para generar automáticamente tanto código como sea posible para el complemento. Por ejemplo, el complemento de vCenter Server proporciona una herramienta de ese tipo.

## Complementos para sistemas orientados a objetos

Los sistemas orientados a objetos ofrecen un mecanismo de interacción basado en objetos y RPC.

El modelo más usado para un sistema orientado a objetos es el de un servicio web con SOAP. Los objetos dentro de este modelo tienen una serie de atributos relacionados con el estado de esos objetos y ofrecen un conjunto de métodos remotos que se invocan en el lado del sistema de destino.

**Figura 6-6. Complementos para sistemas orientados a objetos**

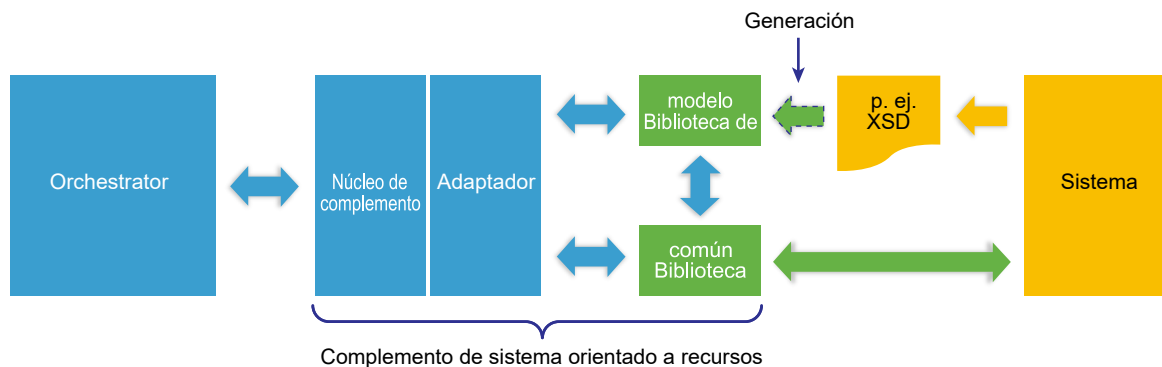
Al implementar complementos para sistemas orientados a objetos, le conviene tener en cuenta lo siguiente:

- Si usa SOAP, puede utilizar el archivo WSDL para generar un conjunto de clases que combine el modelo de objetos y el mecanismo de comunicación.
- Este modelo de objetos es casi lo único que tiene que exponer dentro de Orchestrator.

## Complementos para sistemas orientados a recursos

Los sistemas orientados a recursos proporcionan un mecanismo de interacción basado en recursos y operaciones sencillas que usan métodos HTTP.

El modelo más representativo para un sistema orientado a recursos es REST combinado, por ejemplo, con XML. Los objetos dentro de este modelo tienen una serie de atributos relacionados con el estado de esos objetos. Para invocar métodos en el sistema de destino (mecanismo de comunicación), debe usar el método HTTP estándar, por ejemplo GET, POST o PUT y seguir ciertas convenciones.

**Figura 6-7. Complementos para sistemas orientados a recursos**

Al desarrollar complementos para sistemas orientados a recursos, le conviene tener en cuenta lo siguiente:

- Si usa REST o únicamente HTTP con XML, obtiene uno o más archivos de esquema XML para leer y escribir mensajes. A partir de esos esquemas, puede generar una serie de clases que definen el

modelo de objetos. Esta serie de clases solo define el estado de los objetos, ya que las operaciones se definen implícitamente con los métodos HTTP, por ejemplo según lo definido en el complemento de vCloud Director, o explícitamente con mensajes XML específicos, como es el caso en el complemento de Cisco UCSM.

- Debe implementar el mecanismo de comunicación en otro conjunto de clases. Este conjunto de clases define un nuevo modelo de objetos que interactúa con el modelo original. El modelo de objetos para el mecanismo de comunicación se compone únicamente de objetos y métodos.
- Puede exponer tanto el modelo de objetos original como el modelo de objetos para el mecanismo de comunicación dentro de Orchestrator. Esto podría aumentar la complejidad, según cómo se expongan ambos modelos de objetos y también dependiendo de si se van a combinar objetos relacionados de ambos lados (para simular un sistema orientado a objetos) o se van a mantener separados.

## Implementación de complementos

Puede usar técnicas y recomendaciones útiles para estructurar sus complementos, implementar las clases de Java y los objetos JavaScript requeridos, desarrollar los flujos de trabajo y las acciones del complemento y proporcionar la presentación del flujo de trabajo.

- **Estructura de los proyectos**

Puede aplicar una estructura estándar para los proyectos de complementos de Orchestrator.

- **Elementos internos del proyecto**

Al implementar un complemento puede aplicar diversos enfoques, como usar objetos de caché y objetos en el fondo, clonar objetos, etc. De esa forma se puede mejorar el rendimiento de los complementos, evitar problemas de simultaneidad y mejorar la respuesta del cliente de Orchestrator.

- **Elementos internos del flujo de trabajo**

Puede implementar un flujo de trabajo para supervisar operaciones prolongadas realizadas por su complemento de Orchestrator.

- **Flujos de trabajo y acciones**

Para facilitar el desarrollo y el uso del flujo de trabajo, puede seguir ciertas recomendaciones.

- **Presentación del flujo de trabajo**

Al crear la presentación de un flujo de trabajo, debería aplicar una estructura y unas reglas determinadas.

## Estructura de los proyectos

Puede aplicar una estructura estándar para los proyectos de complementos de Orchestrator.

Puede utilizar una estructura Maven estándar con módulos para que los proyectos de complemento distingan dónde se encuentra cada función.

**Tabla 6-45. Estructura de un proyecto de complemento**

Módulo	Descripción
/myAwesomePlugin-plugin	Raíz del proyecto de complemento.
/o11nplugin-myAwesomePlugin	Módulo que compone el archivo DAR del complemento final.
/o11nplugin-myAwesomePlugin-config	Módulo que contiene la aplicación web de configuración del complemento. Genera un archivo WAR estándar.
/o11nplugin-myAwesomePlugin-core	Módulo que contiene todas las clases que implementan cualquier interfaz de módulo estándar de Orchestrator y otras clases auxiliares que utilicen. Genera un archivo JAR estándar.
/o11nplugin-myAwesomePlugin-model	Módulo que contiene todas las clases que permiten integrar tecnología de terceros con Orchestrator a través del complemento. Las clases no deben contener ninguna referencia directa a las API de complemento estándar de Orchestrator.
/o11nplugin-myAwesomePlugin-package	Módulo que importa un archivo de paquete externo de Orchestrator con acciones y flujos de trabajo para incluirlo dentro del archivo DAR del complemento final. El módulo es opcional.

## Elementos internos del proyecto

Al implementar un complemento puede aplicar diversos enfoques, como usar objetos de caché y objetos en el fondo, clonar objetos, etc. De esa forma se puede mejorar el rendimiento de los complementos, evitar problemas de simultaneidad y mejorar la respuesta del cliente de Orchestrator.

### Objetos de caché

El complemento puede interactuar con un servicio remoto, y esta interacción la proporcionan los objetos locales que representan a objetos remotos en el lado de servicio. Para lograr un buen rendimiento del complemento así como una buena respuesta de la interfaz de usuario de Orchestrator, puede poner en caché los objetos locales, en lugar de tener que obtenerlos de un servicio remoto cada vez que se requieren. Puede decidir el ámbito de la memoria caché, por ejemplo: una caché para todos los clientes del complemento, una por cada usuario del complemento y una por cada usuario del servicio de terceros. Al implementar el mecanismo de caché, se integra con la interfaz del complemento para encontrar e invalidar objetos.

### Usar objetos en el fondo

Si tiene que mostrar listas de objetos largas en el inventario del complemento y no tiene un método rápido para recuperar esos objetos, puede usar objetos en el fondo. Para ello puede, por ejemplo, tener objetos con dos estados: `fake` y `loaded`. Teniendo en cuenta que los objetos `fake` son muy fáciles de crear y proporcionan la información mínima que mostrar en el inventario (por ejemplo, el nombre y el ID), sería posible devolver siempre objetos `fake`, y cuando se necesita toda la información (el objeto real), la entidad que lo requiere o el complemento pueden llamar a un método `load` automáticamente para obtener el objeto real. Incluso puede configurar el proceso de carga de objetos para que se inicie automáticamente después de que se devuelvan los objetos falsos, para prever las acciones de la entidad que los usa.

## Clonar objetos para evitar problemas de simultaneidad

Si usa una caché para su complemento, tiene que clonar objetos. Puede haber efectos imprevistos si usa una caché que siempre devuelve la misma instancia de un objeto para cada entidad que lo solicita. Por ejemplo, la entidad A solicita el objeto O, y la entidad visualiza el objeto en el inventario con todos sus atributos. Al mismo tiempo, la entidad B solicita el objeto O también y la entidad A ejecuta un flujo de trabajo que empieza a cambiar los atributos del objeto O. Al final de la ejecución, el flujo de trabajo llama al método `update` del objeto para actualizar el objeto en el lado del servidor. Si las entidades A y B obtienen la misma instancia del objeto O, la entidad A ve en el inventario todos los cambios que realice la entidad B, incluso antes de que los cambios se validen en el lado del servidor. Si la ejecución se realiza correctamente, no debería haber problemas; pero si la ejecución falla, no se revertirán los atributos del objeto O para la entidad A. De ocurrir eso, si la caché (las operaciones de `find` en el complemento) devuelve un clon del objeto en lugar de la misma instancia continuamente, cada entidad ve y modifica su propia copia, y se evitan problemas de simultaneidad, al menos dentro de Orchestrator.

## Notificar cambios a otros

Se pueden producir problemas si se usa una caché mientras se clonan objetos. El problema más grave es que el objeto que usa vistas de entidad podría no ser la versión más reciente disponible. Por ejemplo, si una entidad muestra el inventario, los objetos se cargan una vez pero, al mismo tiempo, si otra entidad está cambiando objetos, la primera entidad no ve esos cambios. Para evitar este problema puede usar los métodos `PluginWatcher` y `IPluginPublisher` de la API de complemento de Orchestrator para notificar que algo ha cambiado para permitir que otras instancias de clientes de Orchestrator vean los cambios. Esto también es aplicable a una sola instancia del cliente de Orchestrator cuando los cambios de un objeto del inventario afectan a otros objetos del inventario y también es necesario notificarlos. Las operaciones que más tienden a usar notificaciones son las de añadir, actualizar y eliminar objetos cuando estos, o algunas de sus propiedades, aparecen en el inventario.

## Activar la localización de cualquier objeto en cualquier momento

Debe implementar el método `find` de la interfaz `IPluginFactory` para encontrar objetos solo por tipo e ID. El método `find` se puede llamar directamente después del reinicio de Orchestrator y la reanudación de un flujo de trabajo.

## Simular un servicio de consultas si no existe uno

El cliente de Orchestrator puede requerir consultas para algunos objetos en casos específicos o que se muestren no en estructura de árbol sino como listas o tablas, por ejemplo. Esto significa que el complemento debe poder realizar consultas de conjuntos de objetos en cualquier momento. Si la tecnología de terceros ofrece un servicio de consultas, tendrá que adaptarse a usar este servicio. De no hacerlo así, tal vez pueda simular este tipo de servicio, aunque podría ser una solución más compleja o de rendimiento inferior.

## Los métodos de búsqueda no deberían devolver excepciones de tiempo de ejecución

Los métodos de la interfaz `IPluginFactory` que implementan las búsquedas dentro del complemento no deberían devolver excepciones de tiempo de ejecución, ni controladas ni no controladas. Esto podría causar extraños fallos de error de validación *error de validación* durante la ejecución de un flujo de trabajo. Por ejemplo, entre dos nodos de un flujo de trabajo, se llama al método `find` si una salida del



primer nodo es una entrada del segundo nodo. En ese momento, si el objeto no se encuentra por haberse producido una excepción de tiempo de ejecución, puede que lo único que obtenga en el cliente de Orchestrator sea un *error de validación*. Después de eso, obtendrá más o menos información en los archivos de registro dependiendo de cómo registra excepciones el complemento.

## Elementos internos del flujo de trabajo

Puede implementar un flujo de trabajo para supervisar operaciones prolongadas realizadas por su complemento de Orchestrator.

Puede implementar un flujo de trabajo para supervisar operaciones largas, como la supervisión de tareas. Este flujo de trabajo se puede basar en eventos en espera y activadores de Orchestrator. Tenga en cuenta que un flujo de trabajo bloqueado a la espera de una tarea se puede reanudar en cuanto se inicia el servidor de Orchestrator. El complemento debe poder obtener toda la información requerida para que el proceso de supervisión se reanude correctamente.

El flujo de trabajo de supervisión o la tarea que puede usar internamente deberían proporcionar un mecanismo para especificar la tasa de sondeo y un posible periodo de tiempo de espera.

El proceso de depuración de una porción de código de script dentro de un flujo de trabajo no es una tarea simple, especialmente si el código no invoca ningún código Java. Por esa razón, a veces la única opción consiste en usar métodos de registro ofrecidos por los objetos de script predeterminados de Orchestrator.

## Flujos de trabajo y acciones

Para facilitar el desarrollo y el uso del flujo de trabajo, puede seguir ciertas recomendaciones.

### Iniciar el desarrollo de los flujos de trabajo como bloques de creación

Un bloque de creación puede ser un flujo de trabajo sencillo que requiere unos cuantos parámetros de entrada y devuelve una salida simple. Si tiene un conjunto amplio de bloques de creación, puede crear flujos de trabajo de alto nivel fácilmente, y ofrecer mejores herramientas para componer flujos de trabajo complejos.

### Crear flujos de trabajo de nivel más alto basados en componentes menores

Si tiene que desarrollar un flujo de trabajo complejo con varias entradas y pasos internos, puede dividirlo en acciones y flujos de trabajo con bloques de creación menores y más simples.

### Crear acciones siempre que sea posible

Puede crear acciones para tener más flexibilidad al desarrollar flujos de trabajo.

- Para crear fácilmente parámetros u objetos complejos para métodos de creación de scripts
- Para evitar repetir constantemente elementos comunes de código
- Para realizar validaciones de interfaz de usuario

## Los flujos de trabajo deberían invocar acciones siempre que sea posible

Las acciones pueden invocarse directamente como nodos dentro del esquema de flujo de trabajo. Esto puede contribuir a un esquema de flujo de trabajo más simple, al no haber necesidad de añadir bloques de código de script para invocar una sola acción.

## Rellenar la información prevista

Proporcione información para cada elemento de un flujo de trabajo o una acción.

- Proporcione una descripción del flujo de trabajo o la acción.
- Proporcione una descripción de los parámetros de entrada.
- Proporcione una descripción de las salidas.
- Proporcione una descripción de los atributos para los flujos de trabajo.

## Mantener actualizada la información de versión

Al crear versiones de complementos, añada comentarios relevantes con datos como actualizaciones importantes aplicadas al complemento, detalles de implementaciones, etc.

## Presentación del flujo de trabajo

Al crear la presentación de un flujo de trabajo, debería aplicar una estructura y unas reglas determinadas.

Use las siguientes propiedades para las entradas de flujo de trabajo en la presentación.

**Tabla 6-46. Propiedades para entradas de flujo de trabajo**

Propiedades	Uso
Show in Inventory	Use esta propiedad para ayudar al usuario a ejecutar un flujo de trabajo desde la vista de inventario.
Specify a root object to be shown in the chooser	Use esta propiedad para ayudar al usuario a seleccionar entradas. Si el objeto raíz se puede actualizar en la presentación, es un atributo o se recupera mediante un método de objetos, tendrá que crear o configurar una acción adecuada para actualizar el objeto en la presentación.
Maximum string length	Use esta propiedad para cadenas largas, como nombres, descripciones, rutas de archivo, etc.
Minimum string length	Use esta propiedad para evitar cadenas vacías de herramientas de prueba.
Custom validation	Sirve para implementar validaciones que no sean simples con acciones.

Organice las entradas con pasos y un grupo de visualización. Dicha organización ayuda al usuario a identificar y distinguir todos los parámetros de entrada de un flujo de trabajo.

## Recomendaciones para el desarrollo de complementos de Orchestrator

El cumplimiento de ciertas recomendaciones durante el desarrollo de diversos componentes de los complementos de Orchestrator contribuye a una mejor calidad de los complementos.

**Tabla 6-47. Recomendaciones útiles para la implementación de complementos**

Componente	Elemento	Descripción
General	Acceso a API de terceros	Los complementos deberían proporcionar métodos simplificados para acceder a API de terceros siempre que sea posible.
	Interfaz	Los complementos deberían proporcionar una interfaz estándar para los usuarios, aunque la API no lo haga.
Acción	Objetos de creación de scripts	Debería crear acciones para cada creación, modificación, eliminación y los demás métodos disponibles para un objeto de creación de scripts.
	Descripción	La descripción de una acción debería describir qué hace la acción, no cómo funciona.
	Creación de scripts	Al usar la creación de scripts para obtener las propiedades o métodos de un objeto, puede comprobar si el valor del objeto es distinto de null o undefined.
	En desuso	Si una acción está en desuso, la instrucción comment o throw debería indicar la acción de reemplazo; o la acción debería llamar a una nueva acción de reemplazo para que no fallen las soluciones basadas en la versión en desuso.
Flujo de trabajo	Operaciones de interfaz de usuario en la tecnología orquestada	Debería crear un flujo de trabajo para cada operación disponible en la interfaz de usuario de la tecnología orquestada.
	Descripción	La descripción de un flujo de trabajo debería describir qué hace el flujo, no cómo funciona.
	Propiedad de presentación mandatory input	Debe configurar la propiedad mandatory input para todas las entradas de flujo de trabajo obligatorias.
	Propiedad de presentación default value	Si desarrolla un flujo de trabajo que configura una entidad, la presentación del flujo de trabajo debería cargar los valores de configuración predeterminados para esta entidad. Por ejemplo, si desarrolla un flujo de trabajo denominado Configuración de host, la presentación del flujo de trabajo debe cargar los valores predeterminados de la configuración de host.
	Propiedad de presentación Show in inventory	Debe configurar la propiedad Show in inventory para tener flujos de trabajo contextuales en objetos de inventario.
	Propiedad de presentación specify a root parameter	Debería usar esta propiedad en flujos de trabajo cuando no sea necesario examinar el inventario desde la raíz del árbol.
	Validación de flujos de trabajo	Debe validar los flujos de trabajo y corregir todos los errores.
	Creación de objetos	Todos los flujos de trabajo que crean un objeto nuevo deberían devolver este como parámetro de salida.

**Tabla 6-47. Recomendaciones útiles para la implementación de complementos (continuación)**

Componente	Elemento	Descripción
	En desuso	Si se deja de usar un flujo de trabajo, las instrucciones <code>comment</code> o <code>throw</code> deberían indicar el flujo de trabajo de reemplazo; o el flujo de trabajo debería llamar a un nuevo flujo de trabajo de reemplazo para que no fallen las soluciones basadas en la versión en desuso.
Inventario	Desconexión de host	Si el inventario contiene una conexión a un host y este deja de estar disponible, debería indicar que el host se ha desconectado. Puede hacerlo asignando otro nombre al objeto raíz, añadiéndole <code>-disconnected</code> , o quitando el árbol de objetos debajo de este, como lo hace el complemento de vCloud Director.
	Propiedad <code>Select value as list</code>	Un objeto de inventario debe estar seleccionable como <code>treeview</code> o como <code>list</code> .
	Administrador de host	Si el complemento implementa un objeto <code>host</code> para el sistema de origen, debería existir un objeto raíz <code>hostmanager</code> principal con propiedades para agregar, quitar y editar propiedades de <code>host</code> .
	Obtener o actualizar objetos	Si hay un servicio de consultas en ejecución en la tecnología orquestada, debería usarlo para obtener varios objetos.
	Detección de objetos secundarios	Si tiene que recuperar objetos secundarios uno por uno, el proceso de recuperación debe ser de tipo multiproceso y no bloqueable por un solo error.
	Cambio de objetos de Orchestrator	Todos los flujos de trabajo que pueden cambiar el estado de un elemento en el inventario deben actualizar este para evitar que haya objetos desincronizados.
	Cambio de objetos externos	Puede usar un mecanismo de notificación de cambios producidos en la tecnología orquestada como resultado de operaciones realizadas fuera de Orchestrator. Si esas operaciones provocan la desaparición de objetos de la tecnología orquestada, tendrá que actualizar el inventario según corresponda, para evitar fallos o pérdidas de datos. Por ejemplo, si una máquina virtual se elimina de vCenter Server, el complemento de vCenter Server actualiza el inventario para quitar el objeto de la máquina virtual eliminada.
	Objeto de buscador	Los objetos de buscador deberían tener propiedades utilizables para diversos objetos. Estas propiedades son las que suelen estar presentes en la interfaz de usuario.
Objeto de creación de scripts	Implementación	Se debe implementar el método <code>equals</code> para asegurar el funcionamiento de la operación <code>==</code> en el mismo objeto, ya que en ciertos casos podría haber dos instancias de un objeto.
	Propiedades de objetos de complementos	Los objetos que tienen objetos principales deberían implementar una propiedad <code>parent</code> .
	Propiedades de objetos de complementos	Los objetos que tienen objetos secundarios deberían implementar métodos <code>GET</code> que devuelvan matrices de objetos secundarios.
	Objetos de inventario	Los objetos de inventario deberían admitir búsquedas con <code>Server.find</code> .

**Tabla 6-47. Recomendaciones útiles para la implementación de complementos (continuación)**

Componente	Elemento	Descripción
		Todos los objetos de inventario deberían ser serializables para poder usarse como atributos de entrada o salida de un flujo de trabajo.
	Constructor y métodos	En la mayoría de los casos, los objetos de scripts deberían tener un constructor o ser devueltos por otros métodos o atributos de objeto.
	ID de objeto	Los objetos con un ID emitido por un sistema externo deberían usar un ID interno para asegurar que no hay duplicaciones de ID al orquestar más de un servidor.
	Buscar objetos	Los métodos <code>search</code> o <code>find</code> deberían implementar un filtro para encontrar el nombre o ID especificado, en lugar de todos los objetos. Por ejemplo, el servidor de Orchestrator tiene un método <code>Server.FindForId</code> que permite encontrar un objeto de complemento por su ID. Para ello, el método se debe implementar para cada objeto localizable en el complemento.
	Activador	De ser posible, debería haber activadores disponibles para objetos cambiantes, de modo que se activen políticas de Orchestrator al producirse diversos eventos. Por ejemplo, para determinar cuándo se agrega, enciende, apaga, etc. una máquina virtual, Orchestrator puede supervisar un activador o un evento en el complemento de vCenter del objeto Datacenter.
	Propiedades de objetos	Los objetos que residen en otros complementos deberían tener propiedades que faciliten la conversión de un objeto de complemento a otro. Por ejemplo, los objetos de máquina virtual deben tener un <code>moref</code> (ID de referencia de objeto administrado).
	Administrador de sesión	Si se va a conectar a un servidor remoto que puede tener otra sesión, el complemento debería implementar una sesión compartida y una sesión por usuario.
Activador	Activador	Todos los métodos de bloqueo y operaciones largas deberían poder iniciarse de forma asíncrona con una tarea devuelta, y generar un evento de activador al completarse.
Enumeraciones	Enumeraciones	Las enumeraciones para un tipo determinado deberían tener un objeto de inventario que permita seleccionar entre los diversos valores de la enumeración.
Registro	registros	Los métodos deberían implementar distintos niveles de registro.
Control de versiones	Versión de complemento	La versión de complemento debería ser acorde con los estándares y actualizarse junto con el complemento.
documentación de las API	Métodos	Los métodos descritos en la documentación de API nunca deberían devolver la excepción <code>no xyz method / property</code> para un objeto. Deberían devolver <code>null</code> cuando no hay propiedades disponibles y documentarse con detalles cuando estas propiedades no están disponibles.
	<code>vso.xml</code>	Todos los objetos, métodos y propiedades deben documentarse en <code>vso.xml</code> .

## Documentación de API y cadenas de interfaz de usuario para complementos

Cuando escriba cadenas de la interfaz de usuario (IU) para complementos de Orchestrator y la documentación de las API relacionadas, siga las reglas de estilo y formato aceptadas.

### Recomendaciones generales

- Utilice los nombres oficiales de los productos de VMware que forman parte del complemento. Por ejemplo, utilice los nombres oficiales de los productos siguientes y la terminología de VMware.

Término correcto	No utilizar
vCenter Server	VC o vCenter
vCloud Director	vCloud

- Termine todas las descripciones de flujos de trabajo con un punto. Por ejemplo, `Creates a new Organization.` es una descripción de flujo de trabajo.
- Utilice un editor de texto con un corrector ortográfico para escribir las descripciones y luego páselas al complemento.
- Asegúrese de que el nombre del complemento coincida exactamente con el nombre de producto de terceros aprobado con el que esté asociado.

### Flujos de trabajo y acciones

- Escriba descripciones informativas. Una o dos frases son suficientes para la mayoría de las acciones y los flujos de trabajo.
- Los flujos de trabajo de nivel superior podrían incluir descripciones y comentarios más extensos.
- Inicie las descripciones con un verbo, por ejemplo, `Creates....` No utilice lenguaje que se haga referencia a sí mismo, como `This workflow creates.`
- Coloque un punto al final de las descripciones que sean frases completas.
- Describa lo que hacen un flujo de trabajo o una acción en lugar de cómo se implementan.
- Los flujos de trabajo y las acciones se suelen incluir en carpetas y paquetes. Incluya también una breve descripción de estas carpetas y paquetes. Por ejemplo, una carpeta de flujo de trabajo puede tener una descripción similar a `Set of workflows related to vApp Template management.`

### Parámetros de flujos de trabajo y acciones

- Inicie las descripciones de flujos de trabajo y acciones con un sintagma nominal descriptivo, por ejemplo `Name of.` No utilice frases del tipo `It's the name of.`
- No coloque punto al final de las descripciones de parámetros y acciones. No son frases completas.

- Los parámetros de entrada de los flujos de trabajo deben especificar una etiqueta con nombres apropiados en la vista de presentación. En muchos casos, puede combinar entradas relacionadas en un grupo de visualización. Por ejemplo, en lugar de tener dos entradas con las etiquetas Nombre de empresa y Nombre completo de la empresa, puede crear un grupo de visualización con la etiqueta Empresa y colocar las entradas Nombre y Nombre completo en el grupo Empresa.
- Para los pasos y los grupos de visualización, añada descripciones o comentarios que también aparezcan en la presentación del flujo de trabajo.

## API de complemento

- La documentación de la API hace referencia a toda la documentación del archivo `vso.xml` y los archivos de origen de Java.
- Para el archivo `vso.xml`, utilice las mismas reglas para las descripciones de los objetos de buscador y de creación de scripts con sus métodos que las que utiliza para los flujos de trabajo y las acciones. Las descripciones de los atributos de objetos y los parámetros de método utilizan las mismas reglas que los parámetros de flujos de trabajo y las acciones.
- Evite los caracteres especiales en el archivo `vso.xml` e incluya las descripciones dentro de una etiqueta `<![CDATA[insert your description here!]]>`.
- Utilice el estilo de Javadoc estándar para los archivos de origen de Java.

# Creación de complementos mediante Maven

# 7

Orchestrator Appliance proporciona un repositorio con artefactos de Maven, que se pueden utilizar para crear proyectos de complementos a partir de arquetipos.

El repositorio se aloja en `https://servidor_orchestrator:8281/vco-repo/` o `http://servidor_orchestrator:8280/vco-repo/`, en caso de que su versión de Maven no admita el protocolo HTTPS. Esta ubicación está integrada en el archivo `pom.xml` de los proyectos de complementos estándar Maven de Orchestrator. Solo es posible acceder al repositorio si se ha implementado Orchestrator Appliance.

Este capítulo incluye los siguientes temas:

- [Crear un complemento de Orchestrator con Maven a partir de un arquetipo](#)
- [Arquetipos de Maven](#)
- [Procedimientos recomendados para el desarrollo de complementos basados en Maven](#)

## Crear un complemento de Orchestrator con Maven a partir de un arquetipo

Puede crear un complemento estándar de Orchestrator Maven a partir de un arquetipo mediante la ejecución de comandos en la interfaz de línea de comandos.

### Requisitos previos

- Verifique esté instalado Orchestrator Appliance 5.5.1 o una versión posterior.
- Verifique esté instalado Apache Maven 3.0.4 o 3.0.5.

### Procedimiento

- 1 Cree un proyecto en modo interactivo eligiendo un arquetipo.

```
mvn archetype:generate -DarchetypeCatalog=https://servidor_orchestrator:8281/vco-repo/archetype-catalog.xml -DrepoUrl=https://servidor_orchestrator:8281/vco-repo -Dmaven.repo.remote=https://servidor_orchestrator:8281/vco-repo -Dmaven.wagon.http.ssl.insecure=true -Dmaven.wagon.http.ssl.allowall=true
```

---

**Nota** Solo es posible acceder al repositorio Maven si se ha implementado Orchestrator Appliance.

---



- 2 (opcional) Si no puede acceder al repositorio por HTTPS, inténtelo por HTTP. Si accede al repositorio por HTTP o dispone de un certificado SSL válido, puede crear un proyecto sin utilizar el indicador `-Dmaven.wagon.http.ssl.allowall=true`.

```
mvn archetype:generate -DarchetypeCatalog=http://servidor_orchestrator:8280/vco-repo/archetype-catalog.xml -DrepoUrl=http://servidor_orchestrator:8280/vco-repo -Dmaven.repo.remote=http://servidor_orchestrator:8280/vco-repo -Dmaven.wagon.http.ssl.insecure=true
```

- 3 Acceda al directorio de proyectos y compile el complemento.

```
cd dir_proyecto && mvn clean install -Dmaven.wagon.http.ssl.insecure=true -Dmaven.wagon.http.ssl.allowall=true
```

Si el proceso de compilación se realiza correctamente, el archivo `.dar` del complemento se genera en el directorio `target/` del módulo DAR.

## Arquetipos de Maven

Se puede utilizar un conjunto de arquetipos de Maven como plantillas para desarrollar complementos de Orchestrator.

La siguiente tabla describe los arquetipos de Maven predeterminados disponibles en Orchestrator.

**Tabla 7-1. Arquetipos de Maven predeterminados**

Arquetipo	Descripción
<code>com.vmware.o11n:011n-plugin-archetype-simple</code>	<code>com.vmware.o11n:011n-plugin-archetype-simple</code>
<code>com.vmware.o11n:011n-package-archetype</code>	Un proyecto de Maven solo de contenido, que se puede usar para mantener paquetes en forma de origen para una mejor interacción con RCS, diff, posprocesamiento, etc.
<code>com.vmware.o11n:011n-client-archetype-rest</code>	Una simple herramienta de línea de comandos, que se comunica con la API de REST de Orchestrator y llama a un flujo de trabajo.
<code>com.vmware.o11n:011n-plugin-archetype-inventory</code>	Un complemento que demuestra el uso del inventario. El complemento implementa un repositorio, un adaptador y una fábrica para un solo tipo. El inventario se almacena en un archivo en un disco.
<code>com.vmware.o11n:011n-archetype-inventory-annotation</code>	Un complemento cuyo descriptor <code>vso.xml</code> se genera encima de las anotaciones.
<code>com.vmware.o11n:011n-archetype-spring</code>	Un complemento que utiliza SDK basado en Spring proporciona un entorno habilitado para DI y añade servicios de nivel superior en comparación con las API de complemento estándar.
<code>com.vmware.o11n:011n-plugin-archetype-modeldriven</code>	Un arquetipo que genera un esqueleto de complemento para crear complementos con ModelDriven.

## Procedimientos recomendados para el desarrollo de complementos basados en Maven

Puede mejorar el proceso de creación de complementos de Orchestrator con Maven mediante una serie de tareas.

### Usar un administrador de repositorios

Si va a crear complementos en una organización de gran tamaño, use un administrador de repositorios empresarial para configurar el repositorio predeterminado de la solución Orchestrator Appliance que añadir como repositorio proxy. El uso de un repositorio central mejora la administración y la colaboración de proyectos con complementos. Al completar la primera compilación en el nuevo repositorio, el administrador de repositorios guarda en caché los artefactos del repositorio de Orchestrator Appliance y usted puede desactivar el repositorio predeterminado.

### Bloquear flujos de trabajo

Después de verificar que todos los flujos de trabajo del complemento funcionan según lo previsto, bloquéelos para impedir que se hagan modificaciones no autorizadas. Al bloquear flujos de trabajo se asegura la efectividad de las funciones básicas del complemento. Si los usuarios tienen que modificar un flujo de trabajo predeterminado con un propósito específico, pueden crear una copia del flujo de trabajo original y editarla.

Existen dos formas de producir compilaciones de versión con flujos de trabajo bloqueados.

- Envíe el parámetro `-DallowedMask=vf` a Maven.
- Edite el archivo `pom.xml` y cambie el valor del parámetro `allowedMask` a `vf`.

```
<allowedMask>vf</allowedMask>
```

### Usar un certificado de firma de paquetes

Use un certificado autofirmado o uno firmado por una entidad de certificación, para asegurar la integridad y autenticidad de los complementos. Almacene el certificado en el almacén de claves bajo el alias `_dunerssa_alias_`, importándolo a JDK mediante la utilidad `keytool`.

Hay dos formas de especificar la ruta del archivo de almacén de claves y la contraseña del almacén de claves.

- Defina los parámetros de línea de comandos `-DkeystoreLocation` y `-DkeystorePassword` para la variable `MAVEN_OPTS`.
- Edite el archivo `pom.xml` para insertar los valores manualmente. Por ejemplo:

```
<keystore>ruta de acceso al archivo de almacén de claves</keystore>
<storepass>contraseña de almacén de claves</storepass>
```

Si no se importa un almacén de claves, el archivo `.package` se firma con el archivo `archetype.keystore`.